MUS171_Miller_Music_Science_Perspectives

Announcer: [0:28] ...and Max as well. Miller studied mathematics at MIT and Harvard, as an NSF and Putnam fellow, but became a computer music researcher at the MIT Media Lab, and then at IRCAM, where he wrote Max. He now teaches music at UCSD and works on the pure data environment, and on a variety of musical and research projects. Please welcome Miller Puckette. [applause]

Miller: [0:59] I talk to scientists a lot. I am actually a co-worker of Ricardo who spoke this morning at Calit2. And we are non-scientists working in a scientific environment, or in a scientific and engineering environment. Me, I'm not only not a scientist, but I'm not an artist, either.

[1:22] I'm not even really an engineer, in the sense that I don't actually ever specify anything - I just go out and do things. The thing that I've been going out and doing since about 1997 is called Pure Data. I'm not going to talk a whole lot about it, because I'm just going to be using it. And you might be making inferences about what's going on as I do so.

What I really want to talk about is a more general topic, about which I have a very bad understanding - or a very unclear understanding - which is: [1:42] What really are the issues that come up and are important, when you try to make a computer be usable in a musical context?

[2:04] So, the reason I'm thinking and talking about this is because, well, all right... The thing that's correlated with why I'm thinking and talking about this now, is that I've spent the last 25 years of my life trying to make musical instruments out of computers. That was the original goal behind writing Maximus P, and was the goal behind writing Pd.

[2:31] It's kind of beside the point, or it's a nice side effect for me, if not only can it be musically useful. But it can be useful for other kinds of art forms or for visualization of scientific data, or for oralization of scientific data - even better. Or for all sorts of other things that I won't try to list, that people seem to turn this kind of environment to.

[3:03] What is it? I'm just telling you this so I can stop talking about it. If you like, what Max and Pd both are programming environments for using computers as reactive real-time tools. Of course, the input and output that you use - over which you talk to the computer - is a medium. So, in fact, it is an interactive programming environment for messing with media in real-time.

[3:30] Medium, I think, just means "between". The thing about it is, that you're not supposed to know that you're programming when you use it. So, there's nothing to this today. But in the middle environment in which I started all this work, it was a shocking thought that a composer would actually program a computer to do something.

[4:11] Composers were artistes, and they dealt with ink on paper, and made beautiful scores. The boffins - the people on the other side of the aisle - were the ones who took the scores, and turned them into beautiful sounds for the composers. This turns out to be a lousy way to make music. For a number of reasons. In fact, the same principal reason for which it is extremely rare that you will find a composer who can't play a musical instrument at a nearly professional or professional level. A good composer, I mean.

[4:22] There are lots of bad composers who've never touched a musical instrument, but you won't find a really top one. You won't find a Liszt, or a Schubert, or something like that, who can't get around a musical instrument.

[4:47] But it sort of follows that - or doesn't follow -there's no logic here. There's not going to be anything logical this... It doesn't follow, but you might also think that a composer wouldn't be able to make good computer music, or electronic music. Unless it was the composer him or herself whose fingers were actually really on and in the musical tool that was making the sounds, right?

[5:08] So what you're really looking at is a way of fooling composers into doing their own work, which is the spirit in which you have to do things in order to get the good music to come out - instead of just some old elevator music, or who know what. Or some nice academic music, of the sort that they make out on the East Coast. [laughter]

Miller: [5:17] Now, don't get me started on Milton Babbitt, but... Now... [laughter]

Miller: [5:22] Hmm, wait... Milton Babbitt, by the way, is an excellent pianist. [laughter]

Miller: [5:45] So I didn't mean to say that he was either a bad composer, or a bad musician. He's the person who invented the PhD in Composition, and the PhD in Composition, the idea that a composition is research is evil. It's the reason that music in universities has split off from the real lifeblood of music in the United States.

[5:55] This didn't happen in Europe, because they didn't do the PhD in Composition. So, I'm sorry I did get off on it, but I've given you my opinion. Now I can stop that, and get back onto reality here.

[6:36] What I'd like to try to describe today is... What I'd like to demonstrate, in some sense, is a series of attempts - none of which are really successful, and none of which will ever really be successful in any sort of final closing sense - of possible approaches. Perhaps even promising approaches, to trying to make a box like this into something that you really could operate in a musical way.

[7:03] So musical instruments are things that you pick up, and you do things, and outcomes sound. If you do the thing harder, as a general rule, the sound comes out different from if you do it softer. Your ears know how to - or your brain knows how to -sense the amount and the type of effort that you put into operating the violin, or the snow shovel, or whatever you're playing. And the sound that the thing emits when you're actually doing it.

[7:34] Computers don't work this way - naturally. The reason for that is that the actual energy that's going into the computer is coming through the power cord and being generated down in San Onofre or someplace like that. And it's turning to heat here. Most of it is just blowing out the fan as air, but a little tiny bit of it, like a hundredth of a watt, is going down the nice audio cable where another whole huge influx of electricity is turning it into physical motion of speaker cones which can be music for us.

[7:45] But none of this actually involves something that me as a computer operator is making the thing do in any sort of physical real sense to actually make the sound.

[8:09] And this separates the computer from almost any other instrument known to humankind. The only other example I can think of something that is this weird is the pipe organ. I'm probably not thinking of something else important, but pipe organs are that same way. Pipe organs are actually closer to being a proper musical instrument than a computer is, at least in its ordinary state.

[8:43] What is a computer really? Well, no one ever designed a computer as a musical instrument, right? It was designed as a thing to do, well; we all know the first thing was missile trajectories. And then pretty soon people figured out that you could do banking on them. And so IBM was this big company in the fifties and sixties that made these monster machines that were bought by research laboratories, some of them good and some of them evil, and then banks, and then other stuff like that.

[9:08] And the way you use a computer is you go into your office and you sit down at your chair and you start doing office work. For that phrase, by the way, I'm indebted to David Zaccardelli who wrote a paper that I've thought about a long time afterward, entitled something like, well, I'm sorry I don't remember the title of the paper. But you know, when you're at a computer, you're doing office work.

[9:33] If you watch musicians as they live their lives, the pianist kind of musician spends a lot of time at the piano making music. The electronic musician doesn't spend a whole lot of time actually making music come out of the computer. Notice, by the way, I'm not making music come out of the computer right now, right? I'm doing office work. Well, I'm not even doing that right now. I'm not sure what I'm doing.

[10:00] But when I'm sitting down, when I made this patch, there wasn't sound coming out while I was doing the real work. You do the real work, then you stop and the sound's coming out. But when you're doing the work, you're basically sitting in an office typing, right? And that's a transaction that was designed for financial transactions. It was not designed for musical transactions, if I can make that phrase up.

[10:40] And in what sense is it ill-equipped to do that? Well, one difference between music and banking is, with banking, plus or minus a day or so, it doesn't matter when something happens, right? You deposit a check and then you're going to use the money the next day and that's about the end of it. You don't deposit three checks and have the PO [snaps rhythm] rhythm, right? That doesn't happen, right? But rhythm is one of those things without which you couldn't have music.

[11:06] Why is rhythm important? Well, let's not get too deeply into that. Sound is a thing which doesn't spatialize as well as light does, but it's a thing that timeifies, temporalizes. I don't know what the verb would be, but it's a thing in which we can distinguish differences in time that are fantastically subtle.

For instance: [11:43] , if you ask a pianist to play a scale, just play up a scale but play it in fours, and then play it again in threes, just yaga-daga-daga-daga-daga-daga or yagada-yagada-yagada-yeah [vocalizes to demonstrate rhythm patterns] like that but exactly evenly, and then you look at what happened. You will find when the musician was thinking in threes, the timing will be grouped in threes instead of in fours and the differences will be in the order of milliseconds. Furthermore, you can play a tape of this and another musician or even a good listener will be able to hear the difference.

[12:07] This is something you can't imagine doing with your eyes. Your eyes can be fooled by, for instance, presenting 24 images a second as film does, if I've got the right frame or 30 frames a second, to be generous. And it looks like continuous motion. Gives you a headache, but looks like continuous motion.

[12:33] So your eyes are at least about 10 times stupider about time than your ears are, although they are at least, I think, about 100 times smarter. Smarter/stupider is not the right word. More acute in spatial resolution, so you can read something like the letter A up there on the screen, even though the angular size of that thing from where you are sitting is far less than a single degree of arc.

[12:48] But if that thing were making sound, if one side of one of those letters made a peep and I asked you which side of the letter it was, or if I made the whole letter say "peep" like that, would you be able to tell the difference between an A and an F? I don't think so.

[13:32] What that should lead you to think perhaps, is that to make a computer be an adequate musical instrument, it has to be made to be exceedingly reactive in terms of time because that's where the information is. And it's even worse than that, because of course you can write down a sequence of time to the nearest microsecond and say "do this, then do this, then do this" and so on, but humans for some reason can't do this.

[14:02] They can't take subtleties of musical phrasing, how things should be in time, why a thing sounds the way it does, why music says what it says, when all it takes is just oompah. That's a different thing from something else that has seemingly exactly the same values of time.

[14:25] There's some weird magical channel of communication in music that no one really understands, right? It's not even a single channel. You write it down on the piece of paper and it's lost, but a good musician will take the piece of paper and work over it over a period of time, learn it, figure out what really the meaning is behind that mnemonic, which is the score. Nothing but Mnemonic and outcomes the music again.

[15:02] But, you can't actually write down the instruction that you have to give computer, for instance as a musical instrument to turn that into sound that had a believable and meaningful musical phrasing. The only way anyone has been able to figure out is to have someone play it and so you have to somehow take the thing which was the computer. Which was this banking and missile trajectory calculating machine and turned it to something that you can walk up to and play.

[15:30] And if you don't do that, what comes out might be nice sequences but is not likely to touch anything like the full range of musical expressions that humans are capable of and computers might help or might hinder from doing. I am telling you about all the terrible things about computer music. There are good things about computer music and I think there are so obvious that they don't need mentioning.

[16:00] But, it was clear to me even as a junior high school student, when I learned about 4E analysis from my father who is a mathematician. It was clear that OK, you can take sinusoids and build them up and you can make any wave form at all. Wave forms are tambors and you can make any tambor that you can possibly want out of the computer.

[16:29] In BASIC I started calculating things and started making teletypes, draw little waveforms and asterisk that go up and down the page and started synthesizing wave forms. Even knowing that I could never hear them that was the best thing to do. The good side of the situation is that a computer can make any sound as you can possibly describe that can come out of the speaker. That is the limitation. That is the cool part.

[17:20] The bad part of course is that you can't play it as you can only just order it what to do. Send in its' punch cards and get the result back in mail or something. That was the central challenge that lead me to make maximus P and now to make Pd. What I want to do is show you what the issues are. I am still of the opinion that computers are lousy musical instruments. It is not because a whole lot of people haven't tried very hard to make software that will turn computers into things that will respond in a very musical way to gestures that humans make. But we still are a long way from being there.

[17:46] Furthermore, we are at period of time right now historically, where over the last several years -last 10 years say - that situation hasn't really been changing in any substantive fundamental way. Right now, we are on a weird plateau wherein we figured out a certain area of the problem and that is OK. But, then the rest of the problem is still something that may be someone will think of something tomorrow that everything will work out.

Maybe there is hundred years of research still in our future that we can't see yet. That still stands between us and the sort of full realization of the computer music, that it can do anything which you could possibly [inaudible 18: [18:19] 00] and make it sound musical at least up to the musicality of the person using it. I want to try to show you what I think some of the openings might be and none of this is going to be more than openings for a variety of reasons.

[19:06] First things I want to tell you is why pure data is called pure data. I got the idea in 1997 as to what would the original maximus P sort of thing what it was. We had a very good way of describing process, process meaning you have analog to digital converter and then its output will go to multiplier and it is multiply by an oscillator and that will go do the digital analog converters. Something like a patchable synthesizer I say. You could take that thing and make it run real time and furthermore you could make it respond real time to request change the frequency of the oscillator or the amplitude of the aggregate output or things like that.

The reactive aspect of getting the computer there at one level was solved, while at another level clearly was not because you still had to talk to it through weird things like midi ports or keyboards or things like that and that seemed insufficient. But another general thing is that, there is the whole other way musicians think about music which is the storage aspect. Which is the thing I was integrating earlier the business about the ink on the piece of paper? It is not as if just being able to play an instrument is not adequate for being able to be a musician [inaudible 19: [19:49] 46] .

The ability to write things down is also crucial [inaudible 19: [20:28] 54] is enabling they are music that don't require that. But the ability to have a way writing things down multiplies your possibilities like stepping into much larger room. Right now I think by and large what we do with computer to make music is kind of separate into two things. One is real time interactive thing which maximus P and Pd you are doing pretty well.

[21:02] And the other things is the score thing and do that you get off into Sibelius or Finale or Excel. People do all kind of stuff that way and there you are talking about documents. You are doing document preparation, you are doing office work. What would it take to make those two things to somehow manage to cohabit a single kind of computer environment or single kind of situation user would be presented with. This is the reason why I named pure data what I named pure data.

[21:36] Besides slipping couple of jokes and digs, the intent was to go back and think not about the part that seemed interesting in early 90's or late 80's in fact getting the activity to work but getting data to work. Getting it possible to have a way of describing data that was so flexible that you could express any kind of musical expression that could sit in a place that you could sort of store anything.

[21:48] My answer to that is this is not a complete answer. I'm not going to claim to solve this problem at all; I'm just claiming to regard this as a problem that I'm interested on working on.

[22:01] Sorry, I've forgotten where I put the window I wanted you to see.

[22:38] This is me recreating a nice piece of music by Charles Dodge. Charles Dodge wrote a piece called "Earths Magnetic Field" which is a bit of evidence for me that people thought about scientific data and computer music in the same thought for many years. That would have come from the mid sixties, I think it is a fascinating thing to do and wonderful. But this is a pure piece of the east coast, academic, but beautiful music.

This is a thing called "speech songs" and I actually like speech songs so much that I decided to study it to the point I could transcribe it and redo it. This is not actually Charles Dodge speech songs, but this is one I transcribed off the tape and got all the times to the millisecond and put it back on a score. In doing so, my intention was to make the language in which the score is expressed be as close as possible to the musical space that I was able to [inaudible 23: [23:28] 17] Charles Dodge was working in. I don't actually have Charles Dodge's version of it cued up and I think only one of you in the audience is going to know this piece by heart the way I do. [music] "A man.......Man sitting in a cafeteria, and one enormous ear and one tiny one which was fake.........."

[music ends]

Miller: [25:00] Isn't that beautiful? That's not me. That's Charles Dodge.

[25:16] The point behind that was that, this is a very simple example but this is already an example of something that would kind of defy Sibelius.

Sibelius is a commonly used music notation program. Just for the stupid reason that Sibelius does not know how to attach "temporal" elements to things it regards as notes. Now this would be a lousy thing to write Beethoven's 5th symphony in. You can probably tell write off. This was just you will be miserable, because it doesn't know about sharps and flats and doesn't know about measures or reasonable like that, [inaudible 25: [25:51] 47] or a French horn or a piccolo.

[26:14] What it is good at is not pushing you into any corners at all, in the sense that the only thing you see is something that is absolutely essential to what that piece of music is. There is nothing inessential.

[26:43] What it is in effect is a private notation, which is not Charles Dodge notation; it is my private notation of Charles Dodge's piece. It's transcription. The notation was made in a way that has perfect correspondence to what I was able to identify as the musical meat of that piece of music. It would be different from the way I would transcribe any other piece in the repertory or anything that I would have to do personally.

[26:47] Any questions about this?

Student: [27:02] What did you do the first time you listened to the music?

Miller: [27:05] What did I do? Is that it?

[27:44] The poem is by Mark Strand. The first thing I do is what the poem was. This one is 90 seconds long. I got in a sound editor and I looked at the beginning and end of every single sound in the whole piece. Then I measured them out in milliseconds and by ear figured out what parts of the original poem corresponded to what note of the piece.

For instance: [28:23] "Which was fake"..... There is a very strident way that syllables and consonants are treated in the piece. So it was clear when I do it actually find a way to a place where consonants started and stopped and get a separate control of the timing of those events and so I just did that with my ear and sound editor until I was satisfied that I knew it. So it's not such a huge feat for you trying to do this.

[28:44] I didn't do it in real time. I did it in a couple of hours and there's one mistake. It's all the pitches are temper 12 tone regular old western pitches except on that's off that was probably mistranscription.

[29:13] Yeah so oh so why do a piece like this instead of do a new piece of my own music. That's the scientific method. To keep yourself honest rather than you know adjust the problem to whatever solution you have. Whether consciously or unconsciously. They had to really do some things to force yourself to fit into a situation that someone else has externally prescribed on you I think anyway.

[29:35] That might be more an opinion than a fact. All right so there's that and then the next thing would be, well another example of the same thing anyway is going to be this. And my apologies if any of you have already gotten tired of this by yourselves.

[30:09] This is a one of many possible representations of a sound that develops over time. And let me just get this thing cooking and then I'll see if I can make this thing operate. So first off we'll say something. Spaghetti and meatballs. All right, nice two second sample or something like that. And then what I'm going to ask the machine to do is, if I can make this happen.

[30:46] Oh, first of are we listening to it. No were not running why not? Oh, let's not worry about it. Yeah, that looks good. See there's the letter s up there and now I'm making tonal sounds. Oooh we got a couple glitches up there that's nice. Sound is not as easy to work with as image in some ways. By the way you're going to see much, much better than this when Curtis Rhoades gets up here and starts talking. Because he's got tools that are much cooler than this, This is a classical tool.

[31:20] Now what's happening is what happens is each one of these traces you can't see the amplitudes because I didn't know a good graphical way to represent that without making everything get totally messy. But what you see is the pitches of all of the sinusoidal partials that you would have to make a sinusoidal oscillator bank say in order to utter the phrase spaghetti and meatballs the way I just uttered it. Let's see if we can actually hear this. Resynthesis. This is kind of a horrible patch to use.

Computer voice: [31:23] Spaghetti and meatballs.

Miller: [31:25] There it is. All right.

Computer voice: [31:26] Spaghetti and meatballs.

Miller: [31:28] You can hear a lot wrong with that sound right.

Computer voice: [31:29] Spaghetti and meatballs.

Miller: [31:47] But there's something crucially right about it which is it has the same general musical arc as what I put in. Now the cool thing about this is this is the same software as the one that I showed you previously. Except that it has been customized in an entirely different way.

[32:21] So now rather than having things that are notes and you choose a syllable underneath which goes and looks at a bank of analyzed sounds. Here there is only one analyzed sound and you're looking at the analysis yourself. Right. And that's gives you a different collection of things you could possibly do. For instance if you want to make it a question you do this. All right. You can see the interface is just terrible. You shouldn't really make a composer do this to make a crescendo.

[32:35] But I'll do it this way anyhow, why not. Because you know we are just having fun here were not making music yet. And now if I play it back again maybe I'll be offering you spaghetti and meatballs.

Computer voice: [32:35] Spaghetti and meatballs. [laughter]

Miller: [33:05] OK. That's computer music. Well the thing that can be cool about that is that this could give you a tool for actually building up pieces of electronic music by basically putting your hands right in the center of the sound that you're operating on. And the other thing that hopefully is cool about that is there is since there's no boundary between this way of operating and the way I showed you.

[33:25] You could in fact do things that incorporated both of those things at once. The sort of higher level organizing aspect that you saw earlier. Acting on those dimensions that require the higher dimensional organization and perhaps using things like this for describing the details of the things that you need to be able to control it in a detailed way.

[34:03] I don't know that's perhaps easier to suggest than it is to explain in any sort of real way. And I should tell you know that you should probably not start making music with this. Because it is so clunky. And the problems turn out to be so much harder than I'm making them look right now. That when you really get down to the nuts and bolts of trying to make this thing make this thing play with you. It turns out really to be hard enough to turn most people off.

[34:23] So you should regard this not as me selling you a piece of software something like that. This is a way of really showing you what would be cool to be able to do if someone could actually solve this problem in a really friendly malleable way. This is not what this is. This is a really clunky way that I hope someone will overtake.

[34:42] But the principle is there, this idea of data whose view you control so that you can develop the visual language you need as a musician or as a maker of any other kind of art. To get at the aspects of it that you really want to tease the meaning out of.

[35:17] Questions about that? OK. I will just jump on to another wonderful thing that you can try to do and it needs trying to do harder than I've been trying to do yet. Actually I'm going to start with a jockey one or jockey thing. Tell me if this is too jockey and I'll get rid of it. Where did I put it. Oh yeah you just whack the button and it goes maybe.

[35:50] This is something that I assigned a bunch of students in a computer music class that I was teaching. Let's see, are we happening yet. I don't know how to figure out if were happening. Hello, oh yeah there we go. This is a useful application once you've done it. Because if you've ever been a student the way that you act as a student is you sit in a room like this and you open up your laptop and read your email by your professor is explaining calculus.

This is a useful application to have because this will tell you when the professor was actually talking to you and you can put on your buds and listen to your favorite tune [inaudible 36: [36:42] 01] calculus. I actually don't know how many of my students are using the bud on the ears and showing these wonderful things. The purpose of this discussion is... What is this really? Well, this is a normal computer music interface which is actually the stupendous possible music interface. Just put on a microphone or something. Why? Because... so this mike is going straight in to my computer here. Right.

[37:27] People now had to make sound and for them musicians are used to making music by making sound. Because that is what music is made of at some level. One pretty good possible approach to making a computer in to... One good way of making a computer in to a musical instrument or a good general person doing this might be to take what musicians know how to do and strap microphones on their instruments as close as to get on the part that is emitting the sound. And just take that signal which the musician suspends, his or her life learning how to make come up the way they want to come out.

And then turn that potentially into a whole world of other things which would somehow be able to [inaudible 37: [37:48] 32] on a musician's training but then be capable of a wide variety of other musical even non musical activities like what I just told you here. This is a pedagogical example. There are possible real examples of this.

Before I go too deeply in to this, this is only one bunch of other things that you could think of doing. For instance, there is a whole industry out there and research industry of people who are trying to design physical interfaces for computers. things that you [inaudible 38: [38:27] 09] throw or step on or whatever it might be, the computer is unable to sense the electromagnetic [inaudible 38:15] And that become a sort of the control surface over which you make the computer, over which you explain the computer what you want to do.

For example, probably the midi keyboard. But there would also be [inaudible 38: [39:02] 32] things you buy now which were boxes that have [inaudible 38:36] computers get in bunch of changing virtual voltages that can drive a synth or something like that. So that's the whole thing. And I think they are cool things to be done there. Even in the realm of musical instruments, you can put sensors on the instruments to sense things like bow pressure on a violin. They are supposed to putting the sensor which is a microphone which is sensing the output of the instrument.

[39:30] And what I am hoping to suggest here is that it is actually looking at the output of the instrument that might be a promising thing in certain things for certain instruments. You didn't give this to piano. For piano you put the sensors on the instrument itself to try to sense what the players doing, because it comes out of the piano so complicated that I don't think you can use that very easily.

[40:03] But the voice will be opposite in that particular spectrum where you do not want to put the sensors on the thing that is making the voice. Especially the last person I would wish to do that will be a trained singer. Right. They are very careful about their throats. But on the other hand, it is the easiest thing to sense what is coming out or comes out about the same place which is not true in place of clarinet as you can really pick the sound in a local and controlled way and do what I just doing with the bud.

[40:43] And the next thing is that when you use the sound output of a real instrument, be at the voice rather you really using the aspect of what is going on... that the musician's intend is really driving towards. So whether a musician is using a lot of bow pressure or little bow pressure, you know, of course it has some correlation with the musical expressivity of the situation might be. But it doesn't correlate anything nearly strong is the sound better correlate. Because that is the thing. That is the product. That is what they are really making. So it seems like the perfect thing to tap if you want to tap something.

[40:44] Next problem, don't use this for saxophone. Because the saxophone makes so much noise that youâ€™re just going to listen to the sax and the computer is going to follow the sax alone. Do with something whose sound you can cover with the sound coming out of the speakers which you can do very easily with the voice and do extremely well with guitars. You just get nice solid body electric guitar and slap a pickup on it and youâ€™re there. No one even know you are playing the thing except that comes out of the speakers. Examples of possible approaches of using this and this meaning using audio input to a computer as a source of potential musical control. In no particular order, because I was not able to think of natural order to describe all this in. I am just realizing my examples are going to be so weird that it might turn you guys all off. But thatâ€™s just going to be what it is.

[41:50] This is an idea that Iâ€™ve been working on for a decade or so. This is aimed at instruments which have a rich timbral set of possibilities â€“ the voice of course being the best, the prime example. And here the idea would be in this patch â€“ for which I apologize. The idea would be, take the thing that youâ€™re doing, whatever it is, and go look up in a bank of sounds. The thing that most closely resembles it and play that instead. So, for instance, go record a year of alternative radio. Right? And then, give yourself an application where you can just say, â€œPoomh!â€☐ like that, and it would go find the funk tune on Channel X that really have that sound, and it get for you, and play it right then. Wouldnâ€™t that be cool?

[43:07] There are hideous conceptual problems with that idea that I am not explaining to you right now because I don't know how to explain it, except actually by showing you how it doesn't work when you actually try to do it. So, this is actually not bad. It's not a year of sound; it's 40 seconds of sound.

[43:52] And what I've done here is an experiment with three different corpuses of sound. They're not going to be pretty or anything. I was aiming for things as things being as different from each other as possible. Here's me playing violin. Well, we don't even hear it. Why not? Because I'm not doing something right, but what is it? What am I not doing right now? Maybe I have to... OK... maybe I have to do this. Maybe I need to hit the play button which is right here... maybe. Let's see. This patch exists for the purpose of doing research, so it's not a thing you'd want to... yeah, here we go. [violin music starts to play]

Miller: [44:08] All right. That's me. I don't play violin, but that's what it sounds like when I do it anyway. The second thing is that it's always good to get a politician out. So, here's just... [sound of George Bush speaking]

Miller: [44:36] Random phrases taken from here and there, so it doesn't make any sense. And then here's a really wonderful recording I made of Trevor Wishart, who was at Earcom back in the early 90's, I think. I asked him to give me some sound that I could work on. Just to walk up to a microphone and improvise and the result, I have just about memorized this. It's actually a piece of music. [sound of random noises]

Miller: [45:20] And so on. Right. I'll stop it right there. Now just imagine that you could watch TV and there's your favorite politician speaking and you could hear that instead. Wouldn't it be wonderful? Why not? Let's see. Let's see if I can do this. So this is going to be the control source. There's the control source, and here's Trevor Wishart singing instead. And now we're going to do that and turn this on and let's see if it happens. Yeah. [sound playing]

Miller: [45:36] Those are Bush's phrases spoken through Trevor's voice. And then when you get tired of that, we'll make Bush play a little violin. [sound of violin playing]

Miller: [46:15] The test would be, and I think I'm going to fail the test by the way, the test would be could you play the same thing in the control as you looked up and presumably it should find the same thing and play it back out pretty much as it was. And the answer is, I didn't really quite succeed but I'm getting somewhere close. Let's do the easiest to recognize is going to be the politician. Let's see. Now we're going to listen to this one. This is the synthesized result. So, here is the politician controlling his own recording. [sound playing]

Miller: [46:42] Oh, that's completely wrong. I'm doing something wrong here. It doesn't work that badly. Oh that's close. Well I don't know. I don't know if this is succeeding or not. I keep changing the program so you're seeing it in the state it happens to be in right now.

[47:11] But that as a principle seems kind of powerful. This idea that you could just tell the computer what you want it to do just by making it do it by imitatively verbalizing, right? It seems like there's a tremendous space of possibility there to explore. So, really what I'm doing is trying to incite people to go home and do this themselves and see where they get.

[47:29] All right. So that was one possibility. Another is a little bit related. This is another manifestation of that idea, I think. Let's see. First off I'm going to see if we're.... [music playing]

Miller: [48:11] All right. So we got that going. Now what we're going to do is kind of turn it backward. That's radio. You could imagine this could be you and you would be beat boxing. The idea now is not to just try to treat sound in a continuous string, but to try to identify attacks, and then when you get an attack, then rather than go look in a huge collection of sound, just have a drum kit's worth of sound, you know, 20 to 30 things that are somewhat percussive. I happen to have a nice collection of percussive sounds. [music playing]

Miller: [48:38] Never mind where I got this. I only have a rhythmic extraction of that wonderful piece of music we heard earlier. And it turns out to be great with the... [music playing]

Miller: [49:34] All right. Is it clear what that was? So this was reduced in a rather serious way. All we're doing now is rather than trying to do an elaborate, terrible analysis, we're just responding to two aspects of the sound that's coming in. One is just how loud it is. One thing is we're identifying moments of attack. After having done that we're trying to come up with two pieces of information.

[50:10] One is how loud it is. And the other thing is what I call the Westle Number, which is how much the spectrum is weighted toward the high or low end which you would just conceptually move you from the kick drum all the way up to the high hat or something like that. And, that's it. By the way, this is a great way to listen to the radio. On the musical level, this is really only a kind of an experiment. Am I running out of time? I must be. Oh, I'm kind of out of time, yeah. OK, so I'll...yeah. The next thing would be talking about electric guitar. But weâ€™ll save that for another time.

Student: [51:20] I want to ask you something based on your experience. You worked in different environments, for example, in

Carnegie, we have composers and assistants and technicians and people programming. So if you could think of a model where creativity can develop, what kind of model would you choose? For example, you choose a model where different people work together and for example, science, artists, composers, whatever. So different kinds of creativity are connected sort of, through a dialogue, or you think that people should develop, individuals should develop these different kinds of creativity themselves. What kind of model would you think be more satisfactory for generating ideas?

Miller: [52:05] Yeah. Oh boy. That is a great question. Speaking specifically of music, music really grows out of groups of people than it does out of individuals, I think, at least as I have seen it. But, it does not necessarily mean that the right way of doing it, is to establish some kind of hierarchy where there is the composer or the assistant or something like that. In one way of thinking, all of everything that I have done, really has come out of interactions that I have had with composers and other kinds of musicians.

[52:27] So, none of this was actually done in isolation. So yeah, collaborations and especially interdisciplinary collaborations are absolutely crucial to the way I work personally. So, that is only a partial answer because, that is just about me, I guess. But, it is certainly worse to collaborate.

Student: [52:55] I was curious if you had followed the work of a project called Scrambled Hackz, about two years ago they tried to break video and disegmented audio, and said, they could either recreate phonings of Michael Jackson's speech with him as the microphone or MC Hammer in a beatbox re-sampling. What ever happened, did he use your software? I was curious.

Miller: [53:21] You know, I read about it and, I did not actually pay enough attention to find out what his software was, or anything. I do not know, even if it was he, I don't know how they did it and it is an absolutely very cool kind of thing to do. You know, experimenters are doing cool stuff like that all the time and, by the way, always reinventing stuff, because there is nothing like a new idea.

Student: [53:24] Just curious. Thanks.

Student: [53:46] You talked about having some sort of new interface, like a visual interface for composing music, saying there is major ways of doing that, and with your example, the way you drew those on your arms, and how you are doing it.

[54:12] Do you see the future of, where that could go, is more of one piece of software that does what you are doing or is it more of a communication between multiple types of software? So, say something like, the evolution of OSC or where that could go with, you know, communication between, let us say, a visual application and a sound application, as opposed to one piece of software doing it all, as you are doing it.

Miller: [54:34] Yeah, my thinking is actually changed about that. I used to think that because my other tune was totally real time centric. I mean, there are two sides to this question, one is, if you are really trying to get stuff to happen in real time, then that almost forces you to radically limit the variety of applications that you are using at once, because it just won't work.

[55:11] But on the other hand, from a productivity standpoint, it actually makes all the sense in the world, to have software pieces, really be as small, as they possibly can, to have them rich as possible, instead of interconnections. And in some way of thinking, Pd and Max are an expression of that, because, for instance, Plus Tilda is a program and the line is inter-connectivity. But at a larger level, when you think about the whole bind space that thing lives in, you are stuck in there. And that is why you need also to have portals in and out of an environment like that and to other kinds of things.

[55:33] So practicality sort of, at least in the real time context, forces you to down into one environment. But, the best of all possible worlds would be the most supple, possible interfaces and the smallest and most simplest possible applications.

Student: [55:47] So, it would be like having a visual interface that exists on its own and communicates with the sound generator. It is almost having two unique variables.

Miller: [56:12] Well, I would not make the separation between media or between senses like remotes, like sight and sound. I would make it be between algorithm and reactivity. For instance, if you were doing a mark of chain analysis or something, Pd is not the place to be doing it. It is something else. So, the type of work and the type of data you are operating on seems like a sort of good place to, drawing boundaries between environments.

Student: [56:52] If the computer sometimes allows the musical instrument, do you ever see it like a co-pilot, who you then acknowledge how good its output was? So, maybe it's listening and responding, as a synesthetic tool, if it takes in what it heard, and then reproduces it in another mode.

Miller: [57:19] You are intelligent. Well, OK, that is the idea of changing the mode, but there is also the idea of the box itself, having intelligence and that is the thing that fascinates some people and not others. Me, personally, I don't want my piano to think about the music I'm going to play through it, and so my own world view is that the computer should be a conduit and not an intelligent thing, but other musicians think exactly the opposite of that.

Student: [57:20] Thank you. [applause]

Puckette MUS171_01_04

Miller Puckette: [0:03] OK. It's time to start, and so I'm just going to start. This is Music 171, which is known as Computer Music One. I'm Miller Puckette, and the other people in the room whom you might want to know are the teaching assistant, Joe Mariglio, who you know from Music 170 last quarter if you just now took 170, and Joe Deacon who is acting as right now as our cameraman uploader, who has a PhD in math from Stanford University and runs the NSF, the museums and everything else. So talk to him after the class someday and get an earful of very interesting ideas. [0:45] The purpose of this course is to show you how to -- well, show I'm not sure is not the right word -- is to enable you to make your own computer music applications, in the sense of designing electronic music instruments.

[1:02] What that means, in a sense, is making your computer do what a guitar or a drum set does when you do things to it, so that the thing is running in real time. It's making sound, and you walk up to it and do things to it, and that changes the sound that it makes. For instance, it might be silent until you start doing something and then it starts making noise. Then you've got an instrument that does something that responds to how you're trying to get it to do things.

[1:28] So this is a particular... This thing, this idea of using computers to make computer music instruments is, in some sense, sort of the trunk of the whole field of computer music, at least the way I see it. Computer music grew out of, or maybe it's a part of, the field which could be called electronic music, which started, depending on how you think of it, maybe in the late 1900s, maybe -- sorry, 1800s -- maybe in 1948 when the first tape recorder music started getting made. Well, you could put other kinds of dates on it.

[2:04] And the whole field of electronic music is basically people inventing ways of making music with electronic gear as opposed to acoustic instruments. It wasn't obvious, at first, when computers showed up on the scene that computers would eventually, essentially, supplant all of the other electronic musical instruments that exist, which means the tape recorder, the synthesizer, all that convenient stuff.

[2:31] But nowadays, everything that you could have found in an electronic music studio in the '50s, or '60s, or the '90s is a piece of software on a screen on your computer with a couple of very important provisos. Proviso number one is that a computer makes a rotten musical instrument in the sense that you can't strum it, or whap it, or any of those good things that you can do with acoustic instruments.

[2:54] I'm not going to do a whole lot of talking about designing hardware interfaces for making computers that respond more naturally to musical impulses. The reason I'm not going to talk about that is because it's its own subject. And it's also a rather various subject. Different people have completely different approaches to designing interfaces to computers. It's such a wide, disorganized field that it's hard to figure out how to make a syllabus out of it in the first place.

[3:21] So I'm just sort of going to ignore that and, to the extent that I need to actuate my computer, I'm going to be using keyboards, and a mouse, and the microphone. All right. So other than that aspect of just getting inputs into the computer, I think that everything that you do now in electronic music, you at least can do on your computer. So a couple of things about that, OK. First off, what does making music with computers split up into as a set of things that you can do?

[3:58] And my own taxonomy of what you do with computers to make them into computer instruments are that there are three basic things that you might want to know how to do. One is synthesize sounds. What that means, at least what that means to me, is that you write down an algebraic equation and it has a variable in it for time.

[4:19] As time passes you just plug different numbers into the time slot, and out comes a sinusoid or whatever it is that you told the equation to make, and then you get to hear it, right?

[4:28] And if you came up studying mathematics like I did, this is paradise, right? Any equation you can think of, you can listen to. So that is synthesis, synthesizing sound. That comes out of a long tradition of making stuff, like oscillators and filters that have existed for at least 100 years for doing that, before computers really came on the computer music scene.

[4:53] A second thing is what I think people usually call either processing or signal processing, which is a misnomer because signal processing means many other things besides what it means to computer musicians. But at least if you're in a room with computer musicians and when someone says signal processing, what they tend to mean is something that takes a sound in and changes it into something else to go out.

[5:18] The most ubiquitous example, I think, is sampling, where you take a microphone up to something and make a recording, and then you have a button that you press that plays it back. And the only transformation is that you heard it at a different time from when you recorded it. That's a perfect transformation, right?

[5:38] In chapter seven, I think it is, you will find all sorts of things to do with that particular kind of transformation. OK. That's item number two. One was synthesis. Two is signal processing.

[5:52] Three is analysis, the idea of taking a sound that goes in and boiling it down to a set of parameters that describes what that sound is, or some aspect of what that sound is. A very simple example of that is an envelope follower, which will tell you whether someone started playing an instrument or not, or more generally, tell you whether there seems to be sound coming into a microphone right now or not.

[6:15] And you would use that, for instance, if you wanted to find out if someone was walking into a room so you could turn the lights on automatically. Put up a microphone, hook it into an envelope follower, and then have it turn the lights on when the amplitude of the sound reaches a certain level. So that's analysis.

[6:34] That doesn't sound as interesting as synthesis or processing because there's no sound output, there's just sound input. I hope you'll find out that there's a whole world of cool stuff you can do with that as well.

[6:47] In terms of middle mock diagrams, if you want to think about what this all means, synthesis is, you have a box and it has an output. But the input was something that isn't sound; the output is sound. Analysis is, you have a box that has a sound input but not an output, and then processing is a thing where you have both input and output.

[7:08] What I'm going to do to start with is start with synthesis because it's the easiest thing to get your computers to do. Why? Because it's much easier to deal with speakers than it is to deal with microphones for reasons that I don't really understand very well. But I want to give you some time to get used to how to get your microphones and your computers to be friends.

[7:31] That might make it more appropriate to wait a few weeks before, or however many weeks we can afford to wait, before we start doing that kind of... And just to make the gesture, I didn't bring a microphone today, although there will be microphones in the room later on.

[7:48] What do I have to tell you? I have to tell you some organizational things about the course that are boring but that you need to know. There's a website, and the website tells you all the boring stuff that you need to know about the course. The website will somewhat change in time. What it does now is it tells you, week by week, what I believe the topics to be that this course will consist of.

[8:18] Most of the time I'm actually able to do what I was planning to do, but sometimes it has to change for one reason or another. So this is not a guarantee that this is what we're going to manage to do, but hopefully it's what we'll do.

[8:35] What you'll find is that as the quarter drags on there are going to be a certain number of assignments, which are things that you have to do with a computer that demonstrate that you have mastered one or another technique that is the topic of the week. The first one of these is due a week from Thursday, that's to say Thursday of week two, and that is a tight deadline. The assignment itself is very simple, I hope.

[9:03] What that requires you to do is get software loaded onto your computer and figure out how to deal with the mechanisms for turning homework in, which you probably know better than I do. But leave time to figure all this stuff out. What this means is that you should be doing this right now so that when things start going wrong you can ask for help and you can try to figure out what to do to get things to work for you.

[9:33] To that end, there are office hours. Both Joe, the teaching assistant, and I will have office hours on Tuesdays because the homework is going to be due on Thursdays. I think that's the most effective way of running it.

Joe will be here but I'm not sure in what room yet. The room number, I think, might be changing, but he will be here from 2: [9:46] 00 to 3:00 on Tuesdays. I will be here after classes on Tuesdays, which will be when I find everyone most exhausted. Anyway, that's another possible way to find out what's going on.

[10:10] The course has a textbook, sort of. Again, the textbook is -- where did I put it? The textbook is online, and it doesn't look like a book, but here's a PDF version and a PostScript version, and then there's a nice HTML version. You can even download a nice tarball with the HTML version, and you can download all the examples that are described in the book, which are patches in PD.

[10:41] Or you can download -- don't do this -- download all the figures in the book, which are also patches in PD, if you really want to laugh at what PD can do. It would be hard to do it quick. So that is textbook, and what I'm going to try to do, although I've been-- Yeah?

Student: [10:59] What's the website?

Miller: [11:00] Oh. What's the website? The URL is here. That's the URL you want. Although, you can get there very easily because you Google "Puckette" and then you see Courses, and then you see the first one is Music 171. [11:28] I didn't want to insult your intelligences by printing out the syllabus. Well, actually, if you have trouble accessing the web, come see me and I'll print you out a copy. But it won't help you so much because it's going to change.

[11:44] On that subject, I want to not forget to say one thing, which is if you don't have easy access to a computer and/or the network, please come see me after class today so that we can figure out how to get that solved. There are various things that we can do to try to get you to a computer. I don't know what it's going to be yet because we'll just have to do it case by case.

[12:05] The polls say that 99 percent of students now have computers, so I'm going to assume that you do until you tell me that you don't. If you don't, do please come tell me because otherwise you will be in serious trouble, and it is fixable. OK, so that's the course web page.

[12:27] The next thing, this is what you know more about than I do. The system for turning in assignments is WebCT, which probably all of you have suffered through. Right?

[laughter]

[12:38] Yeah. I last touched this in 2004, and it was a real bear. I think they've made it a little better now. It's actually better than anything else that I've seen.

[12:51] The reason it has to exist at all, as opposed to just having everyone put homework up on a wiki, is because legally we're not allowed to let other people see your homework assignments. The whole thing is basically just to protect confidentiality, as far as I can tell. There's no other reason to have all this infrastructure.

[13:09] In fact, I would love it if one of you could try this. If one of you who actually is online, if you could actually go to WebCT and see if you can log in to the course. So this is the WebCT login. Actually, I think you do webct.ucsd.edu. It's not going to do this for you what it does for me.

Student: [13:33] It's not showing up.

Miller: [13:35] It's not showing up? In what sense?

Student: [13:38] : After you log in it tells you what classes you have on WebCT. It's not on that page.

Miller: [13:42] And you don't have 171 as one of your classes?

Student: [13:44] Not yet.

Miller: [13:44] OK. I was worried about that because I asked for the class roster and I got not a single student in it. I have to call in to WebCT to ask them if there's something that I should have been doing that I haven't done yet, which is probably going to turn out to be the reason. Sorry. [14:01] OK. So this is not going to be an urgent issue until a week from Thursday when it's time to actually upload stuff because I'm not using WebCT to make stuff available to you. I'm just using it to collect stuff.

[14:15] So for this week, the thing that really is urgent is another thing that I hope some of you will try because maybe this will fail, too.

[laughter]

[14:23] Which is see if you can download PD and get it to run. This is going to be a little bit less obvious because I'm going to have to show you some things before you can find out whether you're even successfully running PD. So go back and say something I didn't say. There is a software package that you will be using for the course which is Pure Data, or PD, and you get it from my website.

[14:55] It will run on your computer, unless you have something really strange. It will even run on your iPhone, but that version of it is not on my website for that one. I'll tell you if you care. You can run on Android, too. So you can have a lot of fun with this, but right now we're just going to be using the standard one on the computer and making things easy.

[15:19] So to do that, you do this. Or there's several things you can do. I'm going to show you what I normally do, but your mileage may vary. The link is on the website, although you can also find this through my home page if you want to do that.

[15:37] There's all this good stuff, and here is Pure Data. You can be conservative and use version 42, which works, or you can have fun and use version 43, which sort of works.

[laughter]

[15:52] But which does all sorts of new stuff. Yeah. OK. There's one thing I know that doesn't work in 43 which you're not going to get for another week, so I will try to fix that by the time you get it.

[16:09] Anyway, I'll tell you with this one I can, which is when I told you what the object is that doesn't work right. Anyway, I'm going to be using 43. In fact let's just do this. If you have a Macintosh that's more than six years old, you will want this funny version.

[16:32] Otherwise, you will want one of these, Mac OSX. Can I ask for a show of hands, this is just out of curiosity, well, actually it matters somewhat but mostly curiosity, how many of you have, as your primary computer, a Macintosh? Wow.

[16:50] OK, how many of you have the primary computer of a PC running Windows software? OK, so maybe 80 to 20, something like that. How many of you are running something else? One, two, three.

[laughter]

[17:09] Very good. The reason I brought the Macintosh today -- actually there are two reasons -- the honest reason is that a Linux box doesn't have DVI out so I'm kind of stuck with it now on compatibility mode.

[17:23] The other reason is that I want to look like you guys are looking today, but then by Thursday you're going to be watching me play with Linux instead of OSX. All of the OSX lore, unless I decide really to punish you and bring the PC in.

[laughter]

[17:36] We'll see. No promises though. We're going to be Macintosh today. We're going to grab PD, the scary one, and I think... I don't know what you do with these things. Let's just tell it... I know.

[laughter]

[17:59] I usually save it, and then I get into a shell, and then I type TAR, space, XZF, space, blah, blah. You probably don't want to know how to do that so I'm going to try to pretend I'm a regular computer user. One of you is trying this, right, so that you can see if it's actually working? What did it do?

Student: [18:22] I don't know.

Miller: [18:27] I think I just... I thought it opened?

Student: [18:29] No, your window just froze.

Miller: [18:30] See I just did that. It already did that to somebody already.

Student: [18:32] It's in the new Balance folder, I think.

Miller: [18:34] What is that? It probably threw it either on the desktop or in the home drive. Oh, I'm running PD. [laughter]

[18:39] Oh, look. It looks like I've got all this good stuff and now I don't know which of these is the one I just downloaded.

Student: [18:49] Right there. [laughter]

Miller: [18:50] Let's get maybe this one.

Student: [18:52] Left hand side, left hand side.

Miller: [18:53] This must be it right here. All right, this is the one that I had to start with today. [laughter]

[18:58] Sorry, I don't think it will hurt you to have more than one. Then you just do this. That's the easy part and then maybe this will happen, maybe not.

[19:12] One thing that I've noticed, the first time you do this on any computer, sometimes it seems to take 30 seconds for PD to start up. So if you click it and it does nothing for 30 seconds, I don't know what that is, but that's Steve Jobs doing that for you.

[laughter]

Student: [19:25] Will PD Extended work?

Miller: [19:27] Yes. Oh, thank you. Another thing that you can do, which will be more fun, is go get PD Extended as opposed to PD PD. In fact, it's so much fun I'm going to do this for you, too. The problem is I've forgotten where the... Oh, so we just do... [19:47] Get in the browser, and then we say, PD Extended. PD Extended, Pure Data downloads, PD community site. I don't know what the difference is between that and that. All right. This is the redoubtable Hans-Christoph Steiner, who is a person who aggregates-- well, does many, many things for PD including actually spearheading PD's Release 43. But he's also making the so-called PD Extended installers.

[20:28] For those of you who know what's going on with PD and/or Macs, they have various kinds of objects in them. PD itself ships with a couple hundred objects, and PD Extended ships maybe with a couple thousand objects in it. So you have lots and lots and lots more stuff to play with in PD Extended, if you can figure out where to find it.

[laughter]

[20:48] Once in a while I actually reach... You know, I don't want to make a Butterworth filter. They've got Butterworth filters in it. So there are things which you care about which you can get in PD Extended that are sometimes really worth getting.

[21:03] The other thing about that is when you want to make graphics, PD has an extension called GEM, the Graphics Environment for Multimedia, which will allow you to make graphics and also to shoot video and analyze it. Basically do with video the same things that PD will do with audio. It's not really part of this course, but PD Extended has that. You can go make movies or whatever you want to do with it. I'll show you a little bit of that just as a teaser in week 10 when we're reaching out a little bit in the subject.

[21:34] So, PD Extended. The last time I did this it was very easy, so I'm hoping this will still be here. So download PD 42. This is the one that works. PD Extended 43 is up there somewhere, too. So if you want PD Extended in its natural state, you can do that.

[22:08] But anyway, I think what I do is click this, and it says, "Go to virtual online application." Oh, yes. I want to open it with... Oh, it's a disc. All right. I'll say something interesting for 38 seconds. Actually I sort of know this is going to work because I already have one of these things. It worked the first time.

[22:37] Meanwhile, nothing will happen until... Now it's doing a clean-up. Neutralize that. Just want to point to these things here. Oh, no, we don't want to do that. That was the disconnect. No!

[laughter]

[mumbling] [23:05]

[23:09] No, OK. All right. So I don't know why that was just faking me out. All right, we're done.

[23:24] So disc images are things you click on like anything else in that. New? OK, I'll leave.

[laughter]

[23:29] And ta-da, we have a disc that consists of, well, no one.

[laughter]

[23:39] As it starts, you can ignore it. It's quick when I do it directly. So this is the PD Extended application and I didn't do that, I just did this.

[laughter]

[23:58] Why? Because you don't really want to throw stuff into your applications folder. I won't explain all the reasons you shouldn't mess with your applications folder. You'll have to guess.

[laughter]

[24:14] Then it takes too long to do runs. This is all the stuff that it either loaded or didn't load, and that's good. But now we're running PD Extended. More about that later if you want to find out about that.

[24:37] Let's get out of here now and get back to being vanilla. Take that, get rid of it, take that, get rid of it. Get rid of this. All right. It's all free so you can throw it away any time you want.

[laughter]

[24:57] So, next step. Now you've downloaded PD. Has anyone actually done this? So, next step is, see if it's working for you. Of course, it should start when you click it and it should also make sound when you ask it to make sound. Actually, that's the real step that means you're doing computer music.

[25:21] To do that, to find out whether that's happening, there are two places that you should think about looking. I always go to the impatient place first. The impatient place is, go to Media and say "test audio and midi" and up comes a PD patch.

[25:42] This is a PD document, first one you've seen so far I guess, and this has indicators that say whether sound was coming in to your computer. These are numbers and decibels which you learned about in musical acoustics last quarter. These are in decibels with 100 being full blast. I don't have a microphone so this is the noise level on the audio input device in my computer. There's nothing plugged in.

[26:13] So, I have a signal-to-noise ratio of, compute that and it's...

Student: [26:23] Four minus.

Miller: [26:27] So the loudest signal I could get would be 100 here and I'm looking at 28. So the signal-to-noise ratio is 100 minus 28, which is 72. Which, there's not audio hardware supplied. That's bad. [26:40] OK. Now the other thing that you want to know... OK so, but sound is coming in. I like to seeing that better than I like seeing zero. What I really like seeing is one or two, which means I've got decent audio hardware. Now I can make sound, which is to say I can ask the test tone to go on, and this is in decibels too, again, with 100 being full blast. So a good place to start is 60.

[tone sounds]

[27:02] Now you hear a nice A440. Or here's 80.

[tone sounds]

[27:07] I always do 60 first because you never really know where the speakers are set.

[tone sounds]

[27:14] While I was... Yeah, there it was.

[tone sounds]

[27:16] Now, what I didn't show you was, before most of you came in, I connected my computer to the audio system in this room along with the projector. So what you're hearing now is the computer's line output...

[tone sounds]

[27:30] ...talking to my stereo. And any of you who has a stereo can do the same thing, and that's a better way. That or headphones would be a better way to operate than using the little speakers that are on the computer. Yeah?

Student: [27:45] Did you say the lower the number the better?

Miller: [27:48] Well here, yeah. If there's nothing plugged in, the lower the number the better. But if you have a laptop, your laptop might have a microphone. So you might not just be looking at the electrical noise level on your equipment, but you might actually be looking at sound. [28:03] If that's the case, then when you save things that number goes, up. Then you get really happy because you got audio and then you can start making cool processing actions. I'll say that this will happen to you, the first audio process you actually make will suffer from horrible feedback if you're using the microphone with speakers on the...

[clicking noise]

[28:21] Yeah, like that.

[laughter]

[28:24] Because the mike is very close to the speaker, right? And so the sound comes out of the speaker and back through, like that. If you want to control that, plug in a pair of headphones, which usually will mute the microphones. Then you can listen to what it's doing and the microphone will work properly, I think. Depends on, you know, your mileage may vary.

[28:46] The other thing, just telling you about this, I want to just tell you the basics about getting started. When you do this...

[tone sounds]

[28:55] ...and that happens, it's great. But it's possible to do this and not have the sound coming out.

[28:59] Then there are things that you might want to do to figure out why, whether you have sound or not, and that all is here, under PD. So this window popped up when I set the test audio and midi. By the way, this will be possible to do but not useful. I could have two of these up at once and they'll be fighting each other. So don't do that.

[29:25] So then in PD, that was in media, audio and midi. In PD, you get preferences which have audio settings. We're not going to talk about midi today. And audio settings are what sample break we're running at and a magical number, which I should tell you about, and what audio devices, and what numbers of channels.

[29:48] Now I can do things that will cause everything to break. Let's have eight channels about right here. All of a sudden, nothing happens. Maybe, I hope, I have an error message. I have lots of error messages.

[laughter]

[30:06] It didn't even give me the proper error message so I can't do it. So this is the "can't do it" mode. You don't see anything here and you don't hear anything coming out. That just means that your audio device didn't get opened. That could happen for all sorts of reasons, which are hard to disambiguate.

[30:26] But in that case it was me asking for something impossible like that. Also if I ask for megahertz out, I don't think it's going to agree. Can't do that. So you have to ask for something reasonable. And the standard CD sample rate is 44K1.

[tone sounds]

[30:54] Now we're back to being happy with the input now.

[30:59] The other thing that can go wrong is you could... You can't make it not be happy right now. You can have this thing dialed in on a device that is no longer plugged into your computer. You buy a USB audio device, you plug it in, you tell PD to use it.

[31:17] Then you unplug the device, it no longer exists, PD starts up, you can't find it. Then you see here it's just a little circle which has nothing in it. You just have to click on that and select the thing that you really want.

[31:35] The other thing that I want to tell you is this number here, the delay, this is the spooky setting that matters but which is hard to figure out how to deal with. This is a number which is 80 milliseconds or up, if you're using Bill Gates' software. Or it's 20 to 30 if you're using whatever his name is, Bill Jobs', Steven Jobs' thing.

[laughter]

[31:59] Or you can get it down to about 10 on Linux. This is the amount of time that passes between when sound comes in the machine and when it comes back out. And if you try to make this too low, PD shouldn't be showing you errors, which I'll see if I can find here.

[tone sounds]

[32:21] Do you hear that? Let's see here. I'm running 43. On 42 you would see a red light saying digital IO errors. I'm trying to resize the window. It's too big for this stuff. Can't do it. OK.

[32:51] No, it's not there. All right. Never mind. I don't know where you see the error. You just hear the error. Here it is.

[tone sounds]

[32:59] And that's because I asked for a delay that is smaller than my hardware can provide. Oh, I did a 15 and now it's cleaned. But now let's see if we can do 15 to 1.

[33:17] So the smaller that number is, the faster the tablet. That matters because you don't want to do something to your computer and then wait a second before you hear the output. You want it to happen as -- Well, you want to have it happen with a small enough delay that it sounds like it's happening at the same time.

[33:35] Which, depending on your musical chops and which instrument you play, might vary between five and 30 milliseconds. What this means is that Macintosh latency's 15 to 20 milliseconds, maybe, or 25, are barely acceptable and the Window's latencies that you get are basically unacceptable.

[33:57] And I can tell you that that's only the built-in audio hardware on those devices. I have seen Windows boxes get very little latency by professional audio hardware you put on it. So if you're a real gear-head and want to buy the gear, you can gear your way out of the problem. Although you can also just take this, plug it into your machine, and turn it into Linux, which is what I would do.

[tone sounds]

[34:26] Sorry to belabor all this, but this is important because you have like eight days, nine days to get this all happening and be turning in homework. So I want to make this as painless as humanly possible. Questions about all this? I know I've forgotten things. Yeah?

Student: [34:45] So what you're telling me is that the latency... Which one's the latency? Like between Windows and Mac, is it the hardware on there or the processing speed?

Miller: [34:56] No, it's certainly not the hardware because you can fix the problems by loading Linux on the same hardware. I can't even generalize and tell you something that's really true in every possible case, but in some sense the audio... Well, audio systems consist of layers of stuff on top of stuff-- the driver and the API and the PD itself -- and they all have various amounts of buffering they do, buffering meaning the amount of memory that they allocate in order to deal with being on time with everything. [35:34] So when you write something to a computer's audio output, you don't just write the next sample that has to go out. You write several or many milliseconds in advance so that the audio hardware can be throwing them out while you're off thinking about email or something. So that then, when you get back to writing a sample, you're still ahead of what it's doing.

[35:54] So there is a first in/first out buffer sitting in your audio output driver. It's throwing stuff out here, and you're preparing stuff for it to throw out, and you're staying ahead. But you're stopping every once in a while because the OS is not treating you right and it is still reading.

[36:13] If it reads something before you wrote it, then you will hear bad noise. In fact you'll exactly this kind of bad sound now.

[tone sounds]

[36:19] In general, you'll hear this sort of bad noise I'm just giving you. This is a paradigmatic sound.

[36:35] So why would one operating system or one audio application program interface require more buffering than another? You have make enough buffering to deal with whatever your operating system can do for you, in terms of calling you back in short periods of time, and that is in OS.

[36:58] But also, different writers of audio software are sometimes more or less conservative in the way they design these things. So in truth, Windows is overdesigned. It could be a great deal racier, maybe one time in a million, fail. They can't fail one time in a million because they'll get phone calls. So they just make the buffer real long so the phone doesn't ring.

[37:24] So, there's that. Now I can start doing stuff, I think. Are there questions before I actually start doing stuff?

[37:40] So do, please, before Thursday, get this downloaded and running so that you're not discovering that you can't do your homework next weekend or something.

[37:49] So next thing is this, what is this thing good for? So what I am going to do is make a patch that makes a sound. Then I'm going to go back and do some theory, simply because I think it might be better to see the thing happen first and then make a theory out of it.

[38:12] So what I am doing also is I'm simultaneously surreptitiously teaching you how to use your data. The real content, of course, isn't Pure Data. It's the technique of audio synthesis in practice and analysis, which in fact you could do in software packages other than PD. If you want to know about lots of possible software practices, I know them all. I can tell you all sorts of stuff you can do with a computer, in some other context.

[38:45] I am going to just select a ridiculous font to start with. The basic thing you do is you put stuff on the screen so there's this nice menu I can click. What I am going to do today is going to be limited to two kinds of things that you can put down.

[39:04] One is going to be objects. Of course, that really means 200 different things because I have to type in what kind of objects they're going to be. So that's going to be where I live most of the time. The other thing is I'm going to need a button later on. So first off, I'm going to make an object. It shows up and I can... OK. Here's the thing.

[39:26] This has a dotted outline that says that there's nobody in there right now. In fact, if I tell it, let's be some object that doesn't exist, it'll still say, "Nah, there's nobody there." In fact, it even got mad at me. Now I'll just ask it to do something it knows how to do. There's an oscillator, and oscillators take as an argument... Yeah.

[39:53] So I'm going to ask it to play A. So you've had musical acoustics and you all know that 440 hertz is A above middle C, right? That's one of those physical constants, like the speed of light, that people just don't touch. You just have that.

[40:08] Now we're going to say what amplitude we want. So I'm going to put in another object. I'm doing this in the slow way now. I'll show you the fast way later. Put another object and put it down here. Then I'm going to type times tilde, I should say, and ask it, let's only be a tenth of a hold for the blast sine wave.

[40:32] OK. I'm going to crack the book in a moment and show you in wave forms what we're talking about here. But for right now just talking over this, this is putting out a full blast 440 hertz sine wave. By the way, you might know this intuitively, but these things are inputs up here and this is an output. I'm going to hook the output of the oscillator to the input of times 0.1.

[41:00] What that is going to do is it's going to take the amplitude of this and reduce it from full blast to a tenth of full blast. What's full blast? 100. Then I'm going to say put another object and this one is going to be -- this is kind of not well named -- it's going to be the digital analogue convertor. That's the person in your computer who takes those numbers and turns them into voltages. Now I'm going to say...

[tone sounds]

[41:31] Oh, wow, it just worked. Take the output of this thing and put it into speakers. That's to say make it available to the audio output of my computer, which by the way is connected to the speaker.

[41:50] Now how do I make it shut up? There's this control here which says whether you're computing DSP or not. DSP, I don't know if that's a good name, is digital signal processing, and that turns the network on and off.

[42:11] That's the fastest way to get silence if something's happening too loud. That's important so there's a key accelerator. The slash turns it on and period turns it off. Oh, command slash is on and command period is off, which you can think of as mute. It's not really mute, but you can think of it that way for now.

[42:32] All right. Now the other thing that I should have mentioned is that when you start PD, this thing is turned off. The reason it was on just now is not because I surreptitiously turned it off, but because the test tones, which I've already had out, automatically turns the DSP on so that it can make noise.

[42:54] As a result, I was using the fact that DSP was still running, even though I'd closed the test tone. So this thing stays on regardless of whether I have the patches open or shut. I can close this patch and it won't change the status of whether DSP was running or not.

[43:10] So this is more software. No, this is half software and half theory now. DSP running, what that means is every object whose name ends in a tilde, if DSP is running, is computing 44,100 numbers per second. Or a number of numbers per second equal to the sample rate, I should say. But 44K1 in and out.

[43:41] What that means is that when this is turned on, this output contains a stream of numbers, one every 44,100th of a second. Let's say one every 22 microseconds.

[43:56] All right. And furthermore, each of one these things is doing that. It's using all of its inputs. It's receiving inputs at the same rate. If nothing is connected to one of these inputs, the input is... All right, that's a complexity.

[44:16] If nothing is coming to an input that expects audio, the input is zero. I'm going to have to repeat in several different ways distinctions between these streams of audio and things which happen sporadically, which sometimes we call control or not audio.

[44:37] But what you're seeing right now is connections between the audio output of the oscillator and the audio input. And what you have to know is this input expects audio and this input expects not audio. It expects messages, which I will tell you about later.

[44:55] All right. So this network is -- I should say, these connections are like carrying numbers when it's turned on and they're not carrying numbers when it's turned off. This input actually does expect an audio. It expects this audio signal.

[45:14] For instance, if I have this on I can break this. To cut a connection, select the connection, which turns it blue, then hit command X. So if you want to try the other output, do that, or both. You can have fan out if you want. While we're at it, you can have fan in.

[45:46] What's the interval between this frequency and that frequency? Any takers? What's the ratio between those two numbers? Three to two. That's what interval on the channel I'm using this for? Fifth.

[tone sounds]

[45:36] Ta-da, mathematics turned invisible. So the reason I did that was not to tell you what a fifth was, but just to show that you can hook two people into an audio input and it will just add them for you. Over here, here's another thing you can do to demonstrate psychoacoustics effects.

[tone sounds]

[46:23] All right, so I'm going to shut this up and talk a little bit more. Is this all clear, what I've done so far?

[46:44] To do this, basically you do what you do with a computer, which is you sort of flail with stuff and find out what it does. But let me do a little bit of the flailing for you so that you can expect things to happen when they do.

[46:59] The most confusing thing that will happen is this. You will reach to move something, like this, and it will move. And you will be happy. Then you will release it and then you will click it again. Then you won't be able to move it anymore. It won't move.

[47:21] Now I'm editing the text. What do you do? Well, if you're in this state, which is editing the text, and if you want to move the thing, deselect it and then move it. This is second nature to me, but everyone has to do this the first time and it will confuse you for a second.

[47:42] So you can immediately move something that's not selected, but when you select something, when you release the mouse, the text is selected for you to edit the text, which is more than likely what you're going to want to be doing. But in case you really just wanted to move the thing, then you have to deselect it so that you can move it after you deselect it.

[48:00] Also, you can select something by clicking on it, which selects the text, or you can select something as part of a region, and that doesn't select the text. That just selects the objects. Then you can move things.

[48:17] Am I going too slow? Yeah. All right. OK. So also, you can select a single thing as a group using the group selector thing, and again, it just selects the object.

[48:36] OK. Next thing is this. I want to show you what this actually really is, and to do that I have to introduce two new objects. While I'm at it, I'm going to tell you there is, of course, a key accelerator for putting an object and it's command one, and then I can say print.

[48:58] This is object number four. So I believe in the first week you're going to see about 10 kinds of objects. What I try to do is limit it to five a day. First day is going to be iffy because we're already up to four. But theoretically, we will not be learning lots of objects all at once, but they will be coming out at a steady rate.

[49:20] So right now, we've seen the oscillator, OSC tilde, we've seen the multiplier, we've seen the output, and now we've seen print tilde. What I'm going to do is I'm going to show you what the oscillator is doing by hooking it up to the print.

[49:34] Now logically, the first thing that you would expect this to do would be to print out 44,100 numbers a second, but it turns out that that would choke any computer in the world to try to print that stuff. Plus you wouldn't want to see it.

[49:49] So instead of doing that, what it does is it waits until you tell it to please print the next glob of data, and it prints it globs at a time. So now what we're going to do is we're going to put the bang under it, which is a button. Oh, let me do that slower.

[50:04] So put, I've been putting objects, but I'm going to put this thing down now. And it is a thing which... And now I have to let out more of the truth. OK. I'm being very careful, trying to let out bits of truth very slowly. So see now that this line that I connected is only one pixel wide, instead of two pixels wide, where this one is.

[50:26] In other words, these are nice dark lines here, but this is a lighter line. That is to tell you that this is not carrying an audio signal, but is for control. It's for sending messages. Messages are things which happen at specific times, as opposed to signals or audio signals, which are happening continuously.

[50:52] The message that this thing sends out is every time you click on it, out comes a message. The message just tells it to do their thing. In this case, it says do you print, please? What has happened? Oh, because I have this turned off. Now I'm going to turn it on.

[tone sounds]

[51:12] And now it prints out. Every time I whack it, it prints out a new collection of some PD.

[tone sounds]

[51:25] So print tilde's job is every time you ask it to, it will print you out the next block of data. So there's built-in knowledge about what PD is doing here, which is that PD doesn't actually really just compute one audio sample at a time. It computes them in batches of, by default, 64 samples. And let's see if we can get this thing shut off.

[51:59] It should make a nice space with these numbers printed out. So this is 64 consecutive numbers of a sinusoid, which is to say, a sine wave, which is the thing coming out of OSC tilde. This is basically the first and most truthful tool at your disposal for finding out what's going on inside of a patch that's doing audio.

[52:27] It's clunky and stupid, because this amounts to about 1.45 milliseconds of sound. So looking at this wouldn't actually tell you much about what really is coming down in there. But if you tried to see it, it would be too much data.

[52:44] Anyway, you can see that good things are true about this thing. What's the maximum amplitude? It's about one. Here's an almost one right there. So what you're looking at is just numbers, but if you graphed them you would see a rising and falling part of a sinusoid.

[53:06] Now let me get you to the book and show you what this is in a picture. You can make PD make pictures, but I don't want to teach you how to do that yet because there's too much detail involved. So I'm just going to provide data off of them .

[53:21] Where was I? Don't want to do that. I want to do this. Yeah. All right, good. OK. So I told you there's a textbook. This is the textbook. You can buy this if you want to spend, I think it's $79, and they did a good job of printing it. But you don't need to buy it because you can just look at it on the web, which is more convenient.

[53:47] But if you want to read it in a hammock, you can buy it. You can spend $80 and buy the thing, or print it out. But don't tell them I told you to print it out. I'm skipping some stuff, which maybe I should go back to, but here's a picture of what a digitized audio sample looks like in graph language.

[54:09] There are two pictures here because this is what you want, in some sense. What you want to make the speaker come do is move like that. The speaker comes live in continuous time, but time isn't split up into trenches of 22 microseconds a hit. So the computer's representation of this, however, is split up into screen time and therefore if you graph it, it would look something like that.

[54:37] This audio signal has a frequency and an amplitude. This is in fact exactly what would come out if you gave the appropriate frequency to an OSC tilde doc. It varies between positive one and negative one. That has no units. That's an arbitrary scale. But I should tell you that if you put something that's more than one or less than minus one at your audio output, that's to say if you feed something that's out of that range to DAC tilde, then your computer will not be able to play it correctly. It will click.

[55:15] So this is the full audio range of your computer's audio output. How does PD know that? PD just asks the computer, what range do you want to feed your DAC in? And it normalizes that to one. All right. The frequency that you would do this at is manifested in how many of these samples it takes for the thing to make an entire cycle. This is all acoustics, right?

[55:43] In fact, what this is if you give it an equation is one of these things. It's an amplitude times the cosine -- you could use sine, but I'm using cosine here -- of the frequency times the sample number plus a phase.

[56:04] So if you take one of these things and graph it, you will see something like what you saw graphed down there. Furthermore, you can change the numbers A, which is the amplitude, or omega, which is the frequency, or phi, which is the initial phase, and you can change the way that it looks in one way or another.

[56:28] N is the sample number, and that is the horizontal axis here. I'm insulting your upper intelligence here. This is all it is though. All you do is you do this and say we'll change that equation and we get all confused. I've done this for 30 years and it never gets old.

[56:51] So what is omega here? Well omega was enough so that after 20 samples the thing comes around and cycles. So omega is two pi over 20. Omega is the frequency out there. So N is the number of the sample and this is the thing which controls the frequency, but it's the physical frequency of the thing as an array of numbers. It's not a heard frequency, and you can convert that to the frequency-frequency by a simple formula.

[57:32] The frequency you hear is the omega, is the angular frequency is what that's called. Time and sample rate divided by two pi, and that's how you make a sinusoidal.

[57:43] So if you want it to be louder, change A. Or -- and here's why, for this I have to go back to the patch -- if someone gives you a sinusoid and if you want to change its amplitude, all you have to do is multiply it by the ratio of the two amplitudes. That is, multiply it by the gain you want, gain meaning the difference between the two amplitudes. So what that means is what is coming out of this equation, what's coming out of this oscillator right now. Omega is two pi times 440 divided by 44,100, whatever that number is, and A is one.

[58:42] The amplitude of the output of this thing is one. So what this is really putting out is the cosine of omega times the N, and forget the phase for now. There's times when you're passing and you don't know what the phase is right now.

[58:54] But if we want to change this amplitude, if I gave you just cosine of omega, if you said, "No, I want 0.1 times the cosine of omega," in other words, I want something with an amplitude of 0.1 instead of one, then the solution is to multiply the thing by 0.1. That multiplies it this way. It changes the amplitude. It doesn't do this. That would be... Yeah?

Student: [59:21] So if you were in your print command, like your oscillators, then just having them separate, would that, instead of going all the way to one and down to negative one, would it just go up to 0.1 and down to negative 0.1?

Miller: [59:32] Yeah. Thank you, because I actually meant to do that but didn't. So I think what you're asking is, "What if I just print the output of this?" Right?

Student: [59:44] Right.

Miller: [59:44] Yeah. OK, good. So it'll do that, and I forgot to turn... Oh, so nothing happened because the audio is turned off. So I'll turn audio on and it will say, "Oh, I need to print something." [tone sounds]

[59:53] Yeah, there. Now what we see is kind of ugly. I'm sorry the spaces aren't worked right. But what you see is something that's going up to about 0.1. It is 0.9998, instead of one. So these numbers are these numbers divided by 10. Except that I asked it a different time and so actually they are like them but they're not exactly the same as those divided by 10, some of the phase. All right? Is that all clear? OK.

[60:34] Now without anything besides those things, what have we got? We'll do those two. I'm going to raise the total count of objects to five, but not in a very interesting way. I just need an added...

[61:06] Now what I'm going to do is I'm going to take the oscillator and have it be zero plus 440. Let me check if that gives us the same thing. Yeah. So this is stupid. There's zero coming in here and we're adding 440 to it, so it comes out here as 440 volts, if this were an analog synthesizer, 440 volt signal.

[61:31] The oscillator then is giving us a signal that's plus or minus one volt, but it's changing 440 times that. The reason I did that is so that I can do this, take another oscillator, or get another oscillator.

[61:44] Oh, I'm doing this without telling you what I'm doing. I'm selecting this object without selecting the text and I'm hitting command D, which duplicates it. It duplicates it and leaves it selected without the text selected, so that's very convenient for me to move it.

[62:00] This one I'm going to say six. And by the way, the machine did sample it, too. Look at this from a different perspective. Come on. Yeah, by the way, let's multiply that by something. No, let's not yet. Let's just leave it. See what we get.

[62:29] Anyone want to guess what this is going to sound like? All right, I'll show you.

[tone sounds]

[62:37] So it's the oscillator on that. Oh, I can just connect it and show you. There is the sinusoid and here is the sinusoid. Its frequency is changing once a second. It's going up to... OK, so here's an oscillator and it's going at six cycles per second, and what's its amplitude?

Student: [63:05] One.

Miller: [63:06] One, right. OK. So then when we add 440, out comes not the 440 volts, but a varying voltage which varies from 339 to 441. That variation repeats six times per second because this thing is happening at six cycles per second. [63:30] But in fact, to make this an easier thing to hear, I will say let's multiply that by five. This can be ugly, but we're going to do it. This will be quite audible.

[tone sounds]

[63:48] Not quite as ugly as I want it to be. So now we're varying between 435 and 445 hertz. Now of course, since it's a computer, you can tell it to do anything you want.

[tone sounds]

[64:04] It sounds like it's doing two pitches at once, to me anyway. But I'm in a weird place because I'm getting an echo from the speaker sounding. OK, let's do this.

[tone sounds]

[64:21] Or, no...

[tone sounds]

[64:24] OK, let's not do that. So it's a computer, it'll do anything you tell it to, if it was a good idea or not. It doesn't matter to it. And furthermore, it won't hurt you because what comes out won't be more than outside the range of the DAC. So as long as you don't crank your stereo or your headphones, you won't injure yourself doing this.

[64:56] I think that what's going on here is it changes its speed to vibrato. I need it more to make this obvious. Right. I'm sorry, this is ugly now. So this is just how fast it's going, once per second, twice per second and so on.

[65:13] You know what I didn't tell you? When you start typing in an object like this, it doesn't immediately change it to the new object. It only does that when you click off of it to deselect the text. And furthermore, if you do something bad like this and then it would say, oh, I couldn't rate that.

[65:40] Then it prints the dotted line to tell me the object would be bad. But it kept the connection so that I don't have to remake a connection when I fix the problem. The problem here is that OSC tilde has a name that only works when there's no space in its interior.

[65:58] For those of you who are scientists, space is the delimiter. That is essentially the only delimiter that you have to deal with. So don't try to make an object if its name has a space in it.

Student: [66:16] So just a question about the setup. So the amplitude for the OSC tilde is one, right?

Miller: [66:22] Right.

Student: [66:23] And we times it by 30.

Miller: [66:25] Right.

Student: [66:25] So that makes the amplitude for the 440 between 410 and 470. Is that right?

Miller: [66:31] Right, and that's changing three times a second. Then that's becoming the frequency for the oscillator. I didn't tell you something important. Frequently, objects will give you the choice of specifying their input or connecting to their input to set it. Here I've said "oscillator" which means we're just going to take a signal and specify what our frequency's going to be. [tone sounds]

[66:58] But here I'm saying "oscillator" but I know what the frequency is. It's three, so I'm just going to keep it on. There's another way in too, which is that you can change these in messages, but I'm not going to try to tell you that.

Student: [67:10] Is there a map of all the names of the outputs that we learned?

Miller: [67:13] If you really want to see it, you say Help. Right-click on it and you can get help and help within a patch, which tells you everything you want to know about it. OK, so that was help. So if you want to have multiplier help, you do that. Then if you right-click on the canvas and say help -- the canvas meaning the document but not any of the objects in the document -- then you will get this lovely patch that someone else made. [68:00] It will tell you everything in this very carefully organized order. But this will only be the first 200 objects, which are the ones that you get before you get PD Extended. More than one...

[laughter]

[68:16] That's funny. I didn't see any specific examples but I'm just about sure that there are two copies of this thing here. Never mind, I'm sorry. There really is this much stuff. Well, sorry, it's just what it is. Maybe there are more than 200 objects now.

[68:49] So that will tell you everything that you might need to know. If we're doing 10 a week, at the end of the 10 weeks you'll know 100 of those objects. You don't need to know them all. I know them all, but you're not me.

[laughter]

[69:09] Basically, with about 100 of them, you can do a whole lot of stuff. And then there will be an occasional thing that you can't do with those 100 that will require that you find another one out and thereabouts.

[69:20] So what happens is that there will be a period of intense learning objects, like 10 a week. After a while, you won't need 10 new objects, there'll be even more and things we'll call down. Other questions? Yeah.

Student: [69:35] How do you get the print thing to work again?

Miller: [69:39] OK. So oh, yeah, there's a thing I didn't tell you, which is fundamental. The patch can be in two different... Sorry. The interface of the patch can be in two different states, which are sometimes called run mode or edit mode. [69:56] If I try to click this thing now, I'm just editing the patch, and that doesn't click on it. It just moved it, right? So what I have to do is put myself into run mode, which I do here. Let's get out of edit mode. Now edit mode is no longer... Whoops!

[laughter]

[70:16] It is still on.

[tone sounds]

[laughter] [70:19]

Miller: [70:22] OK. Well, this is version 43. You tend to get what you pay for. [laughter]

[70:25] Anyway, the indication is what the cursor looks like. So right now what you see is an arrow. And if I do that again, you will see an arrow again.

[laughter]

[70:35] Like now. Now it's just being happy.

[laughter]

[70:38] All right. You cannot get out of edit mode.

[laughter]

[70:44] That's cool. OK, well, I'm expecting to see stuff like this because we're in pre-release.

Student: [70:52] Does the shortcut work, that link?

Miller: [70:54] Oh, the shortcut works great. So the shortcut, you just hit DSP, command E, or Apple B. And then it goes back and forth between modes, except that this is a thing that Hans has driven, and just torn hair out of his head over it. You don't actually see the new state until you move the cursor. Because some smart person at Apple thought you would never have the cursor change unless you reached it, unless you changed wherever the cursor is. [71:21] So what you have to do is change the mode, but then you have to jiggle the cursor to see that you are in the other mode. Isn't that horrible? That's only on Macintosh, so only 80 percent of you are going to have this trouble like we're having today.

[laughter]

Student: [71:34] 90 percent.

Miller: [71:35] OK. So anyway, we're just moving to make sure this is what you think it is. When you're in the run mode, which is not edit mode, you can click this thing and get it to do its thing. And of course, sorry, we also have to turn on audio. [tone sounds]

[71:51] Of course, there's a reason why I'm not on audio. Let's see. Let's just do this. So now we can turn the thing on but not hear it. Now we're running so I can do this. But if I can get back into edit mode, like this, then I can click it all I want.

[72:13] Although, sometimes you can hold the command key down and click it. And it will say, it's as if you were in run mode. So the command key operates as a sort of shift into run mode thing, if you can remember that. I never remember it, so I just toggle the mode. Other questions? Did that answer yours?

Student: [72:39] Yes.

Miller: [72:41] OK. Yeah?

Student: [72:42] I just want to make sure I understand print and DAC tilde. Print within run mode, when you click it, it creates a graphical mathematical representation of that patch, is that right?

Miller: [72:55] Well, not even graphical. It just prints the numbers out.

Student: [72:58] All right. And then the DAC tilde, then the time is...?

Miller: [73:02] OK, so the DAC tilde, that takes whatever the signal is... [tone sounds]

[73:05] ...and puts it there. So it causes it to appear as an audio output. So this is, print the values out so I can see them, and this is, play them so I can hear them.

Student: [73:17] Is that abbreviated for something?

Miller: [73:19] DAC? It's digital-to-analog converter, and that used to be what people called it. There actually is a DAC in your machine, but people never seem to use that term anymore. So, yeah. Yeah?

Student: [73:32] It only prints the first 64 though, right?

Miller: [73:34] It only prints the next 64, until you whack it. Of course, if you really wanted to... No, never mind. I could ask it to print more. Yeah?

Student: [73:48] Do you have a limit in the inputs and outputs?

Miller: [73:50] You mean as to amplitude or the number of includes or....?

Student: [73:55] Just how many things you can enter.

Miller: [73:56] Oh, no.

Student: [73:57] You can have more inputs than objects?

Miller: [73:59] Yeah. Oh, wait. Add more than just an object, meaning... I think what your question was, was how many other things could I run into this...

Student: [74:06] Right.

Miller: [74:06] ...into these fixed objects. But I can do that and you'd never have to stop. But could I make the object itself have more inputs? Each object has its own schematics about what its inputs and outputs mean. Some of them actually do have variable numbers, but you won't see those for a couple of weeks. Other questions? These are good questions, by the way. Yeah?

Student: [74:31] You said that the right input was an input for messages? So if you try to put an input, like with an audio, to that, it won't work?

Miller: [74:38] Yeah.

Student: [74:38] You have to keep it to the left.

Miller: [74:39] Yeah. So for these particular objects, the right input is... Yeah, OK. So we'll get there, because there'll be other things where there will be more than one audio input. Sometimes you'll want to multiply two audio signals or something like that. And I'd be scared to tell you that right now. I'll tell you about that on Thursday.

Student: [75:03] Is this only one channel right now? Like, left only?

Miller: [75:06] I've only been using the left side, mostly. When I'm working at home, I use both sides... [tone sounds]

[75:09] ...because it irritates me to hear every sound out of just one side of the thing. You know, it's just what you like. Yeah?

Student: [75:20] So there's no spaces unless you put a number in there? You just have the space to put the number.

Miller: [75:25] Right.

Student: [75:25] Because otherwise you just get the domino effect.

Miller: [75:28] Right. Yeah. One way I can have it fail is to add a space right in one. So I'd be looking for an object named OSC and I didn't see one. The other thing I could do wrong would be to not put a space there and say would you look for an object named OSC tilde three?

Student: [75:46] Are there any defaults if you don't put a number in there?

Miller: [75:48] Yeah, zero. [laughter]

[75:51] You can add zero to something, you could multiply by zero. But if you don't fill that number in, then the other inlet becomes an audio inlet. Then you can run an audio signal in there instead. I wasn't going to tell you that. If I just want to just multiply by something else, then I just don't say what multiplies by this thing and then it becomes an audio input. Then you can be multiplying two of the audio signals. That's really for next time, but that's the thing you would do. Yeah?

Student: [76:38] I have a question. Why does the pink object not have two inlets? Why are you going just inlets down this way to the right?

Miller: [76:46] Yeah, isn't that stupid? OK. So inputs to objects can have various functionalities, and one of the particular things that you can send an input is an audio signal. But there are other things you can send as an audio signal input as well. [77:06] Of course if a thing had two different audio signal inputs, then we'd have to have two different inlets in order to be able to disambiguate them. But if it takes two things that are different, like the message in an audio signal, then you get away with these in the same inlet. If there was less clutter on the screen, you'd just combine them. That's the simple answer.

[77:30] Other questions? All right. Go look at the homework assignment. I don't know if my machine is going to be able to play it. But it's going to be to do this.

[tone sounds]

[77:40] Firefox... I don't know if I bookmarked it.

[tone sounds]

[77:47] And somewhere down here... You get your assignment here, which is to do this. Now I don't know if this is going to play correctly, so...

[tone sounds]

[78:03] It's lame. All it is, is just a musical fourth that gets louder and softer. It's checking whether you can control amplitudes and frequencies, and understand the difference between them. And it's checking whether you can actually get around the oscillator, and the multiplier, and the adder.

[78:27] Basically what this amounts to is understanding oscillators, frequencies and amplitudes, and being able to kick PD on one, which is helping you do the hard part.

Student: [78:39] When we turn in homework we'll be turning in all of this?

Miller: [78:42] Oh. To turn the homework in, just upload the patch you made. And I will give you more details about how the patch should act in order to conserve the TA's sample. There should be a clear way to turn it on, that sort of thing. But more about that next time.

MUS171 01 06

Instructor: [0:01] What I want to do is several things at once. I need you to just look at the practical aspects of running and surviving .pd. How many of you are trying to run .pd and not succeeding? Three. OK. One of you emailed me, I forgot who, and didn't have - four - didn't have sound coming out.

Student Questioner: [0:29] I figured that out.

Instructor: [0:30] Oh. OK.

Student Questioner: [0:32] It was weird. It didn't show me any numbers though with the outputs. But when I did the test tone after a little bit of a restart on the computer, it came up. It was kind of strange.

Instructor: [0:43] Something like that's been happening to me today, it hasn't happened before, which is that I had to try twice to get it to run.

Student Questioner: [0:53] I use the version 43 instead of 42 because that looks more familiar from the one we looked at in class.

Instructor: [1:01] I think I'm running 42 right now. I'll tell you one thing that doesn't work in 43 in case you run the 43. This is something I still can't figure out how to fix. For who it's not working, can you tell me what symptoms you're getting?

Student Questioner: [1:23] It wouldn't allow me to put in objects.

Instructor: [1:25] It wouldn't allow you to put in objects?

Student Questioner: [1:27] On the clip menu everything was [inaudible 1:30] .

Instructor: [1:35] Maybe you're looking at .pd's window here and trying to do put, and for that you need to be actually talking to a real document. I didn't actually say this but this is .pd's print-out window. You can have this but it won't be doing anything until you have some number of patches open, and you can have one or more patches open and they're all running, all at the same time. Furthermore, they can talk to each other, so you should be aware of that possibility.

Student Questioner: [2:13] I can get single sign-wave to play. When I try to put in another oscillator I have to get crazy. I can clip and then it's just gone.

Instructor: [2:28] I think I might know what happened to you and that's something that I'm going to try and address today. It could be that what they're doing was numerically outside of the range of the possible values that you can convert, and there were ways that you could do that that would cause it to make silence. [2:46] And that's a "gotcha" that I want to try to help you avoid today if I can succeed.

Student Questioner: [2:54] I'm just having problems downloading .pd on my computer.

Instructor: [2:58] PC?

Student Questioner: [2:59] Yeah. PC.

Instructor: [3:01] I got a PC today and I'm not sure if I'll have the same problems as you, but if you see me doing something that you're not doing, that might help. Otherwise, see me after class today.

Student Questioner: [3:15] Do you know when a 64-bit version of [inaudible 3:17] ?

Instructor: [3:32] The last I heard, someone had a machine and they were going to compile it on but no one really has yet. You might have to do it yourself. OK, so, next matter. I have another thing to sort of just check off which is, in class didn't exist as far as web suits, he was concerned on Tuesday, but that should be fixed now. Is it decently clear how you would upload a [inaudible 4:12] house. I have one slight comment to make which is that it's possible to get confused downloading patches on the web. [4:23] I actually don't know if I'm on the network, so I don't know if I can show you this, but I can tell you this. If you see a patch on the web such as, for instance, the patch that I saved from Tuesday which is on the website for the course. You could click it and it will download you a nice patch, or you can click it and you will see this bizarre text in your browser. You could click it and it will just download you a nice patch, or you could click it and you will see this bizarre text in your browser.

[4:49] If you click and see the text in your browser that's because PB patches are, in fact, text files, and if your browser see's that it's text, it might just decide to show you the text instead of saving it to another file. This is not a problem. Just save it as a file anyway and make sure it ends in .pd and then tell your computer the .pd things are your data documents, and then you're happy again. I tried to download this patch and I just saw gibberish on my screen. If you're curious, by the way, you get to look and see what patches are. They're just text files and they just have gibberish in them that describes how you can make a patch.

[5:33] And, furthermore, those of you who get too excitable too late at night, if you learn what those messages are you can generate those messages from .pd and you can make patches that build themselves. I'm not going to show you how to do that, though. I'll have to figure that out, or, everyone on the web is doing it. Just so I can shut this window down, I'm going to put this up as a sort of review for today. What I want to show you now are two other objects forgot something. I actually gave you six objects last time because there was also the push-button.

[6:24] This is my resume from last time and what we're doing this time is another control which is a number box. It suddenly means now that you can make wonderful analog synthesis type sounds. Arrays which are graphs which you can do this with. These things are functional objects which I will grab and use as needed as we get through today's stuff. Today's stuff is mostly going to be figuring out what went wrong with the last time. So now what I'm going to do is, watch this, it's control, alt, backspace, that is equivalent of, save as. I am going to give myself a new file name so that I can make a new check point.

[7:27] This will be built and I will try it, I will try to save these things before I erase major portions so that you can see, in a progression what happened as we went through.

[7:41] Now, review, let me just make the patches that last time very quickly and show you how you can see there doing and then go on from there. So oscillator 440 hertz please, and then I will say, oh...

[8:09] If you have an object selected and if you hit the key accelerator for making a new object, it doesn't just make a new object but it makes a new object and connects it the previous one. It doesn't matter to you now, but late at night when your make hundreds, and hundreds of objects. You will get to like this feature, most people like it. You're going to multiple by 0.1 to the tenth and then I am going to send this out to the digital analog converter.

[tone]

Instructor: [8:44] I am going to turn the beep off, this is the cooler "hello world" hash. Now, what I am going to do, is show you not just how to print stuff but how to grab it. To print stuff, which is from last time is a print object, which is a print till by the way because it prints with a single input. Now, I am going to talk to this 0.1 thing and then I am going to make it a push button. [9:23] This is the thing that I forgot to tell you first the first seventy minutes of the other class. It is that you're not going to get very far just trying to click this button like this because I am in edit mode and I want to get into run mode, and then I can click it and have stuff happen.

[9:38] Except, nothing happens because you have cue on. [tome] These are the number that we would respond to one 64 sample of digital sound and my apology about the horrible format.

[9:53] Now, another thing that you might wish to be able to do is see it, as in a oscilloscope or as in a sound editor. I am going to introduce that because I am going to be using it to go back and make sure everyone understands about amplitudes, frequencies and modulation again. Even what the word modulation means, so to do that new materials start now.

[10:18] There is a wonderful object, called an array which you can get down here. This is unfortunate, there is a thing called the graph with is a rectangular that you can through arrays inside. This is a array which is the thing you throw inside the rectangular, which more like the thing that you want, because this will just give you a empty rectangle and no way to stick an array inside it. Well, there are ways; I am not going to tell you yet. Given an array like this, and it immediately says "I want to know all this nonsense about the array" The important nonsense is, what its name going to be.

[11:05] I am not going to tell you yet all of them how .pd keep naming, how .pd treats names is a subject all to its self. But, I am just going to use a name for now. ID factor rate one sounds good to me right now. At size this is the number of points that the array is going to have, so that number would be, for instance, at our sample rate if I want a whole second of sound I would have to ask for 44,100.0.

[11:35] As a general thing, pd doesn't know much about sound. It does know that a second of sound require 44k one point, anyway that number could change because the sample rate of the computer might change while pd is running. It doesn't make sense to ask for a array to hold a second of sound.

[11:52] So, you have to go on and tell it numerically how much sound you want in the array. In the same spirit, that you had to tell the oscillator how many cycles per second it had to vibrate in order to make you a nice A440. Well, what I call A4440 is just a concert A4.

[12:10] Here I don't want 44,100 points, I want a 1,000 points juts for now. This is the aesthetics but I prefer points to Polygon's, Polygon's means it draws little segments between the points, and points means it just draws the points. So I am going to choice points because it's me and that is my preference. And I am going to say OK. And it says OK and it draws me a thing. So, I am not in edit mode, so lets get in edit mode and look around. It says "Hi my name is array one and my values are all zero"

[12:49] By default these a values range from minus one to one, which the same range as audio is, which is a good thing. For instance it is a good thing because I can now use that to graph what's coming out of this network and show you what it looks like as an audio signal.

[13:10] Let me do it in RAW view again, in the same way that I did it with the RAW. I am going to need another push button and I am going to need an object whose name is Tab Right. That an ugly name but it fits in a series with a bunch of other names that has to be name the way it is. Then I am going to say "What table," that's to say array we are going to write to.

[13:39] Nomenclature in computer music arrays were called tables, this has been true since 1958 and there is confusion in .pd as to whether something should be called a table to be true to its computer music roots or to be called array which is what the thing really is, which is just bunch of things all the same types. The name kind of flops, back and forth, between things that say table or tab right and a thing that says array. I apologize you never know what's going to happen if you develop something for 20 years.

[14:15] Now, I am going to listen to this thing by making your hear it, notice that I mention the last time. This really skinny wire that hold or carry messages and these are fat wires that carry signals. Different signals are happening all the time. and messages are happening only sporadically. Now I'm going to click this forgetting that I have to lock the patch. So I'll lock the patch, then I'll click it, and nothing happens. Why?

Student Questioner: [14:40] DSP is off.

Instructor: [14:42] DSP is off! Go up here, turn on DSP. That's why I left this thing on the screen. Then, ta-da [tone] . We are looking at [inaudible 14:49] . OK, so now DSP is off but we just wrote into the array. In the same spirit as for the print tilde object, this thing has an audio input but what it does is something that it does sporadically, that's to say when you want it to do it. You have to send it a message trigger to say do your thing. [15:17] Doing your thing amounts to commencing to record the audio signal that is coming in and continuing to record until you reach the end of the array, at which point you stop. It doesn't loop around. You can make it loop around, but by default it just doesn't.

Student Questioner: [15:39] What was the shortcut for DSP again?

Instructor: [15:41] What?

Student Questioner: [15:43] What was the shortcut for turning it on?

Instructor: [15:44] Oh, the shortcut for DSP. I don't even know if this is documented. Control slash turns DSP on [tones] and control dot turns it off. Control dot is fairly standard Macintosh language, and the slash is just next to the dot. [15:57] Yeah?

Student Questioner: [15:58] How did you get the array to look like that?

Instructor: [16:00] To look like that?

Student Questioner: [16:01] Yeah, you did it by writing the top [inaudible 16:03? When I turn my DSP on it didn't do that.

Instructor: [16:07] Yes, OK. I did two things. One is, I clicked the tab, I clicked this button. I did that while I was out of edit mode.

Student Questioner: [16:15] Oh, OK.

Instructor: [16:16] In fact if I do it again... Oh, nothing happened because DSP is still off. We'll turn DSP on.

Student Questioner: [16:23] OK. So you're just in edit mode. [tone]

Instructor: [16:28] OK. With DSP on, each time I click it it'll make a new recording. You know what, I should really... It's all right. Do it the easy way. [16:42] Yeah?

Student Questioner: [16:43] How do you edit array one?

Instructor: [16:45] How do you edit array one.

Student Questioner: [inaudible 16:48] if I want to change it to [inaudible 16:51] [16:48] tab.

Instructor: [16:52] Oh, I see, if you want to change the size. Right. OK. I was going to forget to say this and, by the way, this is very confusing. If you have more than a one button mouse, left click. If you have a Macintosh that only gives you one button, I think you command click, or option click. I forget. It gives you properties or open or help and do properties. Open means, hi, I'm a sub-patch and I only contain this. [student laughter]

Instructor: [17:27] That's good for other things. Properties is going to do this: It's going to give you (watch out) two windows because there really are two things here. There's the array, which is the squiggly line, and then there's the graph, which is the rectangle [inaudible 17:42] . So when I asked it to make an array, it made me an array and a graph and put the array in the graph. By the way, there's no intellectual content in any of this. This is just .pd lore. [17:58] So this is the array. This has to do with the points in there. Since there are thousands of them that I can ask it, oh, let's have 2,000 of them. Then I'll say apply here. It's going to be a little embarrassing because you can only have 1,000 good points and then the other 1,000 are now zeroed. It did have the decency to change the bounds of the graph so that the array still fits in it, but that's about all it did for us.

Now I'll do back to 1,000, like that. OK means apply then disappear. Then the other thing is--what's that graph thing doing? Just left, so let's start over. Oh, and I'm drawing a polygon [inaudible 18: [18:25] 41] . OK, now let's get the properties back so we can see not just the arrays but the canon. Now, here we can say the X... OK, X and Y really mean the horizontal and vertical axis. X is going to range from zero to

1,000. That means this point here is point zero and this point here is actually 999 because there are 1,000 points. If I change that, you don't need to remember this, you get the following embarrassing result-- Your graph wants 2,000 points but the array only has 1,000 so it looks stupid.

[19:22] I could also say, oh, the range is only 700, say. Then we get... It doesn't do it. Oh, yes, because I'm running 43 and there's a bug, so we say no. If the array doesn't fit in the graph... At 43 it doesn't draw it. That's bad, and it's going to take me hours to fix it because there's something subtle wrong there. If your thing isn't drawing, it's because it doesn't fit in the graph. My apologies. Go back to 42 if you really need to see it. Somehow I didn't realize that was running 43.

[19:59] OK, we're going to cancel this. Oh, except I'm going to do this. The Y range goes from one to minus one. Isn't that ugly? That's because computers think numbers go up that way, whereas mathematicians think numbers go up that way, or graphs think numbers go up that way. You have to say, yeah, I could say it goes from minus one to one but everything would be upside down and it would be confusing. For right now I'm going to say we'll go from two to minus two so that you can see the thing drops in size because now it's two and that's minus two. There are ways of getting the thing to show you what its bounds are but, let's leave it that way. I'm going to just leave it like this.

[20:50] Just to belabor a point, let me disconnect this so I can have DSP running and not listen to it. I just have to graph the output of the oscillator and not the output of the multiplier so you can see what a full blast oscillator looks like. Now recall, I have the graph that's the rectangle going from two to minus two. The signal, the oscillator signal, ranges in value from minus one to one, which is full blast as far as the computer is concerned.

[21:30] You can make signals that are more than full blast. These numbers are all floating points, so you can have numbers that go up to 10 to the 38th, 37th power or something like that. Your speakers can't play those. And it's a good thing, because you would vaporize the planet if you could.

[laughter]

Instructor: [21:50] But as long as what goes out is between minus one and plus one, then your computer will, I hope, faithfully turn that into voltages that your earphones or your stereo can... [22:01] All right. Just for, hadn't got these things... OK, this is the moment I think perhaps, to save this patch and continue. Played it, and I'm going to save it as the range, the signal range.

[22:30] All right. I'm going to now show you what happens when you... Oh, I'm going to turn the volume down in the room before I do this. And I'm just going to play the oscillator full blast into the speaker, or in the mixer. But the mixer's volume is going to be down, so we won't lose our eardrums.

[22:49] And then, I'll show you what happens when you add another one, which will cause things to malfunction in a novel way, which actually, you might have already heard a couple times. So, let's see. I don't want this anymore. Don't want this anymore. And I'm going to turn the volume down. OK, then I'm going to connect this to this. Actually, what I should do is turn them off.

[tone]

Instructor: [23:32] It's too much for the mixer. You can hear already, there's not a clean sinusoid. But let's, we'll pretend it's a clean sinusoid because I'm about to make it even worse. [laughter]

Instructor: [23:41] I'm going to say, "OK, that's good." And I also want to go and get 550, which is a perfect third of a 440. First off, I'll get... [tone]

Instructor: [23:52] OK, hear that? Now, ready, set... [tone]

Instructor: [24:00] What happened?

Student Questioner: [24:03] Clipping.

Instructor: [24:04] Clipped. Yeah, yeah, OK. There's some, those of you with digital audio experience know what's going on here. [24:10] But I'll graph it for you to show you what's really happening at the level of the signal, and to do that I have to introduce the final object for the day, which is clip tilde. And this is a, one of those minority objects which, it just, sometimes you just need it. But those times are maybe only once a week or so. So here it is, click.

[24:36] This is the, clipping is, it's a term that... I don't know how old it is, but it certainly dates back to the old analog electronic days. It simply means what happens when signal goes out of the range of the audio device that is receiving it.

[24:49] So if the standard thing about clipping is you can hook an electric guitar up to an amplifier and overdrive the tubes. And if you overdrive the tube, well there is a maximum or minimum current the tube can put through. And beyond that, it just says, "Well, I'm clipped. I can't go any further, so I'm just going to stop right where I am."

[25:08] So it's like in this building, if you ask for floor minus one or floor four on the elevator, you won't get it. You only get floors one through three, because that's where the elevator goes. It's the same deal.

[25:19] So for instance, I'm going to clip between minus one and one, which is an exact imitation of what actually happens when the audio goes out of your computer, because the range of possibilities is minus one to one, and if it's out of the range, it is simply clipped to the range. And if I knew that, and if I for instance, had these two oscillators together...

[25:43] Oh, before I do that, sorry. Before I do that, I'm going to do this. Push that one, here we go. So here's the first thing.

[tone]

Instructor: [26:03] Oh, right. We're only listening to one of them, so let me play you both of them and show you both of... [second tone]

Instructor: [26:09] So what's really happening is the periods of the two oscillators that we have are short. If each of them is fitting 20 or 25 cycles, then the thing... [26:20] What you can see is that the thing itself is repeating at its much lower rate, which is in fact the, what is it now? It's the greatest common factor of those two frequencies, if you like, or the least common multiple of those two periods. OK? Or, now, finally, I'll show you what the computer... Let me show you.

[tones stop]

Instructor: [26:44] Or to show you what the computer really is playing, let's look at it this way. Rather than add it right into the tab where I will add the clipper and the tab right, and then do that. Aha. Now, the signal that you saw before, even though, yeah. Even though the signal that you saw before was clearly periodic with this period, you didn't hear that period because in fact, in its internal structure, it really only had two components, each of which had a much shorter period, and your ear resolved those. It didn't hear, it didn't make a difference to... It just heard the individual [inaudible 27: [26:58] 25] to confuse them, right? At least my ear couldn't.

[27:30] But if you clip it, you'd make that be no longer true. There's simply no possible way you can hear the signals having any period other than the period in this that it's got. Yeah?

Student Questioner: [27:41] What level does toggle use?

Instructor: [27:43] These? Oh, this is not a toggle. This is a, this is the button which is called Bang. Oh, and if you go to toggle, you'll get another rectangular thing, but it's a toggle switch which goes on and off when you press it. That's for later. All right? [28:00] So this is clipping, and this is what your audio hardware does to you. Now, let me show you how you can make your life even worse.

[laughter]

Instructor: [28:10] I think this happened to one of you, but I'm not sure. I'm just operating on a guess now. So I'm going to say if you've got an oscillator like this... And now, I'm going to listen to my nice A440. [tone]

Instructor: [28:28] And then I'm going to add another oscillator here. Ready? [tone]

Instructor: [28:32] Oh, that's not what I wanted. Oh, yeah. I miscalculated. Add another one. Oops. What happened here is this. Let's see. OK, so let me put in the, let me get that something into, yeah now we are clipping. Actually I can save some steps by just listening to the [inaudible 29: [28:40] 04.

[tones]

Instructor: [29:06] Now we get this, and this a problem because it is the sum of a sinusoid and another sinusoid that has zero frequency and zero phase, which means output is one volt... right? And the result is.... that half of the cycle is below one still - the half that was from zero to minus one is now going from one to zero, and the half that was going from one to zero is now going from two to one, and it's getting clipped. [29:46] OK. In fact, when you learn how to control this, you can have a lot of fun because you can do this controllably and you can change the timbre of sounds by selectively clipping more or less of it, and this, for you electric guitarists, is the bias knob on your MPEG amplifier thing [laughter] . Fender doesn't give you the bias knob, but the other manufacturers do. OK. Now I'm going to add another one. And what do you know?

[laughter]

Instructor: [30:17] The patient died, [tone] and the reason - worse - the reason why the patient dies is because now, the entire sinusoid here is above positive one, and so it got clipped to plus one, and so the result is a signal that you can't hear, you an only smell it, because it will melt your speaker. Speakers, theoretically will go down to zero Ohm's of DC and your stereo probably wouldn't do this to your speaker, but if it could, then you would have to call the fire department or something. [30:51] All right, so this is [tone] this is how to make your life hard by making signals that are out of range, and, oops, that's interesting. And so, the first thing you hear is funny distortion, but you don't know whether the funny distortion is your patch, or whether it's just because your earphones are bad or something like that, and then when the signal goes away all together, then you still don't know which it is, but it's very possibly that it might be this.

[31:22] It might be a good idea to equip yourself with one of these things at the same stage as you are making your output. So, for instance, one thing that might be a really good idea would be whenever we do, we'll just put a nice adder at the bottom. It doesn't matter whether you're adding more than one thing or not. This adder really is just here to remind me that whatever's going up the DAC, it's going to go up the- it's going to be graphical as well, so now if I for instance do this, then I can see it. Now, when you're turning in homework. Oh, good.

Student Questioner: [32:25] What's the adder for? Because I didn't see anything. What did it do?

Instructor: [32:28] Oh, what's the adder for? The adder is there because when I change, when I add or take out stuff, I'm going to hook it into the adder instead of hooking it into the DAC, into the tag right. And then that way I'm not going to forget to add something into the DAC that I didn't add into the tag right. So really I'm adding zero.

Student Questioner: [32:44] OK.

Lecturer: [32:47] I'm going to explain more about adders in a second, but all I'm doing really is adding zero onto the signal, so I'm just wasting operations really. The reason I'm doing that at all is so that I can do stuff like this. In fact it doesn't even matter which - you wouldn't need an adder in .pd at all because signals automatically add new waves. Well OK, that's not quite true as I will tell you next. Any questions about that? Yeah?

Student Questioner: [33:18] Wait, so, is the adder unnecessary? Like can you just...

Instructor: [33:23] The adder's unnecessary. It's only there so if I make a connection to it it makes the connection both to the DAC and to the LAN and tag right too, and I could do that in a spiffier way, but I'd have to use another object. So what I'm going to do is I'm going to put this object in a nice state so you might be able to remember what I was doing... This is going to get saved, and then I'm going to make a new thing and do some other stuff... OK, so I'm going to insult your intelligence just for a few more moments and talk once again about amplitudes and frequency then graph them, because.... there is - I sense there is still confusion about this, and I don't know how to simulate this confusion in my own brain, so I'm just going to try to confuse you and hope that you stop me when things get confusing. So, here we go. Now we're going to take this away all the cruft. Oh, I didn't say this, but of course I didn't give this oscillator an argument so that means it's going at zero Hertz until I tell it otherwise, but [inaudible 35: [34:19] 01] .

[35:03] Oh, I didn't mention something else that's going to happen to you. If you open two or three of these patches at once, and if you have an array named Array! In all of them, or in more than one of them, you're going to get complaints because .pd will have two things named Array1 and the name is supposed to let it figure out which one you're talking about. So, if there are two of them, it's confusing, .pd will print you out a warning message and it will choose one of them for you, and it will probably not be the one you thought it was going to be.

[35:34] So what I'm going to do for that is I'm going to now change, I forgot to do one, so I am going to try to remember later. But I'm going to change the thing of this one to Array3. And the variance..

[tone]

Instructor: [35:59] Yeah, were going to introduce that. So now, you don't need the clip anymore. What I'm going to do is I'm going to introduce the last thing that intend to do today; which is the number box. Oh, no. Second to last, sorry. I'm going to start; OK so there are two distinctions which are going to be important for the next half hour or so. One is frequencies versus amplitudes and inputs versus outputs.

[36:33] In other words how you change the amplitude or the frequency of the sound and what it looks like when you change those two things in terms of the graph. That's one thing. And the other thing is that I have to give you some more details or more information about distinctions between messages and signals. And these two things I don't know how to separate them very well, so I'm just going to fold them together into one discussion.

The first thing I want to do is this: [36:57] go back and show you the oscillator, but give it a variety of frequencies. Let's see... [tones] This is my 440 Hertz. And if I start changing it I see this kind of change. All right? So now we got a nice stereo or something. [laughter] You will find this very hard to play musically. And we'll talk more about that later, perhaps. Now that is changing the frequency of the oscillator. That is the same thing in some way as telling the oscillator that it's frequency should be some number like this. You could even do this: but we don't, but you could. You could say, "Hi, such and such, go away."

[38:03] The oscillator is now 440 Hertz. [tone] But meanwhile I'm going to change it to perpendicular. [tone changes] This is perfectly legal, but this is confusing. This is confusing in one of the two ways you could be using a .pd that I know of, that I'm going to cover now. Which is that I initialized the frequency to 440 but then I overrode it, or replaced it if you like, with the value 205. This is not good style. I'm doing to show you how you can confuse yourselves. The reason you can confuse yourself now is that you can think that this number is still 440 because it says so right here. But it's not because I changed it. It would be smarter if I'm going to run something and it's going to change it, is to not give it a value which is not going to be reality after I do something to it. Although it's perfectly legal to do it.

So now, the other thing that I want to do is say...so that's why I'm saying oscillator without an argument now. Because I have a number box hooked up to it now and I'm trying to be hygienic about the way I'm making this patch. So the next thing is going to be: [39:00] now I'm going to starting changing the amplitude. Which, if you recall, one does with a multiplier. What I showed you before is that we can multiply the thing by some number like a 10.

[39:32] And then we would get something that looks like this. What did I do wrong there? Oh, oh boy! This is the oldest mistake in the book. Now what I've done is made a new oscillator, so it's frequency is zero. And I had this nice number box going into to it, so it thinks it's 205. But I haven't actually sent the oscillator the message 205, since I made it again. So let me fix it and then break it again and then fix it so you'll see this. Because this is a thing which could be confusing.

[40:09] So I'll say all right, "go to 220 please." So you're at 220 Hertz. [tone] By the way it's quieter now since I'm doing this other thing. That's good. Now I'm going to say my oscillator [loud squawk] I want it to be something else. No actually, I just want it to be an oscillator, thanks. I hit Control-Z there to do an undo. And nothing, because it didn't really undo the change. It just made me a new oscillator and I told it to build me a clean one. It's clean in the sense that it hasn't received any messages that might change things like its frequency or its phase. So it's sitting there at the frequency zero, so it's putting out the value one. So I should be looking at the value 0.1 here, and I'm sure that's true. And that will be true until I do something like touch this. [tones] And once I touch this the value of 220 is added. Yeah?

Student Questioner: [41:01] Can you engage it too by taking out the in-putter and putting three checking counts?

Instructor: [41:06] No. Yeah so this would not work. Actually another way to break it would be to type a space here. And then click outside and then remake the object. Just typing a space dirtied the text string so it says, "Oh, I've got to make a new one." It makes a new one and it re-formats it so it's like the old one, but it's a new now. Yeah. Taking this out...oops, taking this out and reconnecting it doesn't do a thing. You have to actually make the thing create an output. [41:40] This seems like a horrible bug. And it will seem like a bug for a year or two until you realize that it's actually a feature. [laughter] I'll show you how you can even confuse yourself worse. [tones] Now what I've done is set up and argument between two sources of control. Watch. [tones]

Here's my low A, and here's my high A. Which is it? [tone modulates] Well you can't tell from looking at the patch. So maybe you shouldn't do it. [laughter] You're perfectly welcome to, and there's situation where you would want to. But it's not likely that you would want to display the value coming in from two ways. One of which is almost guaranteed to be wrong and if you get the time.

[42:33] So you might indeed want to get two sources of frequency that are acting at different times. In fact that's an important aspect of why PV is designed the way it is. You can do things from different sources. But if you do that, then you might not want to see what each source is trying to tell the oscillator. You might want to see what the oscillator itself is doing in the patch. So I could do that by either clarifying or further confusing, depending on your point of view. Let's do this.

[43:11] Now notice the oscillator is still going on at 447 Hertz. It doesn't care about any of this stuff that I've been reconnecting. Because these are messages, the messages are not flowing until I do something. I'm just making a patch. But, now, if I do something, like say, 440, please, then I've got 440 but at least I can see what it is [tones] . This is maybe a little bit better, and this will be OK until I confuse myself again by announcing directly into here. This is not computer science. This is music. So, let's get rid of that because this is just easy. I just did that to warn you how you can confuse yourselves.

[44:00] So, this is frequency and, while I'm at it, of course I've all ready shown you, I think, that this makes these kinds of changes [tones] . And, the amplitude is a similar situation except that I wouldn't dare do something like put 440 in here. Would I? Now, remember, I have the mixer way, way down right now. If I had done that at home with my stereo at its usual setting I would have jumped out of my chair [laughter] . What that did, well, I can even graph it for you, though I clip it to graph it.

[44:47] So, I'm going to say "clip", minus one to one so that, in fact, I'm simulating what my computer's really putting out. And now, as long as I'm between mics one and one, everything is good. But when I say I want 440 volts, please... [tone] . That didn't graph terribly well.

Let's try and do the polygraph again. This is what Jimi Hendrix got when he played a note on his guitar. Basically, the tube was always either saturated and then you got what is essentially a square wave. This is a computer music technique. It's been elevated by having a name; it's called "wave shaping". You will hear more about this or read more about this in chapter five of the book. We will hold off on all the gory details until later.

[46:02] So, different ranges of numbers might be appropriate for selecting frequencies as opposed to selecting amplitudes. Now we're multiplying by zero. No matter what you multiply by zero you get zero and it looks like this and sounds like what you're listening to. I meant to say something and didn't finish saying it. You've seen me do this to number boxes and, of course, if you're in edit mode you're doing that or getting frustrated.

[46:40] But, you can also type the number in, and then if you hit character term "enter" the number is generated as output. So if I do this, the output is going to be four, it will be four in fact if I hit enter, but if I want 440 I'll do this. Now, I did that so I can do this. I'm going to go here and I'm going to type the number zero-point-one and hit enter. Now we've got a complete computer music instrument with amplitude and frequency control. Now this is kind of stupid because if I reach for this number box - of course if I just click on this and start scrolling or dragging them down [tones] . It's impossible for me to get a nice sound with this. I can't get anything between zero and one, which is completely quiet and full blast.

[47:49] It is true that you can hold the shift key down and ask this thing to go up and down in hundreds. But if I were designing this thing as a patch I'd want to do something a little bit more user friendly. I would take the thing and divide it by one-hundred or something reasonable like that before I started messing with it. That's going to take me outside of my budget of five object types, although it's nothing but time. What I'm going to do is, now, go back and show you a little bit more about manifestations and differences between signals and messages.

Student Questioner: [48:37] I'm sorry. How do you actually type the number value into the number object?

Instructor: [48:41] Oh, I didn't tell you something important. You have to click on the thing which doesn't give you any visual feedback, which is stupid. It should. Then you can start typing and then you hit enter to make it go in.

Student Questioner: [48:55] And you're out of edit mode.

Instructor: [48:57] If you're in edit mode it won't do it for you.

Student Questioner: [49:01] If you're in edit it won't.

Instructor: [49:02] Yeah. It has to be in run mode or locked, if you'd like. So, the final object that I want to tell you about is "print". What I've been doing on the clip menu so far is mostly reaching for objects. There are three things that I still need to do that don't require just making an object and typing into it because these are user interface objects. They're things that make surface of the patch. They are the number box which is here, the bang, which is here, and the array, which is here. Everything else that you see here is just plain old objects. All right? [50:04] Now, what I want to do is save this. So, now we have this print object without a tilde. OK, so it expects not a signal, but a message, or messages. And now, if I change this value, get to actually see what the thing is doing.

[50:26] OK, this is how you debug patches. You want to know what your patch is doing. And the way you find out is if there are messages going through. You can sometimes help yourself with number boxes, but frequently, if you really want a complete record of what's going on, you just hook a print up of something. And then, if you want to know what's going on with the signal, you have things like this tab right tilde, or print tilde, which prints the things out here.

And those two things have single inputs, so they are appropriate for talking to signal outputs, and they will show you how something is varying in time as [inaudible 51: [51:00] 08] . There's something else I needed to tell you, if I may interrupt.

[51:23] So, back to frequencies and amplitudes. Questions about this?

Student Questioner: [51:29] What was the keyboard shortcut for connecting two objects?

Instructor: [51:32] Oh, there's no keyboard. OK. There's one trick to that. There's only one way to make a connection that doesn't require that you actually do the drag thing, which by the way is horrible on a trackpad. And that is, we have an object, and we want to make a new object to be connected to it. So, while it's selected, you hit Control-1, Control-1 in general just makes a new object. But Control-1, if something ... one other object is selected, still make a new object and connect to... [52:14] Other questions? It's all good.

Student Questioner: [52:19] Does print run automatically when you turn on the [inaudible 52:22] ?

Instructor: [52:22] Yes. Right. So print-tilde, you have to click it. With print tilde, you have to send it a message to tell it to operate like this. Let's see, you can do this, it's starting to get messy now. Turn it on. Oh, right, I've got the input down to zero. Oh, cool. Now I've shown you how you can make a patch that shuts up, but it's still running, so you can debug it without listening to it. [52:59] Just make your patch have an amplitude of zero, make it be a control, the patch is going to be muted in the sense that everything I'm doing is initially getting multiplied by zero, so they're going to hear it. But we can still go through things and do things like look at the oscillator numerically, or look at the oscillator graphically. No, we can't do that, but we can do this. Right.

[53:32] We can do that without having to have everything come over our speakers. It's a good thing.

Student Questioner: [53:38] Is there a way to graph in real time?

Instructor: [53:42] Yeah. There's an object called metro, which is a metronome, which will send out a message repeatedly instead of just once when you click it, and then you can have it just grab 10 times a second or whatever you want. I'll get to that. It's out of my object budget right now.

Student Questioner: [54:00] Can you put up a bang on an oscillator, and how to play it with it?

Instructor: [54:06] No. Oh, yeah. So how would you make something play when you click it? Yeah, I have to do some more objects to do that too. So, in fact, to make something start playing, you don't tell the oscillator to start playing, because it's always playing. You have to multiply it by something and make the commitment multiplied quickly in zero. And that in fact is the kind of the crucial thing that I'm hoping to find a way of saying today. Is ... yet you've turned things on and off not by turning them on and off in the sense of computing or not computing, but by multiplying them by numbers that vary from zero to not zero. [54:44] And the reason for that is, obviously you would save competition time if you just turned the thing on and off, but that doesn't make for nice musical beginnings and endings of notes. We want things to turn on and off smoothly, which you'll have to learn how to do. But the only way to accomplish that is by multiplying the outputs as a way of controlling the amplitude instead of simply by turning things on and off.

[55:08] And a wonderful example of that is this. Let's multiply ourselves by not a number, but by another oscillator. [tone] And let's maybe go at one hertz. Now, this is where we're too loud for the mixer again, so we're hearing distortion. So rather than just take that lying down, I know how to deal with it now. So, let's say. Not only are we going to do that, but we're also going to have a multiplier controlling the overall amplitude. And now, we have a patch that does something interesting, but doesn't do it full blast. I'll start paying attention again and being careful about the outputs, that I can turn the mixer up. Whoops, oh my.

[56:22] OK, so now what I'm doing is I'm taking this nice oscillator, that has a pitch of 330, and making it change its amplitude. So, as I click and look at the signal at different moments in time, I get different strengths. That's because this one is just making that waveform, whereas this other one, this one hertz job, if I look at it, now remember. There are 1,000 points in this array. That means the array has a 44th of a second. So, at the speeds that we're looking at, you don't see very much varying at all in a one hertz sinusoid.

[57:08] But of course, you hear variation, because there are 44 of these things going by per second. Faster than the video frame rate of the thing. So, this thing, even though it looks like it's doing almost nothing, is in fact varying, it's going once a second, so what it's really doing is it's going up, down, up, down, up, down. And what you hear is two peaks per second, because you hear a peak when the amplitude goes up to one, and you hear a peak when the amplitude goes down to minus one. Which means it got multiplied by minus one, which is also full blast.

Student Questioner: [57:44] Is this different than Tuesday? Because Tuesday we were doing something with the frequencies.

Instructor: [57:48] Yes. This is a completely different situation from Tuesday's situation where there was an extra oscillator but it was controlling the frequency of the oscillator we were listening to. So, now what I want to do is say, "OK, this is good", but there's also the possibility where I had my nice oscillator. Here's an oscillator, and then I had an adder and I was adding 440, I think, and then I had another oscillator with another amplitude control.

Student Questioner: [59:10] Isn't that one multiplying by 440?

Instructor: [59:15] Oh, thank you. I don't want to do that, do I. Now we have an oscillator. It's going to operate at some frequency, but, unlike last time, to give me more fun, I'll hook up a number there. I didn't have numbers last time, so I had to do it a little more severely. Now lets see what this does, lets do this. All right. Now we're getting complicated. [59:56] This is yesterday's patch with a little bit more controllability but doing exactly the same thing. This is the oscillator that's making noise. And, morally speaking, it's speaking in 440 hertz. But, in fact, we're taking that 440 and we're adding another audio signal to it. And that audio signal is the output of the oscillator whose frequency and amplitude we are controlling. Now, what was happening last night was this oscillator said OSC, tilde, space, six. So, I initialized it to six hertz. This was times, tilde, 30, and it was doing a fixed thing. But, now I'm using the number boxes here to fill in the frequency and the amplitude of this so-called modulating oscillator.

[60:52] Now, if we turn this thing on, like this, let's turn this one off, I have to turn the patch on [tone] .

[61:02] There's 440 for us again. And, now if I say, this oscillator will run in three hertz. There's that. And now this one was doing something different, which was this [tone] . This is a changing amplitude.

Student Questioner: [61:27] What's the point of adding the 440? Why didn't you just put the oscillator to the space, 440?

Instructor: [61:36] If I did that then it would be overridden by the values coming in. It wouldn't be 440 plus the values.

Student Questioner: [61:43] Oh. OK.

Speaker: [61:45] Yeah, so. That's a good question. It's not beautiful yet. I could have tried this. I just want to run this right in and then say 440. But now this 440 is actually being overridden by this. In fact, what I have really is an oscillator that is buried between plus and minus six hertz. It's very easy to make sounds that you can't hear. And the reason you can't hear the sound, you can look at it and see why you can't hear it, is that it's just not moving fast enough for us to hear it.

Student Questioner: [62:47] For the lines you have connected, can you just disconnect it and move it to a different one or can you take the line off an-

Instructor: [62:54] I wouldn't be able to disconnect it either at the output or at the input. It doesn't do it. You have to delete the old line and make a new one.

Student Questioner: [63:02] This is probably a silly question, but, it sounds like another that was [inaudible 63:05] shooting the signal [inaudible 63:07] troubleshoot.

Instructor: [63:12] No, basically, no. There's no other window here except when you've, I mean, you can build things that do that, but there's nothing built in. OK, so, that didn't work but this works [tone] . Now, at this point I'm just confusing you but I think it's 440. Furthermore, you can improve it by doing this. Now everything is up for control [plays a scale of tones] . Don't try this at home. Then you have this which is how fast it's going, this is just how deep it's going [demonstrating new tones and frequencies] . Now you have frequency modulation. This is 1960's computer music [laughter]. [64:25] That is known as frequency modulation. Frequency modulation's taking the frequency of an oscillator and changing it. Modulation is a musical term that just means "change". Well, maybe it means change with time or something like that, or repeated change.

[64:48] What we're doing now is we're taking this oscillator, it's frequency is the sum of a bass frequency, a super frequency if you like and an oscillator. This is an oscillator, the amplitude and frequency are 172 Volts and 82 Hertz.

[65:15] This is taking this oscillator and changing it's frequency by talking to it's input. This is taking this oscillator and changing it's amplitude by multiplying it's output by something, and that, contrast and other things makes it sound like this. We can make sounds with it. This is trigonometry. If this is cosine A, this is cosine B, this is cosine A cosine B, which we all know is half cosine A plus B, plus cosine A minus B.

[tones and laughter]

Instructor: [65:53] So algebra two is just this! Actually we're going to put that backwards. People will tell you that the reason, if you have two oscillators, two sinusoids that are nearby in frequency, they beat when you superimpose them, it's because they're moving in and out of phase. That's the layman's explanation of what's going on. [66:24] The truth is that the sum of two sinusoids is algebraically equal to the product of two other sinusoids, it's the same relationship I just told you and the product of tells you that it's changing in amplitude, which is the beating you hear, and the sum is the two that you put together.

Now I'm doing that backwards, I'm taking an oscillator here and I'm making it beat by multiplying with another oscillator that changes it's amplitude and that's just algebraically equal to those two that you have to have to beat. It's the same stuff you can do it either way.

[66:52] This however you can do to a real signal. Try that at home and see what you get. Find a politician you don't like and multiply him or her by a thirty Hertz oscillator, and then you'll get a pleasant surprise!

[laughter]

Instructor: [67:16] This is an oscillator, changing the amplitude, which is sometimes called amplitude modulation sometimes called ring modulation as an historical term which we'll never get out of. This is frequency modulation, which is taking the same oscillator, I should make a nice comment, emphasize what an oscillator is, the main oscillator if you like... This is taking this oscillator and changing it's input, you can do both at the same time if you want. [67:49] Yeah?

Student Questioner: [67:50] Is the number there zero right now? On the right hand side.

Instructor: [67:52] Yes. This one here?

Student Questioner: [67:54] Is that the amplitude overall?

Instructor: [67:57] This and this, those are volume controls, or gain controls that I have on the outputs of both of these patches, so I can turn them on and off, which is not making the computer not compute, but it's multiplying them by a tenth or a zero [inaudible 68:12] . [68:14] And of course we don't hear them because... [tone] This is a more correct way of showing you what's happening.

Student Questioner: So what is cross modulation?

Instructor: [68:35] Cross modulation...

Student Questioner: [68:37] Is that not an official thing?

Instructor: [68:41] I would use that to mean that I was taking one sound and multiplying it by another sound or maybe doing something else to it depending on the other sound... I don't know if it's a trick definition.

Student Questioner: [68:54] OK.

Instructor: [68:57] It's also, lots of terms... we'll get further along and see all sorts of other things. Any other questions about this? Sorry to labor this but I'm going to labor it slightly. What we have is two ways of looking at signals which are print tilde and tab right tilde going in to an array. This one's more more work but it gives you the best graph. It's more work because you have to make a button and choose when you're going to do it and make a graph that agrees in name with the tab right tilde. I could have five or ten arrays in my patch, and five or ten tabs right tildes and they could talk to individual rays. There's a print without a tilde and that is a thing which takes message input. And by the way, as I showed, it's good for printing these things out but it's not good for printing these things out. Furthermore it can even refuse to allow me to print a nasty message if I do that. In general you cannot take a signal and put it into a message [inaudible 70: [69:43] 13] the message wouldn't know how to deal with 44,100 numbers a second.

[70:21] I can take the message output and put it into a signal input, as long as there's not a signal going into the signal input as well. The message will simply sit by the signal. So numbers which are messages promoted to signals automatically, but not vice versa.

[70:45] What else do I have to tell you about wave review. One other thing, this is the review but this is important, I'm thinking I could have confused you by... If you do this, this input doesn't take signals, it tells it that it's messages. That's to say multiple or single binary operations like time tilde. If you feed it an argument, it initializes the value but also tells that the inlet is expecting messages as opposed to a signal. Multiply two signals don't give it an argument just say times tilde.

Now all this should be explained in the help windows for all these things. You can always get help [inaudible 71: [71:32] 39] .

[71:59] Any more questions about this? If you're having trouble running .pd and having to resolve it stick around, let me see it...

MUS171 01 11

Instructor: [0:00] But my apologies. The rule today is null and void. [0:05] OK, just to respond to one question. Especially if you have a PC, if you turn Pd on and you don't get that nice computer sine tone, but get a beeping on and off sound like a truck backing up. That is because PCs have horrible audio latencies and you will need to have this delay number up to some nice high number like that.

[0:32] Or maybe even one time within, what was it, Vista machine. I had to to this on the laptop to get the thing to operate correctly. So lesson one, don't use Vista, and lesson number two is if you don't get a solid audio output then that's a good place to change something.

[0:54] Increasing that will make the audio more robust, but will make everything react more slowly. So it won't be as pleasant to use. Another thing that you can do if your audio insists on hating itself, is you can ask it not to use audio input.

[1:23] You can do it that way or you can do it by saying, "Hi, I have zero channels of input." And that will run Pd normally, except that whatever audio input device you have won't be getting used in your -- you haven't used audio input anyway yet, so it doesn't matter. But when you do you will, you'll make your computer's life slightly easier by not using that.

[1:44] But on the other hand it would be nice to be able to use audio input because it will show up later in this class. So that would be a temporary measure to try to improve things if you're having trouble. So yeah, put some horrible number like 100 or 300 up there. This is in PCs, Macintoshes don't seem to have this disease. Although Macintoshes do want this number to be at least 20 most of the time.

All right, questions about the homework. Three of you came up before class and had specific kinds of things to ask about the homework assignment. So it seems like it's a good moment to see if you have other questions that I could just answer to the group. But there are office hours after class, too. So if you don't want to ask it in front of everyone else just come up and see me after [inaudible 02: [2:10] 31] .

Yeah?

Student: [inaudible 02:34] [2:34]

Instructor: [3:03] OK. So yeah, so why do you have to multiply two oscillators? So this gets back to the example from last time, which rather than dig up, I'll just do it again because that'll slow me down a little bit and make me explain it. [3:16] So what we'll do is we'll make an oscillator. What I'll do is I'll make a nice 440 hertz oscillator which we'll be able to hear. To start with, I'll fix it so that you just hear it without any control at all really.

[tone]

Instructor: [3:49] All right. So this is the way to play an oscillator without losing your speakers. Now what I'm going to do is try to respond to you by controlling the amplitude, which is this. Multiply this thing as a signal. That means with a tilde object by some other oscillator. [4:08] Let's make the amplitude -- all right. I'll just type the numbers in to start with. So here's changing amplitude. All right? And here is changing the amplitude real fast. I think this is what you're asking. It doesn't change the amplitude. It changes the frequency.

[4:38] Now this is psychoacoustics, which means it's not science really.

[laughter]

Instructor: [4:43] What I'm doing here is I'm changing the amplitude of the thing 330 times per second. You could think of that as a time varying amplitude, but it's too fast for you to perceive. [4:52] But now you can use a different fact, which is that the product of two sinusoids is the sum of two other sinusoids at different frequencies. So in a confusing way of describing the situation, you can multiply something by a signal to change either its amplitude or its frequency depending on how you do it.

[5:13] If you multiply it by something that's varying slowly in time... Let's see. Let's shut this thing up. If you multiply by something that's varying slowly in time -- that doesn't have a physical definition by the way, whatever that means -- then you will hear the thing changing in amplitude. If it's changing quickly in time, especially if it's changing repeatedly and quickly in time, then you will hear something else, which is new frequencies being introduced, which is one of the possible things that electronic musicians call modulation, although it's not the only one.

[5:48] Does that answer your question at all?

Student: [5:51] Yeah. Because my original conception of the oscillator object [inaudible 05:55] . Because you know when [inaudible 05:59] kind of change the frequency?

Instructor: [6:03] Right.

Student: [6:03] I was having a hard time understanding in my head the difference between changing frequencies and [inaudible 06:11] .

Instructor: [6:13] Right. OK. So maybe what I should do also while I'm at this is get set up to graph this thing again. Sorry. This is repetition, but I don't think it's bad repetition. I'm going to use the [inaudible 06:25] thing and click on an array. It's going to have a name like... let's see what I'm going to call this [inaudible 06:34] . OK, and I'm going to have a second -- ooh, how long am I going to make this? I'm going to make it a 1000 samples to start with. I like points, so I'm going to say, "OK." Oh, man. And I got to fix it so that it goes from -2 to two because [inaudible 06: [6:38] 58] .

So I fixed the bug last night, and if you download you will have the bug fixed version now, on the test version, if you do, I haven't fixed it here yet. So you're going to be cautious. I made the table now [inaudible 07: [7:01] 11] which means I need a button. Sorry, I know you've seen this all before. And I need to tab right. It needs to know the name of the table it is going to write into.

[7:33] It needs two things. It needs a message to tell it to write and it needs a signal. And nothing happens, why? Because I turned it off. [tone]

Instructor: [7:45] There, oh, look at that, isn't that cool? That's not a sinusoid times another sinusoid in the way you'd look at it. It's a sinusoid plus a sinusoid. But that's because I chose this thing funny. Here's something where you can sort of see it, I think. [tone]

Instructor: [8:11] Yeah. And now, you hear those gaps in the sound? That's because I have a very slow processor. And I have to add to my delay. OK, here we are. [inaudible 08:21] [8:23] OK. So now what we see is that if you can imagine there's a sinusoid being multiplied by this other sinusoid. This one is going at 440 cycles a second, and this one is going 10 cycles a second, and the half cycle's about this long, if I can use the mouse to point to it. What you hear right now is the nether region between beating and hearing two pitches. I hear two pitches, but I also hear beating at the same time. Just, I don't know what, that's getting on the wrong side of the uncertainty principle.

[8:56] But if you want to only hear the two tones, make this number at least 30. [tone changes]

Instructor: [9:02] And then, it looks like a sinusoid with changing amplitude, like your ear resolves the two sinusoids. The sum of that thing is maybe two different pitches that are about a [inaudible 09:12] second apart. [9:16] Yeah?

Student: [inaudible 09:16] [9:17]

Instructor: [9:28] That's what this is doing.

Student: [9:30] Yeah I know, but just something about [inaudible 09:32] .

Instructor: [9:34] OK. So do this.

Student: [inaudible 09:34] [9:32] smaller?

Instructor: [9:37] Yeah. Make it a hundredth. Oh yeah. So it's acoustics again. .1 is -20 dB. .01 is -40 dB.

Student: [9:46] OK. You're right.

Instructor: [9:48] Now the thing about that is, if you're having to pull it down that far, what's going to happen when somebody sends something of magnitude 100 into here? You're still going to get one and you're going to get a nice loud sound [inaudible 10:02] . [10:05] So you also, if you want to protect your ears by adding a clip object here. Maybe from -10 to 10. I'll show you how to do this in more sophisticated ways later, but now what we've got is... Actually, this is when I can turn the volume up in the room. I don't hear a thing because I turned it off anyway.

OK. So now if I put out something with unit amplitude what will come out is -40 dB, and furthermore, even if I make a horrible mistake and the numbers here are in the thousands, which does sometimes happen -- especially when it [inaudible 10: [10:41] 53] , this is going to happen -- then clip it to some reasonable range.

[10:59] And in this case, if we clip it to between -10 and 10, and then if we divide by 100, then we know that no matter how crazy the thing is that we put in, it's not going to be outside of the range, plus or minus a tenth, which might be a safe headphone range for the computer.

[11:18] Rather than do that, typically what I do is I make sure that the volume on my amplifier is down so that no matter how stupid I am making the patch, it can't make more than a certain full blast volume. So my way of protecting my ears is -- which is important for musicians, especially -- make the loudest noise that you care to make.

[11:41] For instance, get out your test tone. Do this.

[loud tone]

Instructor: [inaudible 11:56] [12:00] And get this thing to be as loud as it's ever going to be. [very loud tone]

Instructor: [12:04] And verify that that isn't unduly painful. So I didn't leave it up there for long, but that right now, what you just heard is the loudest thing that's going to come out of these speakers, period, with the audio system the way that it's set up right now. [12:19] And that way, when I do something stupid, like take my white noise generator and stick it right into the jack, it won't make me jump out of my chair. I probably emphasize this more than it's really necessary to emphasize, but computer musicians surprise themselves a lot by turning things up because they don't hear them, turning them up because they don't hear them, and then turning them on and discovering that they've, meanwhile, it's been quite a few too many dB. Maybe some of you have had that experience already.

[12:50] OK? So that's sort of survival technique. Let's see, what else here? Yeah. That's about all I should say about that. Other questions or comments? Yeah?

Student: [inaudible 13:02] [13:04]

Instructor: [13:18] OK. Did you, do you have the 42 working version or the 43 test version?

Student: [inaudible 13:23] [13:24]

Instructor: [13:25] OK. Good, so that's problem number one isn't happening. You have to have DSP on. In other words, check the Pd window. And see that this, ooh, it's off for me right now, but that has to be on.

Student: [inaudible 13:40] [13:41]

Instructor: [13:43] OK. Yeah, it's good. Yeah. All right, then what else could be wrong? If you're actually clicking this, oh, someone said make sure the patch is not in edit mode. Then next thing is obviously if I did something like misspelled this, then it wouldn't happen. But then I would get an error message on the Pd window, I think. [14:15] Oh, yeah, thank you. Oh, that is interesting. [tone]

Instructor: [14:20] Yeah, there we go. So look on the Pd window and see if there are any helpful error messages. Yeah, OK, show me after class, it's got to be something, it's probably something that I'll have to tell you all about Thursday because it's probably something important that I've forgotten to say, but I don't know what it is yet. OK. Other questions? [14:53] It's all good. Yeah?

Student: [inaudible 14:54] [14:54]

Instructor: [14:58] No. As far as I know. Why, [inaudible 14:58] ?

Student: [inaudible 14:59] [15:00]

Instructor: [15:04] What's happening?

Student: [inaudible 15:04] [15:06]

Instructor: [15:08] Yeah, WebCT should just have a pointer to the...

Student: [15:08] OK.

Instructor: [15:13] Yeah. That's the thing. I haven't actually learned WebCT yet. So if you're using the chat thing to try to chat with me I probably didn't see it because I'm not cultivated in that way yet. [laughs] [15:29] OK. Other questions? OK, next things to talk about. So, again, there are two things, one is Pd lore, and one is actual signal processing knowledge, which I'll be developing simultaneously.

[15:53] So a bit of Pd lore now is going to be more about messages and doing computations using messages, which is to say the sporadic control things that are not audio signals that go down the wires that are only one pixel big instead of two pixels big. So this is a control message here, and this is a signal message here, or signal, if you like, that's a continuously flowing piece of audio.

What I want to tell you now is that -- oh, so, first off, one thing that you know from last time is that you can take an oscillator and control it. Whoops, sorry. That's for later. Take an oscillator and control it with a number box like this, and then you've got [inaudible 16: [16:19] 36] that operates like this.

[tone]

Instructor: [16:44] You know what... [inaudible 16:45] maybe that's... is that loud enough for people to hear? Or should I turn it up? [inaudible 16:51] [16:54] All right. So what's happening now is instead of having the oscillator be an oscillator space and then a number to initialize the frequency to something, I'm running a number into the oscillator, which is in the form of messages. The oscillator's input actually can take signals, or messages, either way. But it wants numbers.

[17:14] These are messages, and you can tell that among other things by the fact that it's a one pixel wide as opposed to a two pixel wide one. And that's just the thing about number boxes, they don't know about audio signals, there's just nothing really they can do with them.

[17:29] OK. Next thing is, if you want to do something like, say -- oh, let me do it this way. If you want to fix it so that you can predetermine -- oh, go away. So that you can predetermine some number to put here, you can use another thing, which is called a message box, and I'll just put a couple of values in.

[18:00] And now I have a thing which is a push button. But unlike this push button, out comes a message which has a number in it when you click it. Now, this is different from -- OK, right, so let me do this while we're here. This is a message. This is a number box. Whoops. And this is an object. And the thing that distinguishes them is the borders on it. This is supposed to look like a flag, which is... I mean, using only five pixels, or trying to make a representation of that. This is supposed to look like a punch card, and this is just a box to be the simplest possible shape, because it's used frequently, used thing.

[18:53] What this is about, is the number... OK, now we're not in edit mode. The number does this. You click on it, and you drag, or you click on it and type to give it values. It's a thing which will generate the messages which have numbers in them. This is a thing which lets you type a message in which you just click on and get the message. This is good, because very frequently you want to send something a message, you already know what the message is going to be. You don't want to oblige the user of the patch, such as yourself, to have to type the number in, because you know what the number is going to be.

[19:33] So, now if I want to make a...

[19:39] So now, if I have a collection of pitches that I want to hear, that's, I'll leave this for now. Let's go here. So, things that have frequencies that correspond to pitches of that we all know. How about 261.62, that's middle C, and how about a low A? These are just easier to remember ones. And the higher A. And if you know A, then you know the frequency of E, sort of.

[20:18] I'm avoiding having to actually show you how do the math to do this correctly. That'll happen either Thursday or next Tuesday, I would say. OK, I don't want to, I want this to be zero so not changing anymore, just combine... [tones]

Instructor: [20:34] Now I've got... OK? [laughter]

Instructor: [20:40] Now that would not be a good thing to do with a number box. Because to make those four notes, I'd have to type those numbers in at musical speed, so it wouldn't work. [20:56] So almost the only reasonable thing that you can do if you're going to do something like have a musical scale that you want to use is put the numbers somewhere you can get them. And the simplest way of doing that is put them in a message box. All right. Yeah?

Student: [21:12] Can you go over one more time the difference between putting the number after the oscillator, after the oscillating [inaudible 21:16] versus putting [inaudible 21:18] ?

Instructor: [21:20] OK, good. And in fact, I was inadvertently over confusing here, because I left the number here. This number is sitting there, so that this thing that it was creating was 440, but then I was changing the number by putting these other numbers into it.

Student: [21:38] They're overriding.

Instructor: [21:39] And those are overriding it, yeah. So this is an initializer, and it's a better style. In fact, if you're going to change something, not to initialize it, so that it doesn't look like it's 440. Then for instance, it might be 261 or 162, instead. So that's one thing about that. [21:58] Another thing is that you can put messages or signals into the oscillator. And some of the examples from last Thursday had us putting a signal into the oscillator in order to control the oscillator's frequency. So there again, putting a number into the oscillator's first inlet, which is its inlet for frequency, sets the frequency, be it by a number like this or by an audio signal which is a stream of numbers.

[22:27] Now the next thing to mention is that there are two inlets up here, and I've never told you what the other inlet is good for. It is a thing for initializing phase. And initializing phase does not mean that... OK, let's see.

[22:44] OK, so what is phase? Phase is a number which, if you like, it varies in time and varies from say, zero to two pi as a thing cycles. So if I -- let's get rid of this multiplier now, it's going to just be confusing. Actually, let me save this. Let me save this and give it a name, which will be...

Yuck. This is patch number two, and that's going to be, I don't know. It's going to have to be [inaudible 23: [23:09] 18] , but know I'm going to save as, [inaudible 23:21] three. And now we're going to start talking about phase. All right? OK.

[23:37] By the way, I'm putting these patches up on the website, although I did that a little bit belatedly last time, the patches from Thursday only showed up on Sunday. So if you're looking for them before then, you didn't see them because I hadn't done them yet. Actually, I just forgot. So if you don't see them after class and want them, send me an email and remind me to put them up, because this is the sort of thing that I forget to do.

[23:56] OK, so now, now that I saved it I can get rid of this, because you've got this in the other patch. And in fact, I'm tired of thinking so quiet, so I'm going to do something, I'm going to make it louder. OK? And I'm going to test it, so...

[tones]

Instructor: [24:16] Yeah, there it is. OK, more beautiful music, all right. Now, phase. OK, so phase is a... Oh, right, I want to graph this, I'm sorry. Yeah, can I put this one on this. I'm going to graph it from here. So we're now looking at phase 440. [24:33] So the phase is a number which -- OK, you can regard this thing as starting at any point during the cycle. A mathematically easy place to think of the cycle starting is at the top, because then everything is a cosine, and cosines are mathematically simpler than sines, for a reason I'll tell you later if you want to know.

[24:54] So you can, but since this is arbitrary, but I'm going to assume that the cycle is starting here. And the cycle then proceeds from the top down to the bottom and back up to the top. And there is a number called phase, which you don't see here, which is going from zero to two pi every time the oscillator is oscillating.

So if you like, the old metaphor is, imagine that you're in the dark and this bicycle wheel is spinning and there's a light on it, and you're looking straight down the axle, straight along the [inaudible 25:
[25:15] 27] of the thing, so you see the light going up and down. The phase is the angle of the bicycle wheel, which you don't see. And the thing that you do see is the cosine of the phase, which is the thing that bounces up and down.

[25:41] Later, I'll show you how you can actually generate phases as audio signals and put them into things, in order to control this process in greater detail. But right now I'm not going to do that, it's more detail than we can deal with right now. I'm just going to mention that phase is a thing which you can initialize, but then the oscillator itself maintains the phase with changes in time.

[26:05] And now what I want to do is... maybe, let's... Yeah, right. I probably should have kept the other oscillator. OK, so we'll do this. A duplicate... Oh, yeah.

[26:23] If you haven't found out yet, the fastest way to make an object is to select one, without selecting its text, and then hit 'duplicate', which will in effect copy any amount of patch that you want.

[26:36] So I'm going to now go back, sorry, and make an oscillator with a number controlling its frequency, and then I'm going to play it. So, multiply to control the amplitude, then we'll hear the output. I'll show you why in a second, I hope. OK, so I'm going to set one hertz over 40, play.

[tone]

Instructor: [27:05] All right. Yeah?

Student: [inaudible 27:05] [27:07]

Instructor: [27:11] Command-d, or it's up in edit there. Oh, control-d for me, but command-d for you, all right. [27:20] OK, now next thing is this oscillator, of course, also has a phase which is going from zero to two pi once a second. You can change the phase, but the phase is also always changing. So let's make this number -- let's make it slower, so that you can hear what's going on. I'll make it every four seconds. OK.

[tone]

Instructor: [27:42] Now, I'm going to go into the other inlet. Let's see, get a message box, because I was telling you about that. And I'll make something that just bashes the phase to zero. Come on. All right. Now I've got a nice little attack [inaudible 28: [27:52] 04] . Well, it makes an attack if I hit it at the right moment. But if I hit it at the wrong moment, and the thing was at phase zero anyway it doesn't change anything, so I don't hear anything that it wasn't already doing. Now I'm trying to figure out how I can, oh yeah. OK now I can graph this for you, but I will have to just graph this, I think.

[28:32] And furthermore because it's moving slowly, oh, but I'll still use it here. Because it's moving slowly, I'll make the table be huge, so that you can see things that are happening slowly. So in fact, let's put this back at one. And then, I'm going to make this thing happen at properties that will make it have a whole second's worth of stuff in it. And so 44, 1,000, 100, that's a second's worth of sound.

[29:09] Oh yeah, let's change the name. Actually, I'm just going to... do that, all right. New name, you were right. And furthermore, I forgot I have to make the -- oh that's good, never mind.

[29:31] All right, let's record it.

[tone]

Student: [inaudible 29:33]

Instructor: [29:38] Oh, thank you, yeah. Let's make it talk to the [inaudible 29:37] . Now, if I tell it to record, I could wait a second before I see it. It doesn't show the thing to you until it's finished recording into the array. Which is why you make the arrays kind of short if you want to see things quickly. [29:55] So, now what we're seeing is, every time I whack it, I'm going to see one second of just this amplitude controlling sinusoid.

[30:07] Now, I am going to set the phase of it. You here that stuff? That is this. Let's see. Now I have to click this and then click that within a second. So I'm going to move them really close to each other.

[laughter]

Instructor: [30:19] There. Look at that. This is what happens when you have a nice sinusoid going and you say, "Now make the phase be zero, please." [30:31] It makes the phase zero, all right, which means there is a discontinuity in the sound. Which, in computer music lore, means that you will here a click. So, discontinuities, or step functions, are clicks or one source of clicks.

[30:45] Yeah?

Student: [inaudible 30:45] put arrays in a message box [inaudible 30:47] [30:46] ?

Instructor: [30:51] Sure enough. Yeah. So I could do this. I'll do even a little bit better in a second, but let's do that for now. So now let's listen to that again. [tone]

Instructor: [31:00] Now I'll say, "Set the phase, please." [31:04] It set the phase and started graphing it at exactly the same time. So now, any time I hit it, I will hear something.

I'll hear [inaudible 31: [31:13] 31] sound. Actually, you are hearing two. One is when I make a discontinuity by doing this. The other is a second later when it [inaudible 31:21] the table [inaudible 31:22] .

[31:29] So now I have made a triggered oscilloscope, for those of you who have studied physics.

[31:34] Now, let me do something even better. Introducing the delay object. So this is going to be an object that's called delay. I am going to say, "Delay 400." 400 milliseconds of delay.

[31:55] So now what I am going to do is have my nice button start graphing and then, 400 milliseconds later, it is going to set the phase of the oscillator to zero.

[32:16] Now, no matter what I do, I hear this continuity four-tenths of a second after I whack the button. I notice this part of the table is changing, but at a fixed 400 milliseconds into the table, it sets the phase to zero and thereafter -- whoa!

[32:38] Every once in a while, something like that will happen. Thereafter, the thing is always the same. That's to say every time I whack the button I will have a nice consistent result starting 400 milliseconds in.

[32:54] All right, so you cannot put your guitar into this delay. There's another one for that, which I'll show you later. This is a delay for messages. What that means is that, I'll show you.

Student: [inaudible 33:06] [33:06] ?

Instructor: [33:10] Yes. Thank you. [33:14] So delay, what it does is when you send it a trigger, it sends you a trigger, the amount of delay that it... So if you've been following me, what I'll do now is I'll say, "This delay, let's make it a longer delay, like two seconds." OK, now zero, one, two. Whoops, I didn't count right. All right, OK. And now if I graph that...

[tones]

Instructor: [33:52] One, two. Why was that stupid? It didn't hit until after the thing had finished writing into the array. The array is one second long, the phase is getting set after two seconds. It's hopeless, I'm not going to see anything. You can hear it though. OK, good. [34:15] So, are people following this? This is important, because this is how you would set about making sequences, things that have an order in time. For instance, this is going to get crowded. OK, let me save this, I'm going to do a save as again. So now we'll make a dumb sequencer. We'll make a smart sequencer later.

[34:49] The dumb sequencer is going to look like this. Get rid of this, don't need that. So I have this nice A Minor chord here, so I'm going to make the thing arpegiate, all right? It's going to be easy, right? All we're going to do is we're going to have a nice button... Let's see, I don't want this anymore. I'm going to have the button bash us to low A, and then, I don't know, 150 milliseconds later...

[35:26] How do I know what number to use? I've done a lot of this. And then another 150 milliseconds later, and then another 150 milliseconds later, and then let's just go back down. Yeah. OK, this is, let's see. Take the output of that, the input of that, then just go... Let's see, this might work. All right, so now we'll play it.

[musical tones]

Instructor: [36:06] All right, now we've got Beethoven. [laughter]

Instructor: [36:09] Not really, OK. And of course, the punch line. Let's just take this one and connect it over here. Now we've got... [musical tones]

Instructor: [36:21] Whoops, I did something wrong. Oh yeah, I know what I did wrong. I need another delay before I loop it, don't I? [tone]

Instructor: [36:33] So another 150 milliseconds later, I'll go back around. [musical tones]

[laughter] [36:38]

Instructor: [36:45] OK. Do this at home and not here, all right? [laughter]

Instructor: [36:49] In fact, I will try not to play that anymore now. OK, let's... [36:53] Oh, there are little things that maybe you could want to be able to stop this. There's only one way I can stop this right now with what you know, which is to break one of the connections and wait for it to run out. There are ways, but we'll get there when you get there.

[37:12] But now what's happening is the following thing. There are two parts of the patch. There's the part of the patch that's giving control, there's a part of the patch that's doing signal processing. The part that's doing control...

[37:25] And by the way, that's jargon. Control and signal processing aren't really two different things that you can do, they're all part of one thing. But in Pd Land, you think of it as two things as being two different things simply because they're two different computer science-ish constructs that represent them. OK, so let's see. I don't need this.

[37:46] OK, so what's happening is the control stuff is all this. The signal stuff is all this and maybe this, I don't know how to characterize the array. And the control computations are happening at a specific instance in time. In fact, it's happening about seven times a second, because this is about a seventh of a second. And what is happening here is...

[38:10] Oh, let me show you what's happening here. Let's get rid of one of these buttons. Hit speed and clear, put that button here. Yeah, hit next, let's see. Let's not do that, let's do this... That little mistake, oh, it's OK. I can do this. I need a start button, but then I could always just have buttons that show me what's happening. Yeah, this is going to be painful. Let's not do the whole thing, but I'll just do these two. All right, OK.

[38:57] So now, there are messages flying around in a very particular choreographed way. So each one of those delays is the source of a message, if you like, and what that message does... So the message's formal name is Bang. Oh, I can print it out for you. Let's make an object, we just call print, and you'll see what this message looks like, and just we print it into there. All right.

[39:31] So bang is just a word that means I don't have any numbers for you, but do it anyway. In other words, well, you wouldn't just save a space or something like that to say a trigger. You have to have something to print there. So bang is just a verb that says whatever it is you normally do it's time to do it right now. It's a trigger, if you like. And this message bang is coming out of the delay and it's doing three things. It's hard to see because of the messy crossing lines.

But one thing is it's causing this bang object to flash, this push button on it. Another thing that it's doing is it's sending a bang to this message box 261.62 and what is that doing in exchange? It is putting out -- oh my God. Oh yes, right. So there are two messages coming into here, from this delay and from that other delay further over. Anytime either one of those things goes off, it's getting printed here, and we're getting [inaudible 40: [40:04] 38] .

[40:42] Yeah?

Student: [inaudible 40:41] [40:43]

Instructor: [40:46] It'll complain. It won't even connect it because print expects control messages. However, I could put a print tilde after that, and then I would be looking at the audio. You know 64 samples worth of the audio signal. Am I going too slow? No, all right, OK, good. So to go back, if you like the outlet of this delay, this -- whatever you hook up here talks to a tree. That's to say a graph without any loops of stuff. And the stuff is what happens when that thing goes off. And the tree stops whenever you either change from being a control message to a signal. Or it stops whenever you put it in to something that doesn't do anything as a result. I'll show you lots of ways [inaudible 41: [41:05] 42] the result of things that's going to happen.

[41:47] So if you like, the tree of things that depend from this bang outlet consists of this delay, which by the way doesn't do anything so the tree doesn't go further through that. So what this tree is is everything that happens right when that bang happens. And not stuff that happens later as an indirect result of it.

[42:10] So this delay, when it receives a bang, its job when it receives a bang is to schedule itself for 150 milliseconds into the future. So it does that scheduling job which is a side effect in computer science language. And meanwhile, it returns, which is to say that's the end of that arm of that graph for that sub-graph, or sub-tree, I should call it.

[42:37] There are three things. First off there's the bang, and then there's the delay. Then there is this message box which puts out a number and that number goes down to the oscillator. And that's the entire chain of events that takes place when this delay goes off. So I should -- OK. Let's see... shift. Just to select everything that is in the tree, hanging from that delay object.

Make that be a tree and not have loops in it. Because if you put a loop in there, then Pd will try in a zero amount of time to do an infinite amount of stuff, which is traverse the loop as if it were a tree. And Pd will then think hard, and then after a while, depending on the speed of your machine, it'll complain and say stack overflow [inaudible 43: [43:11] 33] . So for instance if you want to see me do that, I'll get a nice number and hook it up to another number and do it back like that.

This is illegal. So Pd can't really sense this so it didn't stop me from editing it because the editor doesn't know that something bad is going to happen, but something bad is going to happen. When I put a number in here... Oh, yeah, I get stack overflows. It's actually pretty good at detecting [inaudible 44: [43:52] 08] . Sometimes it's better than other times depending on the OS and in particular the things you're dong. So your mileage will vary. You can bring Pd to its knees this way.

Why was this not a good thing to do? The first message box, sorry the first number box, the one on the left, if you like, tells the other one be 96, and by the way, output 96 and then return. After that you're done, right? For computer scientists it's a depth-first tree traversal [inaudible 44: [44:19] 42] . So the message box on the right then says "Oh, I just got a message 96, what I'm going to do now is tell the... sorry, I'm going to tell the number box on the left to change itself to 96." It already was but no matter, change it, repeat it anyway.

[44:57] And then that one says, "OK, before I'm done I want you to change to 96." And say, "OK, that's good, before I'm done I want you to change to 96." And so on and so forth. It's an infinite loop.

It's actually a recursive loop, but it's theoretically incorrect, and Pd couldn't deal with it because eventually it ran out of memory. In other words, recursion involves pushing [inaudible 45: [45:05] 15] . OK.

[45:21] You can also not make loops out of signals; signals hate loops too, but they hate them in a different way. So when I do this... let's take a nice signal and multiply it by five.

[45:41] OK. There are a lot of reasons you should not try to do something like this. And after that's done I'll multiply by five, and after that's done I'll multiply by five again and so on. And now -- why am I getting an error? Oh, because I took it off.

[musical tones]

Instructor: [45:56] It says DSP loop detected. Some tilde -- and then something I can't read, you can do it and find out. [46:02] This is not the same kind of loop as that because notice we have fat connections here, and thin connections there. The error is found at different times. Here Pd was smart enough to figure out that there was an error because it's much more able to analyze what will happen as a result of the signal analysis, or signal flow than messages. Why? Because if messages contain decision making objects, and to analyze a message computation, you would have to analyze a Turing machine, which is formally impossible.

[46:35] Signals are analyzed as a graph. You simply know that a signal crunches a number every sample. So there's no decision making to be done which can be second guessed. And so you can predict in advance how it's going to happen. This is important because it allows Pd to operate on the signals very efficiently. What it really does is it -- it doesn't exactly pre-compile it, but it essentially figures out in advance what it's going to have to do for the signal network, and it sets it up to be very optimal.

[47:09] So it analyzes this. It knows how this thing is going to act. And so it knows how to do it. How to linearize it, how to make it happen in order. And, gee whiz, it discovers that there's no possible way to do it because you have to have finished each one them before you can start the other. And so neither of them can ever be run. So that's an error, and that can't be done. What will happen to you when you make that error is those two things just won't be run, and the rest of your patch will run.

But if you see that it means that you're doing something wrong, and you should probably find out what it is. By the way, you can always say, "find last error," and if Pd is able to figure out what object [inaudible 47: [47:37] 48] the last error it will go to that window and turn that object [inaudible 47:56] , select that object for you. That's a useful tool.

So that's a loop, and this is a loop and they are [inaudible 48: [48:05] 12] . Now this is not a loop in that sense, even though it looks like one. [inaudible 48:20] like one very well. But maybe I could do this. Well, I don't know how to make this patch [inaudible 48:26] , you can see it.

[48:29] But what's happening now is each delay's output is going to the next one's input and so on, ad infinitum, until the last one's going back into the first one. That's not a loop because the delay, when it receives a message, does not as a result put a message out. Instead it schedules a message for the future.

So in a sense it's the end of the line. It doesn't have any direct effect to put it [inaudible 48: [48:44] 48] to the delay. Questions about this?

I promised you I wasn't going to play it anymore, but here's how you make the thing have a controllable tempo. You would just put these numbers into the delay. I'm not going to do it because I've already had enough of that.

Students: [49:04] Aw. [laughter]

Instructor: [49:21] If you want to hear more of that you have to make it yourself, but then use headphones. [laughter]

Instructor: [49:28] All right, questions about this? That is a lot of information, some of which was a little abstract. And I'll try to find ways of stubbing our toes on this again later. Yeah?

Student: [inaudible 49:39] Can you put a number? Is that [inaudible 49:43] [49:40] ?

Instructor: [49:49] Ooh, thank you. Yes, OK. So I should have told you that right when I was telling you about setting phases of oscillators. The only thing I ever set the phase to is zero. Let's see, let me save this, and did I...? Yeah, OK, I can go back to this example. So I'm going to go back a little bit. Oh, what did I just do. I'm going to save the changes and cancel whatever that was. So let's save. I'm getting nervous now because I just made a window, no thank you. Oh, this is saved, zero objects, right. Good. OK. So the dumb sequencer, we're now going to close and [inaudible 50: [50:13] 47] for a while.

[50:50] And now I'm going to go back here and start setting phases of the oscillator. We'll go back to only waiting four-tenths of a second so that -- ooh, interesting. Oh, right.

[tone]

Instructor: [51:05] Let's see. This we need to be a one, I think. Yeah, there it is. OK. [51:14] So this number is not in radians, but in radians over two pi. It's a relative, it's in cycles. So if I want to have this thing look like a sinusoid, I'd want to go to three-quarters of a cycle in, .75. Now what I've done, OK, let me see if I can explain this. Now what I've done is I've set the phase of this oscillator to three-quarters, which means another quarter of a cycle in we will be at one.

[51:56] And, let's see, equivalently I could have set the phase to minus one-quarter of this. This is exactly the same thing.

[tone]

Instructor: [52:15] So minus a quarter is also a quarter of a cycle before the beginning of the cycle. And this is how I would set the thing to the phase of the sinusoid. If I say positive a quarter here, it sets us up to one quarter past the peak, which to say, it's at zero now, but it's at zero on the way down. And if I set it at a half, then it will be at the negative peak. [52:48] So that's how phase goes. Now if you want to make two things be a sine and a cosine, so you can make something move in a circle parametrically, set one of the phases to zero and the other phase to -- that's the cosine, and the minus a quarter, or plus three-quarters for the cosine.

And then you'll have two things that are out of phase. It's 90 degrees out of phase, if you like, or pi over two radians or one-quarter cycle, which is the way we [inaudible 53: [53:04] 12] .

Student: [inaudible 53:13] [53:15]

Instructor: [53:18] Oh, all right. How would you get two of them to be out of phase? Why don't you do this -- why don't you set one of them to... OK, you're going to have to have two oscillators, one of them... So you can set their phases to whatever you want to. So what's out of phase? If you want one of them to be at [inaudible 53:35] while the other one goes through zero, then give them two numbers that differ by a quarter cycle. [53:53] Or, if you want one of them to peak up while the other peaks down, then set them a half cycle apart. Yeah, and this you'll want to do using message boxes. Message boxes are the right tool for doing this, because that's where you can put a number in that will happen without someone having to type the number in while they're using the patch.

Student: [inaudible 54:05] [54:11]

Instructor: [54:21] No, we're not going to have a good grading policy the first time around, because you're just going to be sighting the thing in. So basically, for the first homework, if the thing works it's full credit, regardless of whether you were elegant or inelegant. And then, we will discuss after that, or the TA and I will talk, Joe and I will talk, and we'll try to figure out what to do after that. [54:47] We'll either decide to make simple, elegant patches be worth more than ugly, horrible patches, or we'll decide that ugly horrible patches are worth more than elegant patches...

[laughter]

Instructor: [54:58] ...depending on what seems to be pedagogically the most appropriate choice. But for right now, just get it to work, and that's good.

Student: [55:06] On the [inaudible 55:07]

Instructor: [55:25] Sounds distorted? Oh, so you're probably not doing it right, yet. [laughter]

Instructor: [55:33] OK, so good. So there are two things there. One is: what's the difference between those two sounds and can you hear it, which you've already got that, which is it sounds distorted, and it also sounding at the wrong pitch. [55:47] Why is it sounding distorted is for reasons that I explained Thursday, you're overloading something. I don't know what, because I don't see the patch. But either you are putting two sinusoids at full blast into the back, its range is from minus one to one. And if you are adding two full-blast sinusoids into it, you are going outside of the range of minus one to one, and there will be clipping.

Student: [inaudible 56:10] [56:11]

Instructor: [56:14] If you put a clip on, then it'll just clip, and you'll still hear the distortion. You have to actually put the amplitudes down to where you can play it without distorting. [56:22] The other thing is, even if your patch is formally correct, depending on your audio hardware, it might distort. But, your TA knows this, and so his audio will not be distorted. So if it sounds distorted on your computer, it's just conceivable that it will sound clean on his. But if it sounds distorted on his computer, it already sounded distorted on your computer. There's a syllogism in there somewhere, but I can't spit it out right now.

[56:56] All right. Other questions?

Student: [inaudible 57:01] [57:01] go about.

Instructor: [57:10] How would you offset. Oh, oh right. So, how do you find out what... OK. I think what you're asking is how do you find out what the phase of an oscillator is while it's running. So that you could change it to something else, and [inaudible 57:19] . [57:22] You need another object, but I have two other objects that I haven't told you about to be able to do that. They are phasor tilde, because you actually have to deal with the phase of the number. And snapshot tilde, which is the look at a signal to see where it's at object.

And those will come up probably next week. Maybe phasor shows up tomorrow, but -- sorry on Thursday -- but snapshot not until next week. So you can set things right now, but you can't, like, query your patches to how the signal processing is doing for the [inaudible 57: [57:36] 50] .

Student: [inaudible 57:52] [57:53]

Instructor: [57:57] WebCT believes that it's due at 3:30. So that means that if it doesn't show up at 3:30, WebCT will flag it late. WebCT won't even take it if it's more than a week late. And, we haven't yet set the schedule of when we will grade it, but you should get it done before grading starts, because it's much, much easier to grade it all in one batch.

Student: [inaudible 58:22] [58:26]

Instructor: [58:28] My guess is that Joe won't be able to download it until five, because he's going to be sitting in class. But on the other hand, if he gets bored he might actually decide to start downloading it while I'm talking. [laughter]

Instructor: [58:46] I don't think he will. So, it seems like you've probably got until five.

Student: [inaudible 58:49] [58:51]

Instructor: [58:54] Right. [laughter]

Instructor: [58:58] But come to office hours after class and I can help you. Maybe you can be done before 3:30. Yeah. Other questions? [59:07] Oh, right, we haven't decided how late late is, or we don't have a late policy yet. All this we have to figure out how the class works before we can set things. Other questions?

All right. I want to quickly show you another object just because... well, you'll see why. So this is the delay object. There is also another one which is called metro which is the metronome. And this is an object which takes two numbers in, let me get a number box. [inaudible 59: [59:21] 55] Oh you know what, let's do save as, as this is going to be confusing now. So now we're going to say five metronome. And I don't know whether to leave all this stuff, so right now I'll just leave it. OK.

This is an object which when you turn it on, does this. And when you turn it off it, does that. All right, that's useful, and furthermore you can control -- sorry, yeah. I'll leave this [inaudible 01: [60:16] 00:41] for now. But you can control the number of times per second it happens. So 1,000 means its every millisecond. And then if I want to double the tempo I should halve this number and make it 500 which means every 500 milliseconds which is twice a second.

[61:04] Yeah?

Student: [inaudible 01:01:03] [61:05]

Instructor: [61:06] This initializes the amount that you are -- oh right, I should probably sort of say something about initialization. [61:16] So anytime you give an object an argument like delay 400, or metro 400 or plus 400, or even OSC tilde 400, the argument, -- the 400 or whatever number it is -- is an initializer for the parameter that the object uses if there's one parameter. If there's more than one parameter, you might want to give it more than one number, but you haven't seen that happen yet. I think I've been trying to do things in an order that you'll ask me... to start getting it.

[61:46] So what this is saying is when this object is created it's going to be 400. But when I send this number in it will change it to 500. Now let me confuse you a little bit. In order to try to unconfuse you.

[62:01] It's not universally true that the number that you initialize this to is changed by this inlet. In the case of metro, this is a thing which turns it on and off. And any number that's not zero means on and any number that is zero means off. This is the new value of 400 up here which is now 500.

So right now it's not a metro of 400 [inaudible 01: [62:27] 02:28] its a metro of 500.

Student: [inaudible 01:02:30] [62:31]

Instructor: [62:37] OK, yeah. So this thing here, this is a control, which -- control, what's the word? It's a control in the GUI sense of the word. It's a thing which shows its state and allows you to mouse its state to change it. [62:53] These things, messages and objects are things that you type a text in and it defines what they are forever. So the number, you actually don't type this number in when you create it. In fact, in edit mode you can't even edit that number. It's a thing whose job is to change numbers at runtime. These other things we taught you. We put these things in at edit time, and they are what they are while the patch is running.

[63:24] And so this is, yeah right, so this is when your playing the patch and making the sound change and trying to make people dance. You have the patched locked at that point. You're not developing your patch anymore. So you're using the number box and the button and other things like that to be running the patch, basically, changing the state of the patch. But the functionality of the patch, the computer program that the patch is, is determined by the topology. That's to say, how things are connected together. It's also determined by what particular things you type into the message and object boxes.

[64:08] OK, now to slightly further confuse the situation another example of a thing being initialized that came up earlier is that you can initialize an oscillator to a frequency -- like that. But now to change that frequency you would put messages into the frequency inlet, which is the first inlet.

[64:36] The oscillator as a tilde object, it's a single processing object. It doesn't take messages to start and stop. Or to do it's thing the way the metronome does. It takes, all it does is take messages to modify what it does, which is to say its frequency. It's always running. It doesn't need to be turned on. That's because its a tilde object.

So in general, tilde objects, the first inlet, the leftmost inlet, is usually the thing that you can control by -- sorry, here -- usually the thing that you can control by changing the argument. Whereas in objects like metro, this number is being controlled by this inlet because this inlet is used to turn everything on and off. There are no very good generalizations [inaudible 01: [64:57] 05:23] designed to be as coherent as they can be. But there is a limit to how coherent they could be, because they all have rabidly different functions [inaudible 01:05:32] . So that's something you just have to remember.

Student: [inaudible 01:05:36] [65:37]

Instructor: [65:40] Yes, I did that just to show you what was happening. Yeah, so this would be doing the same thing if I had this connected to it or not. Objects don't know what they're connected to. They just put their output out there, and if there's nobody connected to it then the output doesn't get used. And if there are 50 things connected to it the output does a lot of work. [66:02] All right, well. Yeah?

Student: [inaudible 01:06:05] [66:06]

Instructor: [66:09] Yeah, so a couple of, so let's... All right, I can think of nice and ugly ways of making that happen. Like for instance, what if I actually used this patch. Let's turn it on and listen to it. [tone]

Instructor: [66:28] OK. So there's this oscillator, it's at three hertz. Now I'm going to just set the thing to something every 400 milliseconds, sorry, 500 milliseconds. OK. Now just to be... [tone changes]

Instructor: [inaudible 01:06:45] [66:47] . Now I can make a motorboat sound. What I'm doing is I'm changing the phase of the oscillator. Offsetting the phase of the oscillator which is causing a discontinuity in the sound. OK. Let's see, let's shut this thing up. [67:13] All right. Yeah?

Student: [67:16] On my computer mine [inaudible 01:07:16] .

Instructor: [67:21] Oh, yes, that came up before, but I said it kind of fast. If you hit the shift key while you're scrolling it, you're scrolling in hundredths. And if you don't then your scrolling in ones. [67:38] Any questions? All right. So now, what would happen if you took this metronome and 400 -- well whatever it is. Every 100 milliseconds now I'm just going to set the oscillator to three hertz.

[tone]

Instructor: [67:58] Well the oscillator was already at three hertz. So it doesn't do anything to set the oscillator to three hertz. So it stays at three hertz. So that has no effect. [68:09] On the other hand, don't try this at home.

[tone changes]

Instructor: [68:20] You can have two of them and make them fight. So let's make this one go at some nice other speed like 161.8, and you all know that number, right? And now we'll put that up to... no E, no A, no E, right? That's compositional algorithms 101. [68:57] Yeah, you can think about why that did what it did.

[laughter]

Instructor: [69:02] OK, but I should tell you what it did, which is this. These two things are putting out bangs. They're putting them out at different rates. [69:12] One of them is happening 10 times a second. One of them is happening 10 times one plus the square root of 5/2 times per second, the golden ratio. I just chose that because it always sounds good when the metronomes have the golden ratio. OK.

Now, 10 times a second we're bashing the frequency to 220. And some other number of times a second we're bashing it to 230. In fact, if we want to see what that's doing, we could just say, [inaudible 01: [69:25] 09:36] actually show me the frequency of this oscillator.

[69:43] And then we have this schizoid frequency here.

Student: [inaudible 01:09:44] [69:46]

Instructor: [69:48] Oh, yeah. [inaudible 01:09:47] All right, yeah, there's ASCII art to be gotten here, isn't it? [laughter]

Instructor: [69:55] Right, OK. This is now using, yeah, this is now having two different inputs to the oscillator, which are telling it to do different things. And they don't get added or anything like that. They would get added if they were signals, but since they are messages which happen at different times, it doesn't make sense to add them. [70:16] And so instead, it just becomes a situation where whoever sets it last wins. So if someone is opening a door and someone else is closing a door, is the door open or closed? Well, it depends on who got there most recently.

[70:31] So the last person to set it wins. And meanwhile what you hear is the thing changing between the two values in whatever tempo it makes to get the two things to happen at different times. Is that clear?

[70:48] OK, all right. So this shows, in some way, the essential difference between the sporadic control message computations and signal computations. If you want to do something that has to do with decision making or has to do with events that happen in time, like waiting for network packets, or waiting a certain amount of time, or waiting until a keyboard key goes down or something like that, you are in message land.

[71:23] And message rules are that when something comes out of something, like this metronome generates events and the event traverses the tree of everything that is connected to it until it gets to something that doesn't respond, but simply changes its state or does something by side effect. At which point we have done that tree of messages that depends on this metronome setting.

[71:53] All right. Is that clear? And a good part of computer music is thinking of cool networks for controlling these signal processing networks. One thing about that is it takes brains of an entirely different sort to know how to make good control structures from knowing how to make good signal processing structures.

[72:16] Signal processing structures are -- it's very mathematical. You need to be able to deal with trig and stuff like that. You need to be able to think about spectra of sounds. For doing control, actually knowledge doesn't seem to help you very much. No one really has a good way of theorizing about how people should control computers and computer music applications.

[72:39] And as a result, you just sort of learn a collection of techniques which might involve how to respond to external events. How to make decisions. How to generate random numbers. How to solve problems involving constraints. And so on like that. I don't even know how to make the list.

[72:58] But what that looks like is like the field of combinatorics. Just a whole bunch of different things that are different ideas that you have to know a lot of in order to be effective at it. And you just have to wait until you've seen a bunch of things or invented a bunch of things, and then you have a nice repertoire of stuff you can put together and make a meaningful access.

[73:19] So both of those things are things that take a tremendous amount of work to acquire well. The signal processing you can do in a more systematic kind of a way I think than the control aspect of it. And furthermore, it's a little bit artificial to separate them at all. OK.

[73:38] All right, so review of what happened today. A lot happened today. OK. So the new stuff you saw was these messages. And then I got to show you how you could like make things, like sets of numbers that you could then call up at different times. There were, I think, there should have been three new objects, but I can't remember what they were. They were delay and metronome. And I think that's all I showed you.

[74:12] Oh, I forgot to show you one, and I don't have time now. I'll tell you what it is. I also meant to show you this wonderful object here, line tilde, which is your all purpose ramp generator. So I'll start showing you that in detail next time.

[74:32] Actually, since we have five minutes, I'll show you what it does and whet your appetite, and then you can get help on it. Then I'll show you in detail how to use it next time.

[74:40] This is the better amplitude control object. So we've been using oscillators to control amplitude -- I'll come clean now -- just because I didn't want to introduce another object right away. But it's not really the thing that you do all the time, use an oscillator to control the amplitude of another oscillator. More often what you want to do is this -- let's see, let's turn these things off because otherwise it's going to be nuts.

[75:10] OK. Thanks. Now I have a nice frequency. Now what I am going to do is a -- don't need this, don't need that, don't need this, and line. And line is going to now have a message going into it. And the message is going to have two numbers. I haven't told you about this yet, a value, and a time to obtain the value at. So here's on and here's off in a second.

Now I have the [inaudible 01: [75:43] 15:43] .

[tone]

Instructor: [75:49] There's my 220 hertz. And there's my 220 hertz shutting off. All right. There are several new concepts here. One thing is message boxes can have more than one number. And so messages don't necessarily consist [inaudible 01: [75:55] 16:09] of numbers. They are actually quite free form although a lot of the time messages are just numbers.

[76:17] Line tilde will interpret a message with two numbers to mean, "This is the value we're going to attain. This is the amount of time we're going to attain it in." So that this is a faster one.

[tone]

Instructor: [76:36] As opposed to this. So this is the way we make sounds that turn on and off when we want them to, as opposed to just when the oscillator [inaudible 01: [76:39] 16:46] appropriately.

[76:49] Yeah?

Student: [inaudible 1:16:49] [76:50]

Instructor: [76:52] Oh, it's 1000 milliseconds and that's the amount of time that it took to rise. And what I intended to do and got lost in details instead was graph this for you, the same way as I was doing before. Let's see, where going to [inaudible 01: [77:16] 17:08] . Sorry, this is going to be over in just a second. And we're going to graph the line total.

[tone]

Instructor: [77:27] Ta-da. There's what a line does. It sits there doing nothing until you tell it to do something. And 400 milliseconds later I told it to go up to one and 100 milliseconds. So now what's happening is the line is sitting at zero. [inaudible 01: [77:49] 17:54] .

[tone]

Instructor: [77:58] Oh that's nice. Turn it off. [tone]

Instructor: [78:02] So I asked it now to go up to one and 100 milliseconds, and what it did was, it starts graphing. 400 milliseconds later it says, "Line turn on." It turns on. It takes it a tenth of a second to reach its target value of one. And furthermore, I can now tell it to turn off. [tone fades out]

Instructor: [78:25] And then it waits 400 milliseconds and then goes out. This is your way of getting stuff to start and stop. It's kind of one of those [inaudible 01:18:31] things that musicians need to be able to do with their sounds. [inaudible 01:18:37]

[78:39] And more about this next time, maybe. It's time to stop for now. Office hours, I will hang around as long as people have questions.

MUS171 01 13

Lecturer: [00:00] Back to where we were. The new objects for today are going to be... my memory's long here...we're going to find out. Six of them since you only got four last time. What I want to do is I have two things in the program today which are not absolutely to develop new theory but to continue showing you manipulative level things that we might want to do. What they are are, first off, oscillators are the same thing as, let's see, what's an oscillator look like here. Do not be confused by the fact that OSC and COS are anagrams. Oscillator is oscillator and COS tilde is a thing which just takes the cosine of what we're looking at. Actually, you give the input in cycles so it's the cosine of two pi times the thing. And phasor is the other thing that you need if you want to make an oscillator and someone has to use the cosine function. I'll try to explain that in comprehensible terms later and when you see that then you'll actually understand what an oscillator is and does.

Then, various conversion things, frequency back and forth to midi and RMS back and forth to decibels. These are the things that you really use at least at the first cut in order to be able to control pitches and amplitudes in human readable ways. So up until today I've been giving you amplitudes that look like 0.1 and stuff like that which is a perfectly usable way to operate, but most people would prefer to use some kind of reasonable amplitude units.

So stage two in learning how to do that is knowing the usual cyclical acoustics measure of amplitudes and frequencies which are going to be... or frequencies, maybe, which is not really a unit but which seems to be the easiest way to describe what that is and decibels which you learned about in physics. Alright, so that's units...

So, with that in mind, I'm acutely aware that I used about five minutes of the very end of the last class to suddenly introduce the line tilde object so I want to go back over that review-ishly and try to make sure everyone uses it and wants to use it. There will be more about how to use this thing effectively, that's to say programmaticly, that I won't be able to tell you today because I won't have all the GUI objects to be able to do that. But I can at least keep at it at the sort of level of preplanned, here are the breakpoints, and here's what I want to do on this level. Alright?

So the usual patch that we've been operating on, or make it the usual kind of patch, is take a frequency, multiply it by something to control the amplitude which until now has been an oscillator because we didn't have line tilde but now that we have line tilde pretty much for the rest of time were going to be using this or objects derived from it for controlling amplitudes instead of oscillators which are not usually amplitude controllers really. Alright so line tilde does something which at the end of last class I was hurriedly trying to show that I will just develop that again. Probably somewhat differently in order to emphasize it. So what we have here is a nice table with 44,100 elements in it. So that it holds a second of sound sample at the rate that we operate at.

And that way we can do things like first off lets look at the oscillator and we'll see what 440 hertz looks like. If you graph a second of it. Wait a second before you see it and then you see if I were honest I would tell you that there aren't actually 440 cycles being graphed here. What's really happening is there aren't 440 pixels in that part of the screen so it's graphing some incorrectly unsampled form of that wave form. But nonetheless you just get a sense that it just fills this table that is the rampage between negative one and positive one.

Which is indeed what the output of an oscillator looks like. If you want to see it well you have to give this thing some much lower value. And then you'll see a reasonable number of cycles but on the other hand when I play it to you wont hear anything. Because 10 hertz is below the audible frequency range. Alright, so we go back to this. And now what I want to do is graph the, what line tilde does. So lets turn it on and lets graph it.

Well that was kind of stupid. The graph actually falls right on top of the rectangle that holds it. So if you want to actually see it maybe I should look at this, I don't know how to do this in any good logical way. I could make the table go from minus two to two which I've done a couple of times. There now your looking at, yes?

Student: [unintelligible] [05:44]

Lecturer: [05:46] OK, yes I got to get to that. So the 300 is the amount of time in milliseconds that it takes to obtain the value that I gave it as a target. So here the target is zero and the time is 300.

Student: [unintelligible] [05:58]

Lecturer: [06:00] That's right or from wherever it was to zero because if it is at zero, I asked it to do that which means it has to go from zero to zero which means it just flattens. I've been forced to make it painfully obvious we'll put a delay on the graphing, I'm sorry on the message like this. OK. I didn't have to tell it how much of a delay we'll have. So we'll play another 400 milliseconds. I'm going to turn it off and then, there it is. Alright, so this is what line tilde does. It starts where ever it was and when you send to the message, so the message arrives, 0.95 300 arrived, at this point in time. Because I started graphing 400 milliseconds earlier, then I sent that message and line tilde's way of responding to that message is to ramp up to it's target values, which is almost a one, and do that in the next 300 milliseconds.

It doesn't look like it, but this should be 4/10ths of it, and this should be 3/10ths of it. Something bothers me here. [unintelligible] . Maybe it is.

Yeah. I'm looking at it from real close, too.

Who knows.

Alright! Or conversely, sorry to insult your intelligence, but, he'll do the same thing to get back down. [unintelligible] But we could do a different [unintelligible] leaf.

But now, of course, if I do this nothing happens because it's not DSP.

Oh, this is going to be very confusing. The delay sent the message anyway, and then this thing happened, and it all happened while DSP wasn't running. It effect the same moments, but that wasn't what I wanted to show you, I wanted to show you this, going up in here, here's what it looks like do go down, and here's what it looks like if you go up and down. And pretty soon you will be building synthesizers. Just..forget this page. And this is a smaller delay. [unintelligible] . Lets do this for real. What we're going to do is turn it on after 100 milliseconds and then turn it off after another 400. Wow, there it is. Alright, now someone who doesn't understand this, ask a question. Or do you just need to stare at it, that's a possibility too.

Or is this just clear? Probably not. What's not clear about it, first off is that you can't actually see from a patch what objects are doing anything and what aren't. That's a problem that no one will ever be able to solve. But what's happened was, you could pretend these things aren't here because they are not happening right now. Let's actually cut this. Actually, let's just pull the whole thing up. So now what we have is... I'm going to save this. This is a good moment.

[tone]

Lecturer: So what happened is I hit the button, 100 milliseconds later a message comes out of this delay, a bang message comes out of this delay and does this, which means over the next 300 milliseconds we're going up. Then, how long does it stay at 0.95?

Student: [10:23] 100.

Lecturer: [10:24] The remaining 100 milliseconds between this 300 and this 400 because, another 400 milliseconds later, this message clips. Yes.

Student: [10:34] Bang?

Lecturer: [10:35] Yes. Well. Delay sends a bang, which causes this message box to send a message zero 300 to the same line tilde, therefore it goes down. And the whole thing fits within seconds, so you got to see it all. Yeah.

Student: [10:48] It doesn't make sense. Could you just go over exactly line tilde again, what the function of it does? [laughter]

Lecturer: [10:58] Yeah. OK. So...

Student: [11:07] Use it to replace, something you've done before.

Lecturer: [11:10] Oh. OK. There were examples earlier where we did things like this. Give me a second. [sneeze]

Lecturer: And do this. OS oscillator a couple of hertz. Multiply by that. [tone]

Lecturer: Right? And that's an amplitude control. This... Whoops! Sorry. This is another amplitude control. But it is an amplitude control that let's you... [tone]

Lecturer: ...tell it whatever you want to do really, as opposed to just sitting there and just doing something by itself. So, answer number one is oscillator tilde could be used as an audio generator or as a level control by using it to multiply by another oscillator or something else. Line tilde also could be used as a straight signal and you would hear a thump. Or you can use it as a level control by multiplying it by something that you want to hear. Line tilde makes this kind of waveform, in contrast to OSC tilde, which makes this kind of wave form. OK.

[pause]

Lecturer: Now I need another button to press just for graphing. So there is what the oscillator does. So oscillators make sinusoids. Line tilde makes line segments.

Student: [12:37] Say something about you can take the line tilde and a signal and make it sound like an oscillator?

Lecturer: [12:44] Yeah, Maybe I shouldn't have said that. If you want a kick drum sound? Make one of these. [tone]

Lecturer: Sorry. Make one of what I just did, not one of what you just heard. Make one of what you just saw. I shouldn't be telling you this. This is not good computer music here. But you can listen to the line tilde object, right? And if I do this people won't hear a thing and they can just see the speaker come move a little bit. Maybe, not even. But if you do this real fast like, if I replace these numbers with tiny numbers, like this. Let's replace this by four and make all these numbers, you know, public. Then I've got a nice sound that puts out. That's this. Oh. There it is. That's the sound of a pulse. You can't really see it in detail, but that's a ramp that's going up in two milliseconds, then it's staying at the top for two milliseconds, then it's coming back down. Six milliseconds wide, which means that the bandwidth of it is, well never mind, it's audible. It happens fast enough that you can hear.

Student: [inaudible] [13:59]

Lecturer: [14:03] The first number, yeah. 0.95 is the height here. Amplitude is one of these turns, but amplitude is how big it is in some sense so yes it's the amplitude.

Student: [inaudible] [14:20]

Lecturer: [14:24] Like this. Right? You don't want to predict what that will sound like? Yeah?

Student: [inaudible] [14:31]

Lecturer: [14:33] Looks different, but sounds the same. By the way, there's no reason that this wouldn't fit in the design and essentially sound different. That's a way that you can encode secret information in a signal that no one can hear. You've got the sign as a completely inaudible, but very present parameter. OK. So there's that. Let's get back to the other thing. Anyway, let me go back to your reality, which is I'm going to go back to using this as an envelope. OK. Now you didn't hear this because I just played the output of the line tilde, not the thing that is having the sample controlled by that. But you can imagine that if you took a sinusoid and multiplied it by a signal that started at zero and gradually went up to one-ish and then went back down to zero that you would hear the sinusoid turn on and off. That's what you heard before, which is this sound. Furthermore, it might be helpful, let's get rid of this, it might be helpful to graph that.

Student: [inaudible] [15:57]

Lecturer: Oh. The array is going from minus one to one and it's 44,100 points, which means one second. I have it graphing points and not lines, I think. Oh wait, it doesn't look like I'm doing that right so maybe that's not true. One little thing about that, there's another good thing in the properties of an array which is that you can select whether you want it to...there's someplace where you select whether you want to save it's contents. OK, here it is. If you un-check that then it will not pollute your patch with 44,100 values of whatever is sitting in your way. And then when you load the patch, it'll be zeros and your patch will be many many lines smaller, which might be a good thing as soon as you start putting large values in arrays.

Student: [inaudible] [17:17]

Lecturer: [17:23] On a Macintosh? [inaudible]

Student: [inaudible] [17:26]

Lecturer: [17:30] Yeah, you have the test version from December, maybe you still can.

Student: [inaudible]] [17:39]

Lecturer: [17:42] OK. This sounds eerily familiar but I thought this problem had gone away. I've seen this, I saw this last year so I should go back to worrying about that. I suspect that there might be weirdness for OS 10.4 if you have an old one but now, because I heard some words about that. So I don't know, maybe I should look at it later. I know what it would look like because I think I've seen it but it's obvious it's still happening. So, yes, I'm a Mac. When you do this and get the properties do you have a way of moving the properties out of the way because...no, that's not good. I've had it on Macintosh's sometimes happen that you get the two, these two dialogues show up and the array properties are right behind the canvas properties and, furthermore, you don't get the area that allows you to drag the windows so that you can't move it.

Student: [inaudible] [18:52]

Lecturer: [18:57] You can't. OK, so it's 50/50 whether you can move it or not. [laughs] OK, I better quit looking at this. What you are looking at now is not the output of a line tilde but the result of multiplying it by the ocelot . You've seen what the oscillator's output looks like, it's got a constant amplitude of one and it's batting up and down rapidly. Now what we have is nothing because no matter what's coming out of here you're multiplying it by zero, you're getting zero.

And that's nothing for the first 100 milliseconds until the thing ramps up and then there's a period of 100 milliseconds, it's not too clear where the thing is, and then there's a period of a few hundred milliseconds where [inaudible] . And this is what the output of a well-formed computer style should look like. It should turn on in a gentle way. At least it shouldn't turn on by just turning on, and then it should turn off by just turning off here.

So bad computer music style might be to - be something like this. Actually I'll just simulate it. Suppose I either didn't put the line in or just put the thing right in the multiplier like this - yeah, here. I'm just trying to figure out how not to do this using - Maybe the least confusing thing is I'll put times a zero.

Now we have a computer music instrument that makes a pop when it turns on and off. OK. What you see is every time that I whack it I get something somewhat different. It jumps from zero to some value. Whose value depends on what phase the oscillator happened to be at, at the moment I clicked the button. Or actually, at the moment the computer decided I had clicked the button.

And so at that particular time we didn't get a huge click because the value was relatively close to zero right when I whacked it, but if I whack it again - That one really was smooth on but it was not smooth off. You just get what you get. There it was bad - it jumped almost to full blast right at the outset. OK? This is a good way to make annoying sounds.

As a rule of thumb, this is either a function of psychoacoustics or personal preference, depending on your philosophy. If you give this thing at least five milliseconds to go up and down, you will get something that most people don't perceive as having a click at the beginning and the end.

And that, in my opinion, is about as fast an attack as you should have on something if you don't want to have a snapping sound, or a click. Unfortunately, that number is not a constant of nature. That number depends on the frequency of the oscillator. Low frequency tones, like 50 hertz-ish, you will still hear an ugly popping sound even at this speed, and you'll have to make this number larger.

So then it's a question: how do people who play bass , play their instruments? It doesn't take 20 milliseconds for a bass to start sounding. And there is [inaudible] of 20 milliseconds and yet the bass doesn't click. There is a reason for that but I'll try to explain that later on. It is possible to make things go on quickly without clicking even if there are bass frequencies which are smarter than what I've taught you to be so far in order to pull that one off.

In particular you should make it so that the phase of the oscillator is something appropriate for quickly starting up. So we have some barely acceptable computing instrument here. And while were doing computer music lets make this delay be just the rise time which is five and lets make this delay the half second. Now we have the standard computer music bell.

This is incorrect by the way. There is no bell that decays linearly. What would be the correct decay shape from the well logarithmic that's one way to say it should be a fall mix. Which is to say if you took the logarithm of it you should see a straight line going down. Its an amazing fact that a mass and spring system. Like the ones you studied in Music 170 as they decay they lose a fixed number of decibels per second. So if you believe in decibel as a cycle just to measure, the rate of drop off is actually constant if you listen to it. Which is why bells work as musical instruments in some sense.

This doesn't work it sort of hangs in the air and then ends in this sort of, I don't know what. That doesn't sound right it sounds like it was ringing for a while and then someone damped it. As opposed to someone let it ring. And if I graphed the logarithm of this which would be how you heard it. In other words if I graphed it in decibels. It shouldn't take the log of this. Well it could take the log of the of this. This is now this is showing the input generator again.

If I took the logarithm out of this you would see something that started off not quite level but then suddenly started dropping precipitously later on. Until finally here it hits minus infinity because the law of zero diverges. Questions about what I just put down? I'll probably say this again, yeah?

Student: [inaudible] [25:29]

Lecturer: [25:35] Yeah, how would you make it through logarithmic ooh that's a good one. I intend to bend your ears very seriously about that in the coming weeks. There are five or six ways you can do it. Depending on the exact spin that you want to try. One thing that you could do is you could say, oh you know what I'd have to use an object here. So let me go on to units because once I've talked about the type of things then I can answer questions a little bit better. Other questions, yeah?

Student: [inaudible] [26:17]

Lecturer: [26:27] Yeah. Oooh, that is a good question. OK. Units and PD are confusing. 44,100 is the numbers of samples in a second. 500 is the number of milliseconds in a half second and sometimes time is in samples, and sometimes time is in milliseconds. This is just kind of unfortunate, I don't know anyway around this situation. PD here doesn't actually believe that the axis here is time, this is really just an array of numbers. Which of course you could treat as an audio sample which is the way I'm treating it right now. But it could be probabilities or it could be weather data or anything.

If you want to use this to store a sound, the natural thing to do is have the horizontal axis be samples of which each one is 1/n-th of a second where 'n' is the sample rate. And that number varies - if that number were always the same things would be easier, but in fact sample rates vary depending on what you're trying to do. If you want to find about a dolphins songs you should not operate at 44k1 you should operate at a couple hundred thousand at least.

That explains the value 44,100, which was the size of this array which we set the panel that [unintelligible] . Here, these are values of time, which are interests to line tilde. So another part of the answer to that question is that different objects - for instance, line tilde, or OSC tilde, take their inputs to mean different things depending on their functions. So OSC tilde, it's input is set in the frequency of that object,. And line tilde, when I'm sending it messages like this, the message is interpreted as an output value which in itself is in arbitrary units. Which matters to the next object down. The second is a time which is in millisecond. And that is possible, because line tilde is a thing which actually runs in time, and so it has access to what the real value of time is so it can operate on a time limit as opposed to a unit of silence.

Student: [Inuadible] [28:59]

Lecturer: [29:14] You could use line tilde to set the frequency of an oscillator. The answer is yes you can and why not.

Student: [inaudible] [29:27]

Lecturer: [29:34] Now these are not good frequencies for an oscillator, right? So, 440 and lets make it a half second. All right. So, now I'll turn it on [plays tone and bends pitch] . Now we've got full computer music [laughter]. You can hook anything to anything. The only restriction being is that there is a distinction of type, which is to say something can be a number which is happening at the time of messages, or something can be an audio signal, which is something that comes out of the total objects. Yeah [calls on audience member].

Student: [30:26] Does there have to be a message line? Is there a message in [inaudible] .

Lecturer: [30:31] There is. You could do this. Watch this connection when I change that. Now, you'll get a wonderful effect [plays tone] . Let's make that a smaller number. No, it's too small [plays tone]. You can almost hear it now. Hear an arpeggio? The reason it's arpeggiating like that - I'll put it more in your face [makes adjustments to sound]. The reason you hear those - this thing, which I wasn't going to tell you about is a version of line tilde which puts out messages at a fixed rate. What rate? The rate that defaults to 20 milliseconds. Why 20 milliseconds? Because people used to use this to control midi devices and trying to cram more than 15 messages down a device line can get you in trouble for various reasons. So, this is exists, and sending a message to OSC tilde changes its frequency just fine, but it changes it in a way that happens right when it's going to happen, as opposed to doing it continuously the way a signal would. Another way of seeing that is our old friend, print. First off, there's nothing coming out of one, but if I tell it to ramp up to 880 [making adjustments] ... whoops. It's too high at 880. That wasn't a good example; it might go back down. And, it says, "oh, OK. 20 milliseconds later they're here and here", and this is the arpeggio that you heard which was too fast to be able to hear very well [playing a series of frequencies]. The version I showed you before which is [plays frequency pattern] this movement.

Was that clear to everyone?

Student: [inaudible] [33:08]

Lecturer: [33:11] Yeah, it makes it into an audio signal, which for practical purposes is continuous; in fact I think using the word continuous is a bit of a lie because it's really just continuously at the sample rate. Oh yea and we can do that here to, and this is going to be rotten. Let's do 100 here. I don't hear anything wrong. If we listen to this over headphones you wouldn't like it. Can't hear it in this room I don't think, at least I can't. What you should hear is a gritty sound that's called zipper noise, which is the effect of this thing now. So you have five steps, going from zero to .95, and this thing is changing discontinuously, which is there for a click, like this. Let's see, so here's the crude way to turn things on and off. OK, so if you do it this way you have the same thing, but each of the jumps is only a fifth as large so they're quieter and varied in the sound of the oscillator, but if you do that with your headphones you can hear something bad. It's actually, it's easy to track problems down when things are really bad, but when things are only just a little bad like that, then you will have to listen to your thing very much more carefully and more critically on order to be able to find find the problem. It's better if you can to avoid getting into that situation. So that's mine as opposed to one in tilde, which would read correctly.

Student: [35:09] The message at the top is every tenth of a second after you click it, it takes a tenth of a second to go to 110.

Lecturer: [35:20] OK, let me see. OK. See, you''re referring to this network over here. Understand me? These two, oh OK. So what's happening here is - Right. OK. Whenever I whack it, over the next tenth of a second when I whack it, it ramps from wherever it is up to the values, or down to the values it needed.

Student: [35:44] So assuming this [inaudible] turn on the DSP. So what's the starting value?

Lecturer: [35:50] Zero. So now, for instance if I'm playing the sound and I say 'OK give me a new one of these, it's putting out zeros until I decide to send a message telling it something else. Now that we've done that, remember that I showed you how to do rudimentary frequency modulation? This isn't in. This isn't exactly on side, this is an embellishment. So we save as, and we're going to say three FM again.

And now let's see - lets get rid of - sorry, OK. I don't need this anymore. What I need is another oscillator whose frequency is, I'm going to recreate as much like the old values as I the old ones I can remember. So I'm going to add 440 to another oscillator in order to create this oscillator. And that other oscillator is going to be M oscillator with an amplitude control. So lets get these two things out, oh these three things out. So this is an amplitude controlled oscillator. That's all you really need to do it.

So here now if I listen to that like this I get, whoops I don't have the off button sorry. OK. There's that and now we're going to take that instead of setting it up to an amplitude of one-ish I'm going to set it up to an amplitude of 1000-ish. Do not play this through the speaker. And meanwhile I'm going to make it slower so you can hear what's happening. And what I'm going to do to this, is I'm going to add 440 to it and make it be the frequency of another oscillator. Let's just make sure this is off. And I'm going to key it up a little bit so that you can see it. There that's going on. Alright, Ocsillator.

OK. Now what were listening to is this oscillator right now its playing at 440 hertz and this is going to apply vibrato to it. Vibrato is going to be at the audio frequency also at 440 hertz and its going to have amplitude of zero up to 1000. OK. And that's the sound that I incorrectly said was 60's computer music. This was invented or published in 1973 so this is 70's computer music not 60's computer music. That's a correction from last week. My apologies to John Chowning if he ever sees this video. Oh, John Chowning is the originator of the frequency modulation. I guess anyone who has a cell phones uses FM all day long. For which I once heard John say he was sorry.

OK. Is it, now this is an excellent moment to ask if you understand what's going on or not. Let me tell you one useful thing.

Student: [inaudible] [39:41]

Lecturer: [39:43] I'm sorry?

Student: [inaudible] [39:44]

Lecturer: [39:45] The graph, OK. I didn't actually graph anything just now. The graph parameters are properties. It's got a name, it's got 44,100 and I am saving contents which I probably shouldn't and I apologize despite the fact that I might need points. Whoops, I closed the other thing too. And then the canvas, let's just say the graph for this end is., the horizontal range is set so that it pulls the array. You can change that but-

Student: [inaudible] [40:24]

Lecturer: [40:28] Oh, OK. So arrays actually are indexed starting with zero in this file, so really the graph could have been from 0 to 44,099 but what would that change? That might move one thing over one pixel. Also, I mean you also can lie to it. You can say "I'm just going to start graphing 1000 please." You probably should do that and then it's all very good except the thing is - Oh, look at that. I just destroyed it. What did I just do? Hmm. All right. Can't see it anymore. I didn't want it that way anyway. So what's happening here is, first off this thing is repeating every... hmm, I don't know how to describe this... OK, this is an oscillator anomaly which is operating at 440 hertz and I'm applying vibrato but the vibrato itself is repeating every 440 hertz. What that means is that rather than changing the pitch of the thing, I'm changing the pitch so fast that it's actually changing the waveform. Or, to put it another way, the result is repeating still at a 440th of a second. The thing's repeating at 440 hertz still even though its pitch nominally is changing, it's happening within a cycle and the cycle is always the same period so we don't hear any change in pitch when we start applying vibrato.

[sustained ringing tone at 42:19]

[second, higher sounding ringing tone at 42:24]

Lecturer: [42:29] So, if we're not changing the amplitude. [sustained ringing tones end at 42:33]

Lecturer: [42:34] And we're not changing the pitch then old psycho-acoustics joke - if something isn't amplitude and it isn't pitch, then it must be timbre. So we're making a timbrel variation on the sound and there are ways of describing mathematically what's happening here which I won't go into but I will go so far as to graph these waveforms so that you can see the vibrato in action, and it's a good thing. To do that now, I have now to change the parameters of the data. That's long, but properties. OK. So let's have the thing only be 1,000 points now. Yeah. Let's see. Let's graph just the output of the oscillator without worrying about the amplitude. And it won't do it because my output is somewhere else. This is the old footage we tested. Yeah, question.

Student: [inaudible] [43:41]

Lecturer: [43:48] It does, yeah. Oh, yes. Thinking of it that way, the first oscillator is 2,000 volts right now. It's actually varying from plus or minus 2,000 volts and I'm adding an offset to that, so it's varying from 2,440 and minus 1,700. Yeah?

Student: [44:10] Sorry, could you explain [inaudible] output?

Lecturer: [44:15] OK. This is going to be the target value that line tilde gets, so if I graph the line tilde output it would be 2,000 units north.

Student: [inaudible] [44:24]

Lecturer: [44:30] Oh, what unit is it? It's in whatever units the thing I put it to is taking it to be. In other words, it's really just a pure number. The units only become relevant when you use it for something, which is down here. So it's really just knowledge about the patch that this is all operating in hertz and the reason it's operating in hertz is because this oscillator wants it in hertz, or cycles per second. Yeah?

Student: [44:57] 2,000 is [inaudible] is not connected to the value of the oscillator, right?

Lecturer: [45:02] That's right. And that's why amplitude is such a slippery word. It's both an amplitude, but it's an amplitude that's in hertz.

Student: [45:13] It's a magnitude.

Lecturer: [45:15] A magnitude? Yeah. This is a question. I think an amplitude as being able to be positive and negative and magnitude as being the opposite value of the amplitude. That might be a local usage of mine. It's the usage that's in my book. It's also what the quantum commission would say, I think.

Student: [45:37] If that's 2,000 hertz then it's not oscillating [inaudible] .

Lecturer: [45:42] That's right. And yet the change that I'm making in the frequency of the oscillator is varying, but it's varying in such a way it all adds up to no variation because there's as much positive as there is negative. So half the cycle here this thing is adding to the frequency, the other half is taking away from the frequency, so the average frequency is still 440 even though it's going up to much higher than that and down to something negative. Yeah. Let me see how you graph it. Let me graph it. Hang on to the question for just a second. I'm just going to [unintelligible] here. Change it to two, minus two again.

Students: [inaudible] [46:41] .

Lecturer: [47:23] OK, so now we're looking at the oscillator, but we're not changing the frequency. So now you see the period of the oscillator is from, for instance here to here. So it's about 2/5ths of this number line. OK, I changed the window by the way, so that it's at 250 points now. So it's a very short amount of time in the life of this sound.

Now I'm going to send the oscillator up to amplitude 2,000. I'm not going to graph that, because it would be something brief. But you can still now look at the output of this oscillator, and then you get - OK now, this requires some explanation. It still has the same cycle, but the -

Before I get it 2,000, let me give it some smaller number like 440. No, smaller than that.

OK, here, this is easier to understand. So you see it's still cycling from here to here. That it's going too fast, and then after that , it's going too slow. It's going faster here because at this moment in the cycle, this oscillator was positive, and therefor was adding to the frequency. So it sped up to some frequency much higher than 440 hertz. But then over this period of time, it has slowed down to some frequency much than 440 hertz. And on average, over the cycle, the frequency was 440, and so it actually made it through the cycle in the correct amount of time. But it did it in a non-uniform way, whose result therefor was not at all a sinusoid. And then if you listen to it, you won't hear a sinusoid, you will hear some other waveform that has some other partials, because waveforms have partials. OK. Now what was your question?

Student: [Unintelligible] [49:27]

Lecturer: [49:46] Oh, yeah, there's a couple of ways you could do that. You could take the absolute value. Which means negative values simply indicated so they become positive. Or you could slide the whole thing up by adding one to it. And then you would just see the entire wave form do that. Oh, that would be if you added one to this oscillator. This oscillator right now has an amplitude of 330 but I'm adding 440 to it. So if I graphed it I it would go up to positive in fact. It's all ranging from 110 to 770.

Student: [inaudible] [50:22]

Lecturer: [50:28] Yeah, in my way of using amplitude and magnitude I would say that magnitude is the amplitude because the number is the positive real number. Whose absolute value is itself. But I would never say something like that except to confuse someone. So these are amplitudes which are variously positive and negative. But if you added enough to a sinusoid - if you added enough composite to a sinusoid it would be positive and the samples would all be positive.

Student: [inaudible] [51:03]

Lecturer: [51:06] Oh boy it's always effacing. But they would be different things effacing. But that probably didn't answer your question very well did it?

Student: [inaudible] [51:15]

Lecturer: [51:19] Yeah, I think the gist of it, you're confusing yourself by using the word magnitude.

Student: [inaudible]] [51:26]

Lecturer: [51:35] But how would that help you?

Student: [inaudible] [51:36]

Lecturer: [51:50] Yeah I think I could do that to this and then the frequencies wouldn't - whats the right way of saying this. Obscure it, the results would be complicated. Yeah OK.

Student: [inaudible] [52:03]

Lecturer: [52:10] So OK. So now if I make this value bigger, let's try 1000.

Student: [inaudible] [52:16]

Lecturer: [52:20] Zero. That just means do it right now please. So this is equivalent to 1000 space zero. And now we have the following co-area sinusoid which did get around to phase zero once in the cycle but in fact was going, you can't see it right now. I'm going to just scramble the phase a little bit and see if we get a better one. Oh, there we go. Alright. So at some point it hits zero phase and then it goes racing along until at some point it decides no I went a little bit too far lets go backwards. And so I think at this portion of the wave form I think it's actually a negative frequency going backwards until it decides to come backwards enough to rush forward at superior speed. So by pushing the - OK. So what I did was I made this oscillator have amplitude 1000 which therefor is so great that even after you add 440, you have both positive and negative values. And therefore you see the thing wrapping both forward and backwards in the cycle.

Student: [inaudible] [53:33]

Lecturer: [53:36] Yeah. In fact, you can do more of it. So now I'm going to ramp it up to 5,000 and then graph it. 2,000. Sorry. Yeah, yeah. Alright. Let's make it 5,000. Oh. There maybe I shouldn't be graphing points. Let's go back to graphing this stupid polygon. Polygons are better if you have smaller numbers of points. There we go. This is the classical, pedagogical wave form that we show when you show frequency modulation. The thing is wrapping forward crazily and then wrapping backward crazily to add up to this one cycle forward. Yeah?

Student: [inaudible] [54:37]

Lecturer: [54:44] OK. So we output this oscillator and the oscillator's frequency is averaging 440, but it's varying by 5,000 around that average, which means it's rarely in the vicinity of 440 anymore. It's just being scattered all over the place.

Student: [inaudible] [55:01]

Lecturer: [55:09] Oh, this is getting added to this oscillator. Oh, you mean this 440 on top? Oh, that's a good question. What if I make this 440 something different? Let's turn it off first. So now we're listening to the same thing. Now what we have is something that looks like this, but what does that make it? There's the original tone, which is an octave higher. Now what I'm doing is I'm varying the frequency at a rate so that it evens out completely over two cycles. So the resulting period is in fact 1/220th and not 1/440th. So this is the frequency at which this thing is changing. Now the variations are taking twice as long to cycle, but this is still the center frequency, which could be some other number if we wanted to. Yeah?

Student: [inaudible] [56:23] .

Lecturer: [56:36] The line controls how widely it's varying around the center value of 440.

Student: [inaudible] [56:42]

Lecturer: [56:45] It's how quickly the various [attack - yeah! Yeah, yeah that's it. So the frequency of this oscillator is 440 with disturbances. The disturbances have both an amplitude and it has a speed. So the speed is 220 times a second and the size of the disturbance is 5000 or whatever it is that I set it to.

Student: [57:06] And its 5000 hertz?

Lecturer: [57:09] Yeah, because it's being used as hertz because oscillator is, these magnitudes are eventually finding their way down here and then they're being used as hertz. But I could use this to read a sample or something like that and then it would be something different.

Student: [inaudible] [57:25]

Lecturer: [57:26] Yeah.

Student: [inaudible] [57:27]

Lecturer: [57:37] It is FM modulation. It's frequency modulation. Which is FM. You could even think of it being as overdrive in something too, but I'm not sure what. Yeah maybe, I'll talk more about wave shaping and over-driving and stuff later on. It is a sort of overdrive. Now OK, so gravy on the cake is why don't we just make this thing be something we can control. And Chris...Now I'll go back to the original. The amplitude now is 1000 and now I'll start changing the frequency continuously. Alright, oh so this, now looks like this. And you don't see a period in fact, you have to wait an entire second I think, no, you have to wait a fifth of a second before the thing all wraps around. So now you get something which is a nice enharmonic tone.

And you can analyze this and find out what the frequencies of the enharmonic partials are which I think we'll have to get into week six or seven, but here I'm showing that this is a thing that you can do. Now, of course the amplitude can still be varied, the amplitude of the modulating oscillator can be varied and then you get, hang on, that was good, 19 seconds to get as good as it sounds. Yeah.

Student: [Unintelligible] Instead of making him do it in five seconds can you do it in [unintelligible] [59:54] seconds?

Lecturer: [1:00:01] Oh yeah OK. All right speed it up.

Student: [inaudible] [1:00:03]

Lecturer: [1:00:04] Now, these are going to sound bad with these values [plays computer music] . What I don't like about it is that you hear this little wah-wah effect as it's changing and you can't get that wah-wah effect out, and it's cool for the first five minutes but then you get really tired of it. Most people who use FM, don't use these kinds of values. They're good pedagogically, because there's no way you can miss hearing it. If you keep these values upon the order of this then maybe there's twice as much as this. You don't get that wah-wah, but you still get a timbrel variation. You don't get a whole lot of high partials, so then, if you want high partials but not the wah-wah then it might be a little harder.

There are five or six ways I can play with this thing. But, that happens later. There's a homework assignment for next time which is on the website, but I haven't made the website to upload [unintelligible] . The homework assignment is actually not to do FM, is something that you will need line tilde for, which is to make the collection of four oscillators which makes the tone and breaks them into two tones. After you've enjoyed it for awhile. I don't think I'm going to be able to get my computer to play this.

This is a graph that shows you how you can do the thing. It's playing out some other audio device that I can't control. You'll hear it if you play it. It will start out as a nice tone. This is a time versus frequency plot, which is a way of describing how you might wish the partials of the sound which would be sinusoidal components which would add up to make a sound made up of sinusoids .

So, what I'm describing here is how the frequencies of a bunch of sinusoidal components might change in time. If you played this and, for instance, put the amplitudes, they're not shown here, but if you made the amplitudes all equal, when you play these four you will hear a tone. At least if the four sinusoids start at the same time, you'll hear a tone, which, this frequency would be that of the fundamental. Which I think I suggested might want to be 220 Hertz.

And so if that were true this would be 220, 440, 660, 880. And you would hear a nice tone until this thing happened. At which point a wonderful psycho-acoustic effect would take place. Which is your ear would quit being able to hear this as a tone. You would still hear this and this being fused as a single tone at this frequency, although its timbre would change because it would no longer enjoy even harmonics anymore. And meanwhile you would hear this, these two. Oh, what's the interval between this partial and that partial?

Student: [inaudible] [1:03:44]

Lecturer: [1:03:48] What's the interval? Two to one is the ratio. Yeah, so an interval is a ratio, really. So the interval of two to one is called an octave, in Music Land. So since these are an octave apart, they in fact could also function as a tone at this frequency that has two partials. And you will hear that tone as soon as this thing starts sliding away, because your ear will no longer allow it to hide behind these partials to be considered part of this tone. So what you'll hear is a single tone that bifurcates into two tones paradoxically. One of them being, consisting only of odd harmonics, harmonics number one and three. And the other consisting of harmonics one and two of a different pitch. And that's a wonderful thing to contemplate. I didn't bring it along this time, but next time I'll play you some music by Jean-Claude Risset which uses that in interesting ways. Basically, you can design timbres that you can tear apart and make series of pitches out of. Or collections of pitches out of, it's fascinating.

And, it is indeed 60s computer music, because that was deprecated before they even had access to frequency modulation. So this has nothing to do with FM. You can do this just with additive synthesis is what computer musicians say when they're talking about making a bunch of oscillators and adding their results. So you can make this by adding four oscillators up.

And now that you know about line tilde, you can arrange for the frequencies of oscillators to slide from value to value. And of course you should make the whole thing turn on in a smooth way and then do this, and then turn off. And that will require also that you have delay objects because you want the ramp up to start here but then you want the change in frequency to start here and then you want a ramp down to start over here.

Student: [inaudible] [1:05:45] ?

Lecturer: [1:05:50] Yeah, and oscillators, yeah. It's, basically, you just practice with the objects that you all know about. Yeah.

Student: [inaudible] [1:05:58] ?

Lecturer: [1:06:02] Oh, how would you ramp the thing down? Yeah, you can't show it, I'm not graphing amplitudes here but frequencies. So I would have to make a separate graph to show how the amplitudes would change. And, yeah, just have a line tilde multiplied by the whole rec. And then it, after an appropriate delay you send that to a nice missing zero with a planned value. And then it would turn off. And then when you do that you will have full access to all of the additive synthesis. At that point you can make the complicated; well, OK, your patches will be horrible if you actually try to do it without introducing some automation. But you will at least in principle have control over the detail, over the structure of the harmonics, or enharmonic partials of any sound that you wanna make. Which could be powerful.

MUS171 01 20

Professor: [0:07] Oh, of course I've lost my beautiful sequence. [background noises] Here's the thing I'm going to forget to tell you, so I'll tell it to you now while only the early birds are here while I'm thinking of it. You can set the screen size of tables. So if the one size fits all size that you're all used to is not what you really want...Like in this example you can do something else. In this case my table is too fat and it's not tall enough. So if I do that one, then I have...Oh, maybe that's too tall for my screen. Anyway. Yeah, not good. [2:03] All right lets make it 400 pixels tall and 100 pixels wide. No, looks like I have 300 pixels. All right, there it is. OK. My reason for doing that was because the bigger you make something vertically, the more accurately you can control it with a mouse, assuming you're controlling all the way down to the pixel, which is the way tables work. Actually they maintain floating point values, which are to high precisions. But when you're editing you can only push them onto a pixel value, which might or might not be available, but it's more likely to be available the bigger you make the thing. So that's a trick.

[2:46] This is just a review from last time. Right now what we're doing is we're looking at the first seven values of this table. Bring this up to here. And the reason I know that we're doing that is because I'm taking this phaser which goes from zero to one in value and multiplying it by seven. Which means now your going from zero to seven. And then values from zero to seven truncate down to values from zero to six, when you truncate them to the greatest integer below them.

[3:22] And as a result you're looking at the first seven values of the table which are numbered zero, one, two, three, four, five, six. And the others are just there in case you want them later. You don't have to have them in fact I could resize the table to seven values now, but I might decide that I want a different size sequence later, and I can do that just by changing the number. And now we're looking at six, seven, eight...nine out of twelve values.

[3:57] Notice that it gets faster as I... as could use more values that's just because the frequency that I'm giving the phaser is setting the period for the entire sequence and the more notes I stuff in there, the faster the notes are going to be going. Or to put in another way; the more you multiply the phaser by, the faster the output is changing time because the more it travels ultimately.

Student: [4:19] So what's the [inaudible 04:19]

Professor: [4:21] That is the frequency in hertz of this phaser.

Student: [inaudible 4:27] [4:26]

Professor: [4:30] It... let's see, it does 0.7 of that sequence every second. So if I set this to one, it's one cycle per second, if I set it to two, it's two cycles per second. So here's one. If I say two it'll go twice as fast. Oh, so right now it's every second. Two is not every two seconds, but every half a second. So 0.7, what's that? That's one over 0.7 seconds in period. And I don't know why it shows that value. Actually I know why. I tried 0.5 and it's too slow for my taste. [5:23] OK. So the purpose.... does everyone understand how this example works? Or, can you formulate questions if you don't?

[5:41] Yeah?

Student: [5:42] Why would you have [inaudible 05:43] .

Professor: [5:45] Right, OK, so the multiplier is taking the phaser's output and re-scaling it so that it goes from zero to seven in this case. And the adder, I'm not using at this point, I'm adding zero which isn't doing anything. But if I wanted to instead of looking at the first seven points, look at the points two through eight or three through nine, I would add something to the scaled value and that would be reading the placement table.

Student: [inaudible 06:09] [6:07] .

Professor: [6:09] Yeah... oh boy. It's short of a phase shift, but it's also... it's just a shift-shift.

Student: [inaudible 06:18] [6:15] .

Professor: [6:18] Right, yeah. This is worthy of... what's the right word? This is worthy of just mastering in it's own right. This is the general formula for I want a range and I want it to be seven wide and I want it to start at zero. In general, if you have something that goes from zero to one and want it to go from A to B, you multiply by B minus A, which is the size of the range, and then you add A to slide it over to where you want it to go. And you do this all the time [inaudible 06: [6:47] 50] . You can do the opposite as well, which is you have something that's going from A to B, or A to D, or whatever, and you want it to go from zero to one. Because you then want to give it another range cycle. So, to do that you would subtract A to make it start at zero, and then it goes from zero to B minus A, and you would divide by B minus A, so that goes into your one. Yeah?

Student: [7:11] So, if you start with three as the [inaudible 07:14] core value and move around...

Professor: [7:16] It'll... Neither. If I give it three, instead of going zero, one, two, three, four, five, six, it would go three, four, five, six, seven, eight, nine.

Student: [7:27] What if you don't have enough values?

Professor: [7:29] Sure I do. I've got twelve numbers in here.

Student: [7:31] What if you don't have them?

Professor: [7:32] Oh, if I didn't? Then, OK, then it's up to the object to have [inaudible 07:36] to figure out what to do with when I give it a number that's out of bounds and what it does is it simply clicks it. That's to say it simply limits it, so that it doesn't wrap around. You could make it do that. In fact, if you wanted to make it do that, you would, before this, you would say, modulo twelve." Which you all learned about in high school, right? Some logic about it. Yeah. And you would do something somewhat different in signal land. Because mod works great for integers, but for floating point you might want to have something that's defined a little bit differently from [inaudible 08: [7:58] 17] . So, that gets us down to there, and then it's just the same stuff that you know about from before.

[8:24] The reason I'm bringing this up today is because I want to use this as the starting point, because it might be nice to know different ways of getting values in and out of tables, of which there are a half dozen. And I was having trouble figuring out which one to show you, so I just decided to show you all of them. At least, all of them that I could think of. And because actually it's pedagogically valuable to know them, and we're all about pedagogy here. So here goes. I have a list.

[8:55] The first thing that you might want to know about is a wonderful object called tab right. In fact, I told you about this, but I didn't tell you all about how to use it. Without a tilde, and let me give it the table name, which I'll just get from here. And this is a thing which takes two numerical values, which I'll start out by using the number boxes to specify. And here you say which value in the table you want, three say, and here you say what you want it to be.

[9:30] And, tada, you're moving value three around in the table. So, you can see this thing is going up and down as I'm doing this. OK. This would be kind of--what's the right word?--hideous to have to make a pass that did this to every single value of the table. Because you would need a different tab right object for every tab group that you wanted to set, which would be ugly. So, you want to do it a little bit more smart than that, for which you will need some objects that might generously be referred to as glue.

[10:00] So, there's some glue involved in using things like tab right most effectively. Glue. So, suppose I wanted the values to be the following, which I'll stick in a message box. Is this what I want to do? OK. I'm not going to use this message box yet. I'm just going to have these values here so that you can see them. All right. Do I want seven of these things? Yeah. Let's have seven of them.

[10:33] So, now... I'm just making multiples of 1/10, because since I spend a lot of time in music studios, I like things to actually belong to the western scale. And 1/10 is an A. So, these are easy numbers to just sort of fetch out of nowhere that happen to have at least some reasonable western meaning.

[10:54] And suppose I want to get these values into this table, what would I do? OK. Technique number one. We could use tab right. But of course, I don't really want to type all this stuff in. I want to just have it in a message box. How would I get that to happen? Well, there are things that you can do. One thing is this. ...Message box. Saying that the first number is going to be 990, is the same thing as putting a zero here and then putting a 990 there.

[11:32] And a short-hand for doing that, in PD anyway, is you can give it a list of values. And what this means to an object like this...What does an object like this mean? An object, except for a couple of exceptional objects, will take a list of numbers and interpret them to mean, "Put those numbers in the in-list." So, this is saying, "Set the value at slot zero of the table to the number 990. So, this thing jumps up to there.

[12:06] I don't have a good range with these numbers, do I? Let me change my range, slightly. So, let's go up to 1,500 hertz. All right. And this is OK, except of course now I can't do this any more. I can just click it. So, it might be nice to know how to do this, but do it in a way that has a variable associated with it, so that I can make that not be a thing that's known already. So, that's one thing that I want you to just wonder about for a second while I show you something else, which is this.

[12:42] OK. Now, I can say, here are a bunch of message boxes. And if I had all seven of these that correspond to these numbers, and gave their locations, and if I set them all in sequence to tab right, then I would set the table to these values. So there's the first one, there's the second one. Oh, I didn't connect it. There's the second one, there's the third one. That's ugly.

And of course, now yow know that if you just wanted to have some way of doing them all...in fact, I can [inaudible 13: [13:16] 22] to something else. Let's go like that. I could just give myself a button that activated all these message boxes. A message box actually will respond to any message at all. So [inaudible 13:33] would be perfectly all right. We'll output its contents.

[13:40] So now, let's bash it again. If I whack this button it puts the guys in the table. Yeah?

Student: [inaudible 13:47] connect most of these message boxes to your message box, that sort of thing?] [13:47]

Professor: [13:51] You can do that. I didn't do that because I was afraid of confusing people, but you could do this. I wouldn't regard this as terribly good style. But why? Because, in fact, what this is doing is confusing. Because I'm sending this message to this message box which is ignoring it. Someone could look at that and wonder what on earth I was doing. [14:16] So it's a little bit better to do it the way that I did it, but you could indeed do it that way. Oh, come on. Let me select this other thing. This is still a little bit ugly. So let me show you the next thing that's a little bit less ugly, which is this. I can always say, still using tab right. I can say, "Pass the following messages, please." OK, I'll keep going. OK, I'll stop there because I'm going to run out of screen.

[15:02] Now, what this is, is four messages in a single box which are delineated or separated by commas. So commas are special characters in PD. There are four special characters in PD. The comma, the semicolon, dollar signs, and spaces. Spaces you know about because you just use them to separate things. Commas separate things, but in a way that means to a message box, "Let there be another message now," and it's going to start here, with this thing.

[15:37] So this is different from this message box because this contains one message with seven numbers and this now contains four separate messages which will all be sent in sequence. I have to tell you about how long it takes this to happen, but that's perhaps for a little bit later. It doesn't take any time for this to happen, is the short answer.

And so this is now a short-handish way of saying, "OK, go ahead and send these four messages [inaudible 16: [16:00] 06] ." Furthermore you know what order they happen in, and here you don't actually know what order these three things went off in because it really is just whatever order I connected these things in. The order of the things happen in is sometimes very important.

[16:21] And I've been avoiding getting into situations where the order matters, but I'm going to start making situations like that today, because that's harder to believe. Questions about this? Yeah?

Student: [inaudible 16:36] [16:35] .

Professor: [16:40] Oh. So this is the important... there are twelve points in it and Y... the vertical axis is nothing to 1,500. I made it -1 so you can see zero, and there are twelve points in it. And this is the screen size of the table which I changed also. [17:07] Yeah. All right. Other questions about this? Everybody's clearly happy.

[17:18] Let me show you another thing that will save you some typing. If you really just want to put a bunch of numbers in the table, another short-handish way of doing it is to say... Hm, should I tell you this? Yeah, I can tell you this. OK. Second technique of doing things, the second flavor of glue that exists in PD that you'll want to know about. And I'm going show this to you now because it easily confuses with what I'm doing now.

[17:46] So right now what I'm doing is I'm sending messages to this object tab right. You can also send objects straight to this... oh sorry, send messages straight to this object. And then you're not talking to a particular auxiliary object like tab right or tab tilde, or whatnot. You're talking straight to whatever the table itself thinks that it likes to do with messages. And what it does is design for what it is, and how you get a message to it is you say "Send."

[18:23] So let's get an object, and the object is going to be send, and then I'm going to give it the name of the table as an argument. And then I can actually send it a message which consists of the following stuff. It has a place in the table that we want to put a value in, and then it has the numbers that I want the table to have.

Student: [18:57] So [inaudible 18:57] where you want the sequence of stuff.

Professor: [19:00] That's right. So this is now... that's on the horizontal axis. Start at point zero please. And these numbers, any number of numbers that you wish are the numbers that you're going to put into the table. All right this is a good thing because now there's an easy way just to write numbers into the table. [19:26] All right. So this, sorry I have to think about something here. I'm going to have to start saving parts of this patch because it's too big for the screen. Well maybe this is just what it is for now. OK, so here's technique number one, down here is what looks like a far better technique. Oh, I don't need this any more. That was just to make the example, but here's the thing that actually did it which has all the information.

[20:08] Now... Sorry, I'm just cleaning up. OK, so send is the thing which will allow you to send a message to any named objects. The only object that you've seen so far that has a name is this array or table. Another thing that can have a name is an object whose only purpose is in fact to have a name which is called receive. So, let me give it a different name. Oh, you know what? This belongs in a different patch. I'm going to save this patch.

[20:53] This is going to be two and then three is going to be send and receive...and some other stuff. All right. Let's see, we don't need this. By the way I'm going to get rid of this because... I'll try to develop this as I do it. OK, so first off, send and receive as objects. So, well I don't want to use the word "foo," that's too frequent. So, dog.

[21:49] So here's a message that we're sending to dog. In fact so that we can see it, I'll use another box. Control, three. And then I'm going to say receive dog, and it gets an outlet. Oh, so send dog only got an inlet and didn't get any outlets. Receive dog gets an outlet and no inlets. And now I have this going on. All right. So now what you've got is, I was able to put values in the table by sending to the name of the table, but I can also put values into receive by sending messages to receive.

[22:32] And to be a little bit more parallel to what I did, something a little bit more similar to what we saw above was, I was sending a list of numbers like this and a number box will take a list of numbers and it will pick off the first number, as it can only deal with one.

[23:03] All right. This is all going to require five objects. Yeah?

Student: [inaudible 23:09] receive [inaudible 23:13] [23:07] .

Professor: [23:15] Yeah, how could you retrieve all that? OK, this is another topic... Yeah? Which I'll get to in a moment. I'm afraid to talk about it right now. OK, so send and receive are useful because by now you've probably noticed that your patches are starting to have lots and lots of cords, many of which are short because things are local. [23:45] But occasionally your patch, no matter how hard you try to figure out how to lay it out, will have a line like this. And one way of dealing with that is simply to put a send object up here and put a receive object down there, and that way you get the connection and not the awful line going through the patch. Later you'll find that it's even better than that because you can have patches that have multiple windows in them, and send and receive is a good way to get from some remote, not very directly related window.

[24:15] Yeah?

Student: [24:16] So [inaudible 24:18] between different patches, you can send. [24:20] .

Professor: [24:21] Yeah. So for instance, here's a new patch and here's an object. I'll give it the same name, and then I'll hook it up to the number. Now over here, I say send dog. By the way, one thing that you can like about Linux is that you can do this. In your computers you have to click this thing to the front and you won't see this. Most people hate that though. OK. Yeah?

Student: [25:02] You said it was wireless though?

Professor: [25:04] Yeah. It's a wireless connection. And because it's wireless, it even works from window to window. Oh, and this is why I've been giving the tables all these funny names, because the tables will confuse themselves across patches just as the send and receives do. So if you use the same send and receive name in two different patches and have them open, they will get tangled up into these messages. [25:25] Yeah?

Student: [inaudible 25:26] [25:26]

Professor: [25:30] You have to work harder to do that. [laughing]

Professor: [25:33] And furthermore, the timing semantics will be different if you have two different computers. If you do it in one computer it's instantaneous, but there will be network delay if you do it across two. The objects you need are net send and net receive, instead of just send and receive. [25:49] Yeah?

Student: [inaudible 25:49] [25:49] .

Professor: [25:51] No. Yeah, right, which makes it even easier to get confused, because your patches could be things you develop in different years, right? Or, by different people. And they could still be talking to each other and getting you in trouble.

Student: [26:07] Only if the [inaudible 26:07] .

Professor: [26:08] Only if the patches are in the same PD. You can also run several copies of PD in each case you'll get away with it. But at the same time it will turn out to be good practice some time in the future to figure out how to [inaudible 26:22] the send and receive things, so that your patches don't cross [inaudible 26:24] automatically. That will be a thing later on in the quarter. All right, so, next thing I want to tell you about is packing and unpacking. This is partly in response to one question I think. But it's partly also just a thing we need. So the question was, how could you actually retrieve all this stuff if you wanted to? Actually there are two related questions. First off: [26:44] "I know how to send a variable message with one number in it, but I don't know how to send a variable message with more than one number in it." Which for instance you would want to do if you wanted to change...let's close this now. We don't want to save this. So now suppose I want to have a number box change this fourth number in the table?

[27:13] Well, we could go back to what we were doing. Let's see, of course it erased it, but I'll put it back. Tab right, without a tilde though by the way. If it's a tilde it wants to take a signal and it will write the signal sequentially in the table. But without the tilde it needs two numbers which are where you write into the table and what you write into the table. So if I want to change the number at location three which is the fourth number here to some new value like 1,000, I have to type three there and I have to type 1,000 there.

[27:56] It would be nice to be able to do this and move it up and down without having to set this, or whatever. Or, to be able to have this thing be controlled programatically by something else for instance, that sort of thing.

[28:12] So, ways of doing that, There will be two things that you can do to get this to happen. The first is just using a message box. Oh no, let me tell you the other way first because you need this first. So one thing you can do is you can pack and unpack messages that have multiple numbers in them. So the way of putting two things together...I'm going to get rid of this because we already have it.

[28:46] So I'll go back to patch number two and show you some stuff about that signal processing network a little bit later. But first off, here's a thing. I'll get two numbers and I will pack them into a message using an object which is called pack. And then just to show you what's...Oh, so just to show you what's happening the easy way... And then we will see pack messages.

[29:24] So as a rule in PD, when you throw things into inlets that are not the first inlet, if it is not a tilde, if it's a control object, like pack, you put things in the inlets and it changes the object, it revises the object, in some sense. It changes its internal state. But output in general, or whatever the thing does, is typically engendered by putting something in the first inlet. So now we get this message out, "279."

[29:52] So it has two numbers in it, and this message is those two numbers packed into a single message which you could call a list. It's a list of numbers. All right.

So I can change this all I want and nothing happens, but then as soon as I put in another value in here, I get to see 100 higher order numbers. And now I have that...that hasn't helped me yet, has it? Oh, here is the thing that might help me. If I want to have a way just to write into location four of the table, one thing I can do is say pack 04, and then I will a put a number in there. And nothing happens...oh, because I'm not doing [inaudible 30: [30:05] 46] .

[30:46] OK, where is my tab right here? To try to avoid the confusion, I will make a different copy of the tab right object, and now I have got a message box. It is being converted into a message which is this value and four. I should try to receive it, so you can see it. I told you about print, but I didn't tell that you can actually give it an argument to specify what you print. Sometimes you need that.

[31:27] OK. So now what is happening is that I put a 450 in and this becomes the message 450 and four. I could change the value of four by putting a message in that inlet or I can just leave it and it would just say whatever it is, space four. Yeah?

Student: [31:43] Is zero specifying something about [inaudible 31:19] ?

Professor: [31:47] Oh, thank you. Zero is there simply to say the initial value of this element is zero. But in fact it is never used because in fact I immediately override it with similar values. But if I didn't put it there, it would just be packing the message which consists of four. And then I would just be generating the single number. All right, now you know how to do something that you didn't know how to do before, unless you've been looking ahead, which is I've been making these... OK, so what I'm going to do is save patch number two as patch number four. And I don't know what it's going to be named, maybe add pitch [inaudible 32: [32:33] 44] again. OK. Do you remember that I was making myself a message box to start the thing, and then another one to stop it? Now I can do that much better. This is ugly because this value is not in decibels, it's in linear units. And I would love to be able to use DB to RMS to be able to specify that number in decibels. But DB to RMS only has a single number that goes in and a single number that goes out.

Here's DB to RMS. Lets see, can I get rid of this? OK. I'll have to go back to this later, but for right now I'll just leave it like that. So, recall that DB to RMS does this: [33:27] DB to RMS. You put a number in and out comes the amplitude that you would need to have that number dB in your signal. This is a perfectly good way of turning things on and off. So 70 dB is about .03 because it's spending less time...By the way if your going to do this... I'm just going to disconnect this before I do the next thing. Make this wide. Perfectly reasonable numbers of decibels, like 160, are very bad numbers of amplitude.

So it would be very easy for me, notice I've disconnected this for this very reason. So I'm sitting here at zero, say, and let's turn it on to 70 or so. Whoops, just 70. If I did that, I might have just destroyed my speakers. So if your going to do this, take this message box, I suggest you take this message box and fix the upper limit to something really reasonable. Oh, let me show you that slower. So the message box [inaudible 34: [34:27] 57] boxes also have properties. It has a width... But you can specify a lower and upper value. If this value is above the zero, that means the thing is just whatever it is, and it can be positive or negative or anything.

[35:08] But if I give it values that are non-zero, it would interpret that as a range. And that range will be enforced, which is good. But what this means is I have something that I can mouse at very conveniently. So getting the thing to be zero, is that. Whereas if I did that to a regular old unvarnished message box like this? I'll turn it off and it will do that, which is stupid. At least, that would be stupid for dividing decibels. Zero is perfect for just turning it off.

[35:43] So it's a convenience to have it so that your number box goes to zero when you drag it all the way down to the bottom. To do that you set the range of the number box to be the values that you believe will be reasonable. And then, if you repent later and say, "I really want 90 dB," you can always type it in. It won't enforce the range. It's simply a thing that detects how the mouse operates on it. So, this is now a good volume control except for one little thing, and the one little thing is that it clicks when you turn it off and on.

That's why I was using these things that have time values. Of course, when you have message boxes for the time values you have to decide what the thing was in advance. But now that we have the pack object, we can do it all right. We can now say "pack: [36:28] zero, 50". And now, what's going into this multiplier is the output of line which is getting messages which are "0.2 (space) 50", those two numbers, and that will turn the line on and off in a way that will not click. So, this is a correct or at least a not incorrect way of managing gains in dB.

[37:40] You can do even better than this, but this is the first time I've shown you how to do it well enough that you would actually want to do this to someone else who had to use your patch. Are there questions about how this works? Yeah/.

Student: [37:53] Can you explain the two numbers in the pack?

Professor: [37:56] Yes. So, the two numbers in the pack serve to do two things; they initialize the values of the inlets in the pack. So, this value is initialized to zero and this value's initialized to 50. They also serve to tell you how many inlets the pack will have because, you won't have to do this any time soon, but some day you might have to have a message that has six or 10 values in it and you can tell pack to have more than two things by doing that. [38:30] If you just say "pack," by default it assumes you mean "two". If you want something else, give it numbers, numbers which will initialize the inlets, but will also supply how many inlets you want. And you had a question, too?

Student: [38:50] Yes. Is the number box [inaudible 38:51] ?

Professor: [38:54] No. It's just so you can see it. And furthermore, there are differences of opinion about whether this is good style. So, there are people around who will pontificate about the proper programming style in Max and PD, more in Max, actually, than PD. The PD community is forgiving that way. [39:15] But some people say that you shouldn't do this because someone could walk up to your patch and do this. And this would be in a bad state now. Because if I "moused" on this thing, it would suddenly jump from 0.3 to whatever I just made here, which wouldn't agree with it, right? And that might not be a good thing. Yeah?

Student: [39:36] Can you make the other boxes un-editable, so you can't edit them?

Professor: [39:38] Un-editable? Oh, un-editable. [laughter] . I don't think they're editable, but you can't, no. Oh, wait a second. No, you can't really. You could set the range to be from two values that are equal, but then if you touched it, it would jump to that value. So, actually, yeah, they're always going to be editable. [40:03] But one thing that you can do that sometimes helps people--and this is programming style, you can do it or not--is make the thing hang like that. So, that it is clear that it is only there for the purposes of showing me what the number is, and it's not there for you to mouse on.

[40:20] That won't stop me from "mousing" on it, but at least you can tell that "mousing" on it is not going to do anything for you. So, there are people who argue strongly that you should do this, instead of what I just did. And me, I'm agnostic about it. I do it sometimes one way and sometimes the other.

[40:43] Are there questions?

[40:47] So, that is pack and unpack, which we use for two different purposes. Actually, pack seems to get used a lot more than unpack, because there are objects which you have to send messages to that have more than one number in them or more than one argument, as you might call it.

[41:08] Unpack, you use if you want to make your own thing that takes pack messages and distributes the numbers in some way or another.

Student: [41:17] I don't think you've showed us unpack yet.

Professor: [41:19] Oh, I haven't shown you unpack. Sorry, yeah, better do that. OK, so this is now number four. The reason I didn't show you unpack is because I forgot to finish number three. So, let's go back here. [41:32] So, this is pack. I was just showing you this, and then it wasn't obvious why this was a good thing except for putting numbers on the table. But the other good thing that you could do with pack is controlling volume, controlling volumes in a way that it would operate in EB, which you couldn't do before.

[41:48] Now, unpack is this one. Again, you give it arguments. By convention, I just use zeroes for unpack, and the arguments do nothing but set the number of outlets, and the outlets are going to be the two numbers in the message. Whoops. So now, it's showing me the numbers that I was previously using print to see. The reason I used print before was because I hadn't told you about unpack. But now that you have unpack, you can now make things. Assuming you know when you're building the patch, how many items are going to be in your message, how many numbers are going to be in your message, then you can put an unpack there to show yourself what those things are, or to distribute them in some way that does something useful.

[42:50] All right. So, yeah? Good. So now, having seen that, the next thing to do, I guess is go to window number four again, and using send and receive, bash, you basically know how to do it all now. Except I want to show you one other good thing, which is that you can use text files to maintain numbers in tables, and that is a good thing, because that's one of the easiest ways of getting PB Noters software programs to talk to each other. So do that, we'll go forward to ... no, we'll just stay here, actually. So, the table, here, can be sent a message, which we can do using the send object. So we're going to send to tab, oh, we've got one of those.

[44:05] We've got an object called send. OK, so, since we have that object send tab, bla bla bla, I'll just re-use that object for all the other messages that I'll be sending this thing. There are plenty of things that you can send at the table. This was a thing that got values in just by specifying your numbers, so there are also messages that you can send, which are ... I'm introducing too many things here. First off is fact that you can actually have messages that have commands in them, and second, the fact, what the commands actually are. You can say, read some file, and it will look at the first twelve numbers that are in that file, and put them in the table.

And of course, what I have now is just an error message [indecipherable 44: [44:54] 58] text can't open. OK. So, the other thing that we've got is, right. Especially for those of you who have Macintoshes, it's getting hard to make text files on Macintoshes, because when you open the text editor up, it wants to do rich text for you, which you don't want, usually, if you're doing separate computers. So, sometimes you have to just make yourself a nice seed text file. So here's "Right File 1.txt".

[45:25] And now that I've done that, maybe--oh, it doesn't--let's look at all files. Oh, we don't want to do that. Never mind. Sorry. I'm using an operating system and now I have to remember how you see files in this thing, which is you go get the little finder who-ha. Tada! "File 1.txt". And now, we have a text editor. This will work on your Macintosh, too, or your PC, which has the stuff that was in the table.

[46:12] And since this is a text file, you can now read it. You can read it in the MATLAB or Octave, or what else? Well, you can dump it into HTML, or whatever it is that you want to do with it.

[46:25] OK. Similarly, now if I want I can put new values in here. For instance, if I wanted to I could...Well no, I'll just do it. It doesn't matter whether they're on different lines or not. See, we had 12 numbers in the table, so I'll just give it 12 numbers. It does the right thing if you have the wrong number of numbers. And now, we go back to PD. Where was PD? And say, "Read it." And there's the nice table I just edited. Yeah?

Student: [47:09] Does that stay as text file while [inaudible 47:11] ?

Professor: [47:13] Ooh, thanks. Yes. Well, you could give it a path. So, you could say, "Read ../blah/File 1.txt". And, yeah. But for right now just make everything be in the same directory. Also, you can give PD a search path. So, there's directory stuff to worry about. [47:37] One thing that will come up and bite you is that when you make a new patch, it doesn't know what directory it's in. And as a result, it might not be able to read a file that is in a directory because the patch itself before you save it doesn't--isn't in that directory. It's in no directory at all.

[48:00] So, you might have to save your patch just to tell PD what directory you want the thing to believe it's in, so that you can read files in from that same directory. All right. OK.

[48:24] So, you've seen send. You've seen pack, you've seen message boxes, and you've seen files. And this is probably enough PD bore for a day. So, I should show you some more stuff of relevance in music for a while. I know, I haven't been making a whole lot of noise yet.

[48:42] What I want to do to that end is start out with the assignment for next week, showing you what it is. It's on the web page. But the web page doesn't explain it very well. And so let me explain it better.

So, I'm going to save this. So, the near-term plan... [indecipherable 49: [48:58] 14] . So, where did I put the...Wow. I lost my terminal window. OK.

[pause]

Professor: [49:38] And... [pause]

Professor: [49:47] Oh, great. I'm running two PDs. So, I had it miniaturized some how. I didn't know it. What's going on? Oh, and I had this terminal. Oh, I see, I've miniaturized down to... Sorry, I'm getting myself confused here. OK. Now, I'm no longer confused, maybe. [music]

Professor: [50:09] This is... OK. You know how to do this. Here's a cool thing. Let's see. Turn it up. This is an idea that's attributed to Steve Reich. Am I pronouncing his name right? Steve Reich, who invented the idea of having two tape loops with sounds on them, that had slightly different durations, and then playing them both in a loop, so that you would hear them getting phased differently with time. [50:40] And then, OK. So he did that in a series of famous pieces in the 60s, the first of which I think is called "Come Out," which you should probably check out if you haven't checked it out. And then, he started writing these things like that for instruments, in particular, this one called "Piano Phase," which orders two pianists each to play a 12-note sequence at very slightly different speeds, so that they would phase gradually.

[51:13] And you would hear them juxtaposed at different phases essentially. It's the same kind of phase as the phase in an oscillator, except it's being used differently because it's a phase and a melody. Well, we can do that here.

[music]

Professor: [51:34] Is it doing it? Oh, yeah. I'll let this go for another round or two. [music]

Professor: [51:55] All right. There's all sort of fascinating stuff that you can do with this. Of which I'll show you just a little bit. One thing is, and this idea I believe we do today with Wessel. I'm not sure, I heard about it from him. You can make melodies, and this is the extra credit. You can make melodies that have two different timbres as well as a repeating... Actually, this doesn't sound as good on these speakers as it did at home. [52:38] By the way if I were doing this for real I would probably work on getting that better run. I'm doing this only with objects that you've seen so far. Turned out to be tricky to get it to be perfectly clean, to have some of your objects in your arsenal. But what's happening now is there are two different tables of pitches...

[music]

Professor: [53:01] And what you should hear is the same, what's the word, same series of pitches as you heard here. Except that every third pitch has a different timbre. [music]

Professor: [53:22] So here's the title sequence. All right, so you can also hear... If I slow it down, then you just hear the same as a sequence. You can't actually hear the other point of view on it. Which is that you can hear every third pitch as a separate stream. That only happens at particular speeds. This is the effect that Wessel I believe discovered. So now if I speed it up at some point you quit hearing that as a sequence going at this speed, and you start hearing it as two different things. One of which is the sharp timbre, and the other which is the sinusoid. [54:18] All right, the original David Wessel was three pitches. That.

[music]

Professor: [54:34] So it's going down. It's three blind mice now, right? No it's not, it's going up. It's "Do-Re-Mi." Do, re, mi, like that. Actually, for some reason in this register the sinusoid doesn't sound like it's the same octave as the other ones, but I guarantee you it is. And then if you speed it up, instead of hearing the thing rising like that, you'll hear it going down. OK. It's like now you hear two different melodies which are catty-corner, right? And... [music]

Professor: [55:29] So now, if you're good at math, you can figure out that choose any two relatively primary numbers, the number of notes in the melody, and the interval that you change from timbre A to timbre B, and then you can pick out, basically, permutated versions of melodies that exist within themselves. And then, yeah?

Student: [55:52] What does modulus mean?

Professor: [55:53] What does modulus mean? Well, OK. I actually don't know what the word means in fully general, but in mathematics land modulus means the range of a set of numbers that repeats.

Student: [56:08] OK.

Professor: [56:09] So, if you say five is equivalent to two modulo three, in that statement the modulus is three. I don't know of a better way of explaining it than just by complexifying it like that. OK. And, of course, you can do this with...or can you? I don't know. Perhaps, it isn't working today. Yeah. You should be able also to hear that phased. But I seem to have forgotten to connect something in the patch to let you hear that. So, here is the basic [indecipherable 56: [56:52] 54] phase. Let's do this. And then if you want to make it hard, do that, except that every third note should be a separate timbre as well. Then you'll hear really wonderful stuff. All right, that's going be about that. Yeah?

Student: [57:17] Is there another patch behind?

Professor: [57:19] Oh, is there another patch behind here? I didn't tell you this. If you're working on a patch and you run out of room, you can just say "PD" and either give it a name or not. Usually I give it a name just so I can tell what's what. And up will pop another window, and this window is inside this box. [57:41] And this is what you do. I've been avoiding doing this because I don't think you should be using patches this complicated, that you need this yet. But this is what you do when your patches get too complicated to hold on one screen. You start encapsulating parts of it into sub-patches. And then when you close the sub-patch...oh, let me put something here...when you close the sub-patch, it's still there in this box. And you could see it again by clicking on it.

[58:16] So, if you look at a real piece of music realized using either Max or PD, it will typically have hundreds of windows in it. Because all this easy stuff I've shown you so far fits in a window, but by the time you really want to do something specific and you want it just so and you want to have voices and so-on like that, you're going to have windows on top of windows on top of windows. This is the way of managing them. Yeah?

Student: [58:41] So, is there a reason that the PD no longer has six different connections on it [inaudible 58:47] patch itself behind the scenes right there [indecipherable 58:50] ?

Professor: [58:50] Right. Right, and the reason the patch is behind the scenes is because it implements the homework. [laughter]

Professor: [58:58] In fact, there's another reason, too, which is that I implemented the homework in an exceedingly messy way because I made a lot of mistakes and got confused on doing it and kept changing my mind about what you should do. And so it changed a lot of things besides the homework that I want to go back to later. But I don't want to show you right now. [laughs] Yeah?

Student: [inaudible 59:19] [59:17] text file that you did earlier?

Professor: [59:21] Yeah.

Student: [59:21] Is there a limitation on how long the patch can be before it doesn't work?

Professor: [59:27] I think it's limited to 4,000 characters. [inaudible 59:31]

Professor: [59:32] Should be OK. But message boxes don't like to be more than 80 characters wide, so that it will automatically wrap the file name in the message box. There is a thing which I should have told you about. If there's a space in your file name, PD will think the space means that you have two different file names. [59:53] Yeah. So, spaces in file names are evil as far as PD and Max are concerned.

[inaudible 59:57]

Professor: [60:00] Yeah. Yeah?

Student: [60:03] Do you have a patch file on here that shows how complicated this can actually be?

Professor: [60:10] Oh, yeah. [laughter]

Professor: [60:15] Here's the patch. This will take a moment to load, but... It just did. [laughter]

Professor: [60:32] Yeah. This is a piece that I'm working on with a composer. I'll just talk over this while it struggles to load. Oh, here it is. So here are a bunch of control panels. This is an effects processor and it has a frequency shifter with a bunch of parameters. It has a bank of cone filters. It has all kinds of stuff: [60:44] a phaser, noise generator, phalanger, harmonizers. OK. Then meanwhile, there is a sampler bank and there is a thing called Jack. It's just called Jack because it's a name. Then, Eric, this is talking to the sinful synthesizer which you have upstairs if you want it.

[61:18] The phaser line formula generator which is a synthesis technique that has a bunch of parameters in it and has a bunch of voices in it. I haven't told you yet, but number of boxes can have sends and resends built into them, which you need to do if you're going to make something this gnarly. [laughs] And how is it implemented? ...Oh, there's a mark-up change that can drive anything that you want.

[61:42] But anyway. This machine will not be able to make sound lists match, I'm afraid. It makes great sounds, I'll tell you. So, here's the actual patch. Here's the sampler. The sampler has a few voices of samples, and this is a spatializer that sums the first four voices into one place. Then these get spatialized separately and so on. The sampler itself looks like this, so there's a bunch of these guys.

[62:23] Where's the control thing? Here's the unpack and pack for you, I use that a lot. Here's polyphonic voice allocating. Fill one like that, route.

Student: [62:33] You did this all in an hour?

Professor: [62:34] No. I did this in a couple of years. Inventing all of this stuff is [indecipherable 62:41] . Hardcore stuff.

Student: [62:43] Can we get something [id 62:45] ? [laughter]

Professor: [62:46] Nope. You know, if I drop the sample rate to about a kilohertz I'd be able to do it on this processor. But this is an atom processor and it needs Core two or so to run. Yeah. Then there's other stuff. So, we can make arbitrarily complicated patches.

Student: [63:16] Is there a montage [indecipherable 63:19] ?

Professor: [63:19] Yeah. That might happen later this quarter or it might happen next quarter. Most likely next quarter, but that depends on how things flow. You can make various changes to casting processes, and people do use random computer music all the time. Back to "Floor With Tables" and whatnot. Now I want to find the patch I just had. I must have closed it. [pauses] Where was it? Oh yeah, I know. [64:19] OK. So, now, just to try to tie things together a little bit, I'm going to go back to using tables as wave forms and not as pitches. Before I do that, let me show you something else. On the level of glue, there was something important that I didn't tell you. So, suppose I wanted to put values in the table, but suppose I didn't want to type the values out in cycles per second. Instead, I want to specify a melody and pitch. That would be a good thing to be able to do.

[65:01] So for instance, suppose I want to...Let's see what's the easy way to do this? Yeah. Trying not to introduce too many different objects here. So I had to do it in this way because this is going to be the easy...well let me show you. Let me start doing it, and then try to explain what I'm doing as I'm doing it, because I can't figure out how to verbalize it without destroying it.

[65:33] So, the first thing we're going to do is just say, OK, we're going to put a value on the table, and the value's going to be some number. And it's going to go into slot zero. And then I'm going to have a number box that puts the value in. Ugh. Now I've got this thing going.

[65:59] So now what you can see is this first value on the table has gone up and down, but then--what did I figure? So the message now is 617 and zero. Zero's where in the table? 617 is the value.

Now you can sort of see why tab right takes the Y-value, that's to say the value/value here, and the X-value in the other inlet. It is exactly so that I can do this. And it changes the value of the table instead of maintaining a single value and splashing it across the whole table, which is likely to be a lesson [indecipherable 66: [66:11] 32] .

[66:33] There is a way to get it to do the other thing if you want, but I don't want to show that to you just yet because it's more glue. And now of course, if I want to talk to the next value in the table, I just say one here, and then I'm changing this next value. Oh, wait--connect it. OK, so this is all clear, and now...Yeah?

Student: [inaudible 66:56] [66:56]

Professor: [67:09] Except, that in fact it won't do anything. So here--so I'll say, "Put in 550 please, and put it in location zero." And I'll say, "Put something in location zero." Then I have to bang this, or maybe resend 550.

Student: [67:28] Oh!

Professor: [67:30] And, yeah, there's a thing I have to tell you, which I've been putting off, but now that you've asked, maybe this is the moment. You can...OK, here, I'll do it the wrong way first, and then show you why it's wrong and then do it right. So the wrong thing to do is, I have to introduce another object to do it right. So the wrong way that doesn't require introducing an object is: [67:48] 550, it's a message; I can throw the number in there. And then I can say...

[68:09] Oh, that one. Oh, yes, thank you. So I can do nothing. OK, zero, one, two, three, four. OK, that's good. It's not really quite right, because what I haven't done is I haven't actually verified that the six goes in here before the 550 goes in there. If the 550 goes in first, then it writes 550 to whatever this value had been before, and then writes the thing in afterward. And that would be wrong. I'll make it clearer this way. A sort of textbook example of how to do something wrong.

[68:46] Let's take a number and square it by just using a multiplier. And now what I'll do is just multiply the number by itself. Now I'll say five, and five squared is five, actually I don't even know how that happened. But then six, and then I get 30. Why? Because you don't actually see it from the patch, but just from the way I connected it, what it's doing is it's sending the six to this inlet first. And multiplying six by whatever is in this inlet, which is five.

[69:30] And then it's hiding the evidence by putting six in there. Right? And now, no matter how many numbers I put in, it's going to do it wrong until I happen to put a number in twice, at which point it'll do it right.

[69:44] All right. This is confusing me bad. To make it good, you need an object which will actually force the order to be a particular order. Which is to say you want to put this value in first before you put that value in. And the object that does that for you, this is object number six for today, so I'll apologize, I think. This trigger, which, it's named trigger just because that's the word that Max Matthews used to use for doing something.

[70:27] So, you can think of this as a distributor in an automobile, if you're the type of person who fools with automobiles. And what it does is it takes the value as a floating-point number and outputs it here and then outputs it here as another floating-point number. And again, I specify float or bang if I just want to convert it just to a bang message, or other stuff that I don't have to tell you yet. And now, I've got something that is guaranteed to do the squaring the right way no matter what order I connected it in. Trigger.

[71:02] This is so useful, it's so useful and so long-winded that you can also just say trigger, float, float like that. And this is what you see more often. So, here, what I want to do, really, is I want to take this number six and I want to put it in the pack. And then I want to send 550 in the other inlet. And now I've got the thing in a way that it will work correctly. This does the same thing as the network that I had before except that before I had this number box connected both to the 550 and to the zero. And he didn't know that they happened in the correct order, which is to put it in this inlet first and then put 550 here.

Student: [inaudible 72:02] [72:02]

Professor: [72:06] Here? Yeah, yeah, yeah. So you can say whatever you want. Here would be a better style to say give me a bang out this one. So this is the type of this outlet. And you don't need the floating number because just the message box will just convert it to the number 550 anyway. And so here you would just send a bang out instead of a float. [indecipherable 72:40] . [72:43] So this is the way to get something into an inlet and then make the object do something anyway. Whereas if you just knew that this guy was zero and wanted to control this number with a number box, it's the much easier situation to deal with. Which is that you just take the number box in the pack like this. So this is the easy case, where you're changing this for the number box. This is the hard case where your changing this for the number box.

[73:34] Sorry, this is much too much lore for one day. But lore is good. Rather, lore isn't good in the sunlight, but knowing how to do this kind of stuff is good. Now I have to clean this up for you and put it on the website, so you can look at this in fullness of time. Questions about this? Yeah.

Student: [inaudible 73:55] [73:54]

Professor: [74:07] The trigger object always sends messages to it's outlet in right to left order. And it's right to left and not left to right because typically you want the left-most thing to arrive last for the object afterward to work right. Yeah, I should have said that, that was a very good question. [74:32] OK. Now, that I've done this, I can go back and remind you of the wonderful midi to frequency object. Now I have a sequencer that I can control using midi numbers. I'm running out of room here, but I guess I'll do just as much as I can make room for.

[75:10] We're going to have a seven note sequencer. [laughs] Yuck! OK. Got five of them. All right. Now, the only difference between these is that this one is going to go to cell zero, this is cell one, this is cell two, this is cell three, this is cell four and let's just have a five note sequencer because I'm getting tired of this. Make my five. We'll go one inch per second and listen to it.

[music]

Professor: [75:46] Whoops. Yeah. So now, that's a good melody that has five notes in it. [music]

Professor: [76:02] All right. [music]

[laughter] [76:03]

Professor: [76:17] All right. Now we're making western temporary melodies on the table. These numbers, now, are the frequencies in hertz of the pitches which I specified in midi in these number boxes here. Yeah?

Student: [76:36] When you start, can you do 127 values and if you want to just have it go straight through a [inaudible 76:42] midi and input a midi's frequency number [inaudible 76:51] ? I'm sorry. Instead of storing the rate of frequency, can you store in midi?

Professor: [76:56] Oh! Yes. Of course. Oh, yes. OK. You can do that, but notice that I'm reading it. I would need to use M to F tilde because the output of the table has an audio signal. That would be another way to do it. But, of course, then you're doing the conversion of resample, which is more expensive computational, whereas, this is cheaper.

[77:21] Everyone's rested. [laughs] I can see why. There was a lot of very dry detail today, for which I apologize. Yeah. Next time we'll get back to samples and tables, and storing sounds and tables and using oscillators to get them out. Anyway, that's a bad choice of a pitch, isn't it?

[77:55] That's nothing better. [audio ends]

MUS171 01 25

Puckette: [0:00] ...Which was supposed to be a five minute little, oh by the way... So it really, actually works. So, you can slow the entire syllabus down to a crawl just by asking questions, soon as you hit things that you don't know. If the syllabus is going too slow for you, I don't know of any way that you can make me accelerate. I'm going to be working up to six new objects today if all goes well. But one of them that I just thought of last, I'm going to show you and then not use because I just want you to know this exists. It will come in, in its own good time. But you will need it probably before I need it because I know in advance what I'm going to do, and you don't. So, it's more complicated than my work, all right? [0:48] So if in an object box you type PD, then you get a thing, and when you click on it, you see a sub-window. And the sub-window is yours to put anything that you want in. And it will stay there. And it'll be a part of the patch. If you close it, it doesn't actually go away. It's still there and still doing whatever it was doing before, so it's a part of the running patch. But it is a part of the running patch that you don't see. The reason I'm going to be reaching for it probably first off, is going to be when the table...

[1:24] When I have too many tables running around and they start crowding out the functional part of the thing. So a very normal thing to do is take out all the tables that you are using, the arrays, and stick them in a sub-window so that you don't look at them all the time. But you can look at them whenever you want to check them out. And it's a single click. So I'm trying to be good to people's wrists. OK. So that's all...Yeah.

Student: [1:47] Well, is that something where names will get messed up if you have multiple patches going?

Puckette: [1:54] Only in one very crazy sense, which is that you can actually send a message to the window, and do something like open itself. And in that case, it will actually pay attention to its own name. Otherwise this name actually only functions as a thing that it prints up on the window, so you could tell what's what. And especially since your window manager can tell which window is which on the bottom of your screen, which is useful. So it's a good thing to name them, although you don't absolutely have to. And name them something that actually has something to do with what you want to put in there, so that you can find it later. The other objects, let's see. You might have already seen this and you might not have. There is a thing that looks like a button but isn't. This is actually a kind of number box. I'll use it to turn metronomes on and off pretty soon. But the basic trick is... Let's see. I'm going to make a window and start showing these things off. OK. So, this thing, which you get on the put menu under toggle, is a kind of number box, which makes numbers that are one and... Whoa boy! Which are one and zero. So one and zero is computer language for true and false. Most people who studied Computer Science learned that true and false are constants which have a different from one and zero. But PD doesn't use types so one and zero are either used as [indecipherable 03: [2:17] 26] or as logical values. In fact it's even better than that. Any number that is not zero in PD like four times ten to the minus seventh. No point four times ten to the minus seventh that ridiculous number is actually true because its non-zero. Yeah.

Student: [3:48] So when you use E [indecipherable 03:50]

Puckette: [3:52] Oh you can, I think you can use E. There let's see I think this is about it, yeah it typed, oh, It will read either but I think it will represent using lower case. And most computer languages use either lower or upper E interchangeably.

Student: [4:11] I just wanted to make sure literally I know where there's no [indecipherable 04:15]

Puckette: [4:18] It depends on the language like big E also could be energy so this is four times ten to the minus eight if you ever express a number like that use exponential notation. [indecipherable 04:35] . Use it. And by the way even though the value true when you say it is one. Hello, that is horrible. I didn't write the subject. It should say one when you re-click it which it doesn't. So now I have to fix it. Very good, alright, I didn't know that. Alright. So they put, it transmits whatever value you put into it through which is correct behavior and then when you tell it to be on it should probably say one and instead crazy value like that. Yeah.

Student: [5:03] Is the negative number zero or [indecipherable 05:04]

Puckette: [5:05] The negative number is also true. Hello. Right, I can't type into that one. I have to get into edit mode. Yeah, so zero is the only falsehood. It is a misconception that people get program when they start using floating point numbers when they can't hold values acurately. Like the integer five if you express the floating point number is not exactly equal to five. Five is equal exactly to five even if it's expressed as a floating point number. However .1 is not equal to .1 when you express it as a floating point number because it becomes a repeating decibel in binary and then it will get truncated. But you can represent integers and then it will get zero exactly using floating point numbers and not get, not expect a truncation error. [5:58] Integers do not get truncated below values of plus or minus eight million. After that they do because there's only 24 bits of mantis in a number. And go study computer science if you want to know more about that. OK. So there's the toggle switch which is by the way excellent for turning metronomes on and off. Which I think I've shown you I don't remember if I used the toggle, I don't think I did. So here's a nice metronome and this leads to my next topic. So put a nice bang so we can see the metronome going.

[6:38] All right. And then, so this is a combination of objects that you see all the time. And of course you could use the number box, but then you would be obliged to scroll the number box up and down to re-start the metronome, which is ugly, whereas this is appropriate. All right. I showed you that so that I can show you this.

[7:01] I'm about to call attention to the central source of confusion on all of computer music. Which I've by the way visited a couple of times in the past, and we'll have to visit again a few times too, which is the distinction between doing things with messages and doing things with signals. So, for instance, one way that I've taught you to count to four, or to count to some number, is using signals. I'll do it over here, make a Phaser, let's count to five, for reasons I'll tell you, I'll explain later.

I'll run it at one hertz, and then I will multiply it by five, and then this will generate a Saw tooth wave which ranges from zero to just below five in value, and then I can use that for instance to read values in a table, which I've shown you how to do. So for instance, let's make a nice array. And it will be called... oh, boy. OK, Table 125-A. Oh, I was typing in the wrong place. OK. Call it [indecipherable 08: [7:39] 18] points. Oh, right, and then it forgets those points, so I have to say it again. You are, whoa, that was interesting. You have a row of points, and you have only, let's say five of them now. Did that work? That did not work terribly well. Ah, it worked well enough.

[8:38] OK. Little fence post bug. It didn't really leave room for the last one, did it? Let's fix that. Ta-da! OK Table. All right, and now we can do something like, get those values out and use them as the frequency of a tone that we'll listen to. So, for instance, let's do a tab read, T... January 25th, first try. Oh, right. Tilde. Because we are signals, this is the thing that I'm trying to warn you not to be confused about. And now we've got something that we can drive a sequencer with.

[9:26] All right. In fact, I won't go further than just to say, just to do that. And one thing about tab-read tilde is that it's truncating these things to integers, which you could do in a variety of other ways, but the table is the thing which truncates to integers that you've seen, which is why I hauled a table out just there. All right? Next possibility. Maybe it would be interesting to do this using messages. That would mean rather than each time the oscillator tells you it's time for a new thing that is going to happen regularly. It waits for you to tell it to do the next thing please. According to some other some other source of time or some other source of events.

[10:08] Such as this source right here which is a metronome. But it also could just be me doing this by hand. Right. Well you can do that and the way you do it is this. This is one of those paradigmatic ways of programming in PD that is particular to PD or MACs and this won't help you do anything except survive PD and MACs as far as I know. This is not actual knowledge this is just cleft, OK. So what you do is, so were going to make a counter and it's going to count off zero one two three four, zero one two three four, like that. Which is therefore repeating with a period of five.

[10:50] Had you, well the first thing you need is a number. And numbers are held at, well I haven't even told you about this object yet. Have I? You can have storage. OK. You've seen a lot of storage in PD because when you say plus five or plus which has an inlet. That inlet is a storage item. And so a lot of objects such as plus or the oscillator are perfectly happy to store numbers for you which are the parameters or the arguments of their operation.

[11:25] However sometimes you need to talk about storage explicitly. Which is indeed what is going to happen right now. And so we need to have a thing which just exists for storing numbers. This also could be a, no. I'm assuming it could be a number box but I think it just has to be what it is. For what we're going to do next. And in fact now I have to apologize because our object count just went up to seven. Maybe we could, oh yes, if we didn't mod two. So your new objects are going to be numerous today. So I'll hold off on inflicting new objects on you next time.

[12:05] So now what we're going to do every so the oh, what does float do? Float does this. You put a number in and it says thank you very much I know my number. This is acting exactly like a regular inlet acts. And then when you say bang on this side it says what was the number please output it. Alright, clearly a very useful object. You could almost do this with plus because you could actually send a bang to plus and as long as the first inlet thinks it's zero then it outputs the second because it will add them. But that would be confusing to look at because it looks like it's an adder and it wouldn't be so it's better to actually use the object which is explicitly just doing what we're using it for, which is storing a floating point number therefore called float. Yeah?

Student: [indecipherable 13:09] metro so that [indecipherable 13:10] [13:06] float?

Puckette: [13:11] What will happen then is, let me do another thing which is put a flasher here so that you can see when it's actually outputting because 53 is just 53, but if I send more 53's you won't see anything unless I put this bang here. So now what's happening is the bang is going to flash whenever it gets something and this is going to be the number it actually got, which might not change, right? [13:34] So what happens when we hit this with a metronome? It just says 53, 53, 53, 53 five times a second. Could be useful. But, let's see, here's something that could also be useful. Let's take and every time a number comes out we'll just add one to it. Now we have something cool. We've got the genesis of a programming language. If you can count you can do stuff like, well, you get the idea.

[14:12] Alright. This is a loop and notice a few things about this loop. First off, this is uglier than just writing a loop in C would be because when you write a loop in something like C or any programming language that's based on text you actually see the steps of the loop in the order they're supposed to happen in because programming languages tend to go down, statement one, statement two, and so on.

[14:38] PD doesn't work that way. PD goes along the lines and so you have to do a lot of extra nonsense, which is to actually sew the loop together into a loop, alright? So this is actually not an efficient a way of describing a loop as you would have in C or something like that where you just type eight or nine characters and you've got a loop. OK. Another thing is, just to tell you this, usually when you do this you don't bother with this, you just do this.

[15:08] Oh, by the way, I disconnected it so now it's just 399, but now it's counting again. It's just counting in a way that uses the property of float, but doesn't include the little thing I put in to demonstrate the float.

[15:24] This is better because, as you might have already noticed, the more stuff you have flashing or drawing numbers on the screen the more your CPU time is being eaten by just updating the screen, which, of course, is a useful thing for knowing what's going on, but is also competing for your CPU time with the thing that you're actually trying to do which is make beautiful music.

[15:50] So you don't need to have unnecessary boxes and, in fact, it's better not to. To a point and, by the way, if you do want to have a lot of unnecessary number boxes throw them in a sub-window and keep it closed and then PD doesn't have to draw them and then you won't be eating the CPU time after all. That's an important clue to, OH my patch, which covers five screens and you have to scroll in order to see it, is starting to run slow.

[16:19] Well, OK, put a bunch of it in a sub-patch and keep it shut. Then it won't be drawing the stuff and then it won't run slow anymore, at least for that reason. Alright, so now, the first observation about this is this is a bad way of making loops if you're used to using programming languages, but it's what you get in PD. There's no programming language here.

[16:41] Second thing is notice that it actually looks like a tree, even though it looks like a loop to the untutored eye. This connection is going into the inlet of float, which doesn't cause float to do anything, so that when a bang comes into the float the sequence of things that happens is, in some order or another, the number goes out these three lines, one of which is this line which gets added one to it and that gets put into the float.

[17:17] That does not affect what the other two people got because this message was output and it's there and the float, when it remembers its new value, will use that to change anything that it might do in the future, but that doesn't affect the rest of what happens as a result of passing that message out of its output. That would be re-entrancing or recursion. You can do recursion, but don't.

OK, so the reason this works and doesn't give you an infinite loop is because, in fact, this change of operation actually stops there. If I want this not to work, save this just in case [indecipherable 18: [17:51] 11] . OK, we saved it so now I don't feel bad about doing something that might bring PD to its knees depending on the OS. Here, let me take that other line out and just hook it up here.

[18:24] Then we get error messages about stack overflows. OK and that does horrible things there. That doesn't work because then the float says add one and the add one says OK do it again and so on and every time the instrument bangs it tries to add one to it an infinite number of times. It actually only got up to some number of thousands before it just said give me a break and stopped, rather than allowing my PD or even the operating system to crash as would have otherwise happened.

[19:01] Oh yeah, and now that we're here loops are great, but the first thing you learned in computer science about making loops is there are three steps to the loop. There's the thing I haven't done. One of the three of which I've done, which is updating the thing each time around the loop. I haven't told you how to start it and I haven't told you how to stop it.

[19:21] In PD things aren't started and stopped so much as they're always sitting there latently ready to do one thing, whatever it is they do. They're waiting for events to turn them on, so starting and stopping the loop doesn't make sense in PD the way it does in a computer language. But initializing variables, which is typical of what you do at the beginning of a loop in a real language, is a useful thing to be able to do and you do it like this.

[19:43] You say give me a value of zero please and you put it right in where these numbers go in. Of course, when I whack this zero it's going to happen between two of these bangs because no two things happen simultaneously even though they can happen with a difference in time of zero. So time works in, what's the right word? I don't know the right word. Time works chunkily in PD. So this will set it to zero and then the next time it gets banged out will come zero here. Oh, right. It starts at zero, this loop, or it starts at whatever you put it. It doesn't increment first, it increments after. Alright, there's a loop. Suppose you want it to count zero, one, two, three, four, zero, one, two, three, four, and so on like that instead of doing this thing.

[20:38] There are two ways to do that. There's the way everyone thinks of first, which is to test whether the number equals five and if so to bash it to zero. That is not going to be so great because you're going to see the value five and then the value zero unless you're very careful. So let's do it the smart way which is simply to ask for the remainder after dividing by five.

[21:03] By the way, I've disconnected this so it can't update anymore. It's stuck. Now I'm going to say how about mod five. Here's the mod object coming at us. Now we have our wonderful counter that goes from zero to four like that. Now, in fact, if we want to we can do the same operation as here except now we can actually use a tab read without a tilde and look at it.

[21:38] Now we're looking at the five values of the table and I didn't set the range of the table to be, you know, sequence or type range so it's just what you're seeing. Yeah?

Student: [21:48] So that's all it is, is just [indecipherable 21:49] , mod?

Puckette: [21:53] Well, mod itself is an arithmetic operation. What it is the remainder that you get when you divide the left inlet by the right inlet. It works in this case because zero, one, two, three, and four give themselves, but five give zero.

Student: [22:10] Is it like modulus in Java?

Puckette: [22:12] Might be. I don't know Java, but could very well be the same thing. There's only one correct way to do it. It's almost like the percent sign operation in C, except the percent sign operation in C is wrong because if you say negative five percent three you should get plus one and instead it gives you minus two. So mod actually does the mod thing, which I hope Java's modulus does, but I suspect it doesn't. If you feed it minus one, minus one divided by five gives you a remainder of four. Why? Because what's the biggest multiple of five that's smaller than minus one. Well, it's minus five. Then you have to count up four to get to the actual number, minus one, so the remainder is four. Alright. That's mod. There is a mod tilde. [23:29] Oh, yes. This along with tab reader and along with plus come in control versions and they come in signal versions and there are times when you want to use the one type of thing and there are times that you want to do the other.

[23:47] Advantage to doing things in control land. Advantages, I can think of three right off the bat. Potential advantages for doing things the control way. One is that it is much easier to do things irregularly. For instance, if you want to count incoming network packets using net receive, which I won't tell you about. Well, they come in and you might regard them as an event, such as an event or a bang. Me, I'm clicking on the thing. I want to count mouse clicks, same deal. You don't know when they're going to happen, you don't know if they're going to happen at all, and you don't know how many you're going to get. That's the sort of thing you can do effectively using messages and less effectively using signals. OK.

[24:43] That's advantage number one. Advantage number two is that you can actually hook the things up to a number box and see them, whereas signals you have to work harder to see what they are because they're going by at 44,000 units per second or something like that. So it's not feasible just to put it into a number box and look at it. You either have to say print tilde or throw it into a table or an array and look at the wave form or something like that.

[25:10] So it's much more convenient to debug stuff here. Third thing is it's much easier to do things conditionally or, rather, it's possible to do things conditionally I should say. Signals are always happening, but messages, I haven't shown you the primitives for doing this yet, but you can do things that can take messages and optionally respond to them or not depending on whether they obey certain conditions or other things obey certain conditions.

[25:45] That's a good thing to be able to do, which is, again, central to what computers know how to do or how people think about programming computers and which is much more appropriate to this regime of messages than it is to this regime of signals.. So why bother with signals at all? Well, so that you can do computer music because computer music provincially you have to make sound and that's signals.

[26:12] So this world is less flexible, but it is the correct place to be if you want to do something like have an oscillator. There is no OSC that doesn't have a tilde. It doesn't make any sense really to do that in message land because it depends on having a sample rate in order to know what to do.

[26:35] Questions about this?

[26:40] So this is what Luke looks like and some way of thinking both in message land and in signal land, and now that I've told you that, I want to go back... well, back sideways and start looking at tables again as sources of wave forms because it is time to start doing sampling.

[27:04] Sampling is actually no different from wave form oscillators except psychologically, but at times there are things that you want to do with samples that are more easily done using messages than using just a Phaser that grips through things, including I should say the next assignment, which I don't have the heart to show you because I don't think many of you finished homework 3.

[27:29] I'll show you homework three with the extra credit done correctly this time, at least an output so that you can see what you should be listening for, but homework four, at least do the extra credit thing. I think you are going to really want to do it with messages and not with a steady going signal because there might be decision making to do.

[27:52] All right. So this is what this is, and I'm going to save this and then immediately start doing sampling.

[28:09] OK. So to do sampling, you will need some new objects. In particular, well, yeah you are going to need these too. The sound filer, I don't want to show you just yet how to do live recording in the PD because that's going to be another thing. It is going to be easier and anyway more urgent to learn how to get sound files in and out of the race.

[28:35] I've shown you now I think four ways of getting stuff into a race including reading from the text file, but of course when you're doing computer music, you're going to have these recorded sound files that will be in wave format or AIFF format or whatever it might be, and you're going to want to get it into your race so that you can have them as wave forms so that you can play them as samples.

[28:57] So that is a fundamental thing that you might wish to be able to do, and so I'll show you how you will set about doing it. So let's save this.

[29:06] Actually both before and after I saved it. I want you to, enjoy the fact that it does live in a directory which here is going to be my Linux-style home, and then SP and whatever. And this directory is going to be where PD looks for file names. You just saw when I used the read message to...arrays, in order to read a collection of ASCII numbers, or text numbers into an array. And the same consideration will hold here. The directory that I am in, I've already moved...sorry, I've already moved the sound file into the directory that we're in right now.

[29:52] Which maybe you'll see if I say open, except it only shows PD. Yeah I can't ask it to do all files. You won't see that. I could go figure out using the finder how to show you the directory contents. But anyway, let's just look at it.

[30:10] Alright. So we'll make an array, as we always do, as we often have. I don't know why I bother with that. I'm going to say, let's have a 100,000 points. That's over two seconds of sound. And it going to be table, January 28th. Blah, blah, blah. OK. And of course, there's this bug where it forgets that I told it...that I wanted points. Don't know why, alright.

[30:44] Now, read sound file into is done by a separate object. You don't pass a message directly to the, array. Instead, you make an object which is called, Sound Filer. And you send it a message, which is Read Again, and it's going to have the name of the file...Whoa, there's a dot there. That's the name of the sound file that I've got. Then it needs the name of the one or more arrays that I'm going to fill with...the contents of the file. And, 1.24. I forgot the T, don't need the T. Tada! So what I did what I just loaded the voice.wav thing in here. If you want voice.wav, it's in PD. For you Macintosh users, you option-click on PD and show package contents. And go looking in the documentation, there's a thing called voice.wav. Which is this file. Which is included in the PD distribution, just so that some of the example patches in the PD documentation will work. Also, so that you can have a sample sound file to do stuff with as you start to make patches, before you've worked out you're microphone situation.

[32:16] The microphone situation is kind of miserable for people who own laptops. For reasons that you might already have discovered. For instance, you can't type and record at the same time because typing makes a very loud noise that goes to the microphone and things like that are bad. If you want to do stuff with microphones, give yourself some time to figure out what kind of microphone you need and buy it and how you're going to have to connect it to your computer. It's not going to just happen tomorrow. Sound filer also takes message to write the thing back to a file.

[32:58] It likes WAV files, which are the closest thing there is to a standard PC in file format. It will only deal with PC in files. It doesn't know how to do MP3s and stuff like that, which are data reduced using proprietary algorithms. Well, you can find out what they are, but who wants to write the code to deal with that? And it will also deal with .AFFI files, although AFFI files, I'm frequently confronted with AFFI files, in one sort or another, that where they changed the header a little bit and PD cannot deal with it anymore.

[33:32] So PD is not as successful reading the AFFI, which are the Macintosh files, as it is reading the PC files. So if you can't read your file in just convert it to a .WAV and it'll be perfectly happy to read it for you.

[33:52] The array has no idea what sample rate the thing is that it holds and, with that in mind, sound filer actually ignores the sample rate of the file. A frequent beginner's error is to go off on the web and find some nice file of a dog barking or something, download it, verify it as PCM, but not notice that it's eight kilohertz. A lot of files on the web are eight kilohertz. Then if you try to play it at 44 K1 it will sound some octaves higher than where you wanted it. Yeah?

Student: [34:28] Can you go to the properties of that?

Puckette: [34:31] Properties of that. Oh yeah. I should just sort of systematically do that, shouldn't I? So what I've got is Y is ranging from minus one to one and 100,000 points. So if you're operating at 44 K this is a little over two seconds. The reason I gave it that number was because I happened to know that the sound file it's going to read in was a second and a half or so, so that was kind of a reasonable number of points to give it. [35:05] Oh yeah. Let's get rid of this. Oh yeah. Save contents. If I save the contents now there will be 100,000 numbers in the patch. The patch themselves, I don't know if you noticed, but they tend to be less than a kilobyte at this stage of the game, so this would multiply the size of the patch by more than 10. So when I'm using sound files I don't save the contents. Oh yeah, let's save this.

[35:42] This is going to be three dot sampling one. You still have many sampling patches. Sampling is not just one thing. So now when I reopen this patch from nothing, there it is, but it doesn't have the sample anymore until I do this. Now, the thing I will urge you to do when you're doing this kind of thing is say load bang so that when you open the patch the next time it'll open the patch and then the load bang will say bang right when it loads and then it will read the table in.

[36:27] So this as a complete thing really should have two files. It should have voice.wav and it should have the patch. The patch shouldn't have voice.wav in it, which would be ugly and messy. Instead, the patch should be set up to read the thing in on load up. Alright. So now we have that I'll just actually check that it happens. There it is. I guess your way of verifying that the thing really loaded instead of not is disconnect this and see that it's zero again.

[37:04] Or else if you believe it look at the dialogue window for the array. So this is getting sound files into arrays and, of course, we're doing that so we can act like a sampler. Alright. Sampling is reading out of the array, which, to start with, I'll do the bad way and then I'll do it the good way. So I gave you this Hoorang earlier about non-interpolating table look-up being a bad way of reading samples or tables and to was hard to actually hear the badness because I was using [unintelligible37.41] .

[37:42] It will be easy to hear the badness when I start using real sounds, I think. So here now is listening to this sound file, so what we're going to have to do is say tab read tilde voice.wav. Oh, no, wait. I'm sorry. I give it not the file name, the name of the table. OK. Now I'm going to give it a signal which ramps from zero to 100,000 in some appropriate amount of time. An easy way to do that... Let's do it wrong first. I'll put a number box here and then we'll listen to it that way.

[38:28] Just looking at it, this thing has an amplitude of 0.03, which might be loud in this room, so I'm going to multiply this by another 0.03 to drop it to about 0.01ish. It would be better if I did something controllable there, but this will do for now. And now we listen to the sample. What this means is take the 134th point of the table. That's right about here. If I want to hear what this is like after a second had passed I would have to put 44,100 in there. Here's the sound of sample number 44,100 and, yeah, my audio system doesn't like this very much. Alright. Yeah?

Student: [39:21] Is there any particular place where if I renamed just a random wave file voice.wav and put it in the [unintelligible 39:28] folder. Would there be any reason for it to say it could not find the file, there's no such file or directory?

Puckette: [39:35] Beginner's error would be the patch itself doesn't know what directory it lives in. So save the patch into the same directory. The patch should tell you what directory it is here. If it doesn't say anything there or says slash or something like that then it's looking in the wrong place and the way to fix that is to save it so it will know what directory it belongs to. [40:06] That was a very useful question to ask, because that happens to about two thirds of people the first time they try to read a sample. If indeed that was the thing that just happened to you. [laughs] We'll find out. Otherwise, you belong to the other one third that got past that one and hit the next thing, whatever it turns out to be.

[40:28] So this is bad; you can't just do this [repeated booming noise] and expect to hear a nice continuous sound. Why, because you probably know this, but mice update from 50 to 100 times a second, depending on your OS and how hard you're mousing, and that's not going to sound like a continuous signal. It's going to just sound like a bunch of glitches. You could like that, of course.

[40:59] So what we have to do is do something continuous like this. Maybe we'll say line tilde and figure the message to... Let's see, I'll do this a variety of ways. First off, I'll do the stupid thing, which is feed it a message which says jump to zero and then slide to 100,000 over how much time. What is that in seconds if it's 100,000 samples? We'll find out.

[41:42] This is acoustics by the way, you should have learned this in acoustics class. My favorite calculator program. So what's 100,000 samples, which means we divide it by 44,100, and its 2.2676 seconds long. If I were not doing this pedagogically, if I were just doing this, I would just make my patch compute this thing automatically, but that would be more boxes messing around on the screen, so I wanted to show you this way first. So I wanted to show you this display first. So the appropriate length of time to read from the beginning to the end of the table would be 2,268 milliseconds roughly. Not exactly accurate, but good enough for today. 2268. So now I go back here and say you have 2,268 milliseconds to do this. Then we get samples. That's KC95 in case you don't recognize it.

[42:55] It's KC95 with some announcer who was announcing some 10 or 15 years ago, so you probably don't recognize it. Now we'll just use a thing that I've seen people get confused by. If you just say do that you don't get anything because the line tilde is currently putting 100,000 out. That's 100,000 volts if you were an analogue synthesizer. You would be arching at this point.

[43:29] To slide from where it is to 100,000 is basically to continue doing nothing or to continue putting out your 100,000 volts with no variation. So the correct way to hear it again is to go back to zero and then from there to ramp to 100,000 and you get the nice ramp. Sorry, it's a little loud. Let me turn it down a little bit. Alright. Now, either to clarify this or to further muddy it, depending on whether you personally find this clarifying or muddying, this will be different for different people, this is how you play it backwards.

[44:13] Let's go to 100,000, that makes the line jump immediately to 100,000, and then let's ramp to zero in 2,268 milliseconds. Alright. Then you have the other. Oh, and by the way, notice there was a good three quarters of a second of silence. It's reading the zeros at the end of the table. Alright. That's useful alright. OK. Now is it clear to everyone why this is doing what it's doing?

[44:55] So in one way of thinking, the x-axis here is labeled from zero to 100,000. The y-axis is the voltage, which comprises the sound itself, and we're asking it to go from left to right and we're giving it an amount of time. So now if I want to play it transposed up by a factor of two, that's to say an octave, what would I change and how? Someone said something, but I didn't hear it.

The time, yeah. So if you want to go twice as fast you do the same thing, but you do it in half the time. 113. Look, all the digits are even. I don't know why. And now instead of this: [45:27] continue to soft and relaxing [vocal interference] we get this: continue to soft and relaxing [chipmunk-like vocal interference] . All right? OK, and now we've basically got the sampler functionality.

Puckette: [45:58] And I was telling you this was going to sound horrible. But I have to work hard at this to make it sound horrible, because actually dropping every other points of the sample that would cause fold-over, but that wouldn't cause the bad fold-over that would make you really think that I was doing something wrong. A good way to make it sound wrong, might be to go at one percent wrong speed, like this two to 40, that's about a percent too fast, so here's the good sound: [46:12] Continue to soft and relaxing [volume/audio change] and here's wrong: continue to soft and relaxing [volume/audio change, same as before] . Can't hear the problem with it. Continue to soft and relaxing, continue to soft and relaxing [volume/audio change, still same] .

Puckette: [46:43] All right, I can't make it fail. Let's make it fail. OK, I'll make it fail. What we're going to do, is we're going to put an oscillator in here.

Puckette: [46:52] The oscillator's going to have a nice decently high frequency, but not high enough to be painful to most people, I hope. And I'm going to not do it at full blast. I'll do it at only about point three-ish to have similar amplitude to the sound file. And then we're going to use the fabulous tab right tilde object, and then we need a button [pause with clicking] to tell it to go.

Puckette: [47:28] Yeah, here's a nice sinusoid. All right, now here's the good sinusoid: [high pitched tone]

Puckette: [47:38] Oh my, that was horrible. Oh so even this is wrong, because this isn't the exact number. So once in about 2,000, no once in about 6,000 or 7,000 samples this is not working right, and you're even hearing that: [high pitched tone]

[47:52] And now if I push it a little further off,

[high pitched tone/buzzing]

Puckette: [48:00] That [hums pitch] that's the rate at which its dropping samples in order to get it done in a few milliseconds fewer, all right? And that is not, you know, they'd pay you for that in a club but they wouldn't pay you for that in a classical music concert.

Puckette: [48:16] Right? OK, so this: [high pitched tone]

Puckette: [48:20] ...could be a good reason to reach for this object here: [low clicking]

Puckette: [48:26] Oh, ew! Isn't that horrible? That's me putting out, I'm still putting out DC here, except that when I move this, the computer is having to redraw the table, which means we got zeros because the operating system's getting hungry for samples. And, as a result, I'm hearing a huge discontinuity in the sound. [low clicking] [clicking]

Puckette: [48:50] So that's bad. That would be a good reason to put this thing in a sub-window if I were going to do this during a show. All right, OK, so compare that to this [clicking] let's see, I'm just going to get these, oh, you know what, I'm going to do this right in this one, which has all the good examples, and then I'll make the bad one so you can compare it. The four, as you might or might not guess, means use four point interpolation, which means put a cubic polynomial through the four points immediately surrounding the point that you're asking for to give you, theoretically, a more accurate estimate of where the function would be at that point if it were a four time differential function. [49:40] Anybody's guess whether it really should be such a thing. OK. Now I'm going to still want that in this. So here now is the bad result. Oh my. Here's the good one. Or here's the first error I had. This one. It's the one that made a few dozen errors a second. That's to compare with this. Alright. So interpolation is your friend. Yeah?

Student: [unintelligible 50:23] [50:22]

Puckette: [50:27] When did I put the sine wave in? I used this network right here. So now if I want to get the sound file back I would do this. By the way, this is always playing right now, so you hear clicks when I change the table. You should probably have some way of muting the amplitude if you were doing this for real. Don't worry, you can make things arbitrarily complicated later to make it nice and clean. Then here's the sine played back. [50:59] So the morals of this is use tab read four tilde instead of tab read tilde for reading audio data out of arrays if you want the result to sound clean. Then the other thing to be hyper aware of is it might be as dirty as goats and you don't even know it because the sound itself is complicated, so test your patch out with Sinusoids first to verify that your patch actually does good things. Otherwise you might convince yourself that it sounds great, but it doesn't and you don't know it.

[51:39] Which is not a good state to be in. So it's better to apply stringent tests with very simple signals that you hear things clearly. In fact, now that I'm saying that, now that we've got the voice in here I'm not sure it's going to be audible in this room the difference between correctness and incorrectness.

[52:00] Here's incorrect. That's dropping the samples at the same rate and you don't even hear it. Maybe you can convince yourself that you hear it, but it even might be true that you're just imagining it. I don't think I can hear it at all. That's the wrong one. This is the right one. By the way, let's subtract them and we'll get an error signal. So I'm going to take this one and give it an amplitude of minus 0.01 and this one an amplitude of plus 0.01 and start them simultaneously. And there's an error signal for you. That was an aside. If you didn't understand that, don't worry about it. Questions about this? Yeah?

Student: [53:04] I missed the part where you came up with the third value for the message boxes.

Puckette: [53:08] Oh, right. So the message box has two messages in it and it's describing a line segment. So it's describing a start point, an end point, and a time to do it in and the third value is the value of time in which you ramp. Yeah?

Student: [53:28] So with the previous sample, after you had [unintelligible 53:34]

Puckette: [53:38] Oh, yeah. Right. Yes, if I put a few hundred objects on one side and then three objects on the other will the signal get faster through the three objects than through the few hundred? The answer is no. However, there are ways that I haven't show you yet that you can accumulate delays without knowing it, so there's one particular thing to watch out for there. Which is when you make non-local signal connections sometimes you don't know that they're instantaneous. Yeah?

Student: [54:07] Can you explain the interpolation?

Puckette: [54:11] Yeah. I should have started out by doing that. Back in the '70s when I was studying algebra they taught you this because you had to learn how to read tables of logarithms in order to know how to multiply numbers because people didn't have calculators. People might be forgiven for not knowing this widely now. Interpolation is the following thing. [54:40] You've got a table of logarithms and it's crammed in the last five pages of your algebra one text or something like maybe algebra two. You really want a logarithm to six decimal places of accuracy, say. So how do you do it? You're not going to get it just by looking the lines up in the table because just looking at them one next to the other they're 0.01 percent, say, from each one to the next because there are only perhaps 10,000 of the numbers printed for you and what you want is one that's between those two numbers.

[55:10] So what do you do? Well, you know how far it is between the two numbers so you just draw a straight line segment between them. And you know how far along the segment you want to be and then you can use some elementary algebra or geometry to figure out then where that line segment would be at the exact point that you asked for. So in the memory, and this is in general true about any digital signal processing, you only note the value of the function of the signal at specific discreet points and sometimes you want to know what it would be between two of the points.

[55:46] So how you do it is you say well we'll put a line segment or something like that between the two points of the table and if we're 0.03 of the way between this point and the next one we'll take this value plus 0.03 times the difference from this value to that value. OK. Then it gets better because that works OK, but that doesn't work well enough so that you can guarantee that the error's inaudible, say, for a sinusoid going at a kilohertz in a very good studio.

[56:21] So you could say I know it's between these two points, but I actually want to put a parabola, a second degree polynomial, between three points to estimate not just the slope between the two points, but also the second derivative and that might or might not give me an even more accurate value. That is called the order of the interpolation, the degree of the polynomial you're using.

[56:54] Tambourine four is four point interpolation, which I think is correctly called third order, which is to say it puts a cubic polynomial through the four surrounding points surrounding whatever non-existent point you've got. That's what tab read four tilde is doing for you. If you look in the book, I don't want to go fishing through it right now to find it, you'll actually find quantitative estimates of how high a frequency you can store in the table and get decent values using tab read four.

[57:31] It's actually not really good enough even though I can't prove it to you in this room. As soon as the period of a sinusoid is at least 32 points here it's pretty much a guarantee that tab read four tilde will give you inaudibly small errors. But the nine quest frequency, which is the highest frequency you could possibly represent, has a period of two points and at the nine quest even throwing a cubic thing through the four nearest points has trouble giving you the values correctly.

[58:02] They'll be incorrect. The good news is that people don't have signals that are loud at the nine quest frequency very often, so that's not very often a serious problem.

Student: [unintelligible 58:14] errors when you're [unintelligible 58:18] [58:14]

Puckette: [58:19] Let's see. So if the period's 32 and you're at 44 kilohertz that means components up to about 1500 hertz will have truly inaudibly small error and components above that will actually fold over gradually more and more until you get up to 20 kilohertz, the fold over is getting quite significant. [58:43] And if that's a problem, which is usually isn't but could be, then take the signal and re-sample it, so that its true sample rate isn't 44K1, but it's 196K or a million. I mean, you can sample as high as you want. And, so, you can actually get arbitrarily low errors using oversampling here. But, it turns out that at that point, oversampling is a better strategy than adding more points too quickly. I think. In most cases.

[59:13] Me, in practice, I just use tab-read-4-tilde and don't worry about it. Which is of course what I just told you not to do. But, you should actually check it, and all that kind of stuff.

Student: [unintelligible 59:24] [59:24]

Puckette: [59:26] Sorry?

Student: [unintelligible 59:27] [59:27]

Puckette: [59:29] No, there isn't. Life's too short for that one. If you look at the formula for new, big polynomial interpolation, it's already about a line long in C. And the one for eight points is, you know, a paragraph long. It would be a lot of computation, and it would be a serious head-scratcher. And, by the way, at that point, you get into serious questions whether doing a polynomial interpolation is better than doing something else. And so, then, you would have to have all sorts of different flavors and interpolators, some of which were better suited to some frequency ranges than others, and then you would be in a morass of horrible detail. [60:07] So, my advice is, just don't go there. And I don't even know if anyone has made higher-order interpolators for PD. If you want, when you go looking, and if you find one, tell me, because I'm curious too.

[60:25] All right. So, I think that's all I better tell you right now about sampling. I'm going to tell you a bunch more later. Oh, I should say one small thing about this, which is, just to tie this to what we've seen earlier. This is the message-based way of sampling, which lets you do this kind of stuff. [sound] I just whacked it a few times. Every time I whacked it, it was already running through the sample, but I just made it stop and go back to the beginning, start over again. So, that's kind of ... that would be appropriate, to for instance, you want to play something every time the network packet comes in or a key goes down, and apply their keyboard, or something like that.

You can also drive these things in the other way. Which is to say, make the whole thing be a signal network. Like this. Now I need [unintelligible 1: [61:15] 01:27] . And I really want this to be part of the same window, for pedagogical reasons. So I'll just make the window absurdly big. OK, so, another thing that you can do is, take the tab-read-4-tilde again, and all right, I'll do this. And let's just read it with a Phaser, just like I showed you how to do with tables. Of course, you don't really do this, let's see, let's give it a frequency, which will be a number, to do this. This, of course, will give us not a lot of anything, because it will be reading the first point of the table over and over again, which gets us nowhere until we start multiplying it by something.

[62:20] And now, just to remind you, if I give you a Phaser that goes from zero to one, but I want a Phaser that goes from A to B, the thing that I do is multiply by the size of signal that I want, B minus A, and then add A. Right. So we need now to multiply. That's what I need to do, is rearrange this, so we'll multiply it by something, and I'll just have a message box to decide what, and then we will add something, sorry, number box.

And now, for instance, if I want to say, OK, we'll do 10,000 points worth, 5000 points worth. [sound] That's the [unintelligible 1: [63:00] 03:10] continuous, and now if i want to change where I am ... [sound] that was defensive. Now, if I want to change where I start, I change this number. [sound] All right. [unintelligible 1:03:30] Actually, I should say, these numbers are in samples, right, because [unintelligible1:03:42] needs its input in samples, and number boxes operate as; they're good for values around zero to 1000, because it's a pixel a number.

[63:55] So, it's a good idea when you're doing this to rearrange the number boxes, by multiplying by some simple number, which this morning anyway, seemed like 100 is a good number. Let's do that. And now we have a much better, much more mousable little sampler. OK, so ... [sound] all right, oh, wait. Man. Sorry, but that was wrong. I want to do it on this side, and by the way, I've polluted the value of 100 in that box, so I'm going to retype the number to make sure it still works. [sound]

[65:04] OK, enough of that. So, all right, there might be some cleanup work to do if you want to play this in a classical music setting. But, so, what we're doing now is, this is the same thing as this, [sound] except that here, it's entirely a signal network. It's a Phaser tilde, which is simply scrubbing over and over and over again. It's not going to wait for me to get a message box at this point, it's automated. It's less flexible, but at the same time, it's... Well, it's easier to put together and it's easy to do things like change the frequency in the middle of a segment. Like here I wouldn't do something like, say, have it go up in the frequency as it is going through, it could, but it would be a lot of work.

Here will be easy, I'll just say, let's do this in a half... once per two seconds and we'll say, let's start with a zero, and it will range over two seconds, oh but I have it in hundred so that's 800 [unintelligible 1:[65:51] 06:05] . [Sound]

Puckette: [66:10] So now I can [unintelligible1:06:19] .. And even back.

Puckette: [66:25] Now this will be a part of it [unintelligible 1:06:27] . [Sound] Shut up, all right. So that's just the different image, that's just the different way that you might want to do it. And sometimes you want the one and sometimes you want the other. Frequently and unfortunately you might want aspects of one and aspects of the other you will have to make a difficult decision about which path to take and some aspects to what you're doing will be easier and there're others will be things that you wish you and the other [unintelligible 1:07:10] . [67:09] And that is, as far as I can tell it's sort of a fact of life about computer music. All right, so it's clear what the distinction is I think. And this one you could change what it is doing in midstream easily but you couldn't jump it back to the beginning... well you can actually because it just bash the Phase to the Phaser but there would be things that will be harder to do this way... And there are things that will be harder to... All right, let me then go back, oh what I want to do is ask for... Well just make sure that you hear the assignment three which is due Thursday and understand what's going on there. Then I'll show you assignment four not to demoralize you but in order to show you why I'm showing you all this.

[67:57] Let's see, why read me up, the opposite order let me show you the assignment four really quickly because I will show it to you again in more detail later. And then go back to assignment two... so I'm going to show you this and... and close it... and open... And look for... Homework four is just to... Oh, it doesn't work! Oh because I had a table and many pictures inside here. All right that means I have to close somebody... this one, no... this one... and this... many to close... Let's close everything.

[68:39] Oh I have two copies of this open. Wonderful work. Does it work now? Alright. So we're making the poor radio guy sing a little diddy [sound] . Alright, this is really stupid, but I tried to make it sound good using a very simple set of objects and couldn't. I could either make it hard to do or else I could make it sound bad and I decided to make it sound bad so it wouldn't be terribly hard to do. What's bad about that? Well, a lot of things are bad about that.

[69:24] Here's an alternative thing that sounds a lot better, but is much harder to do, which is to arrange that it actually make four copies of the sound for every pitch, which is going to request synchronization. Then you get this. [sound] So that's a proper sampler in some ways and that will be harder to pull off then just making the thing loop at different speeds, which is the basic homework. So think about how you want to get there and now I'll go back and play you the other homework assignment, which is really the one due next Tuesday. Oh, yeah?

Student: [70:10] Is that going to be on the [unintelligible 1:10:14]] ?

Puckette: [70:14] No because this implements it, so this is my secret now on how I did this. What I'm going to do is put the description of how to do it, like the other home works, I put up the description of what I want the thing to do and sound file showing how it sounds when you've done it right and then I make you work out the details to make it work. [70:37] Otherwise you just copy and paste the patch and turn it in. So here's the homework for this time, which also comes in easy and hard forms, but they're actually part of the same patch this time around. Is it this one? [sound] Yeah. I played you this already, but I didn't have the extra credit working quite correctly. This is just eight notes. By the way, since bringing this out, I told you how to do M to F to convert midi numbers to frequencies, but you can probably hear that these are not midi frequencies.

[71:36] I just made multiples of 110. These are harmonics instead. You don't have to figure out the melody. Just get the effect and make your own wonderful little melody. I will say that it works better, as I discovered by trial and error, it works better if you keep the thing within an octave than if you have the thing flying all over the place because, I don't know. Anyway, somehow the phasing effect is more effect if it isn't, if the intervals when they're out of phase aren't too crazy or big. Which means the melody some has to be in a decently small range. This is within a major sixth I think. Yep that's streaming, that's music 175. Oh an it's also the history of electronic music because it's phase music. OK here then is, no this is, where the stop button. I should have labeled them.

Here's the extra credit, this is really cruelly difficult. Don't even try this unless you want to be frustrated. So here are the ideas: [72:39] take each one of the individual melodies and every third note of it should be a substantially different timbre. That's the same melody as di di di di di di. Plus at this speed [sound] but of course what you hear is [sound] , right. Which is every third one. So you actually hear a sequence of eight things going at this speed instead of the [sound] real speed the things running at.

[73:19] OK, the way I did this without using stuff that I haven't shown you how to do. The mute buttons are not working today at all. The way I did this is I had two different strainers one which was just a sinusoid and the other which had I think third and fourth harmonics and none of the zero harmonic which you can do with cosign tilde and Phaser.

And then if you want to shut it up rather than try to mold it flat by the thing that ramps down to zero. Which would be hard because it's driven by a Phaser. I just bashed the frequency to zero to make it quiet. So each one of them has a frequency that is zero when I want it to shut up and non-zero when I wanted to hear that one. And that's why it sounds so ragged. Those popping sounds that's the sound of the sinusoid or another simple spectrum suddenly having it's frequency bashed to zero. It can't really do that.

[74:21] And then make two of those and make them do the same phasing thing. Let's see, I shut it up and I think these things are starting to phase if I just started a, OK like that. [sound] So now you can even hear this at this speed or actually can hear it at this speed. And you have phasing effects going on in both at both speeds. [sound] and so it takes 30 seconds for it to repeat or come back to the initial phase. There it is now. Oh, there it is. I think. Not sure. Alright. If you can do that then you might have something to listen to. Also, if you play this over your computer speakers you won't hear those glitches quite so badly because they are low frequency and your little half inch computer speaker won't play that.

[75:45] That's a better description of what the assignment is than I gave you last Thursday. This is all up on the website, but frequently I write something on the website and think it's totally clear and then when people read it they don't find it anywhere near as clear as I thought it was when I wrote it. So when that happens just ask, either by email or by asking in class. Questions about this? Yeah?

Student: [unintelligible 1:16:18] or is this just one [unintelligible1:16:24] [76:17] ?

Puckette: [76:27] The way I did it? I wanted to do it in a way that would allow me to change the two numbers, so I had to work very hard with modular arithmetic. I actually did the whole thing with one eighth note table, but then I did trickery to make it drop two thirds or one third of the values out of the table to zero. You actually know the objects to do that, but it might take several hours to figure out. The thing you get out of that is I can now do seven against...uh-oh, that's not seven. [77:11] And so on like that. Now it makes sense to make this number two. So if you're crazy like me you would do it that way and it would be much, much harder than just what you're suggesting, which is just have two tables and make it easy. But you can have a lot more fun with this one.

Student: [unintelligible 1:17:40] [77:40]

Puckette: [sound] [77:45] The original assignment doesn't have anything corresponding to that two. Yeah, I did it in the same patch so it is working the same way. I think I can make this now be five and it should just work. No, it doesn't. Why not? Oh, it went to zero. Why does that happen? [78:19] OK, yeah. And I figured out where in the sequence to start so that it was interesting with any number from three to eight. So I went much too far in messing around with this. It turns out to be very interesting to play with. But, yeah. Stop at what you said. Patch responsibly. Alright, office hours start now and also...

MUS171 01 27

Instructor: [0:00] This one does. Now how will this sound differently from this? any takers?

Student Questioner: [0:08] It shouldn't. It's basically...

Instructor: [0:10] It's exactly the same.

Student Questioner: [0:11] Yeah.

Instructor: [0:12] All it does is instead of going up to 88.2 in 2,000 milliseconds it goes to 100,000 in more. So it just ...

Student Questioner: [0:18] Complete silence afterward.

Instructor: [0:19] ... plays off the end of the table, which does nothing. Right. Well, playing off the end of the table simply sticks at the last point.

Student Questioner: [0:24] Oh, OK.

Instructor: [0:25] So again, if this was.

Recording: [0:27] This is your brain on drugs.

Instructor: [0:29] Then this is the same thing:

Recording: [0:30] This is your brain on drugs.

Instructor: [0:33] OK?

Student Questioner: [0:34] Wait, why doesn't it play the little noise after two seconds, because you stopped it at two seconds?

Instructor: [0:40] There is noise there, but if you wanted to hear that I would have to turn the volume way up. [0:46] So this noise ...

Student Questioner: [0:48] ... is being played.

Instructor: [0:49] Yeah. Whatever is there. In fact, actually, why don't we do this? How you would hear that, and just that, is we could go to, let's see, this is...

Student Questioner: [1:07] Instead of starting at 0 you start at 88.2?

Instructor: [1:09] Yeah. Well, I won't start there, but I'll start around here, so maybe that's 80,000 or something like that. So let's go... where am I going to do this, 80,000, and then I'll just add something and I'll say 2,000. Oh no, wait. 2, 268. Sorry. OK, let me explain this now that I've done it. So now what I'm going to do, I believe this to be 88,000 right here, because I said the table would be 88,200 samples long. So this might be about 80,000, maybe plus or minus. [1:51] And now what I asked to do was start there and go to another point which is 200,000 more and to go there in this amount of time, which is the amount of time I should ... no, I don't have that box any more but this is the amount of time it should take to go to... oh, it's 100,000. Sorry. Here. Go from a point most of the way through the table 100,000 samples further and do it in the amount of time that's correct for 100,000. So instead of hearing this ...

Recording: [2:21] This is your brain on drugs.

Instructor: [2:23] ... you hear this: [silence]

Instructor: [2:25] And now if you want to know what's in there really, then I'll take the risk of turning the audio up, and you'll here the room's reverberation and things like that, maybe. Don't know what that is. That might be the room ringing or it might be me clicking on something right after I said it or something else.

Student Questioner: [2:56] Why did you choose that amount of time, 2268, as opposed to just like two seconds?

Instructor: [3:02] Oh, right. So ...

Student Questioner: [3:04] Is there like a reason?

Instructor: [3:05] Yeah, that happened last time. I computed how much time 100,000 samples lasts. And I did it using a calculator.

Student Questioner: [3:12] Oh, right, right, right.

Instructor: [3:14] That's this thing.

Student Questioner: [3:17] So you just got to do that real time?

Instructor: [3:20] I didn't want to actually re-compute it, so I just used the value I happened to know. But the value I happened to know was, if I want to have 100,000 samples and I'm running at 44 kilohertz .1, that's the number of seconds that amount of that number of samples lasts.

Student Questioner: [3:43] So if you don't put that number in, then it won't be the exact same time like you recorded it?

Instructor: [3:49] Right.

Student Questioner: [3:49] Like, that's real time, right?

Instructor: [3:51] Right. Yeah. Or, real time, that is to say, that's for no transposition. And this isn't a good example. This is ...

Recording: [3:57] This is your brain on drugs.

Instructor: [3:59] So if I change this to something else, like ...

Student Questioner: [4:03] Like what if you didn't put a number?

Recording: [high-pitched] [4:05] : This is your brain on drugs.

Instructor: [4:07] If you didn't put a number at all, then it would go to 0, and then it would jump to 100,000 and stay there ...

Student Questioner: [4:11] Oh.

Instructor: [4:12] ... to hear nothing. Yeah.

Student Questioner: [4:18] So how would you start it right where the recording actually starts, and make it accurate?

Instructor: [4:24] Good question. How would you find the place at the beginning of the table in order to be start right there? There are a bunch of ways you could do that. The way I would do it if it were me is I would do the following. Oh, this will take, I have to add a concept here, but I'll do it. Because I was wondering when I was going to be able to add this concept. So how about instead of ... sorry. I'm going to really do this. [5:02] I'm not going to add any concepts at all. I'm going to do this. I'm going to make a little player that plays little tiny segments of the table. The concept, by the way, that I want to add is using message boxes to make variable messages, like using dollar signs, but I haven't hit a place where I need that yet. So what I'm going to do is I'm going to make one of these, except I'm going to make it be variable. All right?

[5:37] Let's see, I'm going to need some room. I can still reuse all that. And this needs to go to here, so that I have room up here. Pull this down, make the window bigger. So now what I'm going to do is I'm going to make a number box, and I'm going to arrange for the table to play, starting at the number box, a few samples.

[6:06] All right? So, to do that, what we're going to want is to generate something like this, number, comma, and then another number in a certain amount of time. Except that the numbers are going to be variable. So, for instance, the numbers might be ... let's start at 10,000, and then let's go to another 1,000, and let's take a hundred milliseconds. So, these two, if I did them in sequence, would go click, and what this is, sounds nice and metallic. That's me doing a consonant, probably, and it's at the wrong speed, because I have to be careful about speed.

[7:01] Now, how would I do that, except make the 10,000 variable? Well, it's not so hard, all we have to do is we have to use the wonderful trigger object to generate two messages. One of the messages will go straight through, the other one we're going to add 100 to it, and then pack it in order to make a message hit 100. So that's going to be like this.

[7:26] So this one, instead of doing this, we're just going to run this straight into this one. Is going to be a packed message using the pack object, and what we're going to pack is going to start with zero and 100, and this, except we're going to replace the value of zero with something. And, the value that we're going to replace is going to be whatever this thing is, plus 1,000. And now, we have to send a message in here and there, and we should do it in the right order. So, we should use a trigger object. Trigger, OK, these are all objects you've seen, although you haven't seen them done this way, so I'm doing a sort of review.

[8:22] So first off, we're going to send the value as a floating-point number to the line then, and immediately after, we're going to send a value which is 1,000 more. But it's going to be in a packed message, with 100. And now if I say 10,000, I get my message back or my sound back. [sound]

[8:47] It's OK. This isn't moving very much. So, let's do some labor-saving and multiply it by 100. Now, this value is too big. [sound] Now, what we're doing, is we're scratching through the sample. It's such a horrible pitch, you can't even tell what's happening, or I can't, so I'm going to say, do this in about a fifth of the time that I said before.

[sound] Now I've got a pretty powerful tool. Because I can do this. [sound] It's got problems, right, which we're going to have to work on. [sound]

[9:25] But notice now, I can say, is the very beginning, this is your question now. Here's the very beginning of it. If I want to know exactly what the beginning was, maybe I should subtract this 1,000 from here, but it's close enough. And, I don't know of a better way, actually, of finding the beginning... I don't know a better, simple way of finding the beginning of the sound than just looking for it audibly like that. Question?

Student Questioner: [10:20] Sorry, can you go over, figure [inaudible 10:20] one more time?

Instructor: [10:22] Yeah. Good idea. So, trigger, I've introduced and I think the last time it was triggered with two bangs, as a way of making two message boxes be in the correct order. And now I'm using a slight twist on the trigger object, which is that it will take whatever message you put in, and output that message from right to left, in right to left order. Let's see, it looks like this to you. And it will put out variously floating-point numbers or bangs, which are messages that don't have any data, or lists, in case your message has more than one number in it. [10:59] And so in this case, what it's doing is it's interpreting this as a floating-point number each time, which is appropriate. And it's first putting the number 2,700 out here, and then it's putting it out here, so that it can get 100 added to it and get packed with the value 40. So that, if I want to see what's happening to the line with a print object, no tilde needed, just the regular old message print. I can do this. Now, the print is hooked up exactly the same as the line object, which is down here.

Student Questioner: [11:35] What does the number 40 represent in the pack?

Instructor: [11:38] 40 is the amount of time. So what I'm sending the line is these two messages. 2,700, that came from this 27 when I multiplied it by 100. And then the other message is going to be a target value of the time. The target value is computed by adding 1,000 to whatever the start time was, and the time I just gave as 40. And now, of course, 40 milliseconds isn't the right amount of time for 1,000 samples to last. But I just remembered that we went to the trouble of finding out that 100,000 samples is 2,268 milliseconds. [12:25] So, really here for 1,000 we should use one hundredth of that which is really 22.6. It's not gonna be exactly right for technical reasons. [sound] Now I've got a little scratch.

Recording: [12:49] This is your brain.

Instructor: [12:50] Yeah.

Student Questioner: [12:51] I'm sorry. Why exactly do you want to pack this?

Instructor: [12:54] Oh. Why do I want to pack these? So that the message itself can have two values in it, because line tilde wants to get a message which is target and amount of time in a single message.

Student Questioner: [13:06] So that's why your packing [inaudible 13:08] to just...

Instructor: [13:10] Right.

Student Questioner: [13:13] So, I have to automatically replace it to zero?

Instructor: [13:16] Yes. So the zero initializes input but is getting replaced every time I put a new number in. In fact I can replace this by setting numbers in here too.

Student Questioner: [13:33] Did you send the trigger also to the right input of pack and so whatever you are sending to that zero you're not just going to [inaudible 13:40] . You know how you're getting 40,200 plus a thousand...

Instructor: [13:46] You mean make a thing that has this number twice?

Student Questioner: [13:49] Yes.

Instructor: [13:50] You can do that but you can also do it wrong, like this. This won't sound good but I'll do it just for the sake of the argument. I do that and I'll put a number in here and I get the correct value is whatever the previous correct value was. So now we have one of these situations where if you put a message in here that generates output and then if you put a message in there it updates the inlet. But you want to update the inlet first so that that value will be there when the outlet comes.

Student Questioner: [14:22] I forget that it has to be...

Instructor: [14:23] Right, and to do that you need another trigger object.

Student Questioner: [14:26] If you put another grammar in there as well will that create another inlet that you can also control?

Instructor: [14:32] Yes, and I think line Tilda doesn't care about that. I'll do another value. Oh, let's not do this anymore. [sound] Right now it's making messages with three values. The thing about that is that there aren't very many objects running on Pd that can meaningfully deal with three numbers at a time. They are all designed to be as elemental or as elementary as they can possible be. In general, when you are just using objects and forming messages for them pack with two numbers is all you're going to need. [15:14] However, when you make instruments yourselves that might have 100 parameters in them to describe a voice you might find yourself packing all those 100 parameters into a big mondo message. Eventually it's going to be interesting to be able to use pack with large numbers or at least medium sized numbers of parameters.

Student Questioner: [15:33] If that's the case would you use Pd window, that hidden window to do that and just have those with you?

Instructor: [15:38] Yeah, by that point you would be keeping everything under the hood. Yeah, in a big way. And we'll get there but maybe not even in the first quarter of this. Other questions? If everyone is happy with this... I can't believe you all actually understand what is going on. OK, well my plans for the rest of the day is to make this more complicated. Because of course, [audio jumps] because it would be useful to be able to do things like actually control the transposition. [16:42] When you buy a sampler you sample something and then you hit C and you hear the original thing so you hit G and you want to hear it in transposed to fifth and that sort of thing, So I want to talk about transposition and how to do it. Now the other thing which is perhaps, even more fundamentally important, is how to keep it from [sound] making those clicks and there are several ways of doing that. I want to show you two of them.

[17:08] What I think I should do is start with the transposition. The sound will still be kind of revolting but at least you'll see how it's possible to transpose stuff. Then after that I'll start in on trying to work on the clicks. Alright. Transposition stuff. Transposition is not so easy to necessarily hear how much this is being transposed by because you might not know what the original... Does everyone know what transposition is? That's a musical term.

[17:41] That means the change in pitch that you get when you read out the sampler compared to what you put into it. At least that's what it means in sampler language. What I am going to do is get rid of this because we are not going to get here so fast, but I''m going to put that in a future window. [sound]

[18:24] What this is that we will come back to is using a phaser object to drive a sampler. Which, you in fact saw for the first time on Tuesday but I haven't shown you for instance how to do things like affect the transposition in this way of doing sampling.

[18:42] So we've seen two ways of operating samplers in the same way as we have seen two ways of reading sequences back. One of which is this way, which is the signaling way which is generated, as I mentioned, well, which if you look all the way over to the top of the network here, you'll typically see a phaser tilde object.

[19:00] The other way of doing it is to just use messages. In which case, you can do things in more irregular ways. But there is sometimes more to think about when you're doing the messages that just doing the phases.

[19:13] So this, which is maybe even conceptually simpler, I'm going to get rid of and go back to the more complicated thing. Meanwhile, let's see. I'll just set a good example by putting a high pass filter there and this, we might need this later. OK.

[19:32] So what we're going to do to start with is just listen to a nice sample [tone] . It's just me saying"Oh" because I can't sing. This is about G, just by accident. OK, so that's what's in this file.

And now what I want to do is say "OK, that was a nice G, but I want to hear an A." [tone] All right. Question: [19:52] How do you get from G to A?

[20:00] Well, you transpose. How do you transpose?

[20:07] OK, choice. So we know basically how to transpose. We know that rather than do this, we should do this. [tone] We should do it in approximately 12 percent less than this; it's going to be 250-ish. [tone] .

No! Bad, bad, bad, bad, bad. I'm trying to do math in my head and not doing it well. [tone] No. [tones] [inaudible 20: [20:27] 42] for you, it's not really exact. OK.

[20:44] A major second, OK, you learn this in acoustics, a minor second is six percent. It's actually 1.059 to 1, that's the twelfth root of 2, which I use every day. [laughter]

[20:57] And what I was trying to do here was divide by the square of the twelfth root of two, which I didn't quite get. I finally just did it by ear.

[21:05] So here's the original sound. [tone] That's G. This one, if I did it right, would be an A. [tone] Sorry about that bad singing.

[21:14] All right. So what's happening is I could have changed this value of 100,000. I could, say, just go, whatever, 12 percent further than 100,000 in that amount of time. But then when I started making extreme transformations down I might not get to the end of the table and so I might not like that so much.

[21:32] So instead, it's easier to, say, go to a place that I know is beyond the end of the table. In fact, when I'm doing this for myself, I usually take this into millions so that it's beyond any table that I would be likely to use. And then, compute the amount of time that you really should do that in in milliseconds. All right.

[21:55] There are a couple of ways you could do that. I could either do this explicitly with logarithms. Or I could do it the less brainy way, which is I could reach to the MIDI for a frequency and operate that way.

[22:09] So I'll do that because that's easier to think about. Or rather, I don't know, I find this easier to think about. Maybe you will, too.

So here is how you do this. This is going to be [inaudible 22: [22:18] 25] done. So you need the frequency This is a thing which allows you to take a number, which is a frequency. Oh, come on.

[22:42] This is a number in MIDI and it converts it into a frequency. So for instance, if I needed 60, I find the number of hertz in middle C. If I feed it 61, I will be... Whoa, sorry Vera. I'll be six percent faster and so on like that.

[23:03] And now I'm realizing, I'm trying to be as simple as possible, so I'm going to go back on my earlier promise to do this correctly in order to do it more simply in the following way. So what I'm going to do is say, OK, we're going to go to 100,000 in 2268 milliseconds, all right.

[23:22] Except that my value of 100,000 is going to be different. I want to fix it so that when I say 60, it will be 100,000 all right. But if I say 61, it will be six percent more and so on like that. OK.

[23:36] How do I do that? Well, it's easy. I just change 261.6, whatever it is, 100,000. To do that, this rescaling, a simple-minded way of doing it is just divide by what it was and multiply it by what you want it to be.

[23:53] This is the other number I use every day, the number of hertz in middle C. And then you can multiply by 100,000.

[24:05] And then, I'm going to, why don't I set a good example and do this network here. [sound]

[24:24] All right, so I'm going to reach for this trigger. This trigger gives me a value, which goes here. And then it gives me the... I'm going to stick this in here, and what I want it to do is I want it to say zero. And then I want it to say this value packed with the amount of time which is 2-2-6-8.

[24:48] So, lose all this. This will double the print object still. Let's see, what's a good... And here I just want a bang. Don't need this anymore. So, in fact, it would be better if I put all this in one place. There. Right. So, when I get a bang, I want to go to the value zero. This ... that's the beginning of the table, and then afterward I'm going to pack the amount of time, which will replace zero with the wonderful amount of time, 2268, this is all well and ... now, G ... [tone]

[tones]

[laughter] [25:52]

Student Questioner: [26:04] I'm lost. I'm lost with the bang. I got how you're setting and transposing, but why the bang.

Instructor: [26:10] OK, let me answer that first, because that's specific, and then I'll try to answer. I've lost, which is more general. So, the specific question is, why would you put a bang on here, and float down here. OK, so this is a message box, which will put out zero, no matter what goes in. So, in fact, I could have put in a bang, or a floating point number, or a list of numbers, or anything, and I would have come to message zero. [26:38] I put bang here as a matter of style, because I didn't want to put float, because it was just going to be ignored anyway. And so, it was simpler to think of it just as being bang. So, bang is just a message that doesn't have any numerical value associated with it. It's the equivalent of the keyword void in C. So, what happens is, whenever I say a number here, [tones] sorry... Whenever I see a number here, stuff happens, which is why I should print this number out. I should display this for you, so you can do it.

[27:16] 60 again. [tone] Sorry. And the width will be seven, it might be a big number. Tada. Ooh. Truncation error. You won't hear that. All right. So, stuff happens, and this number comes out to 1,000, which is the number of samples ... which is the number I want to put here. Oh, yes. Right, I need my print object again.

So, now, this print object is showing us exactly what's happening to the line tilde. Let's get rid of all of this. We've got this patch, it's not doing [inaudible 28: [27:50] 00] . All right. So, 61 here, and that converter, it got turned into 100,000 and then trigger says first, send a bang to here, which outputs the message zero, which causes line tilde to jump to zero, which causes tab re-tilde to read first [inaudible 28:22] .

[28:24] Then, this gets the number 100,000 because I asked for a floating point number, which is just a number. So, this is now the message 100,002, sorry about the two, and then we pack that in 2,268, and so then what we see is the message, which is 100,068. So this pair of messages got printed out, I think.

Student Questioner: [28:53] So now, we're going from zero, to 2,268. It starts at zero and then goes to 2,268.

Student Questioner: [29:00] Oh, I see. Maybe I don't. Maybe I connected with print app. So, it gets two messages, it's zero, and then 100,000 and 2,268.

Student Questioner: [29:14] So basically, this time, we're transposing by... instead of changing the amount of seconds it takes with the light object, by changing the amount of samples.

Instructor: [29:22] Right. And the reason I did that, was because it made the math simpler, not because it launch better. Yeah?

Student Questioner: [29:30] Is there a samples per second, kind of like, M2F kind of thing? You just kind of bypass the math?

Instructor: [29:38] No. [laughs] No, there isn't. Basically, there are no primitives that would make this easier, although I could tell you how to make it a little bit more complicated. So, now there are two things to understand, I guess. One is, the bit that, this bit here, the stuff that's actually making the sound, which is to say sequencing two messages for the line tilde, and making the sampler read. [30:13] The other thing to understand is how on earth I'm computing this value. So this is the value that you have to stick here, in order to get the right transposition. And now, to answer your question a little bit better, I want this value to be proportional to this value, and not in proportion to its inverse. I didn't want to have to divide by something, I could. But it would be more work.

Student Questioner: [30:41] How do you get the 100,002 again?

Instructor: [30:44] OK. Well, it got it for me. What I did was I said I want 60 to go to 100,000. So, this is... So, now what you're asking is what is the design of this collection of objects. There really are only three objects here that are doing stuff. There's the middle frequency and then there's a rescaling, which is these two objects. We divide by the number of hertz and the middle C, and we multiply by 1000. So, what is happening here is we know that these numbers coming up have the correct proportions, so that we can do musical scales or musical intervals. Why, because M2F, if I, for instance, add 12 to the value here. [tones]

Instructor: [31:30] It multiplies its result by two. All right? So M to F is the thing which takes 12 steps and turns it into "multiplied by two Hertz," or, if you like, it takes the keys of a piano and converts them to Hertz. So that you move up 12 keys on the piano and it doubles the value. [31:51] Now that we have the ability to do that... And that's just this object which does the math for us that does that. It's not so bad, it's -- I can tell you the expression. It's just an exponentiation, except it's scaled correctly. Then, what we say is "OK, I want those proportions. That's to say I want 72 to be twice what 60 is. I want 48 to be a half of what 60 is," and so on like that.

[32:15] But, anyway, I want 60 to give me 100,000. So how do I do that? I just have to multiply by the number, which is 100,000 over 261.62. Well, that's kind of... Ah, what's the right word? I could do that. I could divide 100,000 by 261.62; but I've found it more pedagogically transparent, hopefully, to use two objects so that I divide by 261.62 to get a transposition. And then multiply by this because I want a transposition of one to give me a value of 100,000.

[32:56] So, in fact, I should have been showing you two numbers. First off, this number is the interval from 60. So if I get 60...

[tone]

Instructor: [33:08] .. get one there. I really did that. I think this is 261.626. Those twos are getting my goat. [tone]

Instructor: [33:18] This is all good. Now we're within a part per million or so. A couple parts per million. So we divide by 261.626. That's just so that 60 gives us 1. And, by the way, now we're in floating point land. You're never going to be using this value actually. It's irrational anyway. So instead, we just get as close as we can and float point man to it. [33:43] So now 60 goes to 1. So 72 will go to 2, for instance.

[tone]

Instructor: [33:48] 84 would go to 4. Oops. [tone]

Instructor: [33:53] And 96 could go to 8. [tone]

Instructor: [33:57] And so on, like that. 48 goes to a half. [tone]

Instructor: [34:03] And then harder values. There was 60 again, which goes to 1. [tone]

Instructor: [34:07] Go up a fifth and you go to 67, which is MIDI for G above Middle C, and that turns into roughly one and a half... [tone]

Instructor: [34:15] ...because seven semitones is roughly a factor of a multiple of one-and-a-half to one, but not exactly. [laughter]

Instructor: [34:23] Yeah?

Student Questioner: [34:24] If you go too low, though, it's going to start flipping like this. You don't have enough samples?

Instructor: [34:29] Well, it'll just stop after two seconds, even if it doesn't get to the end of the sample. So if I say, yeah, do 24.

Student Questioner: [34:36] What if you did 30? [tone]

Instructor: [34:41] Stops after two seconds. Oh! It stops at...

Student Questioner: [34:43] While you heard it, it was like tut-tut-tut-tut-tut...

Instructor: [34:47] Right. This is now - what is that? Three octaves below Middle C? So I was droning along at 100 Hertz, and you divide that by eight and you get about 12, which is an audible rate. [sound playing]

Instructor: [35:02] Maybe that's 12 Hertz-ish. Oh, and so that's one-eighth transposition, down by a factor of eight. And an eighth of 100,000 is this number, which is the amount we go in the table, which only gets us to about here. Somewhere in there. And it takes that 2.2 seconds to do it, but it originally it would have taken 100,000 samples.

Student Questioner: [35:28] So it's not playing the whole clip? It's only playing a portion of the clip?

Instructor: [35:31] Yeah.

Student Questioner: [35:32] So you're not really transposing. Transposing would be playing the whole thing, but in a different key or in a different frequency.

Instructor: [35:39] Yeah. So it's transposing it, but then it's cutting the transposition off after two seconds.

Student Questioner: [35:43] So if you would have a [inaudible 35:44] .

Instructor: [35:45] Yeah. I could have done it the other way, which is I could have said, "Go to the end of the table and compute the amount of time that you do." But then I would have had to divide by this number, which would have been an extra trigger object. I didn't want to make it any more complicated than it already is. Yeah?

Student Questioner: [36:04] When you're multiplying by 100,000, what is that number representing an amount of?

Instructor: [36:09] So that's the number of samples at... OK. If you were at unit transposition -- that's to say, if you were playing the original sound back -- that is the number of samples that you would play in 2,268 milliseconds. So the value of 100,000 is, in fact, arbitrary. I chose that in some other context two days ago, and now I'm just sticking with that value because I happen to know that 100,000 samples at 44 Kilohertz corresponds to this number of milliseconds.

Student Questioner: [36:47] Which is a little more than two seconds.

Instructor: [36:49] A bit over, yeah. Somewhat over two seconds.

Student Questioner: [36:55] Can you review tardigrade for tilde?

Instructor: [36:59] Yeah, OK. So tardigrade for tilde is a... Let's see, so it's doing a read into the array whose name is T1 [inaudible 37:09] 278, which is over here. And what goes in are the x-values, that's to say where you want to be in the array. And it is in samples. And what comes out is just what value in the array there, which is the vertical axis in this graph. Yeah?

Student Questioner: [37:32] Is it possible to do a transposition without the time stretch?

Instructor: [37:38] Well, yeah. It's more work. But, actually, you all ready sort of heard it because what I showed you in the previous patch, I was able to do little bits of samples in the table, you heard it going backwards and forwards, but it was ugly. So, to do that and make it pretty is actually hard. We might get there in a couple weeks or we might not, depending on whether it turns out to be a good use of our time. I'm not sure. Yeah?

Student Questioner: [38:11] Are there cases where you want to use tardigrade for tilde routed than tab before tilde?

Instructor: [38:17] Yes. The best example I know of is back in sequencer lab. I think we're good here. People pitch this one [tone] . Now let's make this tab read four [tones] . Well, you could want that, but if you want it to sound like a classic synthesizer you really want that. So, tab read four, on the other hand, that's what you want to use if you want a clean audio sound of reading another sample. Let me show you, now that I'm in this context, one place where it's really good to have this. [40:21] One thing that you can do is run off the end of the table, so let's go back to playing the original position [tone] . Now I have a rather bad thing because, if I disconnect this- you hear that click? Actually, I could make that worse. What's the first sample of the table? There it is. It's not that much worse. The reason that you hear that click is because tab read tilde is constantly giving me the last value on the table.

[41:12] So, the line, tilda, is giving me a hundred-thousand right now. Let's just verify this. We need a bang for this. So, look at that. We have a hundred-thousand coming out of the line. That's appropriate because I asked it to go there in 22.68 milliseconds. Then we say tab ring four and it says "OK. That's fine", and I'm going to give you 0.03, blah, blah, blah, this amount. That's the value of the table and it's location that corresponds to 99,999, which is the last point on the table.

[42:05] And that, when you disconnect it or connect it, sounds like a click because it goes between that and zero. So, how do we get rid of this? We say hit tilde, three. The three is just a good number which is well under 20 which is the bottom limit of human hearing. Print tilde here said these numbers, and then if I talk to hit tilde I get these numbers which are much, much smaller. They're down to termination error. It might even be turning into a zero.

[42:56] So, the high pass filter essentially cleaned up the DC value, the constant value of .03 it didn't have before, as a result of which, I can now disconnect it and connect it without hearing it. Although I haven't told you about hit tilde, I systematically put a three hertz, high pass filter in front of the digital analog converter and before and after the analog and digital converter. Every time I make a patch it's going to deal with audio. If my patch is going to drive a DC motor I don't do it, but that doesn't happen very often. Almost always I have a tilde three somewhere in the signal patch.

Student Questioner: [43:49] I was going to but [inaudible 43:50] it's three hertz.

Instructor: [43:52] It's three hertz. What that means is it's way below 20 hertz, so far below 20 hertz that by the time 20 hertz happens, we're so far away from the filters roll off that it's now 10 hertz. Engineers use values up to about 10 for that. It's a low cut-off.

Student Questioner: [44:13] But if we can't hear it, then what's the point of it?

Instructor: [44:15] Well, a point of it is if I had a bunch of these voices, and they were all stopped in the same way, but if their outputs all added up to something more than one-- not only would I not hear anything, but even if I played a sound out in some other part of the patch, I wouldn't hear it because everything would be above the output max of the digital-to-analog converter. [44:39] Here's a thing you might not want to do-- it's good review. Can we get a toggle and then make a metronome? Sorry about this; I'm going to make something that will annoy you. That will come 60 [tone] , twice a second. Now that's really annoying, we're not going to do that. Oh, and we don't need this print anymore. [beat track sound] OK? So, we know what's coming out of this thing.

[laughter]

Instructor: [45:15] Now what I'm going to do is show you how to make it hurt. I'm going to do another one of these. I'm not going to even remember to connect anything to it, but I'm going to connect it with a bad volume. Now I'm going to start annoying you again. [tones]

Instructor: [45:55] That is the sound of my sample getting shoved up against the very top of the audible range or the possible output range of my converter part. It's always called distortion, but sometimes this is called a bias. This is a knob on your expensive guitar amplifier, the expensive tube amp. It does that to the guitar sound.

Student Questioner: [46:22] How does that relate to the height? How is that related to sequence?

Instructor: [46:25] Oh, OK. So, I might have done this by accident, somehow, but if, [tones] on the other hand, I had a high-pass filter there, then I can give this as much glass as I want, and it filters it out. It's gone, so I can still hear it distinctly. [47:04] What happened there was this thing is putting out 0.06 V. So, this is now putting out 60 V, too much for my heart rate; but this is taking the DC out of it, which is therefore returning it to zero, because everything that's happening here is DC. It's flat-lined; it's not changing with time.

Student Questioner: [47:28] When you're increasing the bias, is the speaker cone also going out towards its maximum treble?

Instructor: [47:35] It should be, except that I have faith in the cheapness of the mixing hardware that we have here, and it's almost certainly AC coupled, so that, in fact, there is an implied HIP somewhere in the hardware.

Student Questioner: [47:49] OK.

Instructor: [47:50] Otherwise I could actually literally push the speaker out its cone. Well, probably the amplifier wouldn't allow me to do that anyway, but it would be bad because DC speakers' resistance goes down, and so you can actually burn your speaker out if it would actually send that voltage to the speakers. [48:08] I have long experience with this brand of mixer because I'm cheap too. I know that it won't actually do that to me, neither will your home stereo. It's more expensive to build DC coupled stuff. Yeah?

Student Questioner: [48:20] Is it possible to write your own objects, like the M2 app?

Instructor: [48:25] Sure.

Student Questioner: [48:26] Someone was saying before about the sample's frequency, because that's just a simple conversion?

Instructor: [48:31] Yeah, it's just an arithmetic expression. You could just make an object that does it. That is the subject, or a subject among others, of Tom Herb's seminar that he gives in the Fall on programming for the [inaudible 48:45] applications. Although, you can just sort of learn how to do it. Just go find a .pdf that someone's written, see how it's done by example, and just go from there. There are some thousands of them on the web. Any questions? [49:09] OK. So, what happened was two things that were kind of jumbled together. One is this high-pass filtering notion, which is a good way of dealing with DC offset problems in signals which will come in. This is one good way of making yourself a DC offset without wanting to-- having a sampler and just have it stuck in a location in the sample; but there are many others.

[49:36] In particular, if I, for instance, just say "Give me an oscillator," it now has a phase of zero because it starts at the top of the cycle. It's putting out "one," constantly. If I put that out my DAC tilde, that will mess up the sound of the rest of the patch.

[50:00] Frequency modulation and wave shaping-- I've shown you frequency modulation, although I need to show it to you again more controllably-- they have a tendency to have DC as part of their spectrum. It's not the only thing they put out, but they will put out some DC as well as everything else. As a result, if you are playing them, you probably want to put one of these things in the chain somewhere.

[50:25] OK. So that's this object and then the other thing was this way of talking about transposition of samples. Done in a way that hopefully is as simple as possible which is to say that you first get yourself from pitch to frequency and then correct. So that its in the range that you want which is 100,000 corresponds to one transposition. And the reason I've decided that 60 should correspond transposition is just because that's a habit. Middle C by default is a transposition on a sample. And 60 is the main value from goal. All right, questions about this? All right. yeah.

Student Questioner: [51:20] Is there a way to change the slope of the filter.

Instructor: [51:23] Slope. OK. In this particular filter is the most simple minded well second most simple minded filter in the world and the only thing you can change is the cut of point. It always has a 3BD proactive roll off. So it's the cheapest, simplest possible low cut. You can get better ones but then you land in a region where they're actually thousands of designs filters. There are at least dozens of them available for you And you would have to spend some time figuring out what characteristics you wanted. I wouldn't be able to just give you a quick answer. [52:08] Other questions about that? OK. So what I want to do now is well first off mention that of course this was playing the entire sample but you can play bits of samples as well.

[52:20] And what I should do is start another window showing you an example of playing a bit of a sample. That will be interesting because you make it much much more important to be able to control the clicks that you have at the beginnings and ends of some things. So that should be kind of the next thing to worry about.

[52:39] OK. So what I'm going to do is save as, going to hang on to this thing. And I thought it was before that but, four. I will get this far I'll start it anyway to do a little bit of the sampling. The first thing I want to do is say OK rather than making 100,000 I want to make 1000 points in a well do it, no, lets do 10,000 points. and lets do it in this amount of time. See how this sounds. [tones] Pretty good.

[53:30] I should say here that this is not a good practice. I'm taking a very short amount of time here and time is quantized. Rather I should say our tilde has a quantization in its actions with the 64 samples. So this is not a terribly accurate way to make a sample because this will be quantized to about one 1/2 which would be noticeably our tier is actually being careful about it.

[53:56] So I'm making, so this is just going to be batting an apple right now because I just want to show you a little bit. [noise] That is that, it clicks every time you use it and we would like to quit that. So how do we deal with that, in the usual obvious way, you multiply by a line tilde. So and the thing that you multiply is the operating tab for tilde.

[54:32] I thought I was past filtering that? Oh I am past filtering that but I'm changing the way I get to the high filter which gets us to the step function which we still need. Anyway, let's get rid of this and let's multiply it by a line tilde. And now there will be two line tilde's running around. And so there will be plenty of opportunity to get confused. The first one, let's see, this is going to be more.

[55:11] All right. This one now what were going to do is were going to say you are at zero please and lets go up to a value of one and do it in a certain amount of time. Which will be more effective. And then after a certain delay we can turn it back off. And now I get something which allows me to put values in there. [noise] What it's doing I didn't want it to do that. Send this to zero. [tones] Ah, no clicks.

[56:05] So before it was this [tones] and after is this [tones] . All right. OK. Now who can tell me why this will stop working. The next thing I do, I'm going to do scroll on this thing and it's going to make my clicks after all. Anyway. [modulating sounds] And why does that happen?

Student Questioner: [56:37] You're going faster than the delay?

Instructor: [56:39] Yeah. I'm not waiting for this, I'm not waiting for a new note before I start another one. So let me back up, and carefully explain what's happening. What's happening is, we're starting the tambourine tool, and we're giving it two messages which tell it to jump to a point and then to slide to another point. At the same time that's happening, we're muting this line. We're making it jump to zero; and then we're ramping it up to one in 60 milliseconds. And then, a 100 milliseconds later, we're making it go back down to zero. So that it's safely back down, before the end of this segment. [57:11] So the thing goes up. It sits there at it's apex for 50 milliseconds because we started, within a 100 milliseconds later, we started sending it back down. But meanwhile, if I don't wait that whole 150 milliseconds before I send another message. [modulating sounds] It will bash the value zero discontinuously, and that will sound like a click. So, actually, it's a little harder than this. What you really should do is duck. What you really should do - I'm thinking of a simple way to explain this.

[58:00] First off, what we have to do is remember the value we want. Because we are not going to be able to use it immediately. So we're going to take the value and stick it in a nice F-box. Like this. That's step one. At the same time as that, we will mute the value of the line tilda. What that means is that we will send it down to zero, and do it very quickly. And now i will go ahead and do bad style and sent the floating point number straight into the message box. OK. Then we will start a delay of five milliseconds. After which, we do the rest of it using the floating point value. Like that. Now I have to clean this up.

[59:15] I don't think we need that zero anymore, because we already did that. OK. So now, what's happening? You know what? Let me make it a little bit better style and put it back in. So that you can see it all in order. So, first thing, first step is to mute the thing. So whatever is happening when I ask it to play a new voice it doesn't do anything except play the old one. Because it might be playing something and the new one is going to start at zero. And if we don't then ramp it to zero, we're going to hear a click because it will jump to zero.

[59:54] So, we spend five milliseconds covering up for ourselves by muting whatever was happening previously. And then; and by the way, the first thing we're going to do is take the floating point number and store it because five milliseconds later, we're going to need it. And five milliseconds later, DEL-S, short for delay; five milliseconds later we will get the value out of the float and do the previous thing, which is bang this to turn the line on and do all this choreography for this line.

[60:27] Now if we do that we have; [sound waveform sounds like sliding fingers across piano keys] perfect, that's the least delay. All right? Now let me move this down so you can actually see where the lines are going. So, this is now an almost complete and almost acceptable sampler. In the sense that I can throw pitches at it and it will play samples back. [sound of sample] And this is...Yeah?

Student Questioner: [61:16] In your trigger bank float bang, is the float doing anything, or...?

Instructor: [61:21] Yeah this is a response to this outlet, and it is being remembered. So the float object - I think this showed up last Tuesday. This is an object which remembers values and then when you give a bang which, the delay will put out, it restores those values.

Student Questioner: [61:40] OK.

Instructor: [61:47] There are other ways to cause a number to get delayed. But in general, when you want to do something after a certain amount of time and have the thing that you had before; you'd have to store it. So this is maybe conceptually something similar. Now the next thing that you might want to do is: gee we got a sampler, let's make it a 100 voice polyphonic. So lets just copy and paste this thing a hundred times? No, you want to do something different. [62:20] So that's coming. But actually I think what's going to have to happen next time is more careful talk about envelopes, and pitches, and transpositions, and stuff like that. Because I'm sure there are a whole lot of misconceptions to clear up.

[62:33] So this is using line tilde and messages, as opposed to using a phaser to look through a sampler. I didn't even show you how to de-clicks samplers that are operating from phasers. There's a totally different strategy for doing that. Everyone's happy? Everyone just wants to leave. [laughter] All right, that will do it for today.

MUS171 02 01

Man 1: So what I'm going to try to do today is push on through the designing samplers of both from the point of view of having a message activated and from the point of view of having to run phasers. So, what happened last time, which might be worth a quick review, is a sampler which is able to make samples with a desired transposition and desired size of place in the sample that you're going to play back, driven from messages situated. [0:34] Using things like start notes, guard samples, these things that play notes. And I just grabbed a copy of that patch. Oh, this is the patch after I edited it to put it up on the website. So this is a bit cleaner than what you saw in class. But this is pretty much where we got. The situation was this. There's a tab read tilde or tab read four tilde, which is playing the sound, and the sounds are being tuned on not by a phaser object. In other words the control that did, but this is the patch coming from tilde objects. From objects that make streaming samples, but from messages.

[1:22] And in this case, there's a metronome, which is spitting out copies of some pitch. [tone]

Man 1: [1:30] There you go. [tone]

Man 1: [1:34] All right. Let's turn that off. Oh, this is now going to be the pitch of the thing. Did you hear it? [tone]

Man 1: [1:43] OK. So what's happening is, there's a metronome twice a second, a number is coming out. Oh right, number boxes! If you send a number box a bang, so it comes out of the metronome it simply outputs what the number was. So also I can also output it just by mousing on the number box itself. [tone]

Man 1: [1:57] Right. And there was work to do, which I always review for you if you are curious about it. But for the moment the idea was to figure out how far you were going to go in a fixed amount of time. And so what happens to the tab read four, is it eventually sends a message to tell it to go somewhere in a certain amount of time. [2:18] The time is computed to be the amount of the number of seconds you want to go

10,000. Sorry. 100,000 samples in. So I just got out a calculator and found that out. There was this number of milliseconds. And now so if you gave it 100,000 that would mean you were playing the thing back in it's own speed. And then also we could compute other speeds and get variables of it. And then it got all Music 170-ish, trying to figure out what numbers to throw it in. OK. Questions about that?

Student: [2:49] How do you select the audio for output?

Man 1: [2:52] Oh. Right. Like I didn't say that. How do you select what part of the audio? Really, there are two messages going in which in fact perhaps I should print. So, let's make a print of it. A regular message print of it because all of this is being done with messages. So, yeah. Actually I'm just realizing, looking at it, that it's always starting from the beginning. So here, we say 60. [tone]

Man 1: [3:26] And what comes out is first the message zero and then the message OK. And then slide to 100,000 approximately in the correct amount of time. And this is the only number in the pair of messages that were changed, and you when have to predict it, we'll compute the value of that. So, in fact I have shown you other ways of making samplers, where you could go in and select where to hit play. And you could easily adapt this to do that. [3:56] What you could do if you want to start somewhere other than zero you would add whatever the number is to this zero. And you would also add it to this number here.

[4:04] And then you would have a thing that has a controllable offset in the sample, right? The other thing that's potentially confusing about this is that you don't hear an entire two point something second long sound. When the thing happens you only hear something that lasts a few milliseconds. [tone]

Man 1: [4:28] And that's happening because the thing is being enveloped. So the signal processing network that you see, really is this line is generating the addresses of the samples in this array here. This line here is turning the thing on and off, I would say fading them in and out. It's being used as an envelope generator bi-cycle synthesizer speed. [4:54] So this is a thing, which starts at zero, ramps up. Oh! I should print that for you too. How about we do this? This is going to be the, what do we call, address. And this is going to be the line controlling the envelope. And into it this line tilde which is doing the envelope is getting messages, here, here, and there. So I should show you all three of those. You won't actually see the timing but you will at least see the sequence. I'll do this again. [tone]

Man 1: [5:37] And I will do it from the PD window, and now what you have is... Oh yeah, I didn't tell you but you can feed print and argument. Ask it to say instead of print colon, something colon. So, the envelope generator is getting set to zero. And then after five milliseconds of wait, of delay, it sets the address to zero and the envelope generator starts ramping up. [6:03] The reason to have that delay is so that you don't hear the address get reset to zero. So what is happening is when you give it a number it doesn't actually start playing a new sample for five milliseconds. Because it has to take care of whatever might have been happening before.

Then after another, oh right so these three things, I'm not sure why they're in that order. But these things should be left at the same time. The address should be zeroed out and then should be ramped to a 100,000. And meanwhile the envelope generator should be turned back on at whatever speed. At a speed which is controlled by whatever [indecipherable 06: [6:20] 53] by one.

[6:51] And then after you're ready to skim the note, which is sometime in the future, you turn the envelope generator back off. And the address is still ramping at that point. It certainly is in this cause, because it's only in a tenth of a second in.

[7:09] But the fact that you multiply it by zero means that whatever is happening to the array the tab before is not being heard. And so the thing is effectively turned off. So again as with oscillators, if you want to turn something off, you don't actually turn the oscillator off in general.

[7:23] But the better way to do it is to cut the amplitude off by multiplying by something that you ramp to zero. And so that's happening here with this line tilde. And now, what bothers me about this, I guess it's all right OK. Are there questions about this.

I have to take the [indecipherable 07: [7:51] 56] out. I think that we're OK with this. So there are two topics today. And they take this thing in a different direction. Each of which is an important direction, but this is the starting point. But they don't actually have anything to do with each other. The first one is, going back and showing how to make something similar but arriving at phasing tilde object which is going to make a looping sample of [indecipherable 08:18] signal.

And the second thing to show you is a more interesting and general thing, which is how to make polyphonic stuff [indecipherable 08: [8:20] 34] . That's to say now I've got a nice voice of this, what would I do if I wanted to have eight of these. And be able to play a chord or plug on [indecipherable 08:42] like polyphonic.

[8:41] And what I want to do is, since it's simpler, is go over the phasing tilde driven thing first. And then that should be easy but then the qualifying voice allocation could be hard. So that I can connect it. So, I can close this and move on to the next thing. This is from last week. There are new objects, but I don't want that I want this.

So, all right, this is... OK, there are things that aren't done here, this is stolen from a patch that sometime last week, I think, and this is the way of reading from a tab ring four if, instead of having a line tilde generated from [indecipherable 09: [9:22] 48] into it, you want to use a phaser to drive it. Now, line tilde will go from anything to anything depending on what messages you're sending, but phaser tilde goes always from zero to one. It's just a phase generator. A phase is considered in cycles.

[10:00] So, with phaser tilde you'll then have to take it and renormalize its output to reach where you want it to go. [making adjustment] so here is the...let's see, we get the table now, too stupid even to close that other one. Oh, still there. So you don't have the...this is what happens when you move a patch from one director to another that uses other files.

[makes sounds] The other thing that's wrong is that I made it bigger in order to put it on the web, now it won't fit on my screen anymore. So what I have to do is go copy this thing. Now, do I have the other window? [mumbles]

[10:32] I just wanted to get this thing to read. Actually that's OK. I'm going to get it along with this stuff which might be using.

I'm going to close that and get over here where we're actually working and put that down, get it out of the way. All right. So, now we're back where we were before. In fact I am going to copy this last here. [indecipherable 11: [11:22] 52] So, this now is a thing where you tell it how many per second you want, so five per second maybe, and then you tell it how big a sample you want it to read in that amount of time. Let's see, it's in hundreds of samples, so I'm going to ask it to read a whole second story. Oh, so this is one per second then. Now we listen to this. [tone]

[12:22] So this is just a looping sampler, but it's a looping sampler that doesn't have a metronome that has to generate messages to make it start up. Instead it's a looping sampler that loops just because phaser tilde likes it to. Here, again, you could wish to fix the problem that whenever it loops it has a discontinuity in the sound.

So to emphasize the discontinuity let me make the thing go faster and have less. That would be 881, 882. Woah. [tone] [indecipherable 13: [12:43] 05] Looking for a bad click. Can't hear the clicks in there. There's a click, but it's getting lost in the sound. Maybe if I make it shorter it'll be more obvious? I'm not sure. Oh, there we go. OK, this is a bad setting because I think I put it right where we're breathing in, but if you start moving around the sample maybe. [tone]

[13:45] I'm trying to find something that has a nice and useful sound and has a click. I'm not succeeding. There we go. All right. OK. So there's a nice sample. This would be a nice thing to be able to have, but it would be nice to be able to have it without the clicks. The clicks, by the way, if we go faster than 30 a second they won't sound like individual clicks, but they'll still be a part of the sound.

[14:12] It'll just make the sound, sound buzzy, so now we've got a nice little buzz generator. OK. So this is a useful tool, but this would be more useful perhaps if it didn't sound buzzy. That buzziness is the same issue as the fact that it was clicking when it was going slower.

[14:43] OK. Is it clear what I'm doing? I don't think I gave you this particular collection of window size and speed before, but what's happening is this number here is being added to the output of the phaser that's already been ranged. So when I'm changing this value I'm changing where in the table it is. OK.

[15:15] Meanwhile, this portion of it is doing nothing but generating a ramp that is repeating at 76 times per second, so that's controlling the pitch that you hear. This is controlling how much of the sample you get. A wonderful thing happens when you change that now. [tone]

[15:36] That kind of stuff. OK, now I'll explain a little bit better why that sounds like that later. You can sort of explain that, although it takes a little bit of work. So let me go back down to a reasonable speed, perhaps 10 a second. At this point I can find the place where it makes it click. Nice. Diesel motor.

[16:05] Anyway, there's a click and now I'm going to try to get rid of it. The way you're going to get rid of it is you're going to envelope, but it's not going to be easy to envelope this using the line tilde because you don't have a source of messages that will tell line tilde to do this thing.

[16:22] With some work you could do it and if you really wanted to you would use the threshold tilde object to try to get a message out when it's phasing across a certain threshold, but you'd have to figure out where on the phaser you'd want to start doing the thing and so on. It would be a lot of work. So less work is just to do the smart signal based thing. Yeah?

Student: [indecipherable 16:45] [16:41]

Man 1: [16:48] Oh, thank you. Yup. This turns it on and off. So this packed zero of 50, this zero is getting overwritten by one or zero depending on whether I turn this thing on or off. That relies on the fact that the toggle switch or the toggle itself outputs numbers, which are one or zero depending on whether it's on or off. OK. Yeah?

Student: [indecipherable 17:13] [17:16]

Man 1: [17:20] Well, I read the comments when I put them up on the website, so it happens afterward, except that last Tuesday never got commented. But if you go looking on the website now you should be finding stuff like this. But they're telegraphic comments. What I'll do is I'll make a copy of this and fix it so you can see the before and after. Oh, you know what? This is the first of the objects I'll introduce for the day, which is make a sub-window please. [18:03] Actually, I've already shown you this, but I'm going to be using it again today so I'll reintroduce it. There we go. Put it here here, All right. Put it in a more decent place and by the way, here we have a sample. OK? So now we have a sub-patch, which has the sample in it so, you don't have to look at it, right?

[18:25] Right. Now, so what we're going to do is again, going to be multiplying tab read four by something, which will make it not click. The only problem is, or the only difference is that the multiplication won't be by a line tilde output, but it will have to be by something else.

[18:43] And what? [cough]

Man 1: [18:44] Well, the answer is deceptively simple. So, phaser. Phaser is going from zero to one and you want it to be zero at the beginning. And then you want to go up to some value like one. And then you want to stay at one for a while, perhaps and then you want to go back down to zero, all right? [19:04] Well you could do that algebraically in a variety of different ways. The simplest way to do it would be this. So, you recall that if you just, Oh!

[19:14] Did I tell you about cosine tilde? I think I threatened to tell you about this, or I didn't, did I? I should've. If I told you about cosine tilde, it would have been first week. This is a thing which takes things from zero to one and turns them into the cosine of zero to one.

[19:32] In fact, at this point, it would be a good thing to have another table to just look at the outputs. So what I'll do is put another table, and an array. It's going to be... I don't know what, scope?

[19:47] And it needs to have, I don't know, some samples in it. I'll make it a tenth of a second. Let's see. OK. Here we are. And now, for instance if I look at what the phaser's putting out... [pause]

Man 1: [20:13] Let's see, OK. Put a button in it so we can see it. Phaser will be putting out zeros until I get it a frequency. Let's give it a frequency of 20 and then we'll see Salter's way, and I made this thing be a tenth of a second long, so I think this will be a little bit, I don't know.. [20:35] Since I made this thing be 20 Hertz, 20 cycles per second, there are two cycles within one-tenth of a second. It's all correct, right? So now, if I just take the cosine of that... [pause]

Man 1: [20:53] I'll get that sort of thing. And that's all right, except.. Oh, and it's value is one at the beginning and into the phaser. So when the phaser amplitude is zero or one, the cosine puts out at one. Right? Yeah.

Student: [21:04] It gives you opposite data?

Man 1: [21:06] Yeah. So now what you need to do is get it to put zero out instead of one. So you need to click upside down. So the way to do this is then to multiply it by a half. [21:21] So I multiply it by minus a half. You can't actually see the fact that it got multiplied by minus a half because you don't see that these points are now the zero points of phase. If I made a graph mode for the phaser and the cosine, you could see that. Now that we've got that we can adjust it so that it goes down to zero instead of going down to minus a half.

You do that by just adding point five. Whoops [indecipherable 22:[21:55] 04] Then tada, we've got a thing that starts at zero, ramps up to one, and then ramps back down to zero. This takes a bit of thought to get figured out, so I should stop here and make sure everyone's with us. Should I try to graph the phaser, too? Why not? What we'll do is let's put another array, but I'll put it in the previous graph. I'll say this is going to be the same size.

[22:53] OK, now we have two things getting shown in the same graph. Now what I'm going to do is make two tab rights controlled by the same button so you can see both of them simultaneously. I'm going to show you the phaser and the result. OK. So the phaser is going from zero to one like this and then jumping back down to zero and the cosine wave that I'm making is going through zero when the phaser is at either end of its trajectory.

[23:30] The original cosine was not suitable because it had two flaws. One is it didn't stop at zero, it went all the way negative, and the second thing is it wasn't at its least value at the transition point of the phaser where you want the thing to be off. This hits its maximum. So the first step is to invert it by multiplying it by minus a half. That gives you this. Now we've got the thing hitting its minimum when the phaser changes phase from one to zero, so this is a good thing, but it's not at zero anymore.

[24:15] It's at minus one half. So now we're going to add a half and then instead of going from minus a half to plus a half it goes from zero to one. If you want to be fancier you could ask for the thing to have a different transition shape or have a different amount of time that it transitions from zero to one instead of the entire half of the cycle that this one takes to transition from zero to one. But for right now I'm just going to keep it as simple as possible with just this. So this is multiplying by minus a half and then adding a half gives you this.

[24:54] This thing, the cosine, is sometimes called a raised cosine. It has a name. It's sometimes called the Hann Window and people use it also to multiply snippets of signal by before they take a 48 transform of it either to do frequency domain of it or convolve it with something else or something like that. So you will see this trick of taking a cosine wave and raising it so that it's tangent to the horizontal axis and then multiplying it by a signal in order to control how it acts.

[25:36] We'll see that again just in the course of the next quarter. So one cycle of this is called a Hann Window sometimes. There's a cycle there. What this patch is doing is doing them end to end. So you can think of the patch not as just making continuous sound but also, if you like, as making a succession of Hann Windows, a pulse train if you like, which is pulsing every time the phaser cycles and the maximum pulse is in the middle of the cycle of the phaser. Yeah?

Student: [26:16] So if you [indecipherable 26:18] .

Man 1: [26:24] If we use this to generate the index into the table that's exactly what would happen and that would be interesting. Yeah. I don't want to hear it right now, but that would be a thing to try. The reason I'm doing this is so we can take the original sample output, the sound I'm multiplying by this, in order to control its amplitude. So instead of going into the tab read four to control the location that it's reading at, I'll take this thing and multiply it by the output. [27:00] Let's see. I should get these two to have the same values. This is 10. This is 14. This is 86. So here's the original one and here's the windowed one. Now, you could either like this more or less. This is not a thing that you have to do because this is right and the other thing's wrong.

It's just that this has a spectrum that more correctly imitates the spectrum or sound of the original sound whereas this has got more highs, and you can like that, but it also has more distortions [indecipherable 27: [27:30] 54] you hear. You hear something that wasn't really present in the original sample. Any questions about this? Yeah?

Student: [indecipherable 29:05] [28:01]

Man 1: [28:08] It is. Yeah, right. Right. So that's another whole thing you could do. Rather than take this thing and take the highs out by windowing it, you could also take the highs out by [indecipherable 28:29] filtering it, but you would also take the highs of the original sample out. Whereas here, if the original sample has highs they'll still be there when you window it. They'll be different in some way, but they'll still be there. Yeah. Other questions? OK. [28:49] So this multiplication by this window, I'm going to clean this up a little bit so you can see a little bit better what's happening. I'm going to make this not collide with itself. So this is the same as that. What I added was this multiplier and that corresponds to the multiplying by the line tilde in the other realization of the sampler.

[29:23] It just had to be done differently because we didn't have a source of...sorry, this line, which is the envelope generator which is controlling the other one, this is feasible because I had a sequence of messages here generated by a metronome.

[29:45] The other one I didn't have the sequence of messages because it was generated by a phaser tilde, which is operated continuously. So I had to do something different. This is perhaps more flexible, but at the same time it's more complicated and there are also some disadvantages to it.

[30:04] In particular, these messages happen between audio samples. In fact, they happen between blocks of audio samples. So what really happens downstairs in PD is that PD grinds out 64 samples at a time in order to be efficient and these messages actually happen on 64 sample boundaries.

[30:25] PD tries to hide this fact, but at the same time what that means is you don't have sample accuracy when the tab read four actually started reading the sample. So if you want that level of accuracy it's more appropriate, I think, to use the signal approach rather than this approach.

[30:43] On the other hand, if you've got midi coming in to start things out you don't have that accuracy anyway and this is the better approach. So they just both coexist and you have to get a sense of when to try one and when to try the other. It's clear what the distinction is between those two.

So that's kind of done this. So now what I'm going to do, not to play with this anymore because now I can watch a whole [indecipherable 31: [31:01] 18] of how to make different window shapes and that can be a lot of fun, but that's for a little bit later on in the quarter I think.

[31:23] But now I want to start working on polyphonic voice allocation so we can turn this thing into fine instruments that we can now play quartz. OK. So to do that the main tool is going to be the fact that you can put things in sub-patches with an interesting twist, and this is all PD bore as opposed to real computer knowledge.

[31:46] The twist is that in PD you can ask it to have a patch loaded from a different file into a sub-window and if you do that then you can have multiple copies of the sub-window and when you edit one of them they will all be edited in a way that stays coherent.

[32:03] This doesn't sound all that important yet because, obviously, you can make eight copies of something and if you want to change it you can just change it in all the eight copies, but it will become important as things become more complicated to be able to keep things coherent.

[32:19] The way to do that is very simple in principle and then in the details it gets complicated, so first I'll show you the simplicity in principle and then I'll make everything unbearable complicated for the next half hour. What I'm going to do is say this. I think, for pedagogical reasons, it would be smarter to start with the other flavor of sampler, which I already closed. So this one. Except that I'm going to rebuild it for the most part.

[32:55] But just for now let's call zero sampling envelope. What we're going to do is make a new patch and then we're going to put an object in. We're just going to say zero sampling envelope. Then, if I open this, I get my nice patch.

Furthermore, if I ask for several of these, I have copies of zero sampling envelope. Notice I'm getting all sorts of errors because I'm doing things that I shouldn't do [indecipherable 33: [33:18] 33] copies of. Things that are named you shouldn't have copies of that have the same name because how do they distinguish it? So that's basically it. I mean, that's how you cause a patch to load other patches.

[33:44] Now if I wanted an eight foot sampler I would load eight of these things and then I would do the hard part, which is figuring out how to get messages into them appropriately to do what it is I wanted to do. So if someone said play three notes I don't want it to tell the same voice to play all three notes.

[34:00] I want to choose three of the voices and assign one note to each of those voices. Then I have to have them be able not to be mixed up about which one is doing one. That takes bookkeeping and attention to detail, which I will now show you how to deal with. Questions on what I've done so far? Yeah? So the first thing I do...

Student: [34:24] With the sub-patch, if you want to re-edit it after you make it is that a problem?

Man 1: [34:28] It's OK, but here's the thing. I can edit this and it won't have been edited in the other one until I hit save in this one and then when I save it the others will all have the same edit. Then the others have the same change. Notice, by the way, this one didn't read the sound files because I told it to read it in a message by name and it didn't know which one of these tables I meant when I named it because they all have the same name, so we have to deal with that. [35:07] Right now, my way of dealing with it is going to be brute force and stupid. I'm going to take this and get rid of it then save so I get rid of it in all three of them and then go back and do what I should have done before. Put it here. By the way, I'll just whack the bang action so we'll get it. So now we have three things, all of which amount to sub-patch. Oh, I just made a bad mistake.

[35:50] I actually saved this patch. Well, it's all right, it's a copy from another day so the original patch is still there. So now we have two different kinds of sub-patches. This one is called by a file name and this one is called by saying PD and this one will be saved as part of this patch. So anything that I put inside here is part of this document and if I change something in here the change is reflected by, let me save this thing so I don't get in trouble later.

This is going to be three. I'm going to be optimistic and call it a polyphonic sampler. All right. Now what we need to do is get messages into sampling [indecipherable 36: [36:25] 48] that would cause it to do things. In fact, since it's a polyphonic sampler, let me do something that's going to be essential for our mental health which is to have the array actually be this one.

[36:58] Whoops, don't have it. All right, I wasn't going to tell you this, but rather than move that one in I'm just going to call it by its relative path name. I'll fix this later. I'm going to have that be the load bang action.

[37:17] That's because I want to be able to have different pitches of it and have you be able to hear it and if it's saying we have another three whatever soft and relaxing it's going to be harder to hear what's going on. So now if I want to hear one voice of it, for instance, I can go in here and say 63. There it is. So now I have a nice monophonic sample. Cool. Now how do I make it polyphonic?

[37:53] First off, we probably shouldn't have the metronome in here, but what we should have is an inlet. Now, when I say inlet notice it puts an inlet in this box. In fact, when I save this box it's going to put an inlet on all three of them and that inlet corresponds to the fact that I said inlet, which is an object whose purpose is to put an inlet on the patch if it ever gets called to sub-patch as an abstraction.

[38:29] OK. So this is one of two ways you can get the messages and signals into and out of sub-patches, which is you just make inlets and outlets in there and connect them. I should say at some point there is such a thing as inlet and also inlet tilde, like this which is the signal version of it which makes an inlet that expects audio signals. That's a different thing than an inlet that expects messages.

You'll see that later in gory detail. I'm going to try to keep things simple today. [indecipherable 39: [39:01] 15] So now this is kind of cool. Save this and close it. Now I've got something where I can say, for instance, number. Whoops there you go. [tone]

[39:31] I think I should go in and make the envelope be a little more assertive I guess. [tone]

[39:45] So 50 milliseconds was too slow for having the voice start in that particular instance. OK. Now, for instance, if I want my major chords I'll say OK why don't we add four to the upper musical third. Actually, let's make it simpler conceptually. So then this is a musical fifth. This is just to prove to you that this thing is actually polyphonic.

Now if I say 60 I get the whole triad. I forget who it was. Colin [indecipherable 40: [40:19] 33] has a lot of music that sounds like this. OK. Anyway, you've all heard this kind of sound, right? Questions about this? Yeah?

Student: [indecipherable 40:43] [40:45]

Man 1: [40:53] Yeah, you have to connect it to the copies or somehow distribute the messages to the copies because this copy is actually going to be doing this pitch and so on. Yeah?

Student: [41:03] Did you say [indecipherable 41:09]

Man 1: [41:09] Oh. Control D would duplicate. Any time you have something that's selected you can hit control D and it duplicates the whole thing. You can even do it to a whole thing like that. Oh, gee, now I've got two of them so I can even do this. [41:30] Well, actually, I can have as many of these now as my computer ha run. You haven't had any trouble with your computer not being able to run things yet probably, but now you will be able to make yourself trouble having your computer run everything. Yeah?

Student: [indecipherable 41:46] [41:44]

Man 1: [41:52] If you like, it's a whole new kind of object that I made today. It's not part of [indecipherable 42:03] , but it's part of my private library.

Student: [indecipherable 42:05] [42:01] or do you see the bottom so it's more together?

Man 1: [42:08] There are two things you can do along those lines.

Student: [indecipherable 42:16] [42:15]

Man 1: [42:20] Yeah, but it would be smarter to have just a single one and have a sequence of different pitches going to it because you don't need them to overlap. In other words, it's easier to control something if you have a modifying synthesizer to start the sequence of things than it is to have a qualifying thing that follows each other. For instance, if you want to say do, re, mi and get a trumpeter to do it you don't get three trumpeters to say each one to play a note.

Student: [indecipherable 42:45] [42:44]

Man 1: [42:48] Yeah. On the other hand, you can, for instance, ask it to do these things [indecipherable 42:58] . This is a slight aside, but there is a wonderful object called pipe Pike, which remembers numbers and puts them out after a delay. So if you want to make rounds with cannons do this...this is not exactly an answer to your question, but it's a related idea. Now if I hit 60 you get [tone] . And now what it'll do, it'll be a major cord. [tone] .

Student: [indecipherable 43:47] [43:43]

Man 1: [43:46] Yup, how is Pike different from delay? Pike will remember as many, OK, first of all delay [indecipherable 43:55] and Pike actually remembers numbers. Pike will remember as many numbers as you give it so that you can do this. [tone] . And you'll get, oh that's not a good example because it's too fast. [44:06] Let's make it be a simpleton. Just a second. Oh wait, there's too much going on, so let me just not have this one. So now I'll say. [tone] . So what's happening, oh I should do it this way. This thing remembers numbers, but it also can remember more than one number at a time so that it will remember a whole sequence of things that happens. So it's actually a memory object as opposed to delay which doesn't remember stuff.

[44:44] This is useful. It's not as useful as you think it's going to be, because it doesn't turn out to be a generally, let's see what am I saying, it doesn't turn out to be generally that frequent that you want to have exactly the same sequence of stuff come out after a delay of time.

[45:06] I mean occasionally you want it, but it's more likely that you want to do something that varies how things change in such a way that Pike no longer becomes the right solution. And there's no way to sort of build it out into something better, it just is what it is. Also, when you send a whole bunch of stuff in then suddenly you might say, "Well actually, why don't you just forget the third and fifth things that I told you but remember all the others."

[45:31] But there's no way in Pike to do that. And so the design pattern in the computer is...it's actually not good to have, to schedule, a whole bunch of stuff under the feature and then have it happen. It's better to always schedule the very next thing that's going to happen in case you want to change the tempo or change some other aspect of what you're going to do, because then you might find out that all that stuff you scheduled you might have to repeat anyway. So Pike does the wrong thing and just schedules a whole bunch of stuff under the feature.

[46:02] So, that was just sort of an answer to the question as opposed to useful information. So here's now, back to the triad generator. Other questions about this? Let's see, why is this not useful? What's the next thing you would want to do? Hook it up to a keyboard maybe, but I don't have a keyboard so that's not a problem for me right now.

[46:35] How about changing the lengths of the notes or changing other qualities of them. Right now they're short and kind of little and maybe we would want to do something that would allow you to say well make me a chord that lasts a half second or a second long, have that be another parameter. That's kind of a typical thing that you might want this to do. So would you want to have a whole bunch of different inlets to this thing?

[47:17] Maybe not because if I decided to have like 10 inlets to control 10 different aspects of how the sample works, this doesn't have 10 controls on it, it has maybe five or six things controlled right now, but by the time you have eight of these and maybe a half dozen inlets on this you have a lot of wires on this.

[47:37] You're going to really want to just pack and unpack, which are things that allow you to take numbers and combine them into messages that have more than one value in them so that you cannot have wires flying all over the place. Each one of them just carries one number. So let me just make the thing now. Why don't I save this one, or rather leave this one the way it is and work on a new one.

[48:05] Save as four. Sample duration. So now zero dot sampling dot envelope, if I go changing it it's going to change it for both of these patches, which could be a good thing, but for right now it's not going to be a good thing. So I'm going to save it as something else. Go in here and say save as. Now it's going to be sampler voice with link duration. That's a terrible idea because now we're going to have to type that whole thing out.

[48:58] If I get one letter wrong it'll fail. OK. Now my plan is going to be I'm going to put lists of two numbers in and the numbers are going to be a pitch and a duration. The pitch will be just what it is and the duration will be in milliseconds. It's going to be easy, right? I'm going to want more of these later, but for right now let's just have one of them.

[49:22] So what we're going to do is here we're going to have to use the pack object to put messages together. To start with, let's just do it the most simple minded way, which is to pack two numbers, one of which is going to supply the duration and one of which is going to supply the pitch. So now I'm going to say duration 1,000 and pitch 60 and it's going to last a second, right? No. Because? Why doesn't this work?

Student: [indecipherable 49:57] [49:53]

Man 1: [49:57] Yeah, it's sort of that. I'm being more simple minded. I didn't change the abstraction to do anything to the second number, so of course it can't do anything with it. I didn't fix it. OK. So go in here. If a number box gets a message with two numbers it just takes the first number. You could do a variety of different things, but that's the way it does it. So what we want to do is just unpack the other number. [50:25] Move this here. So that's all we need is an unpack object, which by default expects two numbers. This one is now going to be a duration in milliseconds, which I think is just going to be the value of this delay. Tada. Oh, delay. The first inlet you send a bang to will schedule a bang to come out after the amount of delay and the second inlet changes the delay time so it receives numbers.

[51:05] As a shortcut, you can feed it a number into the first inlet and that will not only set the delay time but also arm the delay. In other words, also schedule the set off. But this is the more readable way of doing it. I'm just going to set the number here and then all this stuff is going to happen which, by the way, is going to start the board off. So I'm going to save that.

[51:29] Maybe it's already going to work. Great. That's too easy. It's still true that when I tell it something new it has to steal the voice from the old one. Oh, stealing the voice. That's midi talk. Voice stealing is when you have a thing here which is a voice. Voices, that's borrowed from corral talk. Voices is a person standing up saying something, but in computer music talk it's a thing that's able to make one pitch at a time.

[52:06] The idea here is that each pitch is going to be made by one of several identical sub-patches. So that's what voice is. So stealing the voice means if I have the thing playing and suddenly give it another pitch before the first one is done. I want it to cleanly stop playing the previous one and start playing the new one.

[52:34] All the machinery is already in there to do that. What we have to do then is mute the thing by taking the line tilde that drives the output and ramping it hurriedly to zero and then resetting the location of the array to the beginning and starting the whole thing over.

Now we've got everything we need to do this polyphonically except for one other thing which is this: [52:57] now suppose I want to be able to give it a bunch of numbers and have it not steal the same one, but allocate a new voice each time so we can hear them all separately. Then what you need to do, and this is going to be work, then what we need to do is have a bank of these and have each time you give it a new one choose a different one of them to send a message to.

[53:25] It's like an automobile distributor, which will send a spark to the cylinder that's supposed to go off next. So we know how to generate the numbers just fine. That's this kind of patch, which you've already seen. You need a floating point number and then a plus one that makes a loop. Let me just make that loop first and you can see what that's good for. That's counting, but now what we're going to do is we're going to say we want some number of voices.

[54:07] I'll just make there be eight. That's a reasonable number. So what I really want to do is count to eight. Let's say plus one all right, but then modulo eight. Then we're counting zero through seven. Let's see. I have to introduce a new object now. I'll just do that. I won't do the simple one, I'll do the one that we're really going to need. So there's an object sitting here called route, which looks at messages and simply routes them according to what the first number in the message is.

[54:58] So, for instance, if I give it the number zero through seven to look for, if I give it a seven it puts it out this outlet, but if I give it a zero it puts it out that one. One, two, three, four, five, six, seven, zero, and so on. It has, by the way, eight outputs because there's a last one which is what it outputs if it doesn't match any of the numbers I gave it, which is always a possibility.

[55:28] I didn't have to give it exactly these numbers. I could have said I only care about numbers five and six or something like that. But in this case I want all the possible numbers this can generate to make outputs.

[55:40] Now it's even worse than that because what I really want to do is have route, I'm going to just demonstrate this, I want to have route take this kind of message and somehow route that message according to this number. So what I need is a message with three objects in it.

[56:06] So we're going to have to have pack with three numbers in it. The first number's going to have to be the number we route according to. The second and third numbers are going to be these numbers here, except that I want the thing to change whenever I wiggle this number just to make it easy.

[56:26] So what I really want is whenever I get message out of here it's going to stuff these values in here and then make this thing go. So to do that I have to unpack it again. This look ridiculous that we're doing pack followed by unpack, but actually that happens all the time that you have to do that.

[56:50] This is now the value of 1,000, so we'll put it there. This is the value 58 and we'll put it here. Then we're going to bang that, but we care what order we do those two things in because we want to bang this after this number has gone in.

So we need our old friend trigger. I'll use the abbreviation trigger bang float. Let's see. I'm going to bang this, but meanwhile we're going to throw this in here. And then let's just look at that to see how we're doing. Print. All right. Now each time I give it a new number, out comes a triple of numbers which consists of the voice it chose and then the pitch I gave it and then the [indecipherable 57: [57:08] 54] duration. This would be an excellent moment to stop and answer questions.

Student: [58:00] What is the [indecipherable 58:05] ?

Man 1: [58:04] Ah, yeah. I'm not using that yet. What that's going to be good for is this first number is going to be which of the eight sub-windows I want to send a message to and route is going to do that for us. Yeah?

Student: [indecipherable 58:25] [58:21]

Man 1: [58:25] No, it'll strip the number. So that's the next thing to do is just send this thing in here. So I'm going to call this before and after. Oh, sorry about the caps. Caps lock. OK. Now we start doing this. Actually, after came before, but anyway, whenever the message starts with zero, the route zero matched and it came out here. By the way, it gave me the other two numbers, which is what I want to feed the sub-patch with. [59:18] Now we're done because. I'm going to leave that there. Now we just make eight of these and have them talk to this. All right. Are we going to get it all on one screen? I could have chosen a smaller number. But no. So I'm going to make eight of these. You can tell I've done this before. I know which way to organize them. Let's see. I didn't make the right number then yet.

[60:10] Computer scientists hate this because I'm actually manually iterating something and, of course, the program should be smart enough of creating the dimensions for me. [tone]

[60:23] Polyphonic sampling. Alright. Or actually to really prove this is doing what I think it's doing let's give it a nice chord like this. 60. Notice the commas. That means send both messages please. All right, this will be horrifying. You can put it right there. [tone, laughter]

[61:00] Tada.

[61:02] So what's happening here is this is the same thing if I had somehow managed to simultaneously type all these numbers I wanted here. So each one of these things generates a message and, in fact, you see here what it did to them. [tone]

[61:19] I don't know, but whatever that was, six numbers, oh wait it starts here, each got assigned a voice number and then here are the six pitches that you see that I typed in there and here's the durations I asked for. OK. More on this next time.

MUS171_02_03

Puckette: [0:00] I think is the best way of getting people to have these. And they won't be any more intelligent or better organized than the lecturers themselves were, so don't expect miracles. [0:13] The other thing that Joe has done, I don't know if this is on the Web or not, is he's actually made an index in all the patches that appear on the Web of where all the objects appear.

[0:24] So that any of you are curious about an object, if you have this anyway. If you're curious about an object, you can find out the names of all the patches that use that object in one way or another.

[0:35] So in particular if you're looking for asterisk tilde you'll get more than a column of little things. So this is some kind of script that I guess you can run any time you want to.

[0:44] He does W-GET, which is this great program that will just recursively download a whole glob of Web, and then you can get all the files and enjoy them.

[0:54] Other thing about that is, I'm repeating. Since everything is in a new directory, things like sound files, voice.wav, are appearing on multiple ones.

[1:03] You don't have to grab that anew each time if you want to save bandwidth. You can just do one or another of several things, like copy an old one into the directory.

[1:13] I haven't shown you how to make PD search outside of the current directory, so for the moment just assume that you're going to need to have all the files that you're using in the directory that you're saving the patch into.

[1:27] OK. So anyway, that's the story about the DVD. And of course, Joe isn't stopping recording the lectures, so the collection will grow eventually to be 20 of them.

[1:40] There's one sonic aspect of sampling that I didn't stop and explain as well as I wanted to explain last time. So, I want to go back and just do it harder.

[1:55] And that is the business about, if you have a looping sampler, there are two regimes in which you can hear the thing. One is where you have the sampler playing back either looped or not looped but slowly enough that you hear individual starts and stops as events.

[2:12] And that would mean perhaps playing fewer than 30 different things a second. So, here the example would be... Whoops, that's not a good example.

[audio tone]

Puckette: [2:24] So, now you just hear one of the Deuce phonemes six times a second. Right? This by the way, is a copy, annotated and then partly de-annotated, of one of the patches that I made last class. And this is up on the Web. [2:42] And my purpose in making the patch was to show you how to add the envelope-generator shaping to a phaser, but now what I want to do is continuing to look at the phaser-driven sampler, because it's the more appropriate way to look at this.

[2:56] To compare again what happens at low and high frequencies of phaser. What happens at low frequencies is kind of exactly what you would expect, which is ...

[audio tone]

Puckette: [3:10] You hear stuff at a certain number of times per second, and furthermore. OK, so what's happening here is this 34 is the beginning place. Oh, it's in 100s of samples, which means 441 of these make a second. [3:26] So this is the beginning place in, sorry, not in that table, in this table. Whoa, how did that work? I guess I have another patch open and it's reading out of that table, sorry. I'm going to get us all confused if I don't close this. Or I'm going to get me all confused, anyway. Now how do I get rid of that? It could be something here.

[3:49] OK. Come on. All right. Back to normal. All right. So, what's happening is we're looking at this array, and we're looking, starting at whatever 34 corresponds to here. And then we are sliding forward 78 of them. Is that me shaking? How is that happening?

Student 1: [4:15] Maybe it's just a sound of an earthquake.

Puckette: [4:17] I think we were having a nice little earthquake. We just invented a new kind of seismograph. Maybe I'm wrong. Maybe it's something totally anodyne. [4:27] OK. So this 78 is now being multiplied by the phaser, so the phaser is going from zero to one, and now we're making the phaser have a range, which, in 100s, anyway, starts at 34 and goes to 34 plus 78. So, this is the amount of sample you hear each time. So that ...

[audio tone]

Puckette: [4:46] Once you've got the units right, anyway. If this thing times this thing equals 1, you've got the straight-ahead transposition, that is to say, the original pitch. And what would that be? Actually, let me change the unit so that it's easier to think about. I'm going to make this be units of 441, so that these are hundredths of a second. And now this'll be something like nine, and this'll be something like 20. [audio tone]

Puckette: [5:15] Yeah, there we go. So now what's happening is this is .017 seconds, because 441 samples is 0.01 seconds at our sample rate. And this 0.17, that times 6, if you multiply it out in your head, turns out to be very close to 1. [5:36] Actually, one-sixth is, as you all know, 0.1666, so we'll do this. And now we'll get the original transposition. This number in hundredths, that is to say, 0.1666 repeating times six is 1. And now if I choose some other pair of numbers which also multiplies out to 1, such as make that five and make that 20 hundredths ...

[audio tone]

Puckette: [6:02] Then I've changed the speed at which the thing's happening but I haven't changed pitch that you hear, the transposition that you hear, of the sample. OK. So, the transposition that you're hear is proportional both to this number. [audio tone]

Puckette: [6:23] And also this number. OK. Next thing about that is this. This is what you hear these numbers are below about 30. That is to say, the number of things you can hear as happening as discrete events. So even if I push this to 20 and drop this to five... [6:48] Whoops!

[audio tone]

Puckette: [6:50] And find a good place. You hear the event as a repeating thing but if I push this past about 30 you won't anymore. I don't know 30. It's iffy. But of course, this is a very particular spiky part of the sample. If I get somewhere else in the sample... [audio tone]

Puckette: [7:08] It becomes very hard to hear that as a sequence of things and easy to hear as just a continuum. If I push this higher, then at some point your perception of this as having a pitch becomes more important than your perception of it as number of times per second. [7:25] This is just acoustics, if you do anything repeatedly fast enough it becomes a pitch. Right. For instance, if I want to use this Western style musically we'd want a MIDI to frequency converter. Then I can say "Well, play that for me at play it at middle C, please." Which is 261 Hertz and now we've got middle C.

[audio tone]

Puckette: [8:02] I could check it on the piano if I could. Oh yeah, let's go down an octave. That's a little too harsh. Oh. Why does that sound so ugly? There's something going on here that I don't understand. [8:24] That should be a smoother sound than that, but we'll go for it for now. If I ever find my mistake I'll tell you. So, at this point, we're playing five milliseconds of sample at a time. But we're transposing up a quite a bit because we're playing 130 of these per second.

[8:44] So really, this should last something under one millisecond. Should last about point seven milliseconds, I think. Oh, this will sound better now, maybe.Yeah, there we go.

[audio tone]

Puckette: [9:00] All right. Then, we have the sample as before. [audio tone]

Puckette: [9:11] Those are timbres taken from the voice. In fact, I could even try to move this continuously, except I'm going to get in trouble. You'll see. [audio tone]

Puckette: [9:21] It's a little ugly because you hear it click every time this thing changes discontinuously. [audio tone]

Puckette: [9:31] So, let's go find soft. OK. And now. So, I mean, I've proved it. When you heard it coursing over the sample, then you heard the timbres or the vowel and consonants of the original sample going by. So, right now, you can believe that this thing. [audio tone]

Puckette: [9:54] Is the O of the short O of soft. Right? [9:57] That's of course, assuming I'm playing everything at the correct speed. In other words, I've made this and this multiply to approximately one. I didn't check that it was exactly. Oh, I could. I could. Nah, that's enough. I could take this and divide 1000 by it, and put that there. Actually, that would be a good thing to do because I'm going to have to do it later.

[10:20] So, to do that, we're going to divide something, but we're not going to divide it by 1000. We want to divide 1000 by it. And, that's the thing I haven't shown you how to do yet. But, it's not hard. So, what you want to do is you want to take 1000. Control alt two. Sorry, 100. We're going to talk 100 divided by this, right? Because it's in hundreds of samples.

[10:48] So, we're going to take this and put it here. But then, we're going to have to bang the 100. And so, we need our old friend trigger to make the thing get banged at the appropriate time. So now we say, "Trigger, bang, float."

[11:12] Sorry. Let's make this be reasonable. OK. And now, I'll show you. Oh,, rather than show it to you, I'm just going to use it. And, you'll see. So now, I'll say, "48 again, please." And, dada. It computed the exact value, which was 100 divided by this, which is the number of 100ths of a second, which would be the period if this is the frequency.

[11:35] This would be a good place to stop for questions.

Student 2: [11:42] Can you explain the abbreviation?

Puckette: [11:43] Oh. OK. So, this is abbreviation for trigger, bang, float. And, what trigger does is two things. It formats messages for you. [11:58] So, there's a floating point message coming in. And, what we want coming out is a bang here to send this 100, and then actually, before that a floating point number, which is whatever we sent in.

[12:10] So, this divide needs to receive 100 here. And then, it needs to receive this 130.8 here. But, the divide divides when it receives this number. And so, it needs to receive this number after it has already gotten that one so that it will do the division correctly. Sorry, there are two questions. You first?

Student 3: [12:33] So what's the purpose of the bang?

Puckette: [12:36] Well, the bang is simply to get this 100 to send out, because if this 100 doesn't get sent out, this thing will never divide.

Student 3: [12:43] So there's a bang and then speaker's input?

Puckette: [12:47] It sends the float, and then it sends a bang, which sends this 100.

Student 3: [12:53] And that's going to be top down, or?

Puckette: [12:57] Uh, let's see. This is going to be 100 over this floating point number. OK, let me see if I can make this into a more synthetic example. Let's make a copy of this. [13:19] All right. So a number comes in, number comes out. And whatever I put in, what comes out should be 100 divided by this number. OK, so how do I do that?

[13:31] The number goes in, and I ask it first to put the number in. OK, this is the divider, so what I need to do with the divider is first put five in here and then put 100 in there, so the thing will do 100 divided by five. All right?

[13:46] Yeah, lots of questions, let's see, I don't even know who to ask. OK.

Student 4: [13:50] So, the float, how do you know it goes in first?

Puckette: [13:52] Oh, trigger is defined to put its outputs out in right to left orders. And in general, objects that have multiple outlets that spit out messages at the same time. Another example would be unpack. Always do it from right to left order, so that you know what order they're coming into a subsequent box in. [14:21] OK. This order doesn't mean that time is passing. All this happens in one instant of time, or to put it a different way, between two audio samples, but the same two. In other words, it happens in a moment that isn't allocated any time of its own.

[14:37] And yet it has an order, because you have to have this thing go in the correct order, otherwise you could never program it. OK. This PD is not a programming language, and this is asking PD to do a programming language-ish thing, which it can do.

[14:54] But obviously, if you just had BASIC or C or something like that, you could do this particular thing a lot easier. You just say, "100 divided by f," or whatever it is.

[15:06] But since we're in a graphical programming language, it's great for doing things like bashing signals around like that. And it's a little clunky for doing things like saying, "100 over f." This is the easiest way I know how to do that. There are others.

[15:20] Other questions about that? Yeah.

Student 5: [15:25] So, for when you're changing the range for the plus value, so if you wanted to. [15:32] Not put this object value back to this top half.

Puckette: [15:34] This value, OK.

Student 5: [15:35] Yeah. You just have to unpack or on the line? Or if you wanted not to click?

Puckette: [15:41] Wanted not to click. In other words... Oh, you mean, if I wanted these things to be their values when the patch shows up?

Student 5: [15:50] Yeah, like cycle through that 60 so...

Puckette: [15:54] Oh, I see. Yeah. So right now I haven't done anything to automate that. I've just left it as a number box. But in fact, that could be messages coming in from some other place if I wanted to. [16:05] And then I could use the number box to override it, or I could just have it be what it is. But it's easiest, just for the purposes of showing the patch, it's easiest to have everything just be controlled by hand. Yeah?

Student 6: [16:19] Now if you put more floats in there? And you put a metronome in there? Can you create a loop to sequence things?

Puckette: Sure. Well, OK, sure, there would be some work to do, because... If for instance, I had a bunch of numbers in a message, like that. If I sent those to this, for instance, this number would become one. [16:51] And then it would become three and then it would become five and then it would become seven and seven would win. If I wanted it to do those things and have it be one, then three at a later moment in time, and then five and then seven, which would be more like a sequence...

[17:03] Then the easiest way to do that would be to store those numbers in a table or an array, and then read them out of the array using a metronome. Yeah?

Student 7: [17:12] So, outlets, when a send a number like from left to right sending from a point one?

Puckette: [17:20] OK, yeah. Outlets are right to left. Things inside a message box are left to right or in order that they are as text.

Student 7: [17:28] Oh. OK.

Puckette: [17:32] Yeah, that's confusing, isn't it? It never occurred to me to think about those two things in the same thought. Yeah?

Student 8: [17:40] Can we depend on everything other than the trigger having regular notes?

Puckette: [17:50] Well, what you can't do, or what you can't depend on, is this. I'm going to erase this now. I might have shown you this already. I can't remember. I'm going to take this number and just square by multiplying it by itself, and I'll do it that way. [18:17] This doesn't work because this multiplier got these two values all right, but it got this value first and that value afterward, and as a result it did the wrong multiply. So, to make that guarantee you would put a trigger in to make sure everything happened in the order that you wanted.

[18:37] In other words, you would say, "Oh, I'll keep the wrong one and the right one. "You would say, "Oh! Go away." I should look at my screen.

[18:59] Sorry. OK, trigger float, float. And now we know that this float will come first and this float will come second and then it will do my squaring job right, whereas here it didn't.

Student 9: [19:15] So not objects other than trigger when unpacking?

Puckette: [19:20] Yes. If the outlets are all, what's the right word? Well, an example of something that doesn't do that is route because it doesn't ever put something out more than one outlet as a result of a single message. [19:37] But things that do put out more than one thing as a result of a single method, such as trigger or unpack or note in if you're doing MIDI. I don't have a lot of other examples here.

[19:53] Those are all arranged from right to left. That's simply because right to left is appropriate for inlets because it's best to get things left last. Yeah?

Student 10: [20:05] Would it be one multiply then by that parameter and the symbol bar would you get the separation?

Puckette: [20:17] Yeah. It gets ugly. What's a good example of that? Yeah. I have a good example, but it's too complicated and I'd have to explain it. If you want to do A plus B times C, for instance. [20:38] OK, so you put B and C into the multiplier and then you have A and you want to put it into an adder. Then you have to decide OK, do I want that number to change every time I change any of A, B, or C or do I want it to change only when C changes or so on like that?

[20:53] You could want it either way. For instance, if you're playing a MIDI keyboard and you want to transpose, when you change the transposition you don't want it to replay the note. That just means you want the next note you play to have the new transposition. So, in that case it would be appropriate not to recalculate the pitches of all the notes because you changed the transposition. Maybe.

[21:14] But in other situations, you want the output to change when any of the inputs change so it's always correct according to the last inputs. Those are just two different situations. So, you can get PD to do it one or the other. The way to get it always to follow what you do is to use triggers everywhere that you need to in order to make sure that everybody's leftmost inlet gets whacked last and then everything will stay up to date.

[21:45] That was a little abstract, but we'll hit examples of that I think later on, I think, in the quarter. Especially if I go back and show you pitch to MIDI conversion in general. Any other questions? That was useful because I think there were a bunch of things I wasn't saying that I should have been saying that, that helped clear up. OK. I don't want to leave this here, though. This is cluttering my screen. OK.

[22:20] So, now what we have is I've got this number here and I'm computing 100 divided by to put here. So now, I can do this kind of thing.

[audio tone]

Puckette: [22:33] It's not perfectly smooth, but it's now figuring out what I have to put here to put timbre. In order to get the sample to play back at the recorded speed, regardless of how many times per second I'm asking it to play back. [22:49] So, the faster I ask it to play back, the less of it I can play back each time if I'm trying to constrain the speed. Now the other thing to mention about this is that. OK.

[audio tone]

Puckette: [22:58] Now we've got a nice pitch and now we could just go and change this segment size without changing the number of times per second and then we get this kind of thing. Notice there are clicks in the sound. [23:20] That's because when I change this discontinuously or when I change this according to the mouse it changes the thing I'm multiplying the phaser by, so it causes the read into the table here to move discontinuously, which is a click.

[23:33] So if I want to do this correctly I'd have to use a line object to de-click this. Nonetheless, I now have my first nice timbre whammy bar that you can use to make wonderful computer music. Right?

[23:46] In other words, this is the first example, except for that attempt that I made to explain FM, which I shouldn't have ever done. This is the first example of a situation where you can change a control and it will actually make an audible change to the timbre of the sound.

[23:59] Which, of course, is what computer music was supposed to be about back in the day. There will be many more examples of this, because eventually you will be able to make filters and frequency modulation once I get that straightened out and things like that.

[24:14] Right now, the reason this is happening, I can actually show you using this table. So, just for emphasis sake...

[audio tone]

Puckette: [24:27] If we go down to a frequency where you can actually hear the things then what I'm doing is actually changing the size of the sample we're playing, which is changing the transposition. But then if you make that happen at an audible rate then we're changing the transposition again, It doesn't change the pitch that you hear. [audio tone]

[24:47] .

Puckette: [24:54] Because the pitch that you hear is now no longer the pitch of the original sample. But it is the new pitch from just being imposed on it by the fact that I'm reading it at a fast rate. And that is in fact the continuum. [audio tone]

Puckette: [25:11] So for instance you can say, "Now I have a continuous way of moving back and forth between pitchy sounds and sampled sounds." [audio tone]

Puckette: [25:31] And that a lot of people heard for the first time when Fatboy Slim did that song "Rockefeller Skank." That was this.

Puckette: [25:41] OK, questions about that?

Student 11: [25:45] Yeah can you actually get any deeper?

Puckette: [25:47] No. [audience laughter]

[25:48] .

Puckette: [25:52] We're not using Superfly. He was using commercial stuff. All right. [audio tone]

Puckette: [26:04] So now, this which becomes a transposition when we do it slow is a timbre change when you do it fast. [audio tone]

Puckette: [26:19] And another way of seeing that is to take this thing and graph it. Which is what I now propose to do by using these graphics over here. So I'm going to keep the phaser and instead of graphing the envelope, which is what this graph is here to do earlier. [26:32] I just want to graph the output of the tab read object. Oh yuck! Oh, I see. I've got this thing so high that the graphing doesn't look good. So let's change the properties here. I had 44 that's a tenth of a second.

[26:55] I'm going to need like a 50th of a second. So let's make this a thousand. That's actually a 44.1th of a second.

[27:07] And let's see. Will this work? Still looks, oh whoops! I didn't do that quite right, I need to change this to. Oh, and this! Sorry I got two tables in there. So two arrays in there, which is why I got confused.

[27:28] So OK now what we see is there's a nice phaser and it's going at 48 times a second. And each time it goes you see the same wave form, which is in fact the wave form that you've stolen from the original recording.

[27:42] And now if I ask it to read, so if I ask it to graph again, you'll see that it's in a different place, because I don't know how to make the graph start right when the thing wraps around.

[27:51] That's for later, but you can see that it's the same wave form. Now if I say, "OK, do that but make this number larger. In other words, make there be more of the sample." Actually, let's have there be less of the sample first. I'll drop it to one.

[28:05] Then when I graph it, you'll see that there was just less wave form stuffed into there. Is this clear? Now you can do wonderful things because, what would happen if you just stuck a sinusoid into here?

[28:27] Actually, that would be a useful thing to be able to do, so let's do it. So now what we're going to do is we're going to say, "Tab right," and I give the name of the thing Table 203A.

[28:45] Now I'm just going take a nice sinusoid, maybe 110 Hertz, so it won't be too different from the sound of the dude's voice. And I don't want it to be quite that loud because I don't have a good volume control, so I'm going to protect us by multiplying this by some small number.

[29:11] And now we need a button to make this thing happen. Wait two seconds and sinusoid, although you can't really see it as such. And now here sinusoid!

[29:29] Except, of course, the sinusoid has a discontinuity every time we restart the sample. And now the bigger a chunk I read of the sample, the more cycles of the sinusoid have to slap into the same amount of time.

[29:46] If you can imagine what that sounds like, it's got this period, so the pitch of it isn't changing, but you can tell by looking at it that it's got a lot of stuff at this frequency and maybe not a whole lot of stuff at fundamental.

[30:02] So now if we listen to it...

[audio tone]

Puckette: [30:06] We've made ourselves a nice little formant. Is that clear?

Student 12: [30:28] Are those pops because of that repeating of the disc and...?

Puckette: [30:31] Yeah, the pops happen every time I change this. And what that does is that changes the amount I'm multiplying this phaser by. If I happened to do that right when the phaser was at zero, it wouldn't make a discontinuous change.

Student 12: [30:44] So you're not possibly ranging?

Puckette: [30:46] As it is, it gets up to a half or something like that, and then suddenly it re-ranges, and that pushes it off to a different value. [30:54] I'll show you things about that later after I've stuffed all these more fundamental ideas into your heads. The object you want is sample and hold because you want to sample this number only when the phaser changes cycles. OK. But, of course, the thing that I had originally was not the sinusoid, it was the man's voice. Then you get this look and this sound.

[audio tone]

Puckette: [31:44] And, of course, different parts of the sound follow different things, but they all have the same basic behavior. All right. So that is basically a thing about sampling that everyone can use. Questions about this before I go on to other things? [32:11] What I want to do now is go back to polyphony and voice management, which I started talking about last time, but about which there is much more to say. Yeah. Everyone's tired of this topic anyway. Yeah?

Student 13: [32:26] What about the patches?

Puckette: [32:28] Oh Sure. Yeah, this is patched. I have to change the order of these because I didn't plan very well, but all the patches up until last time are up on the web. It takes me a day or two, but I eventually get around to putting good comments on that actually mean something when putting them up. [32:51] Yeah, this is a good trick. OK. So now I have several matters to go over.

[33:02] First off, here's the example of a multiple voice sampler.

[audio tone]

[33:09] That was the thing I showed you guys last time. Right? The basic deal was, the new objects you needed were just the notion of having a so-called abstraction.

[33:22] That's to say, being able to put a PD patch inside a box, which would, in fact, be a sub-window. And route, which is this object, which takes a message in with any number of numbers and looks at the first numbers and, depending on what it is, puts it out one of the several outputs.

[33:44] Since there are eight arguments here there's nine outputs because there's one for it that didn't match anything. That way you can gang several objects together. Although here I happen to know that this number will always be zero to seven because I have this mod eight here.

[34:06] This is an example of polyphonic voice allocation and there are other examples of things that you can do with polyphonic voice allocation, which I will actually show you by taking you through some pre-existing patches.

[34:19] A slight departure from the custom. So, this right now is just a copy of what there was last time. The first thing I want to mention before I go ripping through the pre-existing examples is a thing about abstractions I haven't told you about, which is this.

[34:42] Here's another abstraction. Get in edit mode. You can do something like... Well, I have an abstraction, which I moved into this directory beforehand called output tilde.This is interesting because it's not just an abstraction that has a patch inside it, but it also has controls, which it printed on its faceplate, if you like. So, this is a technique for being able to make things like modular synthesizers

MIDI, which are modules with audio inputs and outputs, but which also have controls on the knobs.

[35:21] So, I'm just going to tell you now that this thing exists and how you can deal with it. First off, it's just an object like any other, so I can retype this and it would become some other kind of object. But this is a particular type of abstraction, which does this for us.

[35:41] If you want to see inside it, it is not good enough to click on it because clicking on it means doing that kind of thing. Oh yeah, by the way, this is an output level in decibels and you would use this if you wanted to have a patch whose output level you could control on the fly, which is a good thing to be able to do.

[36:03] The other thing about this is, of course you might want to be able to see what's inside it, but clicking on it doesn't do that because clicking on it does that kind of stuff.So instead, you control click or right click depending on what kind of mouse you have and say open. Then you will see the contents of the thing as an abstraction.

[36:27] Then it's nothing much more than what you've seen already except there are some tricks here that I haven't told you about yet that I hope to tell you about today.Then closing it's the same as it always is. You just close it. All right. This has a name. It's called a graph on parent abstraction. The reason being that it shows a certain portion of its GUI objects, its controls, on its own surface as an object.

[37:00] You can learn how to make these and so on. Well, OK, properties. Notice here it says graph on parent? You have to click that and then that turns your abstraction into one of these things, but then there are things you might want to set that I have to explain in detail later.

[37:19] I'm telling you that because I'm going to take you through some patches that are going to use this and I want you to know about this thing's existence. If you want to use this you have to get this thing and copy it into your own directory.

[37:29] I'll throw it up on the website, but it is also on all of your computers because it's actually in PD in the help patches. And that is in fact, where we're headed next. I've been studiously avoiding the help patches because I've been building everything from scratch, but by the time we're getting into voice-banking stuff, some of the examples maybe are better just looked at and described than they are built up from scratch.

[37:56] You have to do several things at once and get them all working together. So, I'm just going to close this and open some other patches that do other things. And the answer is no I don't want to save anything.

[38:11] So, in PD, if you say, "Help," one of the things you can get is this browser. And the browser has far too much stuff in it, but if you go looking for Pure Data... Oh, by the way, if you have PD Extended this really has a lot of stuff. Right now is merely has a bunch of useless stuff. Pure Data, audio examples. Oh right, there's a manual. This has a bunch of HTML stuff, which is very telegraphic and, what's the right word? Short. Control examples, later. Audio examples, which are examples about how to do synthesis, processing, and analysis of sounds.

[38:53] These examples correspond to the textbook, which I haven't been referring to very religiously, but we're now somewhere in chapter four of the textbook, talking about voice allocation. The A's are all chapter one, the B's chapter two and so on like that.

[39:10] And so, all of this stuff, if you want to see more description of what it's about and how it works and why, look in the book in chapter four and the book actually talks about all of these patches in some detail. The things I wanted to show you were, for instance...additive synthesis. Here's the reason I showed you output is because we're now going to be starting to use output to set the...

[bell tones]

Puckette: [39:39] Volumes of things. Notice these values are in decibels, which means 100 is full-blast and 75 is likely to be quiet. [bell tones]

Puckette: [39:49] So typically I put these close to 100. [bell tones]

Puckette: [39:54] This means nothing to people who weren't in the computer-music scene in the sixties and seventies, and then it means a lot because this is one of the classical computer-music designs done by Jean-Claude Risset at Bell Laboratories in Murray Hill back in the day when people were first doing things with computers. [40:12] And what it is... let's see if I can make it sound better.

[bell tones]

Puckette: [40:20] It's called a Risset Bell. And it's just Jean Claude Risset having found some parameters somewhere that make a nice... [bell tones]

Puckette: [40:28] ...bell sound, that can do that kind of stuff, or this kind of stuff, [bell tones]

Puckette: [40:37] Or of course, this kind of stuff. [bell tones]

Puckette: [40:43] Actually, better yet... [bell tones]

Puckette: [40:46] Ta-da. It's not rocket science. What it is, the old term for this was additive synthesis. It's synthesis technique that you all know because all you do is add up oscillators. And the oscillators in the examples I've shown you have all been tuned to be multiples of a fundamental frequency so that they all fuse into a harmonic sound. In this case it's additive synthesis in the sense that it's a bunch of sinusoidal oscillators being added together, but the oscillators are imitating the modes of vibration of a bell. [41:22] Although, I don't know if this corresponds to a real bell...

[bell tones]

Puckette: [41:25] ...or something fanciful. [bell tones]

Puckette: [41:30] And, the controls that you get are going to correspond exactly to the controls you saw in the sampler example last time, which is, you get to control pitch and duration. [41:41] Differences between this and the previous example, there are going to be a couple of differences when you get into the voices, but there's a huge conceptual difference. Which is that this isn't a polyphonic instrument.

[41:56] It only plays one note. But there's still voices, adding up to that one note. And the voices are, instead of making different notes, making different partials which added up into a single note.

[42:13] It's polyphony in one way of thinking, because it's realized like a polyphonic instrument. But psychologically, it's monophonic. So it's showing the use of voicing and abstractions, in a non-voice allocating way.

[42:31] A key difference between this and the previous example is there's no route object. In the previous example, you decided every time you asked to play a note, which of the voices was going to play it. Here, all the voices play every single note. Yeah?

Student 14: [42:45] And those voices are that partial voice?

Puckette: [42:47] Yes, right. So now I have to show you what's in there.

Student 14: [42:50] So do we have another test?

Puckette: [42:52] Yeah. Next thing is, notice that I'm feeding the thing arguments. I'll tell you what the arguments mean in a moment when I show you what's inside it. But when you have an abstraction, which is to say an object which is a separate patch being written into a window... [43:11] Usually, not quite always, but in most situations, you're going to want to throw in arguments to specialize it to do one thing or another. For instance in this case, each of these partials has to know which partial it is, so they don't all decide to play the first partial together. They each have to play different partials. So each one of them has to know which partial it is.

[43:33] And to do that, you have to pass each one arguments that disambiguate them. That tell them how they are going to be specialized. All right?

[43:45] Other huge difference and a small related difference to the huge difference, there aren't wires going into these things in inlets. So the mechanism for getting messages inside here is not inlets, it's send and receive.

[44:02] For the simple reason that, and here's the small conceptual difference, frequency and duration and trigger are being sent separately in this design. I thought, at the time, this was actually a simplifying instrument, although I don't know now whether I think it's more simple or more complicated. So in this particular design you give it any pitch and any duration you want and then you say, "OK go ahead and then it does it for you."

[44:31] You could now take a packed pair of pitch and duration and make it act like the previous one if you used an unpack and a trigger in order to send messages in the right order to duration and frequency and trigger. So, what's going to happen inside here is each one of these things is going to have a receive frequency and a receive duration, which will tell it the global frequency and duration of the tone, and then it's going to have another receive for trigger, which will set the thing off.

[45:07] All right. Now, just to hearken back to Music 170 for a moment, this is imitating how a bell actually vibrates.

[45:20] One way, maybe the best way, of thinking how a metallic object would vibrate is you strike it and in striking it you activate the modes of vibration the thing has. Those are functions of the shape and construction of the object itself, not of how you whacked it. They don't move. But the modes each have a frequency and a time constant. The frequency is how fast the mode vibrates and the time constant is how fast it's damped.

[45:46] Whether it vibrates forever, which it would do if it had no damping at all, or whether it dies out very, very quickly. So what's happening here is we're pretending there's a metal object that has 11 modes in it, each of which has a different frequency and a different damping.

[46:06] But then there are global frequency and damping or duration, which is damping, controls that are multiplied by each one's individual frequency and damping factor. Now, go look inside partial, which I've been putting off. Here's how you do a partial. This is being done in gory detail in the most careful explained, boring, pedagogical possible way. Let's see. I can do this.

[46:57] OK. I haven't told you this and probably shouldn't tell you this now, but line tilde generates line segments and if you want a line segment to feel like an exponential just raise it to the fourth power.

[47:12] Then instead of going down like this in time it goes down like this in time. Then it's not exactly an exponential, but it's pretty good. What that means exactly is explained in chapter two of the book, so you can look it up if you don't believe me. But the signal processing aspect of this is being done here. All right, so what you see is an oscillator and a line tilde.

[47:44] I'm raising the line tilde to the fourth power so it will go down more exponentially than just a line segment would. Then I'm multiplying it by the oscillator then I'm introducing a new object, throw tilde, which I'll not dwell on right now. I didn't even tell you about sin tilde. This is a way of avoiding having an outlet by asking a p to set up a summing bus. In other words, what's happening is somewhere there's a catch tilde sum, which is automatically going to get the sum of all these signals.

[48:24] It's a good thing, but it's not a thing that you absolutely need to know right now, so I'm going to not dwell on it but just sort of say that's what's happening. This is like dac tilde. In fact, it could be dac tilde except that I wanted to be able to use the output thing I showed you earlier. So, instead of just putting the dac right here I put a throw and it all collects in this catch and then it goes to this output object where I'm controlling the amplitude for output. Al right.

Student 15: [48:56] Can you play them all at the same time?

Puckette: [48:59] Actually, so I can control the volumes of them all in parallel. Let's see, how can I get this thing on the screen? Alt, alt, alt. Here's a thing that's not easy to do on a Macintosh. OK, so this is controlling globally the amplitude of the thing by virtue of the fact that each one of these partials, using the throw object, has added itself at full strength to this catch, which then is going to be sent to the output, which will control its amplitude. So that's controlling all the amplitudes in parallel. Yeah?

Student 15: [49:38] So it's like a send and receive for multiple things going to the send box?

Puckette: [49:41] Right. So send and receive work for messages. There's a send tilde and a receive tilde for signals, but in audio signal land the situation's a little bit more complicated. Because it would not be clear what to do if you had a bunch of sends and a bunch of receives. In message land, you just send all the sends to all the receives.

Student 15: [50:00] So like a patch base?

Puckette: [50:01] Yeah. It is exactly like the patch base. So here what's happening is all of the throws are throwing to the single catch. There's a fan-in thing, which is a summing bus, which is done by throw and catch. And there's also a fan-out mechanism, which is accomplished by send tilde and receive tilde, which I haven't shown you. Yeah.

Student 15: [50:18] So you can sequence them as well, going into the catch?

Puckette: [50:22] No. It's like cables. So they're happening in parallel.

Student: [50:27] OK.

Puckette: [50:28] Right. So sequence... yeah. Only messages can be sequenced. Signals can't really be sequenced. They're all happening all the time, which is one of the good reasons to have messages around. OK. Now the partial, is showing off... Sorry. I have window-sized trouble here. So the arguments to this partial where these numbers one, one, 0.56, and zero. Those numbers are what showed up here. Where are they? [51:09] Alt, alt, alt. Oh, there. So these numbers one, one, 0.56, zero, which are customizing this particular partial, are displayed here. And they are expanded by dollar one. Where did dollar one go? Here... No, sorry. That was wrong. I don't see dollar one yet. Here in object boxes if you put a dollar sign, it is the argument you got called with as an abstraction.

[51:48] And here there is one of the half dozen truly hard to understand things about PD because dollar signs are also available inside message boxes, except that they have a different function inside message boxes than they do inside objects.

[52:09] This is all for a good reason because conceptually it is all very simple, even though it looks completely arbitrary and stupid. So, what I want to do is show you just to help confuse you now, I'm going to show you message boxes and dollar sign in it.

[52:27] So, one, two, three, we all know what this will do. Let's see I'll put a number in. And I'll print it out. And then, no matter what I put in, the print out says, "Hi, I was one, two, and three." All right. You can't even see that this was getting repeated. Oh, I'll hit alt again. Alt, alt, alt. OK. Now suppose I want these numbers to vary as a function of this, I can say, for instance dollar one. And dollar one means, give me the value that was the number that came in this message box.

[53:13] I haven't shown you this because it was possible to do this already with PAC. So, the easy way of getting this effect, I had to do it once before when I had to do it in order to make triples of those numbers that went to the voices of the polyphonic sampler, two days ago.

[53:31] Right? So this is another function of dollar signs. You can put a dollar sign inside a message and it will insert the number that you put in as part of the message. And the one here, the dollar one, is which of these numbers it is.

[53:47] If for instance, I gave it a PAC message, let's see this example. Now dollar one is 45. That and dollar two is 78. Those of you who have programmed Shell scripts in either Macintosh, LAN or Linux, this is exactly the same argument that holds in Shell scripts.

[54:22] Also, I think Perl may look this way, I'm not sure about that. The computer scientists have a name for this type of expansion, but I forget what it is. Yeah?

Student 16: [54:32] Can you add another inlet to the message box?

Puckette: [54:34] No. You could in MACs but here you actually have to use a PAC object. Whoops. So for instance, here's another perfectly good way to make a message that has a bunch of variables in it. And now actually, at this point, I could now do this. And now I could set dollar two this way, and dollar one down this way. OK. Since you have already seen PAC this is not something that you need. [55:14] You will need it later perhaps, because certain things like... Oh right! Tables, if you want to tell a table what size it is, you say resize and then you give it a number. If you want that number to be a variable, then you have to say re-size dollar one inside a message box.

[55:32] That's the only way to generate that kind of message. So this is a more general facility than PAC. And as a result, it runs a little slower than PAC too, so it's not always the right way to do things in every case.

[55:44] Now I told you that because I'm trying to confuse so that you will later on be enlightened. The confusing thing is this. If you put dollar one inside a message box, or dollar anything inside a message box, it means the incoming message that's making me now send the message, right?

[56:03] Because message boxes take messages in and then send messages out. Object boxes, what you put in an object box, is a message. It's a message whereby PD makes the object and that message can have dollar sign arguments in it, too, but those dollar sign arguments are the arguments to the patch because the patch is running the message to make the object. Yeah?

Student 17: [56:30] Can you send object names that are variables inside? Create?

Puckette: [56:34] Yeah, if you really want to. For instance, apply plus five, if you're a list person. Well, let's make a nice thing called apply. You don't want to do this. Apply is now going to say, "Hmm apply dollar one, dollar two. " Oh, come on, let me type in there. [57:15] All right. Then I'll make a new outlet just for fun. PD hates me for doing this. It has no idea what dollar one means in an object box like this, but now if I retype this I now have a nice little plus five thing. OK, go find a use for that. Oh, does it actually work?

[57:44] I don't think I've done this in a decade, so. It works. Yeah, don't do it. It's stupid. Oh yeah, dollar signs can either be numbers or they can be symbols. Symbols, which means strings, things that aren't numbers like file names or names of objects like plus.

[58:07] Those are different kinds of data. For the most part I've only been using numbers except in the very rare occasions where I've had to specify a file name, like voice.wav. I'm doing that on purpose. I'm trying not to get any deeper into the soup of language complexity than one absolutely has to in order to do computer music. So this is message boxes and that apply example, was object boxes, but it was object boxes in such an abstruse example that I don't want to talk about anymore.

[58:44] Oh, almost saving there, which is bad. Don't want to put it there. Let's just not save this. This is better explained somewhere else probably.

[58:58] That was all to say this. Next it's time to say, "What do these things mean?" This thing, this thing, that thing, and that thing. Well, this is a relative amplitude. This is a relative duration, this is a relative frequency, and this is a D tune, which is to say a frequency offset, a frequency that's added to it.

[59:27] So we're talking about individual sinusoids, which are imitating individual modes at the sound of the bell. So each mode might have a different amplitude.

[59:36] That's actually not acoustically correct because the amplitude would depend on how you struck the thing, but we're just pretending now. So each one is given its own amplitude, its own time constant, its own damping, which, in this example, just because Risset did it this way, is realized simply by changing the overall duration of the amount of time it rings.

[60:01] The correct thing to do, acoustically, would be to have it be a dying exponential and to have it go on forever, but that's not practical, so we just give each partial a duration.

[60:13] So first argument is amplitude, second argument is duration, third argument is frequency. Now I have to tell you something else. All these things, well not all these things, but the frequency and the duration are also controlled globally by the controls in the outer patch, which was here.

[audio tone]

Puckette: [60:39] So there's a pitch and a duration. Those are things, which once they've gotten to the right units, will have to be multiplied by the relative frequencies in order to figure out the real frequency of the partial. [60:53] So this partial, which has a relative frequency of 0.56, what that really means is that it's 0.56 times the global frequency that I asked the bell to be so that when I change this frequency the frequencies of all the partials change in parallel.

[audio tone]

Puckette: [61:19] All right. [61:20] So how do we do that? What that means is we're going to have to take this thing which is being sent and multiply it by this number and that will become the frequency that we send the oscillator. Except I didn't tell you another thing. You get a D tune in here, which is after you multiply this factor by the global frequency then add an offset so that the partials can be paired in beating combinations. This is all just history.

[61:56] You can read this in books like the book by Charles Dodge. Here are two partials that have the same relative frequency, but one of them has an offset of nothing and the other has an offset of one Hertz. So, after they compute their base frequencies, this one adds one Hertz to itself so that it will beat with this one once per second, according to another principle you saw in Music 170. OK. So how do you do that?

[62:28] What that means is that the oscillator that's inside here will get a frequency, in fact, which is 0.56 times this frequency plus this offset. So is that true? So, here we are. Every time we get a trigger we re-compute the frequency. Oh, by the way, the thing is designed so that you hit the trigger and then you can start mousing away at the frequency.

[62:51] And it doesn't change the frequency of the bell while it's ringing. Which, you know, is one possible way you could design the instrument. It's the way the original instrument worked, so maybe it's a good way for us to do it.

[63:02] OK, so this is dollar one, this is dollar two, this is dollar three, and this is dollar four. Those are the arguments that we've got as a sub-object, as an abstraction. So dollar three is a... dollar three will come out... OK, this, we're just going to store dollar three, and then when this thing gives us a bang, we will read dollar three.

[63:31] Oh, this is one voice that I'm triggering now, not the whole bell.

[audio tone]

Puckette: [63:35] And dollar three will come out here, it's 0.56, and it's going to get multiplied by a value, which is received frequency, which got sent from the main patch. [63:43] And then we're going to add dollar four, which is zero, which is the de-tuning. And that is the frequency for the oscillator. And all that stuff got done as messages, you can see the thin traces, and then the oscillator then is the first thing in this chain that makes an audio signal. All right. That's the easy part. Well, it's the easy part. It does have two variables controlling it as opposed to one, but it's easy in the sense that now you just feed it as the frequency of the oscillator.

[64:17] The amplitude control, as you know, things have to get turned on and then turned off, and so the amplitude control has more pussy-footing around setting the amplitude. So I'll show you that next. That's this. So the amplitude is being controlled by line tilde.

[64:35] Now, the amplitude, there are two things that are being told us that are relevant to the amplitude. One is the first argument of the abstraction, this one here, is the relative amplitude of this voice. And the other is the length of time, the relative duration of the voice, because to know how to do an envelope generator you're going to have to know how high to make it, which is the amplitude that you're going to reach, and then you have to know how long to make it.

[65:10] OK. So here's the line tilde that's going to do it, and by the way, I raised the line tilde to the fourth power, I didn't say how, but if you take a signal and square it, then you get the signal to the second power.

[65:22] And then if you take that thing and square it, then you get the signal to the fourth power. And why fourth power, not squared? Just so the thing will really hug zero and go up like that, because that resembles more the exponential curve that this thing would really be doing in real life.

[65:38] It's just hand waving. There is no deep psycho-acoustical truth to this that I know of. OK. So what we have to do is configure or connect two messages to line tilde.This is a little bit like what we had to do for the sampler, except in the opposite order. You have to turn it on and then we have to turn it off. In the sampler actually, we had to do three things. We had to mute and then turn it on, and then turn it off. So this is simpler than the sampler in terms of the sequence of the operations that has to take place.

[66:07] OK. So the attack portion over here is look up dollar one which is our amplitude. The attack portion is just going togo up to the amplitude and do it in five milliseconds, come what may. Why five milliseconds? It's a fast attack but it's not quite a click-y attack. Yeah?

Student 18: [66:25] What is happening to the dollar sign going in there?

Puckette: [66:28] The dollar sign? OK. So the dollar sign tells the float to substitute one of these arguments to the abstraction.

Student 18: [66:37] I mean it's in there trying to get a number?

Puckette: [66:39] Yeah. So dollar one gets this one, dollar two gets that one, dollar three gets that one and so on.

Student 18: [66:43] It's a positional number.

Puckette: [66:45] It's a positional argument. That's the correct word for it. It's a positional argument.

Student 18: [66:48] OK. A positional parameter?

Puckette: [66:51] A positional parameter? I'm not sure. OK. So this one, when it's expanded, rather than when it's built, this is going to be float one. And when it gets a bang, out is going to come the number one. So this is just going to be one. We're going to multiply by 0.1 to save our ears. Everything is going to get multiplied by 0.1 in parallels. And then because we are going to raise this to the fourth power. [67:17] We should probably take the fourth root of this, so when it gets raised by the fourth power, it's the right number. So to take a fourth root, you take a square root twice. And then here, I didn't have to do this. I should have done this. I think in the current context anyway pack zero five, which would then take this value and would replace zero with it and then pack a five which would be the number of milliseconds the line would ramp to that value in.

[67:47] But instead, because I had just spent pages in the book describing this dollar one madness, at this point it seemed appropriate to put dollar signs in messages. And so this is, make a message whose value is this number and then five. And here it is again the distinction between objects and messages and dollar signs.

[68:09] Dollar sign inside an object means is the context of when the patch is created. You have to build this thing, and so the dollar sign is evaluated when the patch is created. And it's created with these arguments. Dollar sign here is in the context of the message that it is sent, which is the thing that causes to send a message itself. So this is in the run time context if you like, and this is in the build time context, or something like that. Yeah.

Student 19: [68:47] Why do you need the trigger there when one of the bang has gone through?

Puckette: [68:52] You don't. The only reason that trigger's there is to make this thing a little prettier. Oh, it's psychological. It's to group these two things. But, in fact, it has no function because you're quite right, this delay would ensure this thing happens after this thing anyway. OK. If only this existed and not that then you would get this. [audio tone]

Puckette: [69:24] The attack shape of the envelope and no decay shape. I'm going to do this to make it shut up. [audio tone]

Puckette: [69:31] As it is, you do that and then you decay and decaying is actually easier than attacking. We wait five milliseconds because the attack took five milliseconds. We wait for the line tilde to get up to its apex and then we start the decay portion of the envelope, which is look up the duration. OK, so the attack is five milliseconds regardless of the duration. The decay, now, we know what the target is, it's going to go down to zero so it'll shut up. But what we don't know yet is how long it's going to take until it gets there. [70:06] That we now have to computer. So we're going to make a message again, but whereas the message in this case had a constant time but a variable amount that went up. In this case it has a constant that it goes down to, which is zero, but a variable amount of time to do it in.

[70:29] How do you compute that? Well, after five milliseconds when it's time to start doing this, pick up the value dollar two, which is this value. It's one here, but it will be different numbers for the different partials. Multiply it by the global duration. That was sent to us there.

Student 20: [70:52] Was it 800?

Puckette: [70:53] Yeah, it's 800. So this duration is 800. But this is now one, so this is going to be one times 800, which is 800 again, so this will be the message zero 800. So that means this thing should last eight tenths of a second. [audio tone]

[71:09] Hard to tell because it dies out inaudibly. Now if I get another one out, this one, this one will be higher and lower.

[audio tone]

[71:23] There's a medium longish one. Here's a short-ish one with a higher pitch and a shorter decay.

[audio tone]

[71:32] That's because here, the arguments are we're louder.

[71:38] We're a relative loudness of 1.46, relative duration only a quarter as long, so this thing lasts one quarter as long as the other one did. Then the frequency is two as opposed to this frequency, which was 0.56, so it's almost two octaves up. Yeah?

Student 21: [72:04] Decays can multiply by the total duration message, I'm still confused about that dollar sign one. I understand that dollar sign two, that's the parameter, but partials, the second one, what's coming out of that?

Puckette: [72:23] So what goes in here is dollar two. Dollar two here is 0.25, so it's going to multiply 0.25 by 800, which is duration, so it will get 200 here. So this will become the message zero 200.

Student 21: [72:39] And the other one is computed?

Puckette: [72:41] The other one here. I don't know what the amplitude is, but it would be that number five. So this is zero 200 and blah five, something five. All right. So if you hear all these things together. [audio tone]

Puckette: [73:05] You get beautiful computer music. [audio tone]

Puckette: [73:13] There's some psycho-acoustics there, too. You can pick this up. This was in the help browser, blah, blah, blah D7additive.PD. This is the first example in the PD examples, I think, of something that does multiple voices and there are two others of note here that I want to show you that I will get to next time. [73:50] Oh, homework. The homework is just what you've heard done. Do I have the patch that does it? I can't remember. Open. Hello!

[74:23] Come on, mister. Close that. This is wrong.

[audio tone]

[74:32] Oh, there it is.

[audio tone]

Puckette: [74:41] Isn't that its own antithesis? [audio tone]

Puckette: [74:44] So you all know how to do this because it's nothing but a looping sample, you're just going to have to figure out how to get it the appropriate original pitch, but last a lot longer than it would have lasted. And envelope it so it doesn't sound buzzy. So, it's a straightforward application of what I've been showing you the past couple of days. And it is 10 'til. We should stop.

MUS171_02_08

Puckette: ...Explain some of the other features about how to make things be working polyphonically, and then if you want to know about the previous homework, which is several ideas old, I can go back and tell you how that patch looks. Otherwise, I can plow forward into other techniques, which are going to be ring modulation and frequency modulation, which are the next audio techniques that I want to show people. So by way of audio techniques, you haven't had a whole bunch of stuff in this course yet. [0:36] All that I've shown you is how to make sinusoids and add them. And done a little bit of frequency modulation, and then done some sampling. There's much more to be said about all those things, especially about sampling, which I've really given short shrift too, which is sort of things infinitely deep well of cool things that you can do.

[0:56] Well, rather than spend just another two weeks doing sampling, since everyone knows sampling is there. I want to plow forward and go for brick instead. So what's coming up most likely is going to be, after we get through the polyphonic voice allocation trickery that would be good to know, ring and frequency modulation.

[1:21] And wave shaping which is also a form of modulation, basically the idea of throwing things through non-linear functions that are planned in whatever appropriate way there is that can change their sounds. And then after that, there's a topic, which I call designer spectra, which is given all of the above how would you actually make a sound whose spectrum is this kind of shape?

[1:51] So, you can actually make your own spectral shapes if you want. So, I might only give that a day or so. And after that it's going to be delay lines, out of specialized delay lines one can make filters. So if I can get through decently well using delay lines and using and making filters then that might be a good rest of the quarter. [laughter]

Puckette: [2:16] OK. So what I'm going to do right now, is just start with PD lore and move into the modulation stuff. But the PD lore here, the multiple voice thing? This corresponds roughly to chapter four of the book. And then when we get into modulation this will be chapter five. Designer spectra are chapter six and then delays is chapter seven. Filter is chapter eight. [2:40] So it's all pretty much according to the syllabus, theoretically. The first thing about that is... I don't know if you guys looked at this or not, but are there questions about how this whole thing works now? So, this what you are looking at is one of these. And in this particular situation, I've avoided having leads going into or lines, patches going into and of the sub-patches, or connections going out of them.

[3:13] I'm using non-local connections everywhere, which are in the form of receive and throw tilde. So one thing that I want to do today is talk about local and non-local connections and what they are good for and how you can make them. Local connections, by that I mean just making lines between things in a single window, and non-local connections meaning things like throw and patch are some of these.

[3:41] I'm just going to proceed by example. So, just to remind you the thing that is most likely to be confusing to you about this patch is an important thing to be confused about, so that you can try to get unconfused about it over time, which is the dollar sign mechanism and how its use spins out differently in object boxes versus message boxes.

[4:08] All right. So an object box is like this one. Dollar three means 0.56 because that's what was thrown. This thing has arguments when it was made. Whereas message boxes, this dollar one is whatever comes down this line because that's what is thrown to it is arguments when that message is passed, which is to say when the message box sent a message as opposed to when the thing is made.

[4:33] The deeper truth is that objects really are messages. These are messages that go to a special factory object if you like, which just makes objects for PD itself. So there is actually one unified thing in PD, by which messages get passed down lines and by which also these objects get built. And if you look a little deeper into it, we'll find out that you can use that factor to make patches that change themselves like that, if you care about that kind of thing. All right. [pause]

Puckette: [5:08] That's the main bit of stuff about this that you might not have known before. Yeah.

Student: [5:13] When you were giving that scratch sheet why would you make multiple lists and multiple objects from them?

Puckette: [5:20] I think the next example will make that clear, and there's some funny stuff I have to show you. Yeah. Actually rather than wait for the next example to make that clear, let me just show you what the answer is and then when the example comes up, you will see it in that context. That might be better.

Student: [5:38] Mm-hm.

Puckette: [5:38] OK. So, abstractions are a form of sub-patch. All right. So, this window is actually inside this box. And you can make this box have inlets and outlets and this window will be able to get the inlets and throw things to the outlets. But you don't have to use the abstraction mechanism to make that be true. So there are two possible ways of having objects inside. Sorry, whole patches inside objects.... [6:06] One of which is this way which is you just name a file. The file's name is partial dot PD. The other of them is simpler. Let's see, let's make a new patch. And the simpler thing is this. Font, 16 degrees. You can just say, P.D. and then give it a name, a name without any spaces in it, as usual, because in P.D., spaces are delimiters between arguments. And now this thing is inside this box. And the same way as was true with the so-called abstractions that you saw in that other example.

[6:49] The word abstraction refers to the fact that you are in fact loading a patch into a sub-window as opposed to just editing it yourself, into D patch, they call it. So here we can do things like this. Let's give ourselves an object and call it an inlet. And that, as soon as I made it, it made one of these things happen. And furthermore, if I make more of them, they show up as I'm making them. And the same is true for outlets.

[7:31] Oh, just to confuse matters further, you can have inlets and outlets, but they can come in control and signal varieties. If you say inlet tilde and outlet tilde, then you can put audio signals into and get audio signals out of the inlets and outlets. Otherwise you're doing messages. And you can't mix them, unfortunately. That's a limitation. So the other thing about that is, well, all right, let's get a nice number box here.

And we'll make a stupid patch, which is, it just takes whatever comes in and throws it out at the output. In fact, I can even do better than that. I can say, oscillate at 440, and I'll make that talk to this. We'll go to this one, because it's an inlet, not an inlet tilde. But this one it can connect to, because that's a signal one. And maybe, if I have the file here, I can see, I'll put, yes, and now we've got a great little, pass through a [inaudible 08: [8:02] 40] , it doesn't work.

[8:41] Oh. Why didn't it work? ... I didn't hear you, but ... yeah. That's it. It's like, duh, let's do this. And then we have a little pass-through thing. OK. Similarly here, we've got messages going in, and again, duh, but let's do it. Now we've got that going on. Now, how do you know which is which? It's stupid, but, it's the only possible way you could do it.

[9:17] These things, however they appear from left to right is how these appear from left to right there. Yeah. So I could do this. And now I've got sort of a little brain-melt device. And you can do that too. That's the only way you could really do it. It'd still work ... in other words, moving things around shouldn't change how the thing functions.

[9:55] And the only way to not change how it functions is to sort of fix it so it functions the same. So it actually switches the inlets and the outlets around in real time as you're moving the objects around. There's no other real way to do it. All right, so that's inlets and outlets, and maybe answered your question about how you figure out which is which. They're proportionally spaced across here.

[10:17] There's one other little thing to know, which is this. If I said P.D. and then had a bunch of inlets and outlets like this, this will happen to you eventually even if it hasn't happened yet, at some point you get to the point where it's just black with inlets and outlets. And then how do you know what you're connecting to? And the answer is really stupid. The answer is, just give it a nice long argument like that, so that they get spaced out.

[10:52] And if the name you gave it wasn't long enough, just add some hyphens to it or something to make it long enough. Some day there'll be a way of controlling the widths, but right now, the width of a box is just the width of the text that it fits into, so if you want to change it, you just change the width of the thing. Yeah. That's enough of that. So that's inlets and outlets and sub-patches.

[11:25] And the other things that I want you to be able to know is that the, sort of collection of new objects that we're dealing with that have to do with getting stuff into and out of sub-patches, one collection of things was indeed inlets and outlets and, oh, I said tilde but, inlet tilde and outlet tilde. And then there's also, I've already shown you throw and catch, which are things that set up summing buses.

[11:54] The place that showed up was here, inside the partial again. The last thing that happens is we do a throw tilde to, and then you give it a name, which is sum. This is like [] read almost, sorry, this is like table, making a table or making a send. This defines a thing whose name is sum. And then anyone who wants to can refer to it by saying catch tilde and then the word sum again. And that in this patch is done down here.

[12:33] Catch tilde sum and then output. All right, this can't be earthquakes, this is just vibrations of some other kind. OK. And there's more, probably, which I explained last time. And you can now look it up on the DVD, so I won't repeat it. OK.

[12:54] So I want to move to another example that just shows you how to arrange copies of the sub-patch, which are made using the abstraction mechanism in a couple different ways just to give you programming paradigms, if you like, or ways of putting things together. None of these is the sort of final answer to how you should do this.

[13:17] These are all sort of alternatives that you will find a personal style that attracts to one or another and so on like that. This one is D07. I think there are two others in the series. Here's a cool trick. This is an added synthesis patch that allows you to draw a spectrum. Why isn't this letting me move it? Come on. Alright, got rid of that. Where are we? I must have miniaturized it somehow. I don't know how to scroll this. Yeah, there we go.

[14:30] So this is not drawing a wave form, this is drawing a spectrum. So I can do this and get stuff that has plenty of high frequencies or I can do this and get the low frequencies instead. Or, if you like, I can put in formats and I can pretend that I'm making vocal synthesis. Vocal synthesis is more believable if it's moving around and things like that. So what's happening here is there must be an abstraction somewhere because we have a whole bunch of oscillators.

[15:11] Each one is playing a partial of this. I think there might be 36 partials in this sound. OK, how do I make this shut up? Over here. Scroll. There. There's an oscillator that, just to be on the sane side, I put in a sub-patch and it consists of an abstraction called spectrum partial of which there are 30 voices. What does the abstraction do?

[15:49] Well, what it has to be able to do is figure out what pitch it should have and then go to this table and find the point which is at the pitch and figure out what its volume should be, so what its amplitude should be, I should say.

[16:05] So what's happening here is the pitches are all determined simply by find out what the fundamental pitch and I know what partial I am I can compute my pitch, but I also want to know what my amplitude is and the amplitude is being set by this table, which I can then change. So this is a way of, I don't know what, it's a sort of a super Hammond organ, if you'd like.

[16:32] So how would you do that? So, again, we're using the throw and catch mechanism for getting the audio signals in and out and send and receive in order to get messages in and out. Again, just as in the bell example, we have a problem disambiguating what we're doing so that each one of them can make its own partial and not have all 30 of them make the same partial.

[16:59] So the one that gets the argument 25 should do the 25th partial and so on like that. So you have to be able to do that using the dollar sign mechanism, that's to say the argument passing mechanism for abstractions, which is that this dollar sign one expands to the number one, but in the window which is this next one it will expand to two and so on like that.

So then what are we going to do? One little thing that I should probably say first because it's a detail, but it's a fundamental detail, is this: [17:23] there's no way in particular in PD that you can get told when a table changes value. It's something that ought to be there, but just isn't, so this thing does the incorrect thing according to computer science of polling.

[17:52] That's to say what it does is some number times a second. Each of the oscillators looks at the table in order to figure out what its amplitude should be. From somewhere, which I can show you later, there's a send poll table that has just a metronome sending bangs that are going out at some 30 times per second I think. Questions.

Student: [18:16] What is Moses?

Puckette: [18:17] Oh, I haven't told you Moses. Yeah. OK, so Moses. Here we go. Moses is this: you put a number in and if it's bigger than the argument it goes out one output and if it's less than the argument it goes out the other. It's a reference to the Red Sea thing. You can change the value tin by changing this argument, so if I want 30 to be this flip point then I do that and things 30 and more go this way and things up to 29 go out there. [18:54] It's also floating points, so 29 and a half goes out there, but as soon as we hit 30 we'll get it out here. So that's Moses. Moses is similar to route.

[19:13] Route is another thing which takes messages in and puts it out in a different place depending on what the message is. The difference is that moses is restricted to only two outputs, because it's a different kind of a thing, and it only takes numbers, it doesn't take entire messages. So there's moses, which, by the way, now that I've done that, I should put on the new patches list. [silence]

[19:48] And this one we actually really saw today, random I'm not sure we'll get to yet. [silence]

Puckette: [20:05] Something like that. OK. Good question. Are there other objects here? No, not that I see. I think I told you about tabread4~, which is a signal object which reads tables interpolatingly. Well, you can do it with messages just by not putting a total there. Dbtorms, I think I told you about. OK. So what happens here it this thing is getting banged 30 times a second and every time that happens we're going to make a decision about what our amplitude is. [20:43] Meanwhile, there is pitch to figure out. The pitch is simpler because the pitch is just what got set to by the number box. The pitch of this thing depends. The frequency of the oscillator which is one partial can be computed by looking at the pitch which is sent from this message box out here. Pitch, right? Now we were looking in the oscillator bank and then we were looking at spectrum partial one.

There's pitch coming back. Then we do mtof to figure out what is Hertz, but then this oscillator is not going to play at some fundamental frequency, it's going to play at some multiple of the fundamental given by the partial method. So here is partial number one, so we multiply by one, but if it was partial 30 we'd multiply by 30 to figure out what the frequency really is that the oscillator is going [inaudible 21: [21:13] 42] . Is this clear?

Student: [21:44] One more time.

Puckette: [21:46] One more time? OK. So there are thirty of these things. What we're going to do is we're going to make a sound by adding up 30 sinusoids which will be two different partials of the fundamental frequencies that we want. Here's the fundamental frequency. Dollar one is the number of the partial that we're now looking at, and this number goes from one to 30 depending on which of these boxes we're in and the oscillator bank. One through 30. And, I'll do this later but that's the frequency in hertz and all were going to see down there is an oscillator. Oscillator times and then there's a throw that you can't see but this is scrolled off the screen. Other questions? Yeah.

Student: [inaudible 22:40] [22:40]

Puckette: [22:44] What does this do? Oh, yeah I'll have to get to that. But I'll be in two words and then I'll tell you later. Basically silence. Zero doesn't have a good zero is good silence and not just zero. So it splits zero off and makes true zero as opposed to converting. Other questions? Or just general global cloudy confusion?

Student: [23:14] How?

Puckette: [23:16] How receives yeah. OK. So this is now going through an oscillator and multiplier which is getting multiplied by the gain of the saucer the amplitude of this oscillator. And then its, then there's a throw down here that you don't see. OK. The fun part is all this stuff where we compute the what amplitude the oscillator should have. And that recalled, I want to be able to draw that on the tape. And so what that means is I should figure out where that oscillator should be in the table. And then do a read in the table to see how high the thing should be above nothing. So how do you do that. OK. So the table is not a choice. This is a designer's choice. What should the horizontal units of the table be. Should they be the hertz or should they be video something else. In this particular case it turned out to be more effective or more appropriate to put it in mini units. So that the table itself when you look at it is arranged by pitch not by frequency. Why? Because if it were frequency you'd be using half the table just to describe everything from data to forte or something like that and then there would be a lot of detail in the lower parts that you wouldn't be able to see because it would be too squashed together. So this is better. [24:41] And I even played with here. These are mini factors which are mini pitches so that 60 here corresponds to 260 hertz. This stuff here is very low frequency. But up here these are frequencies which are reasonable for four months of a vowel or something like that. In fact I did this so that I could just play with vocal synthesis. So if you want to do vocal synthesis with this you would figure out where you want the residence of an imaginary vocal track to be. And then you would put bumps at the points at the frequencies of those residences and then you would make those oscillators that line up the screen. They seem to be louder. Alright. Is that reasonably clear. OK. So again if we want these to be addressed in meeting units then what we need to do is after we found the frequency of the oscillator that you want, OK. We're going to get the pitch back.

[25:45] So what's happening here is every time we get a bang in poll table we're going to figure out an amplitude so we're going to have to then get the value of the frequency. We've seen this before in envelope generator controls where you have a delay, but after the delay you want to set something off with a variable message.

[26:01] The only way to be able to do that is to be able to store the value of the variable that you're going to have to recall after the delay. This is a similar situation where someone's given us the pitch at one moment in time, but someone's asking us to use it in another, so we use a float object which stores the frequency so we can get the frequency back when we need it to compute the amplitude.

So frequency is changing only at the moment when we change the values in the number box which controls the pitch of the sound, whereas the amplitude is changing on a different clock, which is whenever we [inaudible 26: [26:23] 41] the table. Alright. So then we recall the pitch, which is in pitch units. We converted back to pitch here. The frequency, which is in cycles per second or hertz, that's appropriate for talking to the oscillator, but for looking up the amplitude in the table we should be indexed by midi pitch because that was the more convenient way to have the table be arranged.

[27:03] So we just convert from frequency back to midi, so here we have this kind of odd sequence of steps, which you might find in more than one place actually. Look at a pitch but change it to frequency but then multiply it by a partial number and then change it back to pitch. There are alternative ways of doing that, but that's maybe the conceptually simplest way of finding a pitch of a partial of a note. Alright. Then I'll tell you about this a little later.

[27:36] If whammy bar is zero that means nothing happens here because we subtract zero, but that's a way of taking the whole table and sliding it backward and forward it we want, but it's not necessary. We're going to get the pitch back out and we're going to read a proposed amplitude out of the spectrum table. It turned out to be a good idea to have a 50 DB throw from the bottom of the table to the top. Why 50? Because 50 just turns out to be a good number between very loud and very soft.

[28:11] 100 is too much. 100 is the difference between deafening and inaudible, whereas 50 is the difference between up and down in audio. Actually, if you want proof of that go look at a mixer and go look at the DB scale and you'll see they like a throw of about 50 DB as opposed to 100. Sort of standard mixers, typically. There's the Moses object. If we get a positive number out of here. Now, let me show you where that's happening. It's here.

[28:43] So what's being said is if this number's zero or even negative, if I'm being sloppy about it, we want to get true zero out so we can really shut it up, that one frequency or another. But if it's positive we want to take that number and consider it as decibels going from 50 to 100.

[29:03] Why 50 to 100? That's because PD has this sort of informal standard of 100 DB is full blast. That's enforced or suggested by the DB to RNS object, where if you say 100 DB it will way one. It's an arbitrary thing. 100 decibels can be any loudness that you want it to if you're just talking relative levels, but in PD the convention is to have 100 decibels mean one, or full blast.

[29:31] So these values go from zero to 50 on the table, so what we're going to do is add 50 to it to get from 50 to 100 and then we're going to convert that into a linear amplitude and then we're ready to multiply the oscillator by that to make it the amplitude of the oscillator. That happens like this. Get back in the voice of the abstraction. So if it's zero, that's to say if it's less than one, we're just going to make the amplitude be zero.

[30:03] If it's one or more then we're going to add 50 to it and run DB to RNS. So zero comes out true zero, but one will get added 50 to it and then DB/RNS will give us roughly 0.003, minus 50 DB, so there will be non-zero values ranging from 0.003 all the way up to one. In fact, it's not strictly limited to one because if I drew the table out of bounds, above the top, then it would be more than one. Yeah?

Student: [30:38] If you have a negative number will it [inaudible 30:39] or is that still...?

Puckette: [30:41] It will still go out this side. It will still give us zero. OK. Then to make it sound good, pack three or some value to it. Who knows which value is best. What that does is that will mean whatever amplitude we computed will become the first element of a two element message with 30 and that will be appropriate to send line tilde to multiply by. Alright. Questions about that? [31:17] So what's happening here is we're going to look something up in the tab read four. If we want to, in some sense, slide the table over conceptually all we have to do is slide the read point back the same amount we want to pretend the table is sliding. What we would like to be able to do is take the table and move it up and down in pitch, which is to say to transpose the entire spectrum.

[31:41] Just a good thing to be able to do. To do that, we simply transpose, in some sense, or we offset the value that we use as an x-value for reading into the table.

[31:53] So what's happening then is, if the oscillator is playing at middle C, that's all right. If the whammy bar says 12, that means we look back here, or if the whammy bar says -12, that means we look forward here. Before we look it up in the table. And then, the result is... [continuous buzzing noise]

Puckette: [32:12] What the whammy bar does this to the spectrum. [noise modulates in frequency]

Puckette: [32:26] All right, so that's another simple... It's a demonstration of using the abstraction mechanism for making a powerful additive synthesis instrument. If you were doing this for real, like making it stage worthy, you wouldn't want to make yourself edit the table by hand while you were playing. You would want to prepare a bunch of tables and be able to switch them or something like that. That would be a whole thing, to plan out how to you want to do and to learn how to make it playable. [33:01] So this is only a demonstration of the concept, it's not a real instrument yet. Questions about this before I close it and get on to the next thing? [pause]

Puckette: [33:14] No. All right. Closing in on the next thing. This was an aside, but should I save this? Oh boy, wrong place. [sound of punching keys]

Puckette: [33:52] OK. Now we can close that. And I might need this again later. [sound of punching keys]

Puckette: [34:08] OK. so that was the table spectrum dot PD example. Now this is more entertainment than it is actual elucidation. But here's another example of using, oh let's see... [sound like a continuous organ note]

Puckette: [34:36] This is something you've probably heard in Music 170. [note gets lower in pitch]

Puckette: [34:42] I think it's fair to call this the famous Shepard-Risset tone. It doesn't sound like much until you listen to it for a while and then it starts to sound impossible. This is of historic interest because computer musicians were able to make this on an analog synthesizer. And hackers were not able to make this because it requires accuracy and control, on a level that you can't get out of an analog synthesizer. And this is not the world's best Shepard tone. [35:14] This is just me working the studio synth one night... [tone fades]

Puckette: [35:20] What it is, is a spin on what you saw last time, which is that the table... It is not even using a table, but it looks like a bell curve. And then the sinusoidal frequencies are sliding from right to left if you like, and moving up, as you heard it sliding from high to low frequencies and working their way up the bell curve. Then back down in such a way that you always hear the descent but you never hear people disappear at the bottom because they're inaudibly quiet at that point. [35:52] And furthermore, the tones are arranged so that they are each an octave above the previous one. So that no matter what you think the fundamental pitch is or whatever your ear tells you the fundamental pitch is, everybody else is a perfectly good harmonic of it.

[36:05] And so as a result, if especially you don't listen to it too carefully and if it isn't too loud, you have this perfectly fused sound that sounds like it has a single pitch, except that the pitch is paradoxical at some points. You have to change your mind about what octave you are hearing it at. So that's the Shepard-Risset tone. How it's done is basically a spin on the other one, but there is more math in it, so I will spare you all the math because it's the same principle.

[36:37] Oh. I did one other thing here but I will show you this out of the next example, which is rather than using throw and catch. I don't know which of these is better style. You can use throw tilde inside an abstraction and catch tilde outside of it to collect the results of an abstraction. Or, you could do what I call summing bus which is each voice adds itself to to all the previous voices so that the output is the accumulation of all the voices, one after the others. It's easier to see what is going on when we do this.

[37:18] By the way here's a reverberator rev two if tired of all those dry sounds in your headphones and want a reverberator and grab this guy. [laughter]

Puckette: [37:26] I'll tell you about that more in a few weeks. OK. But I will show you how to actually do that in the next example, because it's simpler. The next example is... Oh. Should I do this? [tone] [37:47] No. I'm going to skip that. There's much to know about samplers. Here's a design of a somewhat more general sampler than you've seen so far. The samplers that you've seen so far have been of two flavors. One of which was driven by a line tilde object and started with a message that we start reading from a wave table or an array from a given point to another point over a period of time.

[38:16] The other flavor that you saw was driven by a phaser tilde and that was better adapted to looping. This is the sampler that doesn't loop. And it's an elaboration of the idea of the sampler so that you can control all sorts of parameters that can vary one sound from another.

So it was even there in what you've seen before. Everything except, I think, everything that you see there.

It was implicit that you could change the amplitude of the noise of the sample, or its duration. I showed you how to make them turn on and off using an invalid generator, but of course that means you would have parameters that would actually control that, as opposed to specified [inaudible 39: [38:45] 02] .

[39:02] To start rotation, or perhaps you could call it the onset into the sample, would be, if it was continuous soft and relaxing, is whether you want the word soft or the word relaxing, that's the start rotation.

The sample number - I haven't told you about this, but you can direct tab [inaudible 39: [39:15] 22] four tilde to choose one of a collection of arrays by name, by sending in messages.

[39:30] Duration you know, amplitude you know. So everything else is just what it was. Now, how do I, where are the messages that do things. [Mouse clicks] I lose the sample messages. Well, we can just do it.

[40:08] OK, so you sin to a thing called Note, in order not to introduce yet another object, I'm just going to say send Note and send it something. The things you send it are over here. Oh man, to do this...I'm just learning this window manager still. Oh I see, do this. Aha!

[40:37] So we make a message which is a pitch, an amplitude, a duration and so on like that. So let's just say, pitch, amplitude, duration is going to be 60, amplitudes and dv 80, duration is in milliseconds.

There's a sample number, there's a - I don't know how to do that! [Mouse clicks] There's sample numbers, static location or [inaudible 41: [40:51] 08] here, so sample number; start time would be the beginning, rise time and then stop time.

[41:19] And nothing happens - oh! Still nothing happens, let me see if I can turn this thing on. Oh yeah, yeah. Come on. There it is. Let me give it a little more juice. OK there's a nice sample. Here, just to show you what's going on, you can change now various things. Here's a different pitch. I showed you changing the amplitude before. Here's changing the duration. Here's changing the sample number. Alright, there are two samples in there.

[42:29] This is more subtle because this is a bell sound, but I can ask it to play something in the middle of the bell sound instead of the beginning, two seconds later. That's the sound. The rule about bells is that the higher partials tend to fade more quickly than the lower ones, statistically anyway, so if I go into the bell two seconds I'll get a sound more like that than like this.

[43:01] Finally, if I want to change the attack time, say to a second, then I should make this be two seconds long so that you can hear it, then you get. Same thing with the delay. I can make the delay 1000 milliseconds long, now make the note be short, like 100. That's interesting. I guess it sounded different, but it didn't sound different enough. Let's make it 3000 long.

[43:41] OK. While I'm at this, there is always the dollar sign mechanism for doing things like this. Suppose I want to be able to change the pitch quickly, but just move the rest of the things constant. Do this kind of thing. Nice keyboard. Or if I have the voice in there then I can make a thing that played different parts of the voice depending on the number of options there are.

[44:22] This is, again, the message box, this dollar one in the context of a message means this value, which is 71, is getting a thing replaced with. Now, without telling you all the gory details, I'll tell you a few of the gory details about how this thing is done. So the first thing to note is, oh boy, this is more complicated than it needs to be.

[44:55] The first thing that's happening is, we are making ourselves a bang before all of the arguments of a message. So a note is coming in here, and we're unpacking it. Then we are going to pack something which consists of the note, but we're also going to choose the voice number, which will choose which one of the eight sample voices to play.

[45:16] You've seen this pack and wrap combination before, in the example from February 3rd, I think. Where I was doing the first polyphonic thing with an abstraction already had this, I believe. OK. So what's happening here is that whatever these seven numbers are, we're going to add an eighth number in the beginning and then we're going to pack it into a message with eight values. Then we're going to route according to the first one. That is going to be a message for the sample voice.

[45:53] Now, the sample voice, this is an abstraction, now, which is made for just for this one patch. It's working by adding itself to the previous one, so this is a summing bus again. Now I can show you what the summing bus looks like. There's a lot of stuff here to look at, but I'll start with this.

[46:13] Here's how you make a summing bus. It's really stupid. You just take whatever came in inlet tilde, and add yourself to it. So whatever you had to do to compute the sample it's getting added to this inlet, and then it's going to become the outlet. If you make an abstraction that's designed like that, then you can just stack them up and put them one to the next. Then they add themselves up into the sum of all the voices.

[46:38] All right. So I'm not going to try to tell you how everything in here works because you'll all fall asleep. But the basic idea is the same as what you've seen before. which is that there's an inlet here which is corresponding to what comes in from the route object. Except that's the second inlet. The first inlet is the summing bus. So this thing has two inlets and one outlet. This is a signal inlet which corresponds to the inlet tilde you saw, which was the summing bus inlet tilde. And that gets added to whatever computes in that bus there.

[47:16] Meanwhile in here come messages, one message per note. And each message consists of, what was it, pitch, amplitude, onset, duration, sample number, and then the location of the sample, rise and fall. And then using techniques that mostly you know, but using more mathematics than I've thrown at you before, compute messages that you send to a line tilde. And I decided to make this a V-line tilde, for reasons that I'll try to explain later.

But the basic deal is you just work. You make a patch, and eventually messages go to this V line, which is generating indices into the Tab-V four. This V line is making amplitudes, and this V line is getting multiplied by it to control the overall amplitude. There are two amplitude controls here. OK. And I don't want to give you all the details, because it's just going to be too much. I'm just going to show you this as the overall design strategy for these kind of [inaudible 48: [47:52] 26] .

[48:25] And this is explained step by step in the book if you want to see all this in gory detail, gorier detail than I want to use right now. This is about the craft of making a decent, good, working sampler, which is harder than the basic things that I've shown you so far.

[48:46] So that pretty much concludes the basic tools for making abstractions and how you use the basic mechanisms for doing abstractions, which are the dollar sign mechanism, inlets and outlets, and the route object, and then all those objects like send and receive, throw and catch, and the one I haven't shown you yet is the signal version of send and receive, because we haven't needed it yet.

[49:12] All right. Now, change of subject. So that's abstractions. Oh, before I change the subject, how's the homework going for Thursday? Or that's another change of subject. Should I show you the homework again and see if there are questions about it? I see one nod anyway. Let me do that real quick. Yeah.

Student: [49:43] Is there going to be an extra credit for this week?

Puckette: [49:45] I haven't been able to think of one. The thing I thought was going to work as an extra credit just sounded cruddy. And then I could think of ways of fixing it, but they all were much too much work to ask you to do, so I ended up not being able to think of one.

Student: [49:59] OK.

Puckette: [50:00] OK. OK, so here it is, just so you can hear it and see it again. Do we want to save, no. So we're done with the help browser. Oh dear, am I going to be able to find this now? Don't have any files in there. Oh that's the, OK, that's the wrong place. Still the wrong place. There. All right. Does it work? Yeah. [audio clip plays] [51:02] So that is, I think, I'd have to go look, but it's driven by a phaser, if I remember correctly. No, no, no. I didn't do that. I made it driven by a line tilde, because it was easy to figure out how to do it that way. And basically all it is is a bunch of chunks of a sample that are gradually moving forward. [audio clip plays]

And frequency and size are helping to bring this - which, by the way, is because if you read more of the thing in a fixed amount of time, you get more transmission. The frequency is of course is a hundred percent of this. And part of the trick here is, you've also got something that [inaudible 51: [51:30] 53] .

[52:04] So this is a fast way to have a lot of fun with sound waves.

[52:09] But see if you could just get it to do this thing, [example noise] because then you already will have made yourself able to do all the rest. That's clear? That is for Thursday, and I couldn't think of a good extra credit to ask for. The extra credit I wanted to ask for was to make three of them and have it be in a major triad or something like that. But then when you listen to it, it's just hash. You actually can't tell what going on. It's too thick, sonically. You could fix that but it requires things you don't know about yet.

[52:44] OK. The homework for next week - this is looking into the future - is just a simple exercise - not simple - it's a first exercise about polyphonic voice allocation. [example music]

Puckette: [53:09] Just try to make something as different as possible from the continuous soft [inaudible 53:12] . This is a - this is no more complicated than it sounds like. It turns out that if you pull a sliding stone and then make it decay, it has this sort of wet reverberant sound, just because one's used to hearing the sounds of decaying things, and reverberant spaces, I guess. [53:32] So it has this very sort of dripping sound. Although that's nothing but just plain old simasorta oscillators, exactly like the ones I've shown you. If you leave them running it sounds dry and ugly, but then if you turn it on and then quickly fade it out, then it starts sounding like this. [example music]

Puckette: [53:46] Except that you can probably tell, there's more than one sound happening at once. You wouldn't be able to do this with one oscillator because, in fact, at any given time there are ten of these things sounding at once. This is a 12-voice machine that I'm using for this. [54:04] So what's happening now is there's notes being generated - just being thrown. The situation exactly has ended in sego So that you can make sort of classic computer-driven sounds. And the only thing that's useful about that immediately is that you can choose the bass frequency. This is actually quite easy to do, it's just conceptually fun. It's easier than this last one, I guarantee you.

Student: [54:47] Why is it conditional? It seems kind of counter-intuitive to [inaudible 54:52] .

Puckette: [54:54] OK, so the metro object takes an argument which is the milliseconds between things, so really the question would be why is the metronome object designed that way. It's so that you can get exact values out of it. Since the scheduler works in units of time instead of units of tempo, if you say, for instance, 1,000 to it it really will come down once every second, but if you're doing tempo then if you say 1,001, what's that a tempo of? [55:33] It's 59.9 something, but then there would be a round-off error. So the fundamental metro object does that simply so that you can do things that are exact and repeatable. Why didn't I make this thing do the right thing and be 120 beats, two per second, and so on? It's because that would have been making me work harder. You can do it. It's easy to compute what you should feed the metronome to get a specific tempo.

[56:02] You just take the thing and divide it into 60,000 and I could explain why, but what that means is 60 should go to one second, which is 1,000 and 120 should go to 500 and so on. So you're dividing 60,000 by the metronomic value to get the milliseconds you feed the metronome. But that's adding another step to the homework that you have to do that wasn't really part of the homework or wasn't part of the idea anyway.

[56:33] So yeah, you can do it either way. OK. The one object I haven't told you about that will make this possible, obvious there's a sequence of pitches there. I've shown you how to make repeating sequences, but this is even stupider than that. This is just random numbers. They are random numbers with a particular range, which you probably can't hear, but it's two octaves.

[57:06] So how would you do that? This would be the moment to just show you something. Here's a new patch. 16 point OK and we're going to save it, not here but back in the website since you'll see it. Randomness. Alright.

[57:36] So this is an aesthetic fault line in computer music. The word random or the word stochastic has musical baggage when you use it. So we're going to forget all the musical baggage associated with randomness and just consider this as a technique rather than as a musical statement. So here's how randomness works. You just say random and then give it a range.

[58:02] And then every time you give it a bang in, which I'll just give myself a bang object to do now, out will come a number which is between 0 and 24 inclusive, so that there are 25 possible values. Stupid. Usually that's all you do. Every once in a while you have to actually explicitly give it a seed, because you have two of them, and you want them to have exactly the same random sequence as each other.

[58:37] So weird as that might sound, it does happen that you want to do that sometimes. So you can seed these things. That's stuff that you can find out in the help window. But 99 times out of a 100, just the object itself is what you want. And now, for instance, just to make, first off, to make random pitches, do that and add some base pitch. Oh, right, what does that do?

[59:05] That gives us random numbers between 60 and 84, I think, inclusive, right, 60, 85, but 84 really, because this only goes up to 24.

And now that could be something that we feed to midi to frequency, and then to an oscillator, and then to a output, and now we have Idiot's Delight. [sound plays] Yeah.

Student: [59:42] I'm sorry, why [inaudible 59:45] ?

Puckette: [59:47] Oh. These numbers coming out of here range from 0 to 24. There are 25 possible values when 0 is included. And why 25? Because that's two octaves, including the two end points, if you're talking chromatic. Right, if you're talking all the keys, not just the white ones. [60:06] If you want to work harder, figure out how to make this work only with the keys in a given scale, that would be something that I don't want to tell you how to do right now. You can use modular arithmetic to do it, but you'd have to actually think about music theory to get there. And that would be a thing. So there's randomness. Oh, and this is randomness the way it sounded in the 1960s. [sound plays]

[60:38] And while we're here, it's a good point to mention the existence of micro-tonality. That's the difference between this, oh, let me just make this be a musical fifth wide. OK, so now we have [sound plays] . Stuff like that? That, let me make a nice metronome so you can hear this systematically. And that really wants a toggle to turn it on and off. I'm doing this so that I can compare it to another thing. Toggler, toggler, toggler. [sound plays]

[61:33] OK, now if you could do it fast enough, you could play those pitches on a piano. Because they are all integers. But if you didn't want integers, if you really wanted to sound like an analog synthesizer, you could do this.

[61:49] Now, random number generators in general make random numbers that are integers, even if it looks like it's making something other than integers, the pseudo-random number generator in the computer is really making integers. So true to that, random really does only make integers. But we can say, why don't we have a random thing that goes from 0 to 700? And then we're going to divide by 100.

[62:21] Come here. Oh, should I do that or should I make it 800? Let's do 800 just to be, I don't know why, to be simpler. Come here. OK. So random 800 and divided by 100 is the same thing as random 8, except that here a perfectly good value would be 50, and that would turn into 0.5, which won't come out of this one. So this now is [sound plays] , the chromatic version, and this is going to be the micro-tonal version. [sound plays]

[63:04] Can you hear the difference? That's the genuine, almost the continuous collection of possible pitches, as opposed to this, which is [sound plays] , which is cycling back and forth between the same eight pitches over and over again. OK? So that's randomness and a sort of a note about quantization that might inform how you would choose random numbers.

[63:36] Another thing to think about, another thing that you might want to do here, now that you've got nice random numbers, is use a table to have random numbers that are chosen from a set of possibilities that you might have pre-arranged. And that table you would set up in the same way as you set up the sequencer table from many weeks earlier.

[63:54] And then you could choose randomly from a collection of pitches that you chose, and could even change dynamically, if you want. So now you have easy way to make standard midi art kinds of things. That's randomness, and that is only there so that you can know how to do this, because the only thing that you don't know how to do about this yet is generating all these pitches. [sound plays] And all it was was something like this. Yeah?

Student: [64:25] My output box comes out as a slider. How do I get it to come out as an array?

Puckette: [64:30] Oh, you've got PD extended. Someone wrote another kind of output tilde thing. You have to go get this one. You know what, it's on the website, so if you look at the sample patches from any time in the last week or two, there will be one of these. If you put that in the same directory as the patch you're working on that will be read first and it will give you one of these instead of the one you're getting.

Man: [64:57] It's on the DVD, too.

Puckette: [64:59] Oh, and it's on the DVD. There's probably nothing wrong with the other. It's probably better. Alright, done that. Now I'm going to shift gears entirely and start talking about modulation. The first thing to know about modulation is this: this is something I've mentioned before, but not really made much out of. So we're going to make this four ring modulation. Ring modulation is the following idea. This is really, really simple, except you can do a lot with it. [65:45] Take two oscillators and multiply them. OK. So right now we're just recreating something you've seen already once. So I'm going to make number boxes to set the frequencies. The first one will just be a reasonable frequency for listening to the pitch A. It's going to be 440. The second one is whatever I give it, which will be a medium frequency to change the amplitude, this is from day one, and then higher values make the sound split.

[66:28] So there are several ways of thinking about this. One of them is that you know that if you have two different sinusoids at different frequencies that you hear they'll beat together, at least at the frequency they're close, so you can think of that as a sinusoid multiplied by another sinusoid. In fact, it's a trigonometric identity. But you can apply it backward. You can take this thing and make it beat by multiplying it by an oscillator.

[66:51] What that is is it's mathematically equal to two other sinusoids, one at 338 and one at 442. Then if you make this go faster they split further and further apart until you can hear them at two separate pitches. That's a good thing. The good thing about it is not that you can make two oscillators out of one because, of course, you could've done that by adding them, but that you can take anything you want to and do that to it. For instance, let's just do it live.

[67:39] So now instead of the oscillator I'm just going to use the microphone. Risky choice. Microphone, microphone, microphone, microphone. OK. Does the microphone work? Oh boy does it work.

[68:00] Hello. I'm talking into the microphone. Why do we hear that? Because I haven't turned this one. Hello, hello? This is not good because I'm not going to be able to do anything else than have myself be heard. OK, there's probably a button here. There is a monitor switch. Let's use it. Hello, hello? Good. It went away. Alright, so now let's turn this to zero.

So nothing comes out because I'm sending sound to the computer, but it's not sending out. I had my audio interface on the monitor before. Now we turn this on. OK, so this is my voice being amplified through the patch. So now I can do anything that I want to to it. In particular I can make it start beating. So now [inaudible 68: [68:37] 58] trembling a little. So if I make a nice long tone then you get that. OK.

[69:09] So now the fun part is. So now we've got nice monster voices from a TV show. OK. What happened there is kind of cool. Whatever pitch I'm going at has not just itself but it has a bunch of harmonics in it. If it was just a sinusoid you would just hear two pitches split off, but, in fact, a harmonic tone is a bunch of sinusoids and each one of them is going to get split off, but they're going to get split off by a fixed frequency deviation.

[69:53] Suppose I happen to be talking at 100 hertz, which is a typical droning frequency for my voice right now. So this thing is going to turn that into 100 plus 53 and 100 minus 53. Then the first overtone or the second harmonic at 200 hertz goes to 253 and 200 minus 53, which is 147. So if I put 100 hertz in, the original overtones are 100, 200, 300, 400 and then what comes out is 37, then I get 147, then 153, then 247, then 253, and so on like that.

[70:51] And I'm not sure if this is really weird [voice changes in microphone] but it's, it's not going to be harmonic it's going to be computer busy. [sings a tone] Oh, you know what, if I happen to hit twice 53 which is 106 then I get a nice harmonic sound again. [sings] Aaaahhhh which is an octave down from where I started. [stops singing] Which is cool but if I want to do that in a robust way I would have to figure out what the pitch was and constantly adjust the 53. You can do that; I can show you how to do that later.

Student: [71:23] The [inaudible 71:25] right output, is that for channel stereo?

Puckette: [71:26] Yes. That's the other channel; I'm only running into the left channel on this mic so there is nothing coming out of it. [71:35] O.K. But then if I do a different pitch [sings tone] You just don't have things that line up in harmonic series so you have something that would be more typical of a bell tone, except I can't make a bell sound because I would have to make my voice envelope like that and I don't think it's physiologically possible.

[71:53] But it's a thing that's an inharmonic spectrum as the spectrum of the bell might be. If I took that [sings tone] and sampled it and then enveloped it I could make bellish sounds maybe. Or crude bellish sounds I should say.

[72:11] This is actually a very general and powerful technique even though it looks stupid. You just take anything that you want and multiply it by an oscillator and it takes the frequencies and slides them both to the left and right.

[72:23] With very carefully designed filters you can actually separate the part that slides from the left from the part that slides to the right. But that is stuff from chapter eight; you don't get to see that just yet. As it is it is already pretty powerful. Questions about this?

[72:45] So another example would be; lets make a very simple computer music instrument. So now I want to make, in the simplest possible way, a sound that has some interesting overtones that's just made out of an oscillator. [turns dials on oscillator] So we're going to be doing all of this stuff again. Except I'm not going to do it to my voice, I'm going to do it to an instrument so let's design the instrument first.The instrument is going to be... We'll take an oscillator and then just act stupid with it. We're just going to clip it between some decent negative value and some decent positive value.

[73:38] OK and now I'll just play this now so you can hear it. [pause] Whoa, nothing. [pause] Whoa look at that, that's not going to work. [pause] [oscillator works]

[73:59] Oh OK that's not so exciting let's drop this a little bit. There we go. Now what's happening I showed you this before in a different context. Now were doing is were taking the oscillator which is a sinusoid and clipping the bottom and top. That's a very simple wave shaping way to make a different kind of wave point. And that by the way that's a big topic which I will give you some mathematics about in the next couple of sessions. But right now I'm just going to do that and have it be a nice sound.

[74:33] Alright. and now this sound that we have I want to take and remodulate. So now we have the same sound but now [noise] sound that's being modulated. That's not a very good choice of frequencies. [noise] So this is a very fast way of making in harmonic spectra out of harmonic points. And this you know, it doesn't look like much but if you listen to electronic or especially computer music over the last especially actually the period 60's through 90's. This is going to suffuse everything because people got very excited about being able to make in harmonic spectra after being in this is largely the harmonic spectra plus an occasional bell for most of the history of music.

[75:38] So people make music theory kind of thoughts about how these harmonic spectra could make this in a way they could be you now constantly or distant intervals between sounds that didn't have pitches but had just spectra like this. And so that's a rich source of musical inspiration that was brought on by the electronic and the computer music eras. So this is the sort of general direction that were going now. Having seen the multiple voice thing that extraction mechanism. The next thing is learning how to design sounds using the techniques of modulations and wave shaping. Which is represented here by this multiplier and this clip total. You got this?

Student: [76:28] I have a question about the final project.

Puckette: [76:31] Yeah.

Student: [76:32] What is it really going to do?

Puckette: [76:34] Yeah. OK. So the final project I'm going to try to I'm going to try to make a menu of good things that you could do.

Fiveish sort of different things that everyone isn't working on the same project. That would be kind of boring. And the scope of it's going to be like two weeks it should work. Is there like eight homework assignments in that plus a two week final. Ten weeks of work. So I'm not going to, it's not going to be like writing an hour of music or something like that. It's going to be demonstrate a drum machine or demonstrate something like that. Something you can do in a couple of weeks. I'm going to try to actually have a decent website out about it with a description in the next week or two. Alright, that's it for today. Office hours.

MUS171_02_10

[cross talk]

Instructor: [0:12] So this is now chapter five of the book. Oh, let's get the book out while we're at it. What happens now will correspond pretty closely to the material in chapter five, so that you theoretically can actually find out by looking in the book, what's going on. Which has not always been true up until now because I have been operating in a exploratory, make patches as you go mode. [0:38] But, now that the basic notions of how to make patches and figure out what they're doing can be down, I'm going to try to be a little bit more, whatever you call it. A little bit closer to the written thing in the book, because that way it'll be much easier for you to make correlations between the book and what's going on in the class. So nothing in the book is actually wrong. Nothing in class has been terribly wrong, as far as I know, either. But there haven't been perfect correspondences.

[1:07] In fact, people who have been writing, have been making looping samplers for today's homework have been looking in the book to figure out how to do the enveloping. The book does it differently from how I did it in class, so that now you know two different ways of doing enveloping and you might not know how they're different, which I'm not going to try to clear up right now because it's too weird.

[1:26] But what I do want to do is start referring to stuff in the book as I go through the next few patches, because it's just going to make the next thing a little easier than it would have been otherwise, I think. Book.

[1:46] So the place we're at is this chapter on modulation, and I'll get to this in a second. But to talk about modulation, you have to talk, or be ready to talk about spectra. So think about things as having spectra and I'll tell you more about the words that one uses to describe spectra in a moment.

[2:05] But first I will go back to the patch that I was working just in the last 15 minutes of the class on Tuesday, and go into somewhat more detail about what patch is actually doing and why the sounds that it makes sound the way they do in a very hand waving kind of way, before I show you more quantitatively what's going on.

[2:24] So here's the patch up. This is the fourth patch from last class, and I just realized this morning I haven't put these patches up on the web. I'm sorry, I haven't done that comment and put it up on the web thing, so you haven't seen these patches except in class so far.

[2:42] Basically, what happened in class was, the first thing was take a microphone and multiply the microphone signal by an oscillator to mess up its periodicity. And then it was time to go back and explain a little bit more carefully what was going on. And so, to do that, I had to make a thing that had some kind of wave form so that I could then multiply that by a sinusoid and mess it up and show you how that... how you can think about that. There are many ways of thinking about it.

[3:10] So here's the patch again, cleaned up. Basically, the patch on the left is something that you saw in week one or two, which is about clipping and what it does to wave forms. So if I show you that, this is a nice, clipped sinusoid. Oh, and I fixed it so I could clip as I please, so that you can see how that's going.

[3:37] So, for instance, if I tell it the top of the clip is going to be one, then we're, what we're doing is we're allowing the thing to go down to minus 0.2... Oh, that's just for, just to be clear, I'll put it in by 0.2. Now the value's going down to minus .2, but all the way up to one. And if I made this thing be minus one, then you would see the original sinusoid that didn't get clipped.

[4:03] And this is an example of wave shaping. It's taking a perfectly good sinusoid, or in fact, some other thing, but the first thing to think about what happens to a sinusoid when you do this to it and putting it through some function or other.

[4:19] If the function were linear, or a Y=MX + B kind of function, then you would just get a sinusoid out. It would have a different offset and a different amplitude. But if you give it some kind of nonlinear function, in particular, the function that is represented by clip tilde, then out comes something that's quite different.

[4:38] The function that clip tilde is doing is, right now, if you graphed it, you would... OK, so clip tilde puts the, right now it's clipping from zero to one. Oh, let's get rid of these to make it even clearer, or less opaque, I hope. Clip tilde like that.

[4:55] If you clip from zero to one, you can think of that as a function with a graph. The graph looks horizontal. For negative inputs, it's zero. From zero to one, it follows the input, so it looks like Y equals X, and then, starting at one again, it's flat again, and one. And so it looks like a ramp. Not a ramp, but it looks like a sloppy step function. OK?

[5:22] So it's not linear. And if you, you'll see this in gory detail later, but the basic deal is that when you put a sinusoid through a nonlinear transfer function, as we call it, then output comes out as not a sinusoid. And when it's not a sinusoid, then it has a spectrum, it doesn't have just one partial in it. I guess that's almost the totalogy.

[5:49] So what comes out of the oscillator here is repeating itself every 110th of a second. In other words, it's repeating 110 times a second. So, since this is just a function, it doesn't have any memory in it, what comes out is doomed to repeat at exactly the same period, if not even less. In other words, when the oscillator gives you the same value at two different moments in time, the function has to give out two similar values, too.

[6:16] So if you put a periodic function in to clip, you're going to get a periodic function out, with the same period. So, what that means is that if you listen to the original sound, it has a pitch.

[tone]

Instructor: [6:35] And when you listen to the result of clipping, it's got the same pitch. [tone]

Instructor: [6:39] But it's got partials. This is a special case, actually. Oh, I showed this because it's simple, but it's got a very strong octave just because of the way it's, happened to be set up. And if I do something like that, then you'll hear... [tones]

Instructor: [7:03] Just basically what you heard before, except it has a different timbre, which is to say it has different partials. [tones]

Instructor: [7:10] OK? And this is basic, this is how in electronic music, you make... Well, this is one, this is the most generally used thing in electronic music that I'm aware of, for making things that have partials that have strengths that you try to control in one way or another. But in order to control them, you have to do math. In order to do it, you just throw something in a nonlinear function at will and you get it out. [7:35] The history of this is that guitarists in the '40s and '50s started actually not over driving their amps, but messing up their speakers. I believe the first example of distortion guitar was some jazz guitarist who decided to take a knife or a pencil maybe, and just bash the cone of his amplifier speakers, so that it would sound fuzzy. [laughs] And it works.

[8:02] In a very loose way of speaking, what that's doing is making the amplifier no longer be a linear thing and start being a nonlinear thing. In other words, it's a thing where you put two signals in and what comes out is not the sum of what went in for the two signals separately. And anything that has that kind of property has the ability to infuse new frequencies into the sound that might not have been present there before.

OK, now, it's time probably to start talking about spectra and more graph-y, graph-y [inaudible 8: [8:27] 35] of a way, so that I can now show you something about what's actually happening. Actually, sorry. Before I do that, I have to finish showing you, we've just finished reviewing, or bringing back out the patch from last time, because I didn't show you the other thing that you can do. So here's the...

[tone starts, stops]

Instructor: [8:55] ...thing that has partials. And here's a thing that has partials... [tone]

Instructor: [8:58] ...that is also being remodulated. [tone stops]

Instructor: [9:01] And what I did was, what I played you before was sounds like this. [tone]

Instructor: [9:08] Well, not quite like that. More like that. [tones stop]

Instructor: [9:11] Sounds where you would take some sound in and just destroy its periodicity by... [tones start, stop]

Instructor: [9:16] Basically multiplying it by the wrong sinusoid. Well, wrong, a sinusoid that has a different period from the sound of the original, from the original wave form that's getting graphed. But of course... See, here's the wave forms that we're putting in. Here's this. [tone]

Instructor: [9:37] What this is doing is taking this nice wave form and sometimes sending it through positively, sometimes sending it through negatively, and sometimes going through zero when it's doing its main work. And you just get a funky sound, a funky wave form like that. All right? It's artistic. OK? [9:58] Now, of course, it would be true that if this sound happened to be periodicity, periodically the same period that we started with... Oops.

[tones stop]

Instructor: [10:09] Now, it's still an interesting wave form, but now the wave form looks periodic. And in fact, it has the same period as the sound that we just modulated. Way different wave form, but the same period. Or to listen to them, here's what we just modulated. [tone]

Instructor: [10:26] Sorry, that's the signal that the sinusoid clipped. [tone stops]

Instructor: [10:31] And here's the same thing, remodulated. [tones alternating]

Instructor: [10:36] So now you can imagine taking... Oh, I didn't tell you the rest. OK, so now let's try 660. Well, OK. Let's leave it on and... [tones]

Instructor: [10:45] ...try different multiples of 110. [varying tones]

Instructor: [10:53] Oops. Sorry. Can't type that well. So let's leave it here, and I'll show you that... [tones stop]

Instructor: [11:05] ...and then turn it off for a second. OK, now we've got something that... Hm, still got the same period. It can't help but have the same period because both of these things, although this thing has six periods, or six cycles within the same period of time, which is 1/110th of a second, it's still true that after a 110th of a second, it's come back to where it was. It just happens to be the sixth time it's done that. All right? [11:27] So, it's still true that every 110th of a second, which is about this length, both the clipped oscillator here and this oscillator, which I'm multiplying by, that's the ring modulating oscillator. Both of those have, have come back to where they were before. And so we still have no choice but to have a signal which is periodic... Well, a signal which repeats every 110th of a second, which, except for in special cases, will have... Oops.

[tones start, stop]

Instructor: [11:57] Will have the same pitch, as the original that we started with. [tones]

Instructor: [12:00] So there's the sinusoid. Here's the clipped sinusoid. Oops, sorry. [tones start, stop]

Instructor: [12:05] And here's the clipped sinusoid times ring modulation. [tones start, stop]

Instructor: [12:09] OK? When you learn how to do this, or when you learn how to think about this, you can make literally almost anything that you want. There are all sorts of tricks to, well, mental tricks to try and figure out what you do, in terms of what kinds of functions and what kinds of things to multiply to get specific kinds of effects. And so, first off, I want to show you more theoretical aspects of just, what's happened to the sound from the point of view of the spectrum. And then I'll go through and start working on actually building spectra, according to, [inaudible 12: [12:30] 44] errata, out of this tool box. Those are the subjects of chapters five and six of the book. Probably, this will take a couple weeks. So, the first thing that we need in order to be able to discuss this intelligently, is to be able to look at spectra of signals.

[13:06] I'm going to just ask you to take a certain thing for granted, which is that you can measure the spectrum of a signal and graph it. What I can do is make a sort of definition of what the spectrum of a signal is. Let's see where is my - there.

[13:30] I'm going to ride roughshod over some of the details here. This patch is in "Audio Examples." This is the first patch in chapter five. This is a patch that says, "Sorry, but we happen to do spectra."

[13:45] When it's time to actually measure the spectra of things using a patch and understand how that thing is being done, that's in chapter nine of the book. So, what we're doing is we're borrowing results from the future, in order just to be able to see spectra. And, what do spectra look like?

[14:00] OK, so, signals have waveforms and signals have spectra. What I've done here is just made a very simple additive synthesis instrument that does this...

[tones]

Instructor: [14:11] Oh, yeah. OK. There's a frequency coming in here, it's just a standard receive. And we're multiplying this frequency by the numbers zero through five. Why is zero, for completeness sake, and in order to explain a very strange thing about the spectra of sinusoids that I can't hide from you. I just have to explain it. [14:44] So, I'm going to come out with it, right at the beginning. And, so, the ones that you can hear are fundamental, octave and so on, like that, right?

[tones]

Instructor: [14:55] Now what we can do, in this patch anyway, is we can start graphing the spectra and the waveforms of these things. So, here's the fundamental. It has a waveform, which is just a sinusoid of the appropriate frequency, and it has a spectrum which, one graphs. [15:12] There are various ways that you can do this, but one can graph it in terms of the partial numbers, that's to say, the multiple of whatever the fundamental frequency is that we're playing.

[tones]

Instructor: [15:29] Yeah, I don't know what order to tell you this in. So, let me just make another spectrum and play it for you, or, show it to you. [tones]

Instructor: [15:39] Here, now, I've turned the first, second, and third and fourth harmonics on, and so on like that. Now here's the funny part. I can turn this partial on that doesn't have any sound, because it's just constant, because it has a frequency of zero. [15:52] It adds something to the spectrum, too. By the way, my computer's gagging right now. Just let it gag. Now, there's a weird thing that will basically just kind of bite you once in a while when you're trying to do something and something comes out wrong.

[16:11] A sinusoid that happens to have a frequency of zero, you can assign it a strength in the spectrum, but the most correct way to assign it strength is to give it a strength of one as opposed to one-half for the other sinusoids.

[16:30] That's to say sinusoids of non-zero frequency. Chapter eight will explain why for the first time. I'll tell you what it is for those of you who like mathematics or know about mathematics. Sinusoids actually have two frequencies in them; one positive and one negative.

[16:51] They don't act like quantum theory, where all the frequencies are positive. They can be real-valued and the only way you can have a real-valued sinusoid is to have positive and negative frequencies of equal strengths and negative. Equal strengths talk about the phases later and negative frequencies.

[17:09] So, really, although I don't show it on this table, this oscillator has a peak at frequency one, relative frequency one, and a peak at relative frequency minus-one. You can't perceive it but it's there.

[17:28] And, the reason this is double is because here, those two peaks coincide. All right? For those of you who've gone as far as, maybe, second semester calculus, sin of omega T is E to the I omega T plus E to the minus I omega T, all over two. In other words, a sinusoid has two complex exponentials and each of them has an amplitude of one-half.

[18:03] So, if you don't want to know the equation, here's what it looks like. This is the truth. And there's no possible way that, well, you can sort of pretend it's not true by saying, "Oh, it's just frequency zero and we'll just make it the same height."

[18:16] But, all of the stuff that we try to do later will be wrong if we do that, because DC does come crawling into signals and if you don't account for it correctly you will get wrong answers. All right.

[18:29] That's the thing about the spectra of sinusoids. Oh, yeah and while we're here, this is worth looking at. When you turn all the partials on, you get a wonderful thing which is called a "pulse train". Or, Joe will correct me if I'm wrong, I believe this is the Dirichlet kernel. It's a collection of sinusoids, all of which have equal strengths, except that this one is double because of funny stuff.

[18:55] But, anyway, this is just DC, which I was just talking about, the height of the thing, the DC amount of the thing, which we could make arguments about. The more partials we put in, the more close this will become to a perfect pulse train.

[19:11] Engineers will actually talk about infinitely thin pulses, which consist of all the possible harmonics. You wouldn't do that in computer music, because some of them would have higher frequencies than the Nyquist frequency, and they would fold over and you would have trouble.

[19:29] So, you don't make pure pulse trains in computer music. You try to make band-limited pulse trains, that's to say, pulse trains that only go out to a certain number of partials, in order to have your computer be able to deal with it. And this is what those pulse trains look like.

[19:47] All right. Oh, I turned a couple off so that you could see the progression. Here's a pulse train with three partials, actually, zero, one, and two. And then the more partials that you throw on of equal strength, the skinnier and taller the peak gets, and the more wiggles, ripples the engineers will call that, you will see between pulses. All right?

[20:12] So, that's a pulse train. That's just a qualitative thing to know about, because you will see pulse trains again in the future. Now, the reason I'm telling you this is to be able to tell you what happens when you do things like apply a non-linear function to a sinusoid or multiply some complicated spectrum by a sinusoid.

[20:33] Now, the next thing I'm going to do is... I'll stick to these, and then I'll start telling you more of the whole truth later. I'm going to save this, you probably can't do this, but, if you're actually writing the thing, you can save your own help files. Let's see. Help. Browse.

[20:59] Now we're going to look at E zero two. Ring modulation; now what we're going to do is make a nice pep speech about ring modulation. So here, now, what's happening is the following. We're going to go back, and we're going to look at our nice bunch of sinusoids that has a nice spectrum,like this.

[21:30] Oh, yeah. I can actually ask this one to graph repeatedly on a metronome so that I can change things live. OK. This is idiot's delight now. I can make funny spectrum look at their waveforms in spectrum.

[21:46] Now, what we're going to do, gee whiz, we're going to multiply this thing by an oscillator. The oscillator is going to have a frequency, and - I'm cheating a little bit about the frequencies here. Because, to make it very easy to analyze, I'm choosing a frequency that's the same multiple of the sample rate.

[22:04] So, I'm not going to talk about exactly what the frequency's values are, just relatively. So, if I say in F over 16, if this is the frequency F, if I say - Oh, can we hear this? Let's listen to it. Yeah.

[tones]

Instructor: [22:25] OK, now we're hearing it, and now, if I say, "Well, let's make this thing be eight." OK. Oh no wait, let me get this. [22:37] Now we have a sinusoid, and now we're going to start multiplying it by oscillators. And, as you know, what that does is that splits the sound up into two frequencies,because that's what ring modulation does to sinusoids, as described last time.

One way of thinking of that is, beating is the same thing as having two neighboring sinusoids. It's a mathematical formula. But, it also means that, if someone gives you this: [changes a tone] , and says, "Give me two of those, and make them be split in frequency," you just multiply by a sinusoid that has non-zero frequency and you get that: [new tone playing] [22:54] . All right?

[23:19] Oh, and by the way, now you see why it starts to make sense to talk about negative and positive frequencies. Because, in fact, this thing has negative and positive frequencies in it and that is why this peak that you saw split into two peaks.

[23:35] It's because the negative frequency of one, by multiplying with it, drops the frequency, and the positive one added to the frequency. And, furthermore, when I set that frequency to zero,

[23:44] The two collide and then I get that.

[tone]

[23:47] Now, I'm playing tricks with phase here. If I do something like make these two beat against each other. I just asked this thing to do one-hundredth, one one hundredth. Now, we have the two things beating very slowly.

[24:08] And, what you have is, from one point of view, two sidebands that are separated, but, from another point of view, you have an amplitude that's changing. In fact, now, I can say, "Multiply it by an oscillator frequency zero." But, I no longer have the good situation where these two add up. They add up wrong. Why did that happen?

[24:38] Actually, there are two reasons why that happened. This is sometimes called "interference." This is an interference effect, from one point of view. These two things have phases, such that, when we combine them, depending on when you do it, depending on exactly when you combine them they might have different relative phases.

[25:07] Then, when they add up, they won't add up to twice the amplitude. They'll add up just to some amplitude or other, which might be anywhere from zero to twice, depending on whether they interfere constructively or destructively.

Student: [25:22] You said the [inaudible 25:23] of 16 [inaduible 25:24] ?

Instructor: [25:28] That's just my pedagogical choice of the step to use.

Student: [25:32] How do you set it [inaduible 25:34] .

Instructor: [25:38] So, the patch is computing this thing called "frequency step," and it's actually setting that to a fundamental over

16,and that's hidden in some sub-patch. Probably in here. And the only reason I did that was just so that when you go into the patch and start mousing on it, you get decent range. There's nothing special about 16. All right. Now how could I actually make this thing behave itself. Maybe I just can't then. [tones]

Instructor: [26:12] OK. Good. So I just tried again and got a better max so we can now pretend that the thing's in phase again. [26:21] So then, it follows that if you had a few other sinusoids, [tones] here's why I used 16 so that you can see the original spectrum and you could also see the splitting and they would appear on the same picture and with reasonable spacings. Now I'll say, let's make the frequency step be 1/16 again, go up 2/16. And now what we've done is we've taken each one of those three peaks and split them separately into side bands. Let me shut this up for a second and talk about that.

[26:58] This process, if you think of this as a function of what goes in, so a signal goes in and a signal goes out, and so what's happening is it's some kind of function. That's in a loose way of speaking. It's a linear function. In fact, it's nothing but times tilde. But it's times tilde times a signal not times a scale or a thing that's changing in time. That's a linear operation. What that implies, it implies many things, but for right now what that implies is that if you took two signals in or here three signals in and added them up...oh, yeah. I didn't tell you this did I? If you hook a bunch of signals into a single inlet, they are added automatically. I think I mentioned that at one point, but maybe it's a good moment to say it again. So this is now multiplying the sum of these things by this original oscillator.

[28:00] OK. Now I'm talking about linearity. One good thing about linearity is that, in this case, given to us by the distributive principle. If you call this thing, I don't know what, call these things A, B, C, and F here, then F(A + B + C) is the same thing as (F x A) + (F x B) + (F x C); and that's the distributive rule. And what that is saying here is that if you take two or three signals and you superpose them, that's to say add them, and then multiply them by this modulating oscillator, you get as a result the sum of what you would have got putting them in individually. This was not to be taken for granted.

[28:43] So here what you saw was that we had (oh, I turned it off) [tones] we had originally this signal going in and if you multiply by an oscillator of frequency zero we see at least a multiple of that coming out. And it turns out that although I didn't have to be true, the result of ring modulating this is the sum of the result of ring modulating the individual ones. [tones] OK. All right. Is that clear?

[29:22] OK. Examples of things that are linear in this way, obviously, multiplication although here we're multiplying by something that's not constant in time. And the other example that you'll see later is filters. Well, I introduced a filter quickly but I haven't told you about filters in detail. But filters also are, at least in their usual form, are things that are linear in the sense that you put two signals in and you will get out the sum of what you put in individually.

[30:06] As a detail also, any kind of linear function like this, you can multiply the input by some constant. For instance, double the input or multiply the input by I or anything else that you want and what comes out will be that many times stronger or weaker than the signal went in too. In other words, linear things, the spec changes in amplitude and will give you the same relative amplitudes on output. In general, that's not true of nonlinear things. What's a nonlinear thing you've seen very recently?

Student: [30:45] Wave shaping is it?

Instructor: [30:46] Yeah, the wave shaping example. This clip (oh, where did I put it) this clip tilde operation was not on here and as a result, let's see...well one thing that happened about that was the oscillator that you put in (I haven't said enough to explain this well yet) the oscillator that you put in, if you change it's amplitude you will not just change the amplitude of the output and give you the same thing louder. It will give you a different signal altogether. OK. So I'll go back and belabor that point with you in a few minutes. [31:34] OK. So this is ring modulation and oh, right, special case again. What happens if we pull the zero frequency signal in? [tones] So now we have the same thing as we had before except I threw in frequency zero which has double amplitude. And now when I start modulating that, it does the correct thing which is again, it gives me two side bands [tones] each of which has half the strength but, of course, the original was twice as high. And also, this one we only see one of because the other one is negative frequency.

[32:16] All right. And it's even worse than that because, and this is hard to see very well, but as I start pushing the frequency of modulation up, [tone changes] oops, oh yes. A funny thing happens when you hit a half. So eight, now what's happening is the original signal had peaks here, here, here and here. So, oops, I pulled the table over but I didn't pull the labels over. They might be useful later. Like that, OK.

[tones]

Instructor: [33:01] So, if I modulate it, that is to say, multiply by an oscillator of half the frequency, this thing gets a side band up here and this one gets a side band that is halfway down and those two will collide. And when they do, they will superpose. And furthermore, depending on the phase, they will sometimes superpose into something stronger and sometimes superpose into something weaker. All right. And now the next funny thing is, let me turn the DC off. Actually, let me just have one of them again. OK, so here's the nice original signal. Oops, I didn't do that right. This is more like this. Sorry. [tones]

Instructor: [33:57] OK, so now we'll start pushing the frequency up. Then we'll split it into two partials again. And as we keep going up, what's going to happen when we hit zero? Well, we're going to keep going, but we have a doppelganger who is going to come back the other way. [34:25] So what you saw, if you just sort of think of things in terms of characters, is this peak just bounced off of the vertical axis. What happened mathematically might better be described as you don't see the negative frequencies, but there is also a peak here and a peak there.

[34:43] And this peak kept on charging towards the negative as they were getting split further and further. But the one that was already negative charged back the other way and turned to positive. And there is a special case right when I ask the thing to modulate it so that it goes to zero frequency. Oh, I didn't do it right. Oh yes, that is 32.

[35:08] Then we get a different strength again because now we have two peaks. Again, the negative and the positive frequency peaks coincide. And now we get another situation where the phase is controlling how the two act. So here again, depending on the phase, we'll get one or another strength. Oh dig. I almost got it turned off. And that's just what that is.

[35:35] If you want to control that exactly, you have to control exactly the relative phases of the two things that you're multiplying.

[35:45] All right. Questions about that?

[35:53] So in general, ring modulation, sometimes people use to mean multiplication by any old thing. But ring modulation in the simplest sense of multiplying by an oscillator is putting out a sinusoid.

[36:08] What it does, if you give it a spectrum that just can be described as a bunch of peaks, is it takes each peak and splits them. And furthermore, as the peaks go further and further away from the original, sometimes they bounce off of the zero frequency.

[36:23] And meanwhile, when any two peaks coincide, they coincide but they don't necessarily add amplitudes. They do something found only by the triangle involved.

[36:37] All right. So to make these a nice full picture, [tones] there's kind of a typical ring modulation output spectrum. And if you wanted to really go into it, this is several, this is two and change times the fundamental frequency.

[37:02] And so the DC peak got thrown all the way out here and meanwhile all the other peaks got sort of scattered around in that particular way. OK.

[37:09] All right. Yes.

Student: [inaudible 37:16] [37:14]

Instructor: [37:23] Well yes, negative frequencies... In general, what ends up happening, everything, for technical reasons, ends up being symmetrical about the frequency zero. So that anytime you make a negative frequency you hear it as a positive frequency. [37:40] So the general rule is that the frequencies you hear here are the frequencies that went in plus this frequency and minus this frequency. Except that when you compute the frequency minus that frequency, if that is a negative result, flip it around to positive, take the absolute value of it, is what you would get. OK?

[38:02] Now I want to talk taxonomically about spectra a little bit more so that I can have more terms to tell you more qualitatively what sorts of things you can get out of this.

[38:14] Now this is all just what happens when you multiply a signal by a sinusoid.

[38:20] So one thing that your ears told you was that here, when we multiplied by this sinusoid, oh, I think I have to just turn this all off. I'm not sure I'm going to be able to get rid of this all together. Let's see.

[38:40] OK, so what happened here was when I told it to multiply by a decently low frequency sinusoid, and by the way, I chose the same frequency as the frequency of the thing I'm modulating by, then I get something like that in the waveform and I get something that ... I can't show you the spectrum of this in this patch right now, but it sounds not terribly different from the original sound, which is something like this.

[tones]

Instructor: [39:11] But as the frequency goes up, the sidebands are being pushed further and further out, further more the sidebands that are negative are getting pushed further and further out, furthermore, the sidebands that are negative are getting pushed further and further out because they're wrapping around. And that becomes more and more true as you go up, so that you get sort of a knot of frequencies that gets higher and higher. [39:35] Unfortunately, you can't just use this in its current state to make a nice sweeping filter kind of effect. Because if I slide this from 60 down to 550, was harmonic at the outset and it's harmonic at the end, but it goes through a whole bunch of enharmonic results intermediate. You have to do something smarter if you want then to be able to make continuous changes between these.

[40:06] All right. Now, to show you something about how you can predict that. You have to go make more pictures. But now, it's better to show pictures, that are just dead pictures in the photo, as opposed to this live demonstration. So, here, first off, talking about spectra. I've been using some terms without defining them, and other terms I want to define right now. In general, a spectrum ... a spectrum, at least for our purposes, is going to be a description of how strong the frequency content, or how strong a sound is at all the possible frequencies.

[40:48] This is what a ... this is something that you could talk about it, for a sound or for light. This is a representation that ignores time. So, right now, we're just going to sort of pussyfoot over the fact that time is changing and this spectrum could be changing in time, which is not a mathematically correct thing to talk about, but which is in fact the thing that you have to talk about when you're talking about sounds, because they do change in time.

[41:13] So, we're just going to forget about that, for now, and we'll deal with that a little bit later. Or maybe we'll let Tom Erb deal with that in Music 172, I'm not sure. So, the basic deal is that spectra consist of a description of how loud the various frequencies are that make up a sound, but there's a more fundamental distinction, which is, is the sound to be regarded as being made up of a discrete set of frequencies, or in fact is it a continuous frequency thing like white light, or like noise.

[41:48] So, in sound land, you can make ... you can, by either playing a string instrument or whacking a bell, you can make things that have, perhaps an infinite but at least a countable collection of frequencies in them, and if you restrict yourself to Nyquist's frequency, there will be a finite set of them. Or, you can have something like noise, which I haven't told you much about yet, but noise could be better described as consisting of a solid mass of sound at all frequencies.

And, the distinction there, is between a discrete spectrum, like these two, and a continuous spectrum, like this. This looks like a discrete one, but I'm trying to describe the [inaduible 42: [42:25] 39] . I haven't even shown you how to make noise yet, but just type noise-tilde and the object and you'll get noise, but you won't be able to do much with it yet.

[42:47] So, noise is available. Noisy sounds, which are also sounds that you would get just by regular out bursts like this. Are sounds that you can't describe as being objects that have a fixed set of vibrational modes that sit there and vibrate and you can listen to them. For some deep reason, your ear loves things that vibrate in modes, and is built to be able to separate sounds that are distinguished by the fact that they have different modes of vibration in them.

[43:20] You can argue about why that would be, but it might have something to do with being able to hear people speak. Because there are modes of vibration that you set up in your throat, and your throat makes noise, and you can use that modality to hear someone speaking over a background noise. Your ear has been very well-developed to do that, and that hearing facility, but I think was probably originally for listening to voices, turns out to be what makes music possible as well.

[43:45] Music in the sense of things that have pitches. So, continuous noisy spectra are things that aren't described that way. Aren't described as things that just have modes that vibrate, but rather things that generate sound because of heat or whatever, some kind of random motion, as opposed to a vibrational motion. I'm waving my hands pretty, both metaphorically and physically here.

[44:10] So, that's the difference between a discrete spectrum and a continuous spectrum. And those two terms are not exactly accurate uses of mathematical terms. If you happen to have a discrete spectrum like this, oh, whether you have a discrete or a continuous spectrum, you always have ... you can always pretend that you have a thing which is called a spectral envelope. Which is an imaginary curve that you draw over the spectrum to describe what the spectrum looks like as a shape, as opposed to ... as opposed to what.

[44:44] Well, OK, so the spectral envelope is this line here, or this ... what do we say, this curve here that I drew. Which actually is the same curve for all three of these examples. The envelope is in some sense, an idealized description of what the spectrum looks like shape-ishly, as opposed to in the details of where the frequencies are. So, to speak very loosely, the spectral envelope is in some ways related to the timbre of the sound in a way that's independent of the positioning of the frequency components which make the sound up, which could be discrete or continuous.

So, this would sound noisy, and this would sound pitched. And this would sound - oh, yeah, right. Next thing: [45:28] If you have a discrete spectrum - that's to say a spectrum that can be said to be consisting of a bunch of different frequencies that you can just write out, so that they'd be finite, up to a finite frequency - then you can say, "Is it a harmonic or an enharmonic spectrum?"

[45:49] What that is saying is, "Are the frequencies that we see multiples of a fundamental frequency?" OK. These terms are loose because we're talking psych-acoustics here, in some ways. But, for something to be harmonic, it's frequencies should be multiples of a thing that you can hear as a pitch, which means maybe above 50-ish hertz and below 4000-ish hertz.

[46:17] That's hand wavey, too, because you can hear pitches down to 25 or 30 hertz, but it gets harder . If it is true that all the frequencies that you see our multiples of some fundamental that looks somewhere between 50-ish and 4000-ish hertz, then, multiples, then you can say this is a harmonic spectrum.

[46:36] And, if you took such a sound and looked at it as a signal in time, you would see a repeating waveform. So, there's this great fact about repeating waveforms, which is if you look at their spectrum, you will see a harmonic spectrum.

[46:52] And the frequencies present in the spectrum will all be multiples of a frequency which is the fundamental frequency, which is one over the period of the repeating waveform. OK? And that's acoustics.

[47:09] Oh, and for interesting reasons, both a cylindrical air column and a string stuck between two things, turns out to make harmonic spectra, because miraculously enough, the various modes of vibration of either an air column or a string are all multiples of the fundamental frequency.

Student: [47:31] It's integer multiples, right?

Instructor: [47:32] Oh, thank you. Integer multiples, not just multiples. Yeah. I probably have been saying multiples, meaning integer multiples all day. [laughs] This is why you need two mathematicians in a room. [47:46] The mathematician's worst enemy is unstated assumptions. All right. Enharmonic spectra are spectra whose component frequencies are not describable as integer multiples of a good fundamental. And that would be typical of, say, a metal object that you whack and vibrate it. A metal bar, or a bell, or that sort of thing. Solid, vibrating objects that aren't strings I guess tend to have this effect.

[48:18] OK. So, we saw a patch that imitates a bell, then we say bell patch. And, if you look at those frequencies, the point five six and the one point four, I think, those are not all integer multiples for one good candidate for a pitch, and so, as a result, you hear an in harmonic sound.

[48:35] You can ascribe a pitch to it, but it's a different thing from a harmonic sound. And, if you look at it in time, you wouldn't see a repeating waveform. OK. So, the spectral envelope is a handy way to determine, or just describe the shape of the thing. Then in the shape, you can color it in with either a harmonic or enharmonic discrete spectrum or with a continuous spectrum.

[48:59] And, that's not a complete description of sound, by any means, but that's a working description of sort of a first layer of distinctions that you could make between different large classes of sound for making brutal distinctions. Questions about this? Yeah.

Student: [49:17] You know [inaudible 49:18] we aren't not talking about enharmonic, right? Because last week we talked about enharmonic [inaduible 49:23] . Enharmonic means it's not harmonic, right?

Instructor: [49:27] Yeah, and anharmonic, I don't know what it means. I ought to. I mean, etymologically, it means...

Student: [49:35] It's like techno transposition in notes. It's like C# to B flat.

Instructor: [49:40] Those are anharmonics?

Student: [49:42] En.

Student: [49:42] Enharomonic's, E-N.

Instructor: [49:44] Oh, E-N. Oh, oh, oh, oh, oh, yeah. OK, I'm from Tennessee, where we don't make differences in pronunciation between E-N and I-N. [laughter]

Instructor: [49:54] Yeah, sorry. So, yeah, I don't know about that term at all. That's a music term, and I don't have a license for talking about that kind of stuff.

Student: [50:01] So enharmonic just means it's not harmonic right? It's a different term.

Instructor: [50:03] Yeah, yeah, it just means not harmonic. And then there's anharmonic, which means, doesn't know about harmonicity, but I'm not sure how you're supposed to use that term, so I stay away from it. [50:17] Yeah. OK. So there's that. Now to go... actually, let me stay here. Now, using that language, I think I can go to the next thing. I'm going to skip the equations, and reach for another picture. Nah, not that picture.

[50:38] This is stuff that I just described to you being shown in a good text-y way as opposed to a demonstrate-y way. This is non-moving pictures that just show beat splitting because of multiplication of sinusoids, and all of the cases.

[50:55] But, what I really want to do is get down to this picture, yeah. Here, now, is a way of thinking about what happens to both the frequency content and the spectral envelope of a sound when you remodulate it.

[51:12] OK. So, we're taking a sound here, and for the sake of argument, I'm starting with a harmonic sound, and I'm not assuming that the sound doesn't have a zero frequency component, because that might be a musical thing to have in the sound.

[51:27] And, anyways, some things will get it, regardless of whether we wanted it there or not, for reasons that will show up later. OK? Now, we will take that and multiply it by a nice sinusoid with a low frequency, a frequency that's small compared to the fundamental frequency of this harmonic sound.

[51:46] And, this is now showing the spectrum of an imaginary harmonic sound, right?

[51:50] So, then what's going to happen is, oh, let's see. So, all right. This peak turned into that peak. This peak turned into these two peaks here. There and there. This one turned into these two peaks here and here. This one turned into these two peaks here and here, this one turned into these two peaks and this one turned into these two peaks.

[52:24] And then, either to clarify or to obfuscate the matter, I'm not sure which, I tried to draw a spectral envelope through all the peaks that wraps around through zero frequency. In other words, the peaks that were going down, I drew one part of the curve through, and the peaks that are moving up, I drew the other part of the curve through. In order to describe really, in order to try to represent really the fact that what we're looking at is just the positive frequency portion of a thing, which has negative frequencies as well, but happens to have symmetry about the zero frequency axis.

[53:02] But we can see that, although we lost amplitude here, well, we lost amplitude, but we also got extra peaks. And you could talk about the power, and blah blah. That will come. But by and large, the spectral envelope of this is something like the constant times the spectral envelope of that, and we could argue about whether this spectral envelope should be regarded as a half of this one, or just equal to it, because there are more peaks here.

[53:30] And nobody will ever tell you whether throwing a whole bunch more peaks into something should mean that you should make the spectral envelope look higher or not. No one knows how to talk about that. Spectral envelope is a completely imaginary concept, only useful for trying to make descriptions like this. And the good thing about this way of representing it is that it works when you start talking about modulating by, or making the modulating frequency be very high.

[54:02] So, the thought experiment here is that we're taking this signal and multiplying it by a sinusoid, exactly as in the working patch that I showed you, and now we'll make the sinusoid we were multiplying by be so high that it's actually higher than most of the peaks in the original signal. So here is the modulating frequency here. This is the thing that DC turned into. And this first partial turned into these two. Now, that was these two, but as we pushed this thing further up, this one got pulled into zero and bounced off it, and now became two positive frequencies.

[54:40] And furthermore, this thing pulled most of the spectrum with it, except for this very last peak, which is still ... this peak here still hasn't wrapped around through zero, as some people say, but it's still positive. So now what we have is a radical change in the spectral envelope. We took the spectral envelope, but thinking of the spectral envelope as being a ... as extending into negative frequencies as well as positive frequencies, we're taking the entire spectrum envelope and hauling it out into some different place.

[55:10] Furthermore, if this modulating frequency happened to be chosen to be a multiple of this fundamental, then all of these peaks would land on multiples of the fundamental too. And again, we would get something that had this as a fundamental. It just would have a radically different spectral envelope. Which would be the spectral envelope unfolded and then pulled out. All right? And that is why that thing sounded the way it sounded, that thing being over on the separate page. That's what's happening when we're doing this.

[tone]

Instructor: [55:56] Oops. That's this. [tone]

Instructor: [56:00] What we're doing now, is we're taking this signal, which has whatever spectrum it has, and we're multiplying it by something whose fundamental frequency is seven times the original fundamental, that's pulling it way out into high frequency land, and meanwhile, now we're hearing it and its mirror image. All right. And you can do this with anything. Yeah. Questions? Oh, right. The other thing about that is, here is the waveform, if you want to see that. This waveform also does the right thing as you multiply by ... here's a low one, and here's a very high one. [56:47] As you multiply by a faster and faster sinusoid, our original thing which looked like this, turns into a more and more wiggly waveform, which is also constant, or in agreement with the fact that we're hearing progressively higher and higher frequency content, even though we're not really hearing this pitch as such. The pitch is this frequency.

Student: [57:09] If you're supposedly clipping at negative 0.2 then how come you can see frequencies below? Isn't it cutting everything below negative 0.2?

Instructor: [57:18] That's not a frequency. That's ...

Student: [57:22] Amplitude.

Instructor: [57:22] Yeah. That's an amplitude. So, oh right. So here, why do you see things below there? It's because we're multiplying by this oscillator, that ranges in value from plus to minus one. And so, these large negative values are when the original waveform was up high, but then got multiplied by minus one. So this is minus 0.2, which is getting multiplied by this sinusoid, so it's ranging from minus two to positive two, because this is ranging from one to minus one. [58:01] All right. This is stiff medicine, I know. OK, but if you get your head around this stuff then you can make all kinds of cool sounds, so it's good to be able to do, as a, not quite a sight aside. OK, here's a sound.

[tone]

Instructor: [58:23] If I happen to choose not... Oh, let me go back to, let me go back to 110. [change tone]

Instructor: [58:31] All right, here's this sound. What would happen if I said not 110 and not 220, but halfway between them, which would be 165? [change tone]

Instructor: [58:41] It goes down an octave which, let me graph it for you. What now is going to happen is that every other wave form of this is going to catch this, being negative what it was before. And as a result, it'll be whatever it is followed by itself upside down, followed by itself right side up, and so on. And one way to think of that is that it then has twice the period, because you have to wait for the right side up one to repeat, which takes twice as long. So that would explain why the pitch went down by an octave. [tone]

Instructor: [59:15] See, here's the original. [change tone]

Instructor: [59:16] And here's that. [alternating tones]

Instructor: [59:22] And furthermore, over here, one can explain that by... [tones]

Instructor: [59:31] Come on, that was kind of cool. [laughs] So, let's make this eight. [tone]

Instructor: [59:35] So here's the original sound. [tone]

Instructor: [59:37] And here's the sound being remodulated by something an octave below, that's to say half the frequency. [tone]

Instructor: [59:45] And now we've got... Not only did we get the thing down an octave. But we've got a thing that only has odd harmonics. We have one half and three halves and five halves and so on, times the original frequency. Because each peak, the peak that was one times the original freq-, the frequency became 0.5 and 1.5. The next one became 1.5 and 2.5, and so on. And all the side bands crashed into each other to give you, again, the same number... Well, the same number plus one of partials as we had original. [60:19] Roughly the same spectral envelope, because we're remodulating by relatively low frequency and so, because of this picture, we didn't change the spectral envelope very much. But we changed the placement of the partials. And in particular, as a special case, we replaced the partials with ones that are placed at an octave below the odd harmonics.

[buzz]

Instructor: [60:44] All right? So that's a thing that you can do with remodulation. If you only knew what the pitch of someone's voice was, or the frequency of someone's voice was, from the old frequency, then you could divide that by two and multiply it by their voice and you drop their voice down an octave, and you would roughly be respecting the spectral envelope of the original voice. All right? So, since that's a good thing to be able to do, let's do it. That actually shows up in another one of these examples. [inaudible 61: [61:13] 22] the browser still. Oh, is this it? Yeah, this is the example right here. So, with apologies, here is our favorite radio announcer.

[tone]

Instructor: [61:36] Oops. Let me get rid of this. Shut up. Let's get rid of this now. [tone stops; buzz continuing]

Instructor: [61:49] OK. All right, now continue with soft and relaxing, and come back. OK, I haven't told you about this, but there are - ooh, that's the old one - but there are objects in PD that will try to determine the pitch of sounds. [buzz stops]

Instructor: [62:05] One is called fiddle. Oh, looper, yeah. This is just a sample looper, just like you know about. There's the sample, we've got a phaser. Not even doing anything special, I'm just multiplying the phaser by 44-100 and then reading the table. Very sloppy. And then we're hearing this.

Recording: [62:23] The stock number. Continue the stock number.

Instructor: [62:25] And then if we take that and figure out what its pitch is, using the wonderful fiddle object... And by the way, maybe we should... This is pitch and amplitude pair, there's stuff to do here that I would have to describe fiddle to tell you how to do in detail, but you get the help window for fiddle and see it. [62:41] But anyway, it's pitch and amplitude pairs that we'll unpack to just get the pitch. Then we use Moses to get rid of thing, get rid of pitch estimates that are zero, because that meant it just failed to find the pitch altogether. And then we have a nice number, which we can convert from any frequency and then we can multiply it by a half. So that's...

[63:01] So you take the fundamental frequency of something and multiply it by a half, or an octave down. And then, that can be the frequency of an oscillator that we will multiply by the radio announcer. And then I, just to make it louder, I multiply by two, and then you get this.

Recording: [63:15] Continue with stock number ... Continue with stock number ...

Instructor: [63:17] See, here's the original. [recording starts, stops]

Instructor: [63:20] And here is the octave down, but only on harmonics. [recording starts]

Instructor: [63:24] And then if you want the whole thing, you add them together. [recording stops]

Instructor: [63:31] This is, the funny thing about this example was... [recording starts, stops]

Instructor: [63:35] The fellow's voice actually goes all the way on to 50 hertz anyway. So first off, we're giving fiddler a real workout. But second, the frequencies which are coming out are, that you hear coming out, could be going down to 25-ish hertz, which are just monstrously low, even for this guy. [recording starts, stops]

Instructor: [63:54] The other thing to know... OK, so the general rule is multiply by oscillators with relatively low frequencies, you maintain roughly the same spectral envelope, but change the corers-, the component frequencies, and as they... Oh, as a result, you heard... [recording starts]

Instructor: [64:13] ...the same vowels coming out as the original. You can... [recording stops]

Instructor: [64:18] Its intelligible speech still heard... I don't know how you, I'd have to give you some real speech so that you'd, so you could decide whether it's intelligible or not, because we've heard that so much that, who knows what it is now in your ears? [64:30] But at any rate, it would be intelligible speech if you put intelligible speech in, because the spectral envelope is roughly speaking preserved by multiplying, or remodulating by a relatively low frequency sinusoid. It didn't move things around very much in distance, but it changes the component frequencies.

[64:48] The other thing, or another thing that you can do is say, "OK, let's just multiply him by an oscillator that is 15 times the fundamental frequency." Now, what that'll do is that'll take whatever he's got and throw it up into, well... So, if he's going, he's ranging from 50 to 80 hertz, so that times 15 is... Something like a kilohertz to 1500 hertz.

[65:13] That's taking all the nice, those nice low frequencies in his voice and turning into things that aren't low frequencies, right? And then you get this. Let me turn the original off.

[recording]

Instructor: [65:26] So now you can get, not exactly tip miking, but it is... And still, you can persuade yourself you hear the same pitches. And actually, this is also good to add to the original. This is just... [recording]

Instructor: [65:39] This is sort of monstrous. [recordings stop]

Instructor: [65:41] Something that, I don't know, the sound of someone talking through a paper plate, or something like that. OK, so that's... Yeah?

Student: [65:51] So is this how the octave dividers from the '60s, the pedals they had back then worked? Or is it different altogether?

Instructor: [66:00] I'm not dead sure about this, but I think what they did was something simpler, which was, they assumed the incoming sound was a sinusoid and they put it through a triggered flip flop, a D flip flop.

Student: [66:13] OK.

Instructor: [66:17] And that would be electronics. It doesn't work as well when you do that. But on the other hand, if the guitarist is very careful, he can get it to behave, OK? And there's a wonderful solo in Led Zeppelin to prove it. Yeah, this works much, this is much easier to get to work than that, with the, and if you tried any of those old pedals, you'll know what I'm talking about. [buzz]

Instructor: [66:42] Oh, and another thing about it is that this only works with a monophonic signal. If I gave this a signal that had two different pitches in it, as if, for instance, if you played two strings on an instrument together, this wouldn't be good for dropping that by an octave. For this to work, we're assuming that the signal coming in is periodic, or almost periodic. Otherwise, it'll do something else. [67:13] Put another way, it's a perfectly linear process, so it'll, so to two strings would do exactly what it does individual strings added up, except that the, you've got to choose one pitch to modulate it by, and which of the two pitches of the strings you're going to choose. So you can get one of the strings to go down an octave, and the other string turns into something else. So that's just a simple cheap thrill with remodulation.

[67:47] OK. Actually, yeah. So are there any questions about this before I go back to the original example? Yeah?

Student: [67:55] Can you explain how [inaudible 67:57] ?

Instructor: [68:02] Oh, what? OK. So that's, yeah. I can show you that happening here, I hope. [tones]

Instructor: [68:09] Oh, wait, let me turn this off now. So now what's happening is, as I increase the frequency of the oscillator that modulates by, each of these things... No, this is going to be the sample. Each of these things turns into two side bands. [68:26] If I want this side band to crash into this side band to superpose with it as one frequency, I would make this thing be exactly half of the fundamentals, so that each one of them would come halfway over and then they would meet. And that, we do this way. Oh, there.

Student: [68:47] So if you put in 24, would that all sound similar?

Instructor: [68:51] Yeah, you're right. Except, again, now, OK, so 24, let's see. We're doing half integers times 16, right? So 8, 24, 40, 56. I can't do, oh, 80 might be, oh, 72? Yeah, there we go. Now 80, 88, right? As opposed to the multiples of 16... Turn it off. [tones stop]

Instructor: [69:35] OK, so again... Oh, actually, changing this by a multiple of the fundamental even, of the fundamental frequency will give you another spectrum that, which lies down on the same frequencies in some sense as the other. [69:54] In other words, if I start adding or subtracting multiples of the fundamentals of this, I'll make other spectra which will line up with the spectrum I just got. And that could be the original frequency of these, or actually integers, or it could be half the original frequencies if those are half integers or something else in harmonic, if they were something else, like three.

[tone]

Instructor: [70:20] Now I can make more spectra with these frequencies by adding 16 again, which is... [tone]

Instructor: [70:27] Which is being normalized to one. So one to 35... [tone]

Instructor: [70:33] 35 plus 16, anyone? 41?

Student: [70:35] 51.

Instructor: [70:36] Oh, 51. Thanks. Ooh, sounds better. 41 was wrong, right? 51's good. And so on, like that. Crash. [tones stop]

Instructor: [70:50] Are there questions about that? OK, next matter. This will go on until we're done, I guess. This is going to go on for, probably about four lectures' worth. I should say, there aren't very many things to do; there's just lots of ramifications of a very small number of things. [71:16] So really, all we're doing for the NOS is taking oscillators, running them through non-linear functions, and then multiplying things together, sometimes adding things together. It's now perhaps time to go back to this original clip example.

[tones]

Instructor: [71:50] This is a clip of sinusoid, and, by the way, you know this already, you can now get timbres by changing what you clip by. That's these waveforms here. If you were smart, you could harness that to give yourself a collection of timbres, all of which are harmonic. [72:14] If you do it right, you can start talking about things like, "What's the relative concentration of energy in low versus high harmonics in a sound like that?" That might give you ways of making qualitative changes that you'd want in the spectrum.

[72:32] What happens when you do this? The only way of describing what happens when you do this that is easy to understand-- well, there are two things that are easy to understand; one is that clipping is distortion. That people understood perfectly well in the 50's and 60's.

[72:49] The other thing is that you can also think mathematically about what happens when you apply non-linear functions to oscillators. The simplest non-linear function that you might want to apply is squaring.

[73:04] Where we're going now is we're going to work ourselves up into functions by just talking about polynomials-- that's to say, the original signal, which is itself, it gets squared, it cubed, it to the fourth power, and so on. It turns out that it's very easy to analyze what that does to the frequency content of things.

[73:23] Then, if you get a more complicated function-- if you can approximate it with a polynomial-- then you can come up with a description of what the non-linear function does. Furthermore, if you're smart, if you want to do something, you can dream up a polynomial that might do that for you, or make a function that it approximates, apply that function, and get a desired effect. We'll start out inductively by saying, "Here's an oscillator..."

[mumbling]

Instructor: [74:14] So, now we're just going to try the simplest possible thing, which is to take the original signal and square it. Let's see if we can get rid of this, so I can show you this and the clip one, and give you these for comparison. [74:36] To square something, all you have to do is multiply it by itself. Now, squaring things does funny things to their amplitudes. If I took this thing and doubled it in amplitude, then, after I squared it, it would quadruple in the amplitude. This is a thing which does not respect changes in amplitude. Well, it respects it in the sense of getting louder when it gets louder in some sense, but other than that, it doesn't do the thing that you might wish. It does something better.

[75:23] Here's the original oscillator, which I wanted to graph for you. Here's that oscillator squared. Here the period is about a half of the table, and here, if I show the thing squared, it changed. It's still a sinusoid-- I can prove that it's still a sinusoid; I can play it for you.

[75:56] Here's the original, [tones] , and here's that signal squared [tones] . Isn't that interesting?

[76:10] Furthermore, that agrees with what we see. Here's the sine of omega-t, where omega is the angular frequency, and here is the sine of omega-t, quantity squared. Actually, let's take cosine of omega-t, quantity squared. There's a relation which says that the cosine squared of theta is half of the quantity one plus the cosine of two theta. That was Algebra 2, I think, right? So, everyone's forgotten it.

[76:48] It's a special case of the master trigonometric identity for computer music, which is cosine A times cosine B is a half cosine of A plus B, plus cosine of A minus B. That's the thing that describes ring modulation, and it also describes what's happening here.

[77:04] You see that the half is the fact that it now ranges in value from zero to one. It's now one-half plus a sinusoid of amplitude of one-half, and that sinusoid has twice the frequency of the original sinusoid.

[77:22] The reason it sounds just as strong is psycho-acoustics; it's because your ears are much more sensitive to this frequency than to that one. At some other frequency range, it might sound a little quieter.

[77:37] I'm going to stop there for now, let you think about that as a possibility. Oh, thank you, yes. There are CD's with...

MUS171_02_15

Lecturer: [0:01] ...instrument [musical tones] of which I've made a simple example. I'm sorry, this is a little boring because it's another 16 time sequencer. But this one, if you listen to it carefully [musical tones] acts not quite like an analogue synth. You can't really get an analogue synth to do this exact thing but you can get kind of close to it. So I'll slow it down. [musical tones] [0:26] The basic deal is the timbre of the sound is changing during the life of the sound so it doesn't just go beep, beep, beep like a Hammond organ that the sound is brighter at the beginning than at the end. [musical tones] It actually... I can slow it down some more. There. So you can hear how the sound is brighter at the beginning than it is at the end.

[0:55] Now, there are two fundamental ways in electronic music that one does this, of which you know one. The one that people reach for if they are used to working in studios is a filter. But I haven't told you about filters yet and I might not even be able to tell you about filters in this quarter depending on how things go.

[1:15] The other way is by wave shaping. Just the technique that you've seen so far. So those of you who've played electric guitars probably noticed that if you put your amplifier in overdrive then the volume control on your guitar is actually a tone control because the more you saturate the amplifier the more brilliant the tone becomes in some sense. That's the same techniques as what computer musicians call wave shaping. And that's what's happening in this patch here.

[musical tones]

[1:49] So this is just nothing but a sinusoid going through, oops sorry, going through a skillfully chosen transfer function. Not that skillfully actually. And the thing that changes the timbre of the output is just changing the amplitude of the input.

[2:11] So if you... Well, I'll show you this in the patch. But the basic trick to making timbres with computer music or using computers, the simplest way of doing it is called wave shaping where you take anything that you want which you presume is some kind of periodic function but the sinusoid is perfectly good, control its amplitude, and then pass it through a non linear transfer function. And then probably control its amplitude again so you can turn the thing on and off.

[2:41] The then first amplitude control actually changes the timbre of the sound. And I can prove it my example. Oh, actually, before I prove it by example, what I'll do is start this back up and show you what the controls are that I put on it. [musical tones] There is of course the speed control. The duration.

[2:59] You all know how to make envelope generators. Envelope generators are just line tilde objects with messages that turn them on and off, which you saw in the holophonic example. Actually, the homework that is due this coming Thursday has an envelope generator in it, which you need in order to be able to turn sinusoids in them and turn them off in the way that ramps in time.

[3:25] So this is that same thing almost.

[music]

Lecturer: [3:29] It's actually a monophonic instrument. There's no polyphony in here at all, which is therefore more what like what an analog synthesizer would have done. The sequencer is nothing but looking up a nice table using a metronome, and modular arithmetic to go through the table, exactly like in the previous example, I think. And the timbre variation is happening just using an envelope generator. And I'm not even bothering to control the amplitude except right at the output here. [music]

Lecturer: [3:56] And this is just change in the timing on the wave generator. [music]

Lecturer: [4:04] And it's useful to be able to change this. And that's it. That is the whole thing. And you know you can turn this thing on and run the tape and sound like Morton Subotnick sort of, not quite exactly. [4:16] The extra credit example is similar. Actually, I'm sorry this is not as imaginative as I was hoping. It is the same extra credit as during homework three. But homework three was actually impossible to do.

[4:33] Did anyone actually do the extra credit for homework three? That was the eight-note sequencer where every third note had to have a different timbre? Which was just hard, right? I hope it was. It was hard for me, so I hope that it was hard for you.

[4:47] If you do that but drive it with a metronome, everything becomes a great deal easier. You just didn't have that way of doing things before.

[music]

Lecturer: [5:00] So now it is very easy to do this kind of stuff. [music]

Lecturer: [5:12] And all this changing now, I am not having to do any weirdness about turning oscillators on and off. That is just changing the parameters that go to the envelope generator that is controlling the thing, which I am going to define later as the index of modulation. That is the amplitude of the sinusoid that you are putting into the wave shaping function. [5:31] So that is homework number seven, which is due on the 24th, a week from Thursday. You of course are all still working on homework six, and that's cool. But I want to show you that, so that you will have some context or some sense of direction as I show you again painfully or what do you say, didactically through the ABC's of wave shaping.

[6:00] This is the [laughter] the most popular. If you look at wave shaping in it's most general form, this is probably the most popular way there is making timbres with computers. And so there is a fair amount to know about it.

[6:14] Although, there is a great deal more not to know about it because, it turns out that in most situations, what wave shaping does is mathematically very difficult or intractable to analyze.

[6:26] So, you can use simple examples, to sort of guide your way through designing instruments. And then when you're designing real instruments, the simple examples will give you intuition but they won't give you exact answers that apply to complicated situations.

So the example that you've already seen is this one: [6:42] take a sound, clip it from any two numbers, I think I was using minus zero point two and one, and then listen to it. And out comes a sound that's clearly not a sinusoid. OK. Oh, let me show you the other function that you've seen first and then make an observation, actually, two observations.

[7:22] First observation is the very simplest one which is, how would you make a time varying timbre using this technique? And the answer is just what I said, take this thing and multiply it by something that varies in time, so multiply it by the output of a line tilde. No, you don't need...that's just the sinusoid. We're going to get rid of that so you don't have to listen to that. OK.

[7:48] So we're going to multiply this by the output of a line tilde and then clip it. And the line tilde's going to be control. Let's see, in this case just a plain old linear control is going to be good so I'm just going to take whatever I have and pack it with a decent interval of time and just make there be a number box. Let's have it in hundredths though. All right.

[8:19] And now we have, this is not going to be great or anything like that, this is just going to be sort of basic. So the oscillator itself, of course, when I multiply by nothing, nothing comes out. So if I multiply by say a tenth, here's the oscillator. Oops, it's very quiet. Oh, I'm going to cheat and make it louder at the amp. I'm not sure that's a good idea or not. We'll see. OK. Oh, sorry. I don't want to do that. I'm doing something stupid. This is how to teach yourself how powerful your amplifier is, but I don't want to know how powerful this amplifier is. OK.

[9:04] I'm dividing by a hundred. I did not want to put zero point one, I meant 10. So now I'm going to go back, turn this down and do this right. OK. So 10. There we go. Sinusoid, right? Now you all know this, but as I turn this up if I put 100 here this is multiplying by one and therefore I would get that sound.

[9:26] And, of course, sounds in between. Here are sounds in between. So now I have the very simplest possible whammy bar that would control timbre. Right, oh yeah and it of course if it's negative, negative amplitudes are the sum of its positive.

[9:41] So if I was doing this in a way that, if I were doing this for someone to use I would make this thing have a range. Like I don't know zero to 500 so that it won't go negative and then I can have this. This does not work. All right. Computer music instrument. This is a little dull but it's only dull because well the main reason its dull is because there's only one function so far. Right, so what can you do with this?

[10:14] OK. First off let's try other functions besides clipping. So I'm going to just copy this clip functions so I can get it back later. And meanwhile I'm going to do stupider things like for instance take the signal and square it by applying it by itself. And now I get nothing and as I turn it on I get and as I showed you last time it goes up an octave.

Because when you square something well there are two ways of thinking about this, why that happens.

[10:48] The way I told you last time is oh it's just a trigonometric identity. The square of the cosine of omega T is nothing but cosine two omega t plus one all divided by two. Which means if you square your sinusoid you will get something at double the frequency.

[11:07] If you take the sound of your guitar and square it you will not get a thing that will not give up. That only works for sinusoids. A guitar signal or any other kind of real signal which you record will have overtones and those overtones will when you square them will not just get squared individually they will look at something else.

[11:32] Meanwhile I should tell you something else about this which is this is good but in general with non linear transfer functions. Oh let's get the actually while I'm here let's do the let's get the clip thing back. I can leave this here and add the clip to it.

[11:51] So I'm going to put this over to the clip thing and I"m going to make another output obviously you can listen to that too. Minus something plus something I don't know. Now let's hear it [noise] Oh yeah turn this up so were clipping now. Now here's the thing about that if you take two oscillators and give them two different frequencies and add them.

[12:27] By the way I mentioned this once in passing and I should have mentioned it again. If you connect two signals to an inlet they will be added. Now if I want to give this thing a fifth up so let's get this at 325 oh, no, Oh right, its, yeah. This is it. Sorry that's not a fifth up that's some other variable or whatever that is. OK so that's a thing you enter two sinusoids so now when you push it to a point where they eclipse I get some of this stuff [musical tone] right?

[13:10] Well, OK, clip tilde is a non linear function and when you send signals to non linear functions not only do sinusoids turn into things that are not pure sinusoids anymore, that's called harmonic distortion in the stereo view, the other thing that happens is when you take the two different sinusoids or give a signal that has more than one sinusoid component in it, they will, as it's called, intermodulate.

What that means mathematically, well hand wavely mathematically, is that there will be distortion products which are not just functions of the one and the other but cross products of the two. So to go back to the simple example, so if this was the complicated example would be [musical tone] . Oh yeah, with the clipping function. [musical tone] . [indecipherable 14: [13:37] 03]

The simple example is going back to just squaring the thing. There we go. [indecipherable 14: [13:58] 09] hear anything now. Oh yeah, I've turned this thing off. OK, so I'll push it all the way up. So now we're going to hear [musical note] . Oh yeah, good.

[14:20] So now, with the clip tilde function, by putting a low amplitude in I was able was able to use just the linear portion of the clip function. So the clip function, if you graph it as a function, is flat and then linear and then flat. Because clip just lets the value through until it clips. So if you don't let it clip, you give it a value that's less than the value at which it clips, then everything that goes in will go out. And in particular, here the sum of two sinusoids [musical tone] is just two sinusoids again.

[14:57] Oops, sorry. I didn't turn the volume up for that to be true.

[15:01] OK? And then as I push it up again that distortion products, intermodulation products as the stereo people would call it.

[15:09] So for in the sinusoid case, the thing is not linear anywhere. In fact, if you like the sinusoid is...sorry. If you square a sinusoid that means you look up functions as a parabola. And the place where a parabola is least like a line, if you like, is right at the origin where it doesn't have any slope.

[15:28] So here, 250 and 325, those are these pitches. [musical notes] But here they are [musical tone] not there at all. What there is, is this sound [musical tone] . That's an octave above this. Furthermore, you get an octave above this [musical tone] . OK, so now you've got [musical tone] and oops, come here. [musical tone]

[16:01] And then if I put them together you'll get another pitch which is about [musical tones] and another one which is [musical tone] something like that. I can't get down that low, right? [musical tones] Two other pitches that were not there in the original. Actually, none of the four pitches that you hear.

[16:20] What pitches do you hear. I hope you hear, oh let's make it this is better than a real example because its harmonic to the point that you can actually find [piano] so like that and then [piano] you hear those two pitches four pitches sorry, duh. OK. What happened? Well this oscillator by itself made one pitch and it also make DC by the way. This one also made a pitch and it made DC. But as you all know A plus B quantity squared is A squared plus B squared plus two AB.

[17:05] That means you get the square of the first one and the square of the second one. But even more you get a product of the two. So yeah, one component of the output of this multiple. Of this squaring function is this times this. Crossterm and that crossterm is what's called intermodulation. And these two pitches [piano] those two pitches were just double the original pitches and these two pitches [piano] were the sum of the two frequencies and the difference of the two frequencies.

[17:47] All right, oh, you've already even seen this because if you multiply two oscillators that's ring modulation which is what you get is the sum and the product of the frequency. So here you get the sum of these two frequencies which is whatever that is 575, you get the difference which is 75 and then you get double this and double that. All right.

[18:12] Of course if we gave it a harmonic sound like 500 here then we're going to get just a nice harmonic sound. [piano] there components frequency are going to be double 250 which is 500 double 500 which is 1000. But also, 250 minus 500 sorry 500 minus 250 which is 250 again and 500 plus 250 which is 750. So you actually get 250, 500, 750 and 1000. All right questions about this? Yeah?

Student: [18:56] I don't get why [indecipherable 19:00]

Lecturer: [19:02] Yeah, so it's two times the coast, you know what I really should have a blackboard or should use a blackboard but it's going to be a mess if I do. So this is the, what's coming out of here is the cosine of 250 times two pi. What's coming out of here is the cosine of 500 times pi too. And when you multiply those two it's the cosine of one thing, times the cosine of another which is half the cosine of the sum plus the difference. That's the trig function you need. [19:36] Where do I have that written down? Let's see. That's in the book somewhere, but I'm not going to be able to remember where, right now. So that's why you get the sum and difference frequency. Yeah, I'll go try to figure that out. It should be at the very beginning of chapter five. Actually, you know what? It's such an important formula. This is the fundamental formula of computer music, so it's... let's see.

[20:07] I'll go here, maybe I can actually go back to modulation and go to multiplying audio signals, and there it is. This is more than you need. [laughter] This is showing you a cosine including the phase, and another cosine including the phase term and then it's this mess. But in fact, it is cosine of the sum of the two frequencies with another phase, and it is cosine of the differences of the two frequencies with another phase. OK.

[20:47] All right. And did I find my way back to where I wanted to be? Yeah. OK, here. This is a picture of squaring the signal, and what it looks like, when you square a signal. Gee, it becomes positive. And this is the transfer function. This is clipping transfer function that I was just talking about here.

[21:02] Here, and this is what the waveform should have looked like earlier, except I drew a symmetric one and in the patch I did non-symmetrical. Important detail about that... Yeah, I can demonstrate this, I think. I mentioned that it's a very special case that the oscillator went up by an octave, when I took the sine of it. Oh sorry, when I squared it, but in fact if I did any even function that I wanted here...

[21:37] What's another good even function? Let's see... absolute value? I hope that this winds. Good, yeah. This is not going to sound nice. This is what happens in analog electronics when you take your nice sinusoidal oscillator, and put it through a full wave rectifier. The absolute value is, if it's positive it lets it through, and if it's negative it negates it so that it's positive again. OK. And when we do that...Let's see I did not do this real well. OK. We'll do it like this. Sorry, so this thing is not working right now because it does not have an input. So now, once more.

[22:19] Oh, let's get rid of this, and let's also be able to hear the original so that you can get that original pitch in your ear again. Here's the original pitch. [musical tones] And here's the original pitch with taking the absolute value of it. [musical tones] It goes up an octave. OK, this sounds mysterious until you think about it, and then it sounds stupid. So, why is it stupid? The oscillator itself, I'll graph the oscillators now. OK, there's an oscillator for you. Amplitude one spends half of its time being positive and half of its time being negative.

[23:01] Now we're going to take this and put it through an even function. That is to say, a function whose output for negative values is the same as its output for positive values.

[23:09] An example of an even function is the sine... actually, sorry, is the second of these two examples. Y equals X squared. That is an even function. That is the simplest even function. No, the second simplest even function. How about the function F of X equals one?

[laughter]

Lecturer: [23:41] The voltage you get when you put your electrocardiogram to a dead patient. All right. [laughter]

Lecturer: [23:47] That is an even function. OK, this is the next even function up, if you're thinking in the terms of polynomials, which is one possible series of functions that we could look up. In fact, that is the one that we're going to talk about later. OK. So it's even. That means if you put a positive number in or a negative number in, the same thing happens. And lo and behold, the positive part of the sinusoid that goes in, gives you whatever it is. And the negative part gives you whatever it is all over again, because it's the same. [24:14] So, the result is the same for the first half period as for the second half period. As a result of that it had to go up an octave, because those periods suddenly drop by a factor of two. All right. So that's a more general statement about why squaring a sinusoid, cinches the sound up an octave. And in general, that would happen if you had a sinusoidal input.

[24:42] Simply, or if you had any other kind of input, half of whose cycle was the opposite of its other half. That's... yeah. If you studied acoustics really well, you probably didn't hear this. But those are things that only have odd harmonics in them. If you do that, if you send one of those things into an even function, you will get something that is an octave up. Because what happens on the left hand side is exactly the same as what happens on the right hand side.

[25:10] All right. Now, if it were true that the function for the negative value was in fact, the negative of what it was for the positive values, then you would get the rest of possibilities. The rest is the odd harmonics. Those are the things you didn't hear for even functions and so, hand waving you might think, that odd harmonics would be a thing that you might get by sending a sinusoid into an odd function. In fact, it turns out to be true.

[25:39] An example of an odd function that you just saw was this one. So now for negative values, you get negative of what you have got for positive values. And as a result, when you send a sinusoid in, the result has the same period as you had before. And furthermore, the results still have the property that the second half of the waveform is the additive inverse of the first half of the waveform. So it's something and then it's minus it and then something and then it's minus it.

[26:17] And if you think about what harmonics would have that symmetry. The first harmonic does, it's positive and then it's negative. The second harmonic loses, because it does the thing, and then it does it again. So it's the same for the second half of the cycle as the first.

The third harmonic goes up down up, and then it goes down up down. And so if you squeeze the third harmonic into the cycle, it will again have the property that the first half is the negative of the second half.

[26:47] That will be true for harmonics one, three, five, seven, nine, and so on. All of the odd ones. As a result of which, if you make a wave-form like this, which is, it doesn't have to be positive followed by negative, it just has to be whatever it is followed by minus whatever it is, so that if it's a negative in the first half, it's positive in the second half. Then you will have something with only odd harmonics and the typical sound that that makes, let's see.

[27:12] OK, so this clip was, I made it un-symmetrical deliberately in order to avoid having this happen, because I didn't want to be confusing or something. But I'll now be confusing. I'll clip between minus point two and positive point two, and now we want to graph this so I can prove that it's doing what I'm telling you it's doing. Now we can listen to it. So here's the original.

[musical tones]

Lecturer: [27:45] And here is the wave-shaped one. [musical tones]

Lecturer: [27:52] And it has that, you know, clarinet-y, held-nose kind of a sound that one associates with sounds that have odd harmonics. By the way, don't let anyone tell you that clarinet sounds are typified by having strong, odd harmonics. That's true for the first 18 and a half tones of the clarinet, if I'm not mistaken, after which it's not true anymore. As soon as the little hole goes open, that quits being how it acts anymore. But it is true for the low notes, that you get these kind of notes for clarinet. [28:28] And for the first 30 years of electronic music, whenever anyone made a timbre like this, that just happened to have, for symmetry reasons, mostly odd harmonics, they said, "Oh, it sounds like a clarinet." And so now you can find clarinet voices on your organs or synthesizers or so on. And it's just things that happen to have that symmetry, regardless of whether it sounds like a clarinet or not.

[28:48] To any reasonable pair of years this doesn't sound like a clarinet at all, it sounds like a very cheap computer music instrument. Oh, and what does it look like? Ta-da. What I showed you before. And it does have the correct symmetry for having odd harmonics.

[29:04] Now, why am I belaboring this point? Because now, I've shown you how you can make odd harmonics...

[musical tones]

Lecturer: [29:11] ...and even harmonics. [musical tones]

Lecturer: [29:14] Oh, yeah, I should show you that as a wave form. That was the one where I took the absolute value of the sinusoid. And now the thing can't go through zero without taking the absolute value. So the second half of it is flipped upside down so it's positive again. All right. [29:34] And that of course had to go up an octave. And now and this is an idea that I think that might be originally due to Don Blokah. At any rate the only symph I know that knows about this is Blokah's old modulator sense. Now we have something where we can control the relative strength of the even and odd partialist. [noise] All right, questions about this, yeah.

Student: [30:11] Does it pass through the kind of relative amplitudes like harmonics or because you can't add those signals back together and get.

Lecturer: [30:28] The original sinusoid. No, no. Yeah, so one way of thinking of that is one of these only has odd harmonics and the other one even. So adding them is simply introducing different frequencies which will in fact cancel each other out. However you could make a wave shaping function. [30:51] You could make two wave shaping functions each of which is hardly nonlinear but there's some just be the idea of the function. In which case you actually would have two funky timbres whose sum is the original sinusoid. I don't know if that would be useful but you could do it.

So this is sort of, so this is just sort of phenomenal if that's the word. Experimental, intuitive. This is just very general observations about

what kinds of wave forms might come out of [indecipherable 31:
[31:12] 36]

[31:29] So far what we've got is that putting two things in will cause distortion products which are sums and differences of the incoming frequencies. Or maybe sums and differences of multiple incoming frequencies. That was what we got when we took these two things in and where we added these two together and started wave shaping. And then we're getting stuff like this. [musical noise]

[31:54] Oh right, it's going to make this 350 again. 325 OK. so now we have complicated sound and more complicated sound right here [musical noise] Oh interesting. They're probably no common frequencies although I can't swear to it. So that, oh, let's look at it, isn't that cool.

[32:17] This is the absolute value. This is the sum of two sinusoids full wave rectified. That's old fashioned talk. So you can imagine every other one of these loads being negative signed before it got wave shaped. And then every other one, this one [musical noise] is the same thing clipped so it looks like this.

[32:53] All right, another observation that is just right there on the surface, different kinds of wave shaping functions have different behaviors when you change amplitudes, in terms of the overall amplitude that comes out. So, if you take the absolute value of something, the louder the signal goes in, the louder you will get out.

[33:19] All right. Pretty much, if you double the signal, in fact there aren't very many oh, real value functions. Something where if you double the input, it doubles the output. The only ones that I know of are the identity function and the absolute value and combinations of those. Although if you think in the complex plane you got a whole bunch more. [laughter]

[33:41] And that's a very rich source of ideas. If you clip, clipping things means that no matter what you put in, the result can't get louder than between minus zero point two and zero point two in this case, which means we have a very... [laughter] What was that word?

We have a very predictable instrument in terms of what kind of amplitude will come out.

[34:04] All right, that's good, or that could be good. In particular if you got the, well this is the thing that you get if you put an electric guitar in an amplifier and overdrive it. The cool thing is the amplifier can't push the tubes harder than full on and full off. As a result there's this sort of basic loudest thing that you will get out of the thing no matter what you do on the incoming side, including feedback. You can do feedback, which is you put it through a linear system, could eventually grow without bounds.

[34:34] If you put it through something that's being clipped, you know it will stop somewhere, and then you will get something that hopefully is at the level that you want. All right.

[34:47] It's still true, it's true of all these things that... let's see, what is particularly true of this one, that we still have this nice timbre control which is, [musical tones] louder and gives you a sharper timbre and furthermore, gives you interestingly enough, a louder sound. So, even though the difference in power between this [musical tones] and this is tiny. [musical tones] This is actually louder than that. Not a lot, but substantially, and the reason for that is Psychoacoustic mumbo-jumbo.

[35:29] Specifically, the loudness in which you hear something is in some sense, the sum of the loudness of the signal in all of the different critical bands that it has energy in. And so even without substantially changing the energy of the signal, it can just spread the energy out over more critical bands.

[35:48] The result will sound a great deal louder because the entire thing is in one band. Because putting a quarter as much energy in four different bands is much, much louder than putting the entire thing in one band. So band width makes loudness in Psychoacoustics.

[36:01] And here what we're doing is we have an instrument which changes loudness. [musical tones] More by changing band width than changing the acoustical power of the signal that you are listening to.

[36:14] So if you have, if your bassist isn't loud enough, you don't get very far by just pushing the bass up, because you just hit the limit of your cabinet. But if you add some overtones, this could be a great deal louder. And if you're mixing music and you want the bass to be audible, even if you're playing it on a boom box, don't just push the fundamental, because it won't come out that speaker. [laughter]

[36:39] Add some harmonics, that will make it loud. And people will think, oh, that deep bass even though the low frequencies which are the actual fundamental and the low harmonics of the bass aren't even present. Same thing is true, even more so with ear buds. All right. Learning how to do that is an art, not a science. OK. [musical tones] So there's that. Questions about this? OK.

[37:08] Now, I'm going to get a little more mathematical. So this was all experimental stuff, with stuff like absolute value and clip, which by the way are functions which are not pleasant to approximate with polynomials that are not analytic functions. But now we are going to take the opposite approach altogether and start talking about polynomials because they are the things that we can analyze the most easily when we're talking about wave shaping.

[37:37] So the first example of the polynomial that I showed you was just squaring, the first nontrivial example. And this was the example, or this was the thing that allowed you to just...Whoops! [musical tones] Got to do this. Now this is happening. Got to turn this off. [musical tones] It takes the thing and bashes it up an octave. OK, so this is an even function, so it happened to get bashed up an octave.

[38:02] An odd function would be, take this thing and cube it. So we'll take the square and then multiply it by the original signal again, and then it's cubed. In fact, while we're at it, let's make a few of these. All right. So now the results, the outputs of these multipliers are going to be the squares of the third power, fourth power, fifth power, and sixth power of the original signal.

[38:42] Now raising this signal to the sixth power could give you a huge amplitude, except for the fact that this oscillator gives you values between minus one and one. And no matter what power you raise that to, it's still going to be minus one and one, positive power anyway. And so, let's see. Let's turn this on and turn up the bass and [musical tones] so here we go.

[39:11] So here's the first power of the signal which is just itself. [musical tones] Here's the square, [musical tones] quieter by the way.

[39:19] Here's the cube. [musical tones] They are getting quieter. Why are they getting quieter now? I'll show you. There's the fourth power, fifth power. You know what? They are getting quieter at the point where I want to get. Oh, duh! I'm not doing what I said I was doing. I just want to make this full blast, they have all the same amplitudes. I'm sorry.

[39:41] So that was 24, that was a quarter going into there and so a quarter to the fourth power was getting to be a timbre kind of signal.

[39:50] Now I am going to try this again, but I'm going to turn it way down. OK. So the original signal. [musical tones] Square. [musical tones] Cube. [musical tones] OK. Fourth power. [musical tones] Fifth power. [musical tones] Sixth power. [musical tones] Seventh power. [musical tones] OK. So, seventh power compared to third power is this. [musical tones] All right?

[40:22] So if you just had something that could freely move between the different powers of the function you could have a nice timbre whammy bar. All right. However... Oh and let me quickly graph of what happens when you...let's take a look at the seventh power of the thing. [musical tones] So that is this thing. Now if we look at it we will see, huh?

Student: [40:46] It's breaking over to there. [laughter]

Lecturer: [40:49] Oh. Thank you. I don't want back trace there. There you go. I was getting worried because Math tells me that this thing should still be between plus and minus one, it was reaching outside there. And when that happens usually you're doing something wrong. [41:09] OK. So, here's the seventh power of a sinusoid, and it is sort of looking like a pulse train, where every other pulse is negated. And in fact if I take an even power of it, it looks like a pulse train again so, accept that, now every other pulse is going the same way. Going in a positive direction, every pulse is going in a positive direction. That is the even odd thing again.

[41:38] All right, furthermore let's do this. Actually, not now, let us do that one. So, I'm going to listen to this for a second. [musical tones] As I mentioned, or as I showed with the clipping operation, clip. OK. So, here. [musical tones] As I change the value, or as I change the volume or the amplitude of the sound that I am clipping that changes the timbre. [musical tones] That is going to be true in general of.

[42:07] Non-linear functions. So here too, and I'm really scared of this one. [musical tones] But here too... [musical tones] Whoops sorry. [musical tones] Still hearing this one. Hm. [musical tones] Turn off. There we go. [musical tones]

[42:26] Well, really it is not so clear. [musical tones] But in fact the higher harmonics are somehow growing at a steeper rate than the lower harmonics. I can show you that to you better a little bit later when I show you some more math. So this thing does have a varying timbre but the problem is that the amplitude is changing in such a way that you can't get the timbre to change because the amplitude is dominating it. OK.

[42:58] There might be ways you could deal with that. You might be able to predict what amplitude you should get and divide by it and I'll let you think about that. So now to go back to the equations and pictures for a second. So this is the wave shaping chapter and then ooh let's not do that yet.

[43:21] A little bit further down here is an analysis of what happens when you actually take powers of the signal. Oh yeah here's me figuring out what happens when you take two different sinusoids and square the sum and show the intermodulation products. But, I already talked about that.

[43:43] Here now is, is it? Yeah, here now is what happens when you just take a nice sinusoid puts some omega in organize the amount of samples as well and start raising it to powers. And this is just algebra in it's delight. The first one is cosine times cosine is a half plus a half cosine double right. And in general the cosine of A times the cosine of B is half the cosine of the sum plus the cosine of the difference.

[44:18] And so here what happens when you cube it you can think of multiplying the square by the original one. And so you have to multiply this term by this term and this term by this term separately and then add them. So this one just gives you a half of cosine omega n again. Which to confuse the matter I wrote as a quarter of the cosine of the minus omega 'n' plus a quarter of the cosine of the plus omega'n'. That's because that's the way to write it down so that the pattern will come clear later.

[44:53] And here the cosine of omega 'n' times the cosine of two omega 'n' that's the cosine of omega 'n' plus the cosine of three omega 'n' and that generated the other half of this one. The other quarter of cosine of omega 'n' and one quarter times cosine two omega'n'. We've all seen Pascal's triangle right. We're doing Pascal's triangle in harmonics. OK.

[45:16] So the next one is, all right so the lowest frequency is minus two omega 'n' and the highest frequency is plus four omega 'n'. It's centered around omega 'n' and not around zero. All right.

[45:26] And now we have 1331 all divided by 8. Those are the probabilities of getting zero one two or three heads in three tosses. OK. And meanwhile the signals that have those amplitudes instead of probabilities are minus two, zero, two and four times the original frequency. This is the fourth power so it's an even function.

[45:50] So we're seeing in mathematics what I told you before by handwaving. Which is, you want to see even frequencies so you can take the even function. And now the next one not to belabor the point that's as far as we go. Divided by 16 now and its minus three minus one plus one, plus three and plus 5. So one interesting thing about this is if you look at the highest frequency which occurs all the way to the right. The frequency of the highest harmonic is going up one step each time you raise the thing to the higher power.

[46:28] Now someone I'm not sure who but it might have been Mike Lebraun in like the early 70's. Thought about this and realized that, if you were smart, you could just isolate these individual frequencies by picking up the correct polynomials. In fact the correct polynomials were easy to think of because they have been named for I don't know how many hundred of years. They're called chubby chip polynomials. And chubby chip polynomials are what you get when you say, "I just want the cosine of five omega 'n' and I don't want this other stuff." So how am I going to get rid of it.

[47:12] Well it's easy. I'll get rid of this one by subtracting out twice this whole thing. So instead of taking x to the fifth, I'll take x to the fifth minus twice x to the fourth. And that won't have any of this...oh wait, I'm telling you the wrong thing. There is no cosine four omega 'n' here. There's only three. OK. So there's five, three which comes in twice and then one which comes in twice. So we can get rid of the three. Cosine three omega term by subtracting some suitable mark of this which I can't do in my head.

[47:44] And then we can even get rid of the cosine omega 'n' after that by subtracting that from the suitable notes of the first one. So there is a polynomial out there which is something times x to the fifth plus something times x cubed plus something times x. Which has a property, we could put a cosine in here to get exactly the cosine of five times it out.

[48:02] All right, and since seeing is believing I made a patch to do that. Here's a picture of I believe the fifth one. This is a polynomial that I stuck inside a wave table. How would do you do that? Its work but you can look inside the patch to see how I did it. And I'm just using timbre four tilde the old friend to read values of this polynomial.

[48:36] And then here is nothing but a nice oscillator going in times eight and amplitude control which I'm going to call the index. And if the index is one that's just to say if I just put this thing in amplitude one, ooh, there's stuff here going on, that I'll explain when I have the actual patch out. Then out will come the fifth harmonic. I'll get the patch out and show you what happens. If I can figure out how to get the patch out. Here. So the patch is in the tedious but essential help browser.

[49:11] We're in Chapter five so they start with E and it's going to be the chubby chip dot PD. By the way this is all stuff I think that you've seen. Here it is and here is the sound of, let's turn it all the way up, here is the sound of whatever this is. So what we're doing is we're computing different polynomials to put in the table and I am realizing now that it's too bad I didn't also have a linear polynomial. Let's just draw one.

There's the original pitch. If I had actually made the patch that would be the sound. There's the oscillator going through just the function wide physics. This is going through a suitably designed problem so before you saw more sort of just thoughtlessly [indecipherable 50: [49:53] 20] raise this thing in order to give you the octave. Here this is the polynomial x squared minus one over two. Which has the wonderful property that goes between plus one to minus one to plus one.

[50:29] And you can compute it if you want but if you apply that to a sinusoid you will get the second harmonic term and you won't get the DC term that you got when you just squared with it. So you just square the sinusoid you get double frequency and then you get DC term. If you just subtract the appropriate thing from the thing then you get just the second harmonic term and nothing else.

[50:55] So now I have polynomials that will take an oscillator just take a pure sinusoid and make pure harmonics out of it. OK.

[51:08] Furthermore it's going to be this one. [noise] Depending on the amplitude of the sinusoid you put in you get different timbres. So what I told you is only true in fact for a unit amplitude sinusoid. Of course if I put in a zero amplitude sinusoid I have to give up nothing. I get probably some DC I'm not sure but I sent out. And in between I get a range of timbres which [musical tones] sounds like that. OK. So there starting to make computer music here. All right.

[51:49] And similarly for all of the others this is an even one which similarly will sound an octave up, OK like that. So this is what in the early 70's people thought would revolutionize computer music. Because no one would ever need to do anything besides this for building timbres. Until people realized that actually static timbres aren't interesting. It's variable timbres that are interesting.

[52:19] And furthermore, although you get to control exactly what you want this thing to be when you give it, take it all the way to the end. You don't really get to specify in addition what it does on the way up there. It just does whatever that polynomial does and there's no choice about polynomials. There's exactly one polynomial that will give you that thing at the end.

[52:35] All right. So in fact you have to be smarter than this if you want to make timbres that actually vary the way you want them to. And how smart do you have to be?

[52:45] You have to proceed by special case. Everything is a special case from here on out. Particular kinds of functions, and particular ways of using them, and combining different ones of these will be useful for making different things. And that will be something that I can't really even give you a summary of. It's just a whole field of inquiry. Questions about this? Yeah.

Student: [53:09] So would we be able to freeze physical modeling in its units, are there programs or functions that would be able to... [whistles]

Lecturer: [53:17] What?

Student: [53:18] I was thinking of one synthesizer that I used to have where you could change the modulation and have a waveform definition pattern. Really, it would sound actually like an paranormal entity. It was just an oddity. Let me you what, a really breathy sound?

Lecturer: [53:33] Hm.

Student: [53:33] Sounded like it was breathing. It's got this similar thing to...

Lecturer: [53:38] Mm-hm. That might have been. Was it a Yamaha DL1?

Student: [53:41] No it was a...

Lecturer: [53:42] No, you're not old enough to have one of those. [laughter]

Student: [53:46] I know the DL1 but it was a Korg Platinum synthesizer. [cough]

Lecturer: [53:49] Oh that one, I don't know. Since Yamaha licensed something from Stanford, I don't thing Korg was actually using so-called physical modeling. Although, it really wasn't physical modeling in the first place.

Student: [54:05] OK.

Lecturer: [54:06] So I fact I don't know what is was. [cough]

Lecturer: [54:09] Yeah. But to be honest that Yamaha thing, that's the original synthesizer that described itself, as physical modeling, didn't work at all. It was a completely different principle of operation.

Student: [54:21] Mm-hm.

Lecturer: [54:23] And so this is the easiest way into making varieties of timbres. There are other interesting but more complicated ones too, and you can spend decades learning them all if you want.

Student: [laughter] [54:35] [musical tones]

Lecturer: [54:39] OK. So, there is that. What I want to do is make another observation. OK. So, just continuing to look at special cases, right? So what are some good functions that you could think of trying that aren't polynomials? Oh! So the problem with polynomials is this.

[54:56] You don't see it because I've only shown you the part of the polynomial that happens to be the good part of the chubby chip polynomial. Of course, this is a fifth degree polynomial. It's leading term is x to the fifth and so it's going to shoot out of the screen just as soon as I get two tenths of a point to either side of the part that you are looking at.

[55:18] All right. It's coiled up tight right in that little rectangle, but that's the only place where it is well behaved. OK. So polynomials, even though they are simple to think about, are actually very ill behaved in terms of the amplitude that you get out when you start putting freely varying amplitudes in. That is almost the opposite situation from the clipping functions.

[55:43] I don't know if they even have names, but functions like clip tilde, where no matter how hard you punch the input, the output is limited to a specific kind of value. And there the only thing is, that is not really an analytic function.

[55:56] That is to say, it's not describable very well as a power series, so I can't really tell you by using this kind of mathematical analysis what it's going to do to the signal. In fact, I don't have anything beside hand waving sort of descriptive analyses of what clipping does to signals.

[56:13] There are analytic, that is to say easy to approximate by polynomial functions, which do reach asymptotes like arctangent. But take the arctangent function and take the first ten elements of the Taylor series, and then plot them, and the result will not look like arctangent. It will look like arctangent in the neighborhood of zero and then of course, since it is a tenth degree polynomial, it will shoot off to plus and minus infinity.

[56:47] Oh, sorry. The terms are all odd so it would be nineteenth degree polynomials. So...

[laughter]

Lecturer: [56:53] So as soon as you out get past the well behaved portion, it's going to blow up harder, right? OK. So how do you get something? Actually let's do the simplest analytic function, which just blows up horribly anyway, which is the expediential function? [57:11] Right. [cough]

Lecturer: [57:13] It turns out that, well the simplest analytic function polynomials. The expediential function is a good one to think about because its turn out you can analyze what happens when you send signals through your expedientials and decent well-behaved things happen. So let me show you that. Let's see. Actually, I'm going to cheat and show you this off of the prepared patch [laughter] rather than build this for you. [57:46] Because I don't want to build, well. There is an EXP tilde object so you can just exponentiate anything you want. But there is stuff to do in order to exponentiate things well, as just opposed to just exponentiating them. So, how do you exponentiate something well?

[58:08] So, here's a picture of an expediential table. Since we're doing computer music, we're going to run the thing through a digital analog converter. So we would like these to be bounded, and the way to make expedientials be bounded is just look at the part of where the exponent is negative. And then it's all going to vary between zero and one.

[58:28] So this is the function E to the minus X graphed from zero to ten. So, down here if you could see it this would be e to the minus ten, which is tiny. And now what we're going to do is use this as a wave shaping function, but we're going to be smart about it.

[58:46] And the smartness is this. Rather than to look in the middle of the function, where the thing is about E to the minus five, I mean we ,could do that. We could have a wave shaping function, and the thing would be E to the minus five which is, I forget. It's maybe 100 or so, we multiply the result times 1000. We could hear it. And then you would increase the index, that is to say the amplitude of the sinusoid when we look up.

[59:14] And then we would start going up this thing on one side and then we would start clipping that later or do something bad anyway it wouldn't be correct. Or at least it wouldn't be the expediential. Or the amplitude would grow in some very unruly way.

[59:31] So, rather than do that, the smart thing to do hear is to take the sinusoid that actually don't just center it around zero because then you'll go negative and that will either be clipped or be growing clipped depending on how you realize it. But do the following real simple thing.

[59:49] Have the sinusoid be arranged so that it reaches from zero to whatever point you wish. So now what's going to happen is rather than look, let me show you, OK. So going back to, where did it go? Here, going back to here. So these functions are all being read around the middle of the function. The simple example was just squaring. Oh, wait, that was the other window.

[60:26] Here in this way of shaping example what were putting a sinusoid in and a sinusoid is variously positive and negative. But the result is centered around zero. And were just leaving it centered around zero. Not moving that center as we change the amplitude. When we change and when instead we use OK, can I get rid of this, great. Yeah, this is the one.

[60:56] When we use a function like this its smarter to have the thing grow from the left hand margin of function out. Because then we know so whatever the amplitude the sinusoid is we will adjust the center of it so that it reaches to zero instead of plus or minus something. So instead of reading around the center we'll read starting here around whatever we need to read around to get it started here.

[61:23] How do you do that? You just take the oscillator and add one to it. So then instead of ranging from zero to sorry instead of ranging from minus one to one like the oscillator does it will now range from zero to two. And now it would be correct at that point to divide by two so it will range from zero to one but I'm throwing care to the wind at this point. And now the index is again a number its intense now which controls the amplitude that result before we go reading it in the table.

[61:56] All right. At some point I want to remember to tell you why I'm doing this but this times 100 because this is in fact 1000 points representing ten. In other words I have ten points on the table for every unit of input for the expediential function. So that when I do the look up it will be decent and accurate. So this is correcting for the units at the table.

And then we will read it out of the table and then we will start graphing it. So let's graph it. Here is a spectrum. Oh, you saw the spectrum. This is the spectrum of what you get when you have a zero amplitude reading at a disc which is therefore putting out solid value of one. So the index is zero so this oscillator is multiplied by zero so zero is going in here. So the result is one wave form who looks like that if you don't see and who's spectrum looks like a peak at the frequency [indecipherable 63: [62:24] 06] .

[63:02] And then as we increase the index now what happens is as the original sinusoid...well when the original sinusoid is at its negative peak, so that when adding one you get zero. Then you get one but meanwhile as it reached out to wherever it goes it goes down to some which will get closer and closer to scale.

[63:35] Furthermore the hotter you make the signal going in, the faster or the less time it spends in the neighborhood of the peak here and the more time it spends out here in the neighborhood where everything is almost zero. And hence the skinnier this pulse gets.

[63:57] So I think this is a good way to make a pulse train commuter. But there are other ways of making pulse trains. This is a pulse train, you can in fact compute its amplitudes and they turn out to be best functions of the second kind. If I'm giving you the right nonsense. But basically you can look at them. Let's see here. You can look at them intuitively and see that what's happening is there's a peak here and the energy is moving out so there's an increasing band width.

[64:34] Furthermore, as you tweak this peak, the peak itself lasts less and less time. And so in a very non rigorous way of thinking the frequencies present in that should be growing linearly as the peak gets squeezed which is to say linearly as this number goes up, if I'm not doing that wrong.

So as this number goes up the energy gets spread over progressively more and more hormonals. And so to listen to it you get nothing if you send the index to zero because it's all DC you can't hear it and then you [indecipherable 65: [65:00] 21] musicians hear that and they say oh that's a brass tone.

[65:27] And actually if you look at how brass spectra changes you put more and more pressure into the brass instrument they do actually spread out in this way. See if you can see any harmonics. Well, you probably can't see the harmonics, while I'm chaining it, very well. Yeah, sort of.

[66:03] So the basic idea is the harmonics will spread out fatter and fatter and to a certain point you will not hear the thing get quieter even though the power of the signal is dropping because it's spending less of its time away from zero. Because of the cyclic acoustical affect I told you before the energy is spreading out into more frequencies. So up to a certain index you actually hear a decently nearly constant amount of sound.

[66:32] That will quit being true when I get up past two or three hundred here. In fact, I decided to protect users from this. But now I'm going to unprotect us. Saying let's just go up to whatever value we want. And now we get [horn blowing] that. And then, eventually. [horn blowing louder]

[67:05] All right, let's just change the scale. [horn blowing at different scale] Ta da.

[67:21] That's something that people used to do with low pass filters. The standard analog synthesizer kind of waved form and sent it into a low pass filter, and you can get that kind of effect.

[67:33] Here in computer music land it's easier to get it this way. Although, people still reach for the analog way of doing it anyhow because it has a particular quality of sound that makes people nostalgic for the 50s and 60s, 60s in particular.

OK. But this is now [horn blows] a very simple but very effective way of making one collection [indecipherable 67: [67:50] 58] .

[67:57] Questions about this?

[67:59] Yeah.

Student: [68:02] Is [indecipherable 63:01] .

Lecturer: [68:05] Yeah. It's even better than that. There's a couple pages of mathematical analysis of what this should do. And what I can tell you about that is this. The tailor series for the expediential has this wonderful property that for very small... OK, so everyone knows the tailor series for easy X, right?

Student: [68:25] No. [laughs]

Lecturer: [68:27] OK, I was kidding. I know. It's one plus x plus x squared over two plus x cubed over six, which is three factorial, x to the fourth over four factorial, x to the fifth over five factorial, and so on. So the denominators are going up crazily so that by the time you get to... Well, [indecipherable 68:53] . Anyway, the coefficients of the terms go up very rapidly. [68:56] Now one interesting thing about that function is that if you say, "What's the loudest monomial in that series." So if you put in zero you get one plus a bunch of zero's so obviously one. If you put in values between one and two if you think about it between one and two x is bigger than one. But x is smaller than x squared over two. So between one and two x is the dominant term. Between two and three x squared over two is the dominant term. Between three and four x cubed is the dominant term and so on.

[69:38] So you can think of the expediential series as the tailor series of the expediential as they sort of fall under the degree it goes up as you push the input up. In the sense of the dominant term going further and further out the series as the value of the input is going up. This is a good way of thinking about it. The positive numbers going.

Negative, stranger situation. Because they're canceling each other. So in one way of thinking the bigger number you put in, the higher harmonics you're going to get.

[70:13] Because as I showed you before as as you start raising the cosine or start raising the sinusoid to higher and higher powers you get those. You get those collection of terms that were spreading out the frequency. That was this picture here.

[70:30] Whoa, come back. Oh, where did I put it? I think I put it here, yeah. No, wrong, it was just here. This stuff. So see how the band width of this is going up as you raise it to higher and higher expedientials. Well that suggests that the expediential, if you expedentiate something the you'll get a mixture of these things and whichever one is dominant will be the loudest thing in the mix.

[71:15] So you should expect to see something whose band width increases linearly in fact with the strength of the signal that you're putting in. That's not quite right. But I won't tell you why it's not quite right. The band width does not go up linearly because its standard deviation of a collection of coin tosses.

[71:39] But anyway you see these things. Each rated according to, each rated according to how important this term is in any particular series. That would be true for any tailored series you can think of it that way.

And for the expediential in particular since it has this very simple behavior which term is the most important. You just get a widening and a flattening [indecipherable 72: [71:52] 06] as the signal gets louder and softer, made the thing get lighter. OK. Now, I'm pulling a fast one here of course. Because, that analysis assumes that I was centered around zero. In fact, what I'm doing is pushing the thing over so that it only reaches zero at the loudest point. But in fact, the expediential function, if you slide an input over, you're simply rescaling the output.

[72:38] That is the quality. That is what expedientials are. They are the same as themselves rescaled when you move to the left and right. And so, in fact that I am sliding the thing over in order to control the amplitude is simply rescaling in the perfectly appropriate way to get the thing.

[72:55] Not only to have the band width property that I told you about, but also to have a well behaved amplitude as the index of modulation is going up and down. So at this point, your expediential should be an all time favorite wave shaping function to try. Except for this one kind of inconvenient thing, which is that...

[musical tones]

Lecturer: [73:19] The overall power of the signal can be rather small if you average over the entire length of the period. That is not going to be a problem if you have good audio equipment. But if you don't have such good audio equipment, you won't necessarily be able to reproduce these functions as well as you would be able to reproduce something whose power was distributed nicely in time over the entire wave form. [73:49] So things that are pulse-y, are great until you put them through a boom box, and then.

[73:55] They are not quite so great anymore. So this is good thing for the studio but you might want to mess with phasers or something before you actually put this on a record. Do people use the word record anymore?

[laughter]

Lecturer: [74:07] I don't know. Never mind. OK. [laughter]

Lecturer: [74:11] This is closely related. So what were the other Taylor series you all were made to learn in Calculus besides, expediential? All right.

Student: [74:19] Maclaurin.

Lecturer: [74:21] Oh, well, the Maclaurin series. They are a different series altogether. Didn't you have to memorize sine and cosine as Taylor series?

Student: [74:31] No.

Lecturer: [74:34] No. No one took you over those coals, did they? OK. [laughter]

Lecturer: [74:39] All right. Yeah, and then there's De Moivre's theorem. I forgot their name. Sines and cosines are nothing but expedientials, complex numbers, linear combinations. And in fact, the same kind of reasoning I show you here should suggest also instead of using an expediential as a look up function, you might be able to use sine or cosine and get something like similar results. [75:08] And in fact you do you and it's even better than that because you're going to get results that you have heard before. Because... am I going to explain this, yeah? Because you can reduce mathematically phase modulations to wave shaping by waves which are sine and cosine, the functions that are sine and cosine.

[75:33] So let me just demonstrate that.

[75:35] OK. You've seen basically three classes of functions I think. We have seen things that consist of linear segments that's the clipping function of the value. Those are hard to analyze in terms of frequency content but are easy to describe in terms of wave form. So they're good petalogical things. Also the clipping thing sounds familiar because you've all heard of the drive.

[76:01] Polynomials and then finally transcendental functions which have tailored series which therefore can be approximated to follow in terms of polynomials. So the two transcendental functions that we're going to mess with is first this expediential and second, let's go back to the one that I'm building here.

[76:22] So that was, what we just saw was E5 and E6 here. Now I'm going to quit using the examples and start just sampling myself. OK. So here I'm going to save this and do a save as and this one is now going to be three sinusoid.

[76:49] OK. So here what we're going to do is throw out this nice polynomial machine and do something much simpler which is a cosine and now we have, yeah you heard DC that's the sound of DC right there. All right. And now if I turn the index on [noise] you get this kind of sound. And that sound should be strictly like frequency modulation.

[77:20] Because basically it's essentially the same thing as frequency modulation. All right, now cosine is an even function which implies that whatever I put in, oh sorry, what comes out will be an octave up because yeah, so there's the original [noise] and here's the cosine [noise] . It's not too higher because it has only even harmonics.

[77:45] If you want to change that you just do this. Why don't we add some number to that? And since we have about one minute I'm not going to. Oh, since we have one minute I'm going to really be fast and loose and just duplicate this one. Now we're going to add.

[78:07] I'll have time to explain this better next time but if I give you a quarter cycle through then it's the... actually its minus sign instead of cosine. And now I'm using an odd function so I'll get the odd minus. And in general between here and there we'll get mixtures of the two.

[78:39] So now we have something like this. [noise] All right which is more FMee than it was before. OK. That's something you know you well we all heard that sound. That's the old FM sound that we heard over and over again in the seventies. I'll go back to this example next time because I haven't really described this in enough detail to know what's going on. But now it's ten till [mumbles] .

MUS171_02_17

Man 1: [00:00:00] ...the final projects thing with the web page. There's a month to go, and this should be like a two week kind of a final project. The idea is not to have you working all quarter for it, which is part of why I've been stalling. So the way the rest of the course goes is there'll be one more assignment after... This is assignment seven. Oh, I'm sorry, assignment seven is due a week from Thursday, right. And then there'll be one more assignment after that.

Assignment seven is up on the web, and assignment eight will... well, let's see, I have to think some about that.

And then the final project, I came up with some possibilities, but basically the idea is come up with a patch and be prepared to come show your patch to the class. The patch could be one of a bunch of a different things. Ideas that I had were... I'm not sure this first one is a real good idea, but... make a thing that actually is a patch where you hit start and it plays a piece of music using synthesis techniques that you've learned.

I have to teach you one technique in order to make that possible, which is how to sequence. I've shown you how to do looping sequencers, but I haven't shown you the more general sequencing techniques we need. So that will be one thing that I shoe-horn in at some point maybe in the next week. I'm not sure.

The next one is kind of the obvious one. Make a nice chord for a drum machine slash sequencer. I went and dug one up just to entertain you with. This is good, this isn't real syllabus stuff anyway so I can be doing this now. Here's me making a drum sequencer.

[music]

Man 1: [00:01:54] This uses all sorts of synthesis techniques you don't know. So yours won't sound like this. Notice that it doesn't do the same thing every time through. The interesting thing, the thing that's fun about making these things is figuring out when to throw what probability in order to make it so [inaudible 0:02:13] . Because you can make these very, you can make these lame very easily. [laughs] In fact you might think this is lame. So. No comment.

[music]

Man 1: [00:02:29] That was actually me trying to design a synthesis technique, and the synthesis technique didn't go very far, but the drum machine sounded cool so [inaudible 0:02:40] , so that's... [laughs] I haven't listened to this in about five years. But, drum machines. OK. Even people like me make drum machines sometimes. So there's that.

Another example of something is do the Switched-On Bach, by which I mean go find some nice two-part invention on the web somewhere. Figure out how to get PD to play it. That might be the hard part. What I did, I had to do this once because someone put me up to learning a piece of classical music. And I don't read music, so I went and found it on the web so that I could learn it by ear. [laughs]

So, you can do this... Here. You'll see this in more detail when I start showing you how to sequence again. Here's a nice sequence. This is a text file that contains times, pitches, and cumulative times. You can make these files. You can get them on the web from various sites, and they're in various formats. And then you can teach PD how to read them. So I made a little sequence out of that.

I made a little polyphonic synth that could play that sequence. This really sounds horrible; this'll make you feel good because yours is going to sound better than this almost no matter what you do. Especially since it doesn't do anything! Let's get a pre-set... Ah, here it is.

[music]

Man 1: [00:04:24] That's an old loop piece from the 1500s that I just had to know for some really weird, stupid reason that I can't explain to you. Weird things happen when you work in music departments. [laughter]

Man 1: [00:04:36] All right, so this is the Switched-On Bach style project example where you just, you know, go make something that sounds like Walter Carlos. Except of course it doesn't sound like Walter Carlos, because no matter how hard you try you will not be able to sound like Walter Carlos. And you can indeed, the very best things to grab are those two-part inventions, because they're easy and they sound good. Those two-part inventions meaning the ones by Bach.

There's that. So those were just examples that I happened to have of times that I had to do these things. Just to show you that these are real projects that you might actually profit from being able to do. At least if you're in my line of work. So that was the: make a patch that plays the, blah blah blah...

Here's a cool one that will make you impress your friends. Make a nice little laptop instrument that you can take to a bar and get people to dance to. Laptop instruments I think, are things where... the things that you see when you go to entertainment venues where someone is staring into a laptop. And they see the screen and you don't, or if you do you see a whole bunch of nonsense that's their screen, that means nothing.

The basic deal is you have something where you can mouse and keyboard away, and that changes something that typically is based on a loop that's modified by parameters that you set. Moving sliders up the banner or whatever it might be.

That's a good thing to do. And that could be based on sampling or whatever. Yes. I did put up the following idea, because I haven't though it through well enough to know whether it's a good project or not, do a mash-up.

Fine tune pieces that are incompatible, tempos and keys, and make a looping patch that allows you to superimpose them and switch them around. That would be a good thing. But I'm not sure how to make that sound good and I'm not sure if you do it it will sound good, so I'm not sure that's a really good suggestion.

You don't know how to do this because I'm not going to tell you. Tom might. Think of something you weren't able to do in PD, whatever it might be, and write a C object to do it and make that a new PD object. I won't go into that except to say that there are thousands of these running around.

They're not hard to do and if you want to see some of them they're all over the PD source code and also there are hundreds of examples plus websites that tell you how to do this on the web. But don't do this unless you know how to code C pretty well because otherwise you will have to learn how to code C, which you cannot really do in four weeks.

Make a melodyne. This is fun. It is not easy to do because, well, the hardness is not because it's inherently hard to do. It's actually stupidly easy, but you have to have a pitch shifter handy and I haven't told you how to make a pitch shifter, and you also have to figure out what the pitch is of the thing that's going in so that you can figure out how much you have to correct it.

Then you have to make an algorithm that, given any pitch, figures out what the nearest pitch is that it could have been supposed to be. So what you do is you say the allowed pitches are the C maker scale. Alright. And the current pitch that I'm getting is, who knows, 110 hertz. Well, that's a real pitch, sorry. 120 hertz. OK.

What's the nearest white key to that? Figure that out and then make a pitch shifter to change the pitch by that amount then you can talk into it and it will sing, well sort of sing, it'll talk back at you in the C major scale. Then you'll have a melodyne and you can sell it. Actually, probably not because I think there are probably hundreds of them on the web now.

Melodyne, by the way, is a trade name, but people use it generically just to mean something that whacks your pitch into something it's not. Then you can go from there. You can go wild from there because you can add a keyboard and make it allow you to play things. All sorts of good stuff you can do once you have this.

But the two things I haven't told you is how to find what the pitch of the sound is and then how to change the pitch of the sound. Both of those are techniques that I will try to squeeze into the rest of the quarter, but I don't know if I will be able to. Think of something else. Oh, you know what? Something else.

Go buy one of those Arduino hoo-has and make a little physical device that has buttons or knobs that allows you to give yourself a physical synthesizer control interface.

Man 2: [00:09:33] [inaudible 09:33] .

Man 1: [00:09:35] Oh, the usual one is called Arduino. What I do is I go to sparkphone.com and you can buy processors for like, it depends, but $20-ish and you can build little circuits out of them that are anything that you want. You can make robots or sensor arrays or whatnot. And that really should be the subject of a different course. And, also, you shouldn't do that unless you're able to deal with voltages and solder and things like that. So don't do that unless you think you know what it entails. Yeah.

Anyway, that would be another thing that would be perfectly cool, but I predict that most of you are going to want to make a drum machine or an interactive, playable laptop instrument or something like that and that's just cool because, actually, 30 different drum machines are going to have 30 different personalities. It'll be fun.

Questions about this? This is all up on the web now. Well, what this is so far is up on the web and if you have questions about it ask me and probably that means I should put something else up that says something more about what's going on. OK. Alright. So that's the final project. It's a presentation. We are now still doing modulation, continued wave shaping.

Wave shaping and not wave modelling yet, although I might get into that today depending on timing because what I want to do is make sure everyone is on the same page just about the wave shaping thing first and then we'll proceed from there time permitting.

The other sort of organizational thing that I'm succeeding in not forgetting to say is the following. I have two trips coming up and there will be a substitute teacher the next two Tuesdays.

Cooper Baker who is a graduate student who is also an expert PD programmer and computer musician and circuit builder and many other things who has a lot of good things to say and show will talk, I believe, next Tuesday about frequency modulation, assuming the syllabus works.

Then depending how it all works we'll be talking about delays the Tuesday after that and I will try to make it ducktail with the syllabus as closely as I can. I'm still not sure how that's going to work with the taping scheme. Also, taping. All of these classes are taped thanks to Joe and you can get them, so don't forget if you need to review that's one possible way of doing it, which might be useful.

OK. Right. Questions about all that before I just jump in and patch away? Or things I forgot to say? OK. So now what I'm going to do is I showed you a sort of mathematical way of thinking of all this stuff last time, but this time I propose simply to go straight in and deal with it at the level of patch because just on the theory that if you change your teaching style every week it either makes everything total coherent or else it makes everything totally incoherent. So this is where we go at the end of the last class where, let's see, I'm going to give us 110 hertz which is a low A.

And then we were taking that signal and messing with it's amplitude and then reading the cosine of it and that was giving us this kind of signal. Let's see. What' I'm going to do is give us both channels. I don't know why. Turn this on then turn this on. Yes. OK. So in goes sinusoid, out comes signal like that and the good thing about this signals is it changes tamper when you change the amplitude. There are two things happening here that I mentioned last time.

One is, yeah, non-linear things like cosine don't react to doubling the input by doubling the output. They react to doubling the input by changing the output and that's maybe most easily described as a change in wave form. The other thing I wanted to do was graph this so that you can see some sort of representation of what this is. Woah! Oh yeah. I'm listening to that and I'm graphing that. OK. Here it is. OK.

This might remind you of what happened when I used the exponential look-up last time which is if you give it an index of nothing you don't hear anything because we're multiplying it by zero, this adder's adding zero too, I haven't got there yet, and now we're taking the cosine of zero, which is one, so the thing is just sitting at the top of the table being constant one.

And then as we push this up we start getting sound and the thing starts acting like a pulse train except it's not because instead of reading at exponential we're reading at cosine so that after a certain point, well, so this is the most pulse training we get, but notice that it doesn't sit at zero it does the next thing, which is rising again.

So now the wave form that we've got is we're making a sinusoid and we're reading the cosine wave, but we're reading not only the first lobe of it, if you'd like, which is to say it goes from minus one up to one, that's at zero, and back down to minus one, but we're making it go past that amplitude a half thing and therefore it's giving us the next wiggle, if you like. Oh, I should put this in a measurement so you can see it.

Well, maybe I shouldn't. I will. Let's not do it to crazily fast. And now I need a nice toggle. So, by the way, the reason this is changing, of course, is because the cosine is at a different phase each time the metronome goes off. Meanwhile, as I change the amplitude of the cosine the wave form changes because it's reading more and more cycles of the cosine wave. Actually, I should turn the frequency down so you can see it more clearly.

So now we have nothing turning into something looking kind of sinusoid with a knot and then turning into progressively higher and higher frequencies. And you can almost tell just by looking at this that this should have high frequencies in it, although I don't know a simple theorem that says that the more a thing wiggles up and down the more high frequencies it has.

Even so, it's clearly true that if you make something like this it's going to have some high frequencies that it didn't have when it looked more like this. Whoops. What did I do wrong? I didn't hit the return. Like that. All right. So that's the amplitude to timbre change. The good thing about the cosine as a wave shape is that no matter what you do to this index it gives you out roughly the same power.

So even if you ask it to make some ridiculous index, all right, how about 10,000. All right. Well, OK. But we can tell that it was basically, I mean, it ranges from minus one to one and, you know, the power of that thing is going to be roughly the same as the power of the original cosine wave. That's a good thing for building computer music instruments because it means that you can put it into your amplifier and expect decent results to come out.

The previous example, which was the exponential, sort of did that except that, actually, let me get it out and show it to you. Let's see. I'm going to save this and close it because I haven't changed the name of the table. I changed the name of the table. Let's see. This is 17. Alright. Do that. Now we can actually open the other one and they won't fight. 15. There it was. OK. This one now, oh, wrong. Oh, I'm sorry.

I'm looking in the wrong place. Where I wanted to look was in the help. We were doing this and then we were looking at E exponential. Sorry. This is the real one. Now we have this situation where we can listen to the sound given an index.

Now it makes wave forms like this, but the trick is here if you start giving it extreme values here, which I didn't allow you to type because I didn't want to reveal this right away. 2,000. At some point you're going to notice that this thing doesn't have a whole lot of power and it's going to start fading out. So 20,000. Now it's getting quieter. OK.

So this is not terribly well behaved from the point of view of the power of the signal, although it's not too bad statistical for the usual reasons. OK. Using the cosine is much more gentle to your audio hardware because it pretty much gives you the same audio power no matter what you do to it except in extreme situations like almost zero. OK.

So the next thing about this is this. Let's see. Let's go back down to something reasonable and let's listen to it. So we can also ask for the cosine or for sine by moving by 25 percent down the wave form and then we have this change and then we have this stuff in-between, which is changing from even to odd harmonics.

That could even be a useful thing to listen to. OK. So that's exploiting the fact that ht cosine is even, the sine is odd and if you just graph the cosine and changing the place that you look at, just moving the origin, you can move continuously between being an even and odd function.

That's a good thing. That's a property that the exponential wouldn't have.

As a sort of sneak preview, OK. So sneak preview, this will last about five minutes. How to think about frequency modulation in these same terms. Alright. Let me just do a save here, or save as because I don't have to ruin my patch.

So what I'll do is I'll make two exactly equivalent frequency modulation patches, one the correct way to do it and the other the way that allows you to actually think about what it is and what it does. So frequency modulation is this. First I'll do the classical one. Give yourself an oscillator. Oh, actually, let's do the whole thing. Let's see.

I'm going to get rid of this. This is now an amplitude controlled oscillator here. So I'm going to make myself a copy of that. Let's see. Does this work? I'll give a frequency then an amplitude and amplitude. Sound. Now, next idea is what if we took this and took another one and used this one to control or to mess with the phase of the other one? How would you do that?

So we need another oscillator whose phase we can mess with, so we can't actually do that by just doing the simplest form oscillator with OSC tilde. We have to split that into cosine and phase. So we'll do that. So now this one's going to be an oscillator alright, but it's going to, oh dear. I don't know where to put this.

I really need to put it on the other side. OK. This one I can leave alone and this one I'm going to make the cosine, phaser and the cosine separate. So there's a cosine and then we're going to say phaser. Alright. And this now is just another oscillator. It looks like we can't hear it. Alright. Still can't hear it. Why not?

Am I doing something wrong? OK. Alright. Now what we're going to do is take this oscillator and use it to mess up the phase of this oscillator by adding it to the phase. I should put a plus tilde to be explicit about it, but I'm going to be lazy and not do that. Then I'm going to move this up here so you can see what's going on.

Alright. That's as well as I'm going to be able to make it. So now we'll listen to this one. This one, to start with, I'm just going to make it run at six hertz then I'm going to do that to it. Oh, let's make this higher.

So now what we're doing, this is not what I showed you in the first couple of weeks of class when I showed you here's an oscillator and here's another oscillator changing the frequency of that one.

Here, instead, I've taken this oscillator and split it up into a phaser and cosine so I can add this other oscillator not to the frequency, but the phase. Why? Because that's the way that people do it. You know, there are two reasons to do this. Well, three. Yeah?

Man 2: [00:26:12] Couldn't you accomplish the same thing by adding it into the second inlet of the oscillator?

Man 1: [00:26:19] No because the second inlet of the oscillator takes messages to set the phase, but then the phase starts taking off from there, so the signal itself doesn't add in. It won't take a signal itself. So you have to explicitly do it this way. OK. So why don't you just have an oscillator here instead of separating it and then just changing the frequency? The original reason, I think, was because this was once implement in fixed point hardware and it turns out that if you put the thing up here you have to put it in units of frequency, so you have to have the values in hundreds or thousands and that wasn't a good kind of an amplitude to give this oscillator if you were working with hardware whose maximum amplitude might be one.

So people figured out that by changing the phase directly instead of changing the frequency you got to give this thing a much, much smaller amplitude for the same amount of modulation. So this is almost equivalent to putting this thing in here, except you have to give it much smaller amplitudes.

Here I have an amplitude of 10 and it's already giving me plenty of modulation, much more so than if I had taken that and put it here. I'm going to do that now. Wait a minute. And, again, the more you give it the deeper it gets and then if I change this oscillator to an audible frequency then we get this kind of stuff.

This is what people sell you when they sell you a frequency modulation instrument, although to be completely pedadic about it, this is a phase modulation instrument. Another reason for doing phase modulation instead of frequency modulation is it's better behaved if you're not using sinusoid but other wave forms or if you are using more than one modulator or more than one carrier.

In other words, if you're making more than two oscillators, if you're making a more complicated network with five or six oscillators talking to each other for various technical reasons it's much better to operate on phase than on frequency as well. Alright. So this is the incorrectly called frequency modulator, which is a phase modulation instrument.

I think this is mostly what people will talk to you about next Tuesday, but by smartly choosing these two frequencies and this so called index of modulation you can make all these wonderful sounds. And you're not limited to only two of these. You can make bunches more and then you can be smart about how you design the sound.

There's a book about that that came out in the '70s or '80s so you can find out lots of stuff about just frequency modulation. Well, not to go too far down that route. Why am I talking about this now? Because this is equivalent to one of these, so you can think about that by thinking about wave shaping if you're careful.

So the careful way of thinking about this as wave shaping is the following: this is the cosine of the sum of two signals, one is a phaser and the other is a sinusoid, OK, but the cosine of the sum of two things is equal to the usual formula, cosine (A + B) is (cosine A)(cosine B) - (sin A)(sin B). So we could rewrite this network.

And I'll only do the cosine half, I won't do the sine half just to save sanity. OK. I need a new window anymore. So make new window. Oh wait. You know what? This is a new window. I can just erase this. Another way of thinking about this is we'll take the cosine of this and we'll take the cosine of that and we'll multiply them.

Let's get rid of this because we don't need it anymore. So here's taking the cosine of the oscillator. That's taking the cosine of this side. Now we'll take the cosine of the phaser. That's one of these. Then we'll multiply. So I'll cut these off and then I'll just say times. Total. And here, of course, there's another simplification we can make.

Phaser and cosine, we're not sticking anything extra here, so this actually could just be an oscillator now. So this is equivalent to just saying OSC tilde. And now look what we've got. We've got exactly that wave shaping instrument that I told you about before, which is take an oscillator and change its amplitude and take the cosine of tit and then multiply that by this oscillator. And you've seen that before, that's ring modulation.

So frequency modulation is equivalent to two networks like this because the other one would have to use sine instead of cosine, but basically it's an oscillator, take the cosine, that does this, and then multiply it by some other cosine and then you get these sounds.

Those sounds are pretty much the same kind of sounds as these sounds, although I'm pulling a fast one on your because I'm not really checking that they're exactly the same.

But, morally speaking, that's about the same deal. So frequency modulation you can understand by understanding wave shaping and ring modulation and, again, the word modulation just means change and computer musicians use it to mean all sorts of things.

So don't consider ring modulation and frequency modulation as being in any way related except for the fact that they both use the word modulation for artificial reasons. Ring modulation is this multiplication thing. Ring modulation is linear, but it is not time variant, so it is able to make new frequencies out of old frequencies.

That will get explained in more detail later, I think. Then this is running this oscillator with an amplitude control through a non-linear function and this amplitude control corresponds exactly to this thing, which is called the index of modulation over in this thing, which is where we're doing phase modulation. So now modulation means ring, frequency, phase, and then something else. Questions about this?

Man 2: [00:33:45] I have a question. So like with a DX7 uses this phasing functionality problem.

Man 1: [00:33:53] Yeah. Yeah. Except they have six so called operators instead of two. I don't know why they call them operators. They're just oscillators. Yeah. Then you get to make various circuitry of the six. There's some circuitry that allows you to feed out this signature. But, yes, basically this is what's happening inside a DX7 for those of you who know what a DX7 is. They came out in '84 maybe. Yeah. Now it's called a cell phone. OK. So this is the relationship between wave shaping and frequency modulation, however, it is a special case of wave shaping because it's this cosine function here. If you threw any other function in besides cosine you would no longer have this identity that the sum of it means multiplying two of them.

In other words, this trigonometric identity would work out differently if it were not cosine, but some other function then this wouldn't work. So this kind of wave shaping followed by modulation is a much more general way of doing things, in some respects, than frequency modulation is.

In fact, I guess this is the moment to say this, what would happen if you just made this thing be exponential instead? So what hat means is we go back to our patch. I threw it out. OK so I'm going to save this. What I'm going to do is go back and get help and get that exponential patch again, the exponential wave shaping patch, and do the same thing to it to see what we get.

Whoops, sorry. Go back here. I don't want to build it because I don't want to go through the hassle of making this function. I'm just being stupid because I could just use exp tilde couldn't I? OK. Let me be smart now. I'm going to get rid of this. Oh, wait, I want all this graphing stuff. I'm going to keep this because I want to graph this for you.

So we're going to save this as then we're going to go back over. So this'll be three. Alright. So now what I'm going to do, OK, so reminder. What this path does is this kind of sound and this kind of wave form.

So there's the wave form up there. It hates me because I've got too much stuff on the screen.

There's the wave form for you and here's the spectrum and to make it clear what's going to happen next I'm going to try to move this over so that you can see the whole spectrum pretty much. Alright. Now what I propose to do is to take this. Oh, by the way, the reason you hear these skips in the sound is because it's using CPU time to graph these tables, which is making my little machine hate itself.

If you want to have tables that are changing while you're computing stuff put the tables in a sub-window and close it so your machine doesn't have to graph it so you won't get these skips. But I'm being pedagogical, so I'm leaving everything out here on the main page where everyone can see it.

OK. So now I did the table lookup. Now I'm going to operate exactly as before, which is to say I'm going to multiply this by a nice sinusoid. So what that means is we'll say times. Oh, I made one. Well, get this one. Here's an oscillator, here's an amplitude thing. Oh, but I don't need this.

I just need a number and this is no longer index. This is now ring modulator. Oscillator times, now we look at it and listen to it. So nothing's different yet, but if I start changing this now I get classic ring modulation. Well, it's just going to be what it is. So each one of these peaks is split into two peaks because that's what ring modulation does to a spectrum.

This is the spectrum and this is the wave form and we're multiplying rather slowly. This thing is not really showing the wave form as it's changing because it's really going up and down 15 times a second, but if I make this higher then eventually you'll start seeing. Oh, drat. I want that thing to be loaded. There we go. OK. Let me see if I can get it exact.

Alright. So now what we have is a sinusoid which I tuned just by hearing a beat, the second harmonic of the pulse train, and now if I start pushing the pulse index up I get a hat shaped function, which is centered around this frequency and, in fact, I can make that whatever I want.

I can start this thing wherever I want and have this sort of hat shaped spectrum move to whatever location I want it. So now this is controlling band width and this is controlling center frequency of the spectrum. OK. Band width is a term you will hear a lot in computer music. It just means the width of the band, but bands in this case are ranges of frequency.

That, I think, is radio terminology originally like the FM band. So in this case what we're talking about is a band of frequencies like that. Why would that be a band? Never mind. The center of the band is being controlled by this ring oscillating modulator. This is what happens in DC. So if I send the oscillator to frequency zero the biggest peak is at DC, right?

This peak then got aliased out somewhere here and then these peaks got aliased out to DC plus n minus those frequencies. Or, to put it another way, there are negative frequencies in this thing that I'm not graphing and when we multiply it by a sinusoid which then moves it over, it actually moves it over like that, then we see the negative frequencies, lowest and positive ones, showing up, which is this.

Oh, we don't see any negative frequencies until this one bounces off. Then there's this sound. OK. Now, these are perfectly nice, harmonic sounds if I choose these things just right, like that number I found. OK. So this is going to put out a pitch, which is controlled by this fundamental. I don't know when this fundamental's being computed, so I can't use it.

This fundamental must be something about half this, like 170 something, but I think it was computed using load bang, so I don't think if I use it now it's going to give me a new value. OK. What else can I say about this? What does it look like in the tan domain? I guess what I should do is let me stop modulating it. OK.

So there's our nice pulse train and I'll skinny it up so that the pulse is decently small like that. Then when I start multiplying it by an oscillator, just to see what happens I'll give it a very high frequency here. 5,000. Now, what we see is that there is a pulse train.

Every time this thing makes a pulse, which it does every cycle, you get a pulse times this oscillator and I asked this oscillator to go very, very fast so that you can see it, but what's happening here is just a bunch of pulses, one after the other. Now the center frequency's way off the screen here so you can't see the spectrum anymore. But you can sort of guess what this thing should sound like.

Its period should be from here to here. It's the smallest interval at which you can see repetition, so that period's being controlled by this fundamental frequency, as long as this one's a multiple of it, which it's not so I'm pulling a slight fast one here. Meanwhile, if you think about what frequencies are present in this as a signal they're mostly high.

They're mostly these frequencies here and that agrees with the general observation about ring modulation, which is if you ring modulate by a very high frequency it takes whatever you've got and slides it to where you can sort of see it as a clump of frequencies around the modulated frequency.

So another observation about this is that, questions about this? Is everybody completely confused now? There's nothing complicated about this patch, but the complexity is all in how you analyze what it does and that is typical, unfortunately, of electronic music, which is putting three or for modules together that quickly leads to a situation where it takes hours or days to explain what the thing is even doing.

That's just what it is. And unfortunately you have to go through the explanation because I don't know any other way to be able to design things with a summation of what they might do. All right.

Man 2: [00:45:01] [inaudible 0:45:01] .

Man 1: [00:45:03] Oh gosh! If you really want to. Sounds like something out of poem e electronic. Most of these are enharmonic, but every once in a while I'll hit a multiple disc and get a harmonic sound, but you can't even really tell the difference. Well, there are theories of perception that say, I don't know if they're true, that basically the first ten harmonics are the things that your ears will use to try to determine the pitch of the thing and then after that you'll hear that there's energy there, but you won't use the pitches of the harmonics to tell you what the pitch of the sound is. So here, don't do this, but I could take this thing and add it to the modulated sound and now you hear a nice sound with a fundamental and some nice high harmonics.

And these nice harmonics they don't have anything to do with the sound, they're just whatever frequencies they are, but your ear can't tell that so it just accepts it. All right. Don't tell anyone I told you that.

Oh, actually, it's not a useful a fact to know as all that because, in fact, what you'd really like to do is make things that can change controllably between stuff that has low frequencies and not and, of course, once I change this thing so that low frequencies come in you're going to hear the fact that they're mistuned and then you're not going to believe that that's a harmonic tone.

So you would have to work harder if you want to make a general instrument that allows you to do this general thing. And I want to show you how to do that, but maybe not right now because, why? Because there are two ways of doing it and they're equally important and I don't know how to fit both of them into one half of a class. So what I want to do is show you things that will be useful for doing final projects. OK.

So there's stuff that doesn't fit in the syllabus that's just lore about how to use PD, so what I want to do now for the rest of this class is show you PD board that is of use for just building stuff. The most important thing that I haven't told you how to do yet is sequencing. Well, I've told you how to do sequences, both of which are table based and the thing that you do is make a counter and you make the counter count through the table.

It can either be a phaser reading a table as a signal or it can be a metronome driving something that increments. That's a way of making a 1960's kind of sequencer and it's appropriate for driving monophonic synthesizers and then for polyphonic stuff, OK, there's a certain place you can go with it, but it's not going to do like general polyphonic sequencing for you. So how would you make something that's actually capable of polyphonic sequencing?

There are many ways, but I'll show you one that is just the second most general. The third most general. Something that's about the right level of complexity to get most people's needs, but without having to spend hours and days learning how to do it. OK. So here it is. What I'm going to do is make a new window and give it a name. It's going to be for sequencer.

So the object that does sequencing, wrong button, the object that does sequencing in the most general form that I want to deal with right now is called queue list. This is probably a misuse of the word queue list. Let's make it have a decent format. But we can call it that anyway. What a queue list is is a bunch of messages in PD language that can have, I have to tell you some things I haven't told you.

First off, message boxes. Messages boxes and receives. I've shown you how to do this so far. I've shown you how to do send name1, send name2. Then we'll have receives. Oh, S and R are short for send and receive and now I'm just going to put numbers here so that you can see that this is a way of making a non-local connection. All right. So my reason for showing you this is so that I can now show you the following very strange thing.

Let's get a message. First off, let's make it just have a number in it and do this. Oh wait, that's the same number. OK. That's all good. In fact, I showed you another thing which is that you can have commas and here, if you do that, you will send those three messages and you'll just see the last value, even though the number box actually obtained all three of those values all in a zero period of time. OK. There are, you see one other syntactic element of message boxes, which is the dollar sign.

That's the thing that allows you to have an incoming number that changes the message. There's one other syntactic thing available for message boxes and that is that you can have messages separated by semicolon and by convention I put a carriage return in here. So now if we do this, what we're doing is we're saying the message if 56 and then there's another which is 67, but that message is going to be sent to the object named name two or the objects named name 2.

All of them if there are more than one. So comma means begin a new message. Semicolon means begin a new message and, by the way, this message is not going to go to this outlet at all. It's going to go this other object. Yeah?

Man 3: [00:51:47] So if we didn't have name2 there would it go to name1?

Man 1: [00:51:54] No. It would try to find an object named 67 and that's not a legal name for an object because it's a number and so then I should see an error message. I hope I get an error message. It just says float no such object. Oh, that's horrible. Anyway, yeah. So there's no such object as 67. It's not letting me do that. So you have to give it a name of a destination. OK. Next thing. We don't really even have to use the first one. We can just say no message at all, thanks, but name1 gets a message 67 and name2 gets a message 34 and then when we whack that those two messages go out. Now, this is starting to look useful I hope. Now we can do this kind of stuff. So it's almost a preset mechanism. Not quite. Yeah?

Man 4: [00:52:53] Do you need to have a semicolon in front of it like can you put name1 123 semicolon then name2 221 and that work?

Man 1: [00:53:02] Yeah. You have to have the semicolon otherwise there will be a message name1 123 and that will come out this outlet, so the first semicolon means the name of a receiver follows and the receiver is this. So this is really a strange, ugly syntax. But it's what it is. It's consistent. It's logical even though it looks weird. OK. Now I showed you that so I could show you this. This is cool and this will allow you to have any number of parameters in a patch. Right. So I could now make an FM instrument and give the carrier frequency, well, I'm using names I haven't described. You know, the frequency of the two oscillators, those could have names, and then the index of modulation could have a name, the amplitude could have a name. I could make it six operator and I could have, I don't know, 12 names and that would all be cool.

And then I can put them all in this one message box and just whack the message box once and all those values would go off to all the right things. I probably should have told you this before. You know, this becomes almost inescapably important once your patch reaches a certain level of complexity. While we're here, it's always good when you have a patch more than a certain amount of complexity to have a button which is just reset.

So you probably have already had the experience of starting a patch p and not have it doing the thing it was doing when you last closed it. This is your friend for being able to get things to go back to starters that you know about and it's often worthwhile having one of these things hooked up to a load bang so that every time your patch opens up the values are as you wish them to be when the patch loads up. All right.

So this is not computer music knowledge. This is just PD board and when you change to some other programming language this will be different. Now, about queue lists. OK. So this allows you to do everything you could possibly want except for sequencing and now if you wanted to do sequencing with this I could tell you how to do it using delays. Well, you already know.

You just make a whole bunch of message boxes separated by delays. Yeah. And, in fact, let me show you the first ever Max patch, which I've imported into PD which does exactly that. I'm just doing this to horrify you. So we're going to go back to here. Repertory because this is going to be public. We're going to go look at Pluton. This is on the web, by the way. If you download the PD repertory project you can look at all these scary patches.

This is going to be Manoury Pluton and then we're going to have a patch called Pluton.PD. This is a 45 minute long piece of music. Maybe 48. And here's, I'll get one at random, here's section 31. It has a queue, which is a number, and then it has sub-patches and this sub-patch has events number one, two, three, and four in it.

So we'll get an inlet and we'll select one, two, three, and four and each one of them is going to have message boxes sending parameters to values. You can make message boxes like this. You will tear your hair out after a certain amount of time keeping track of when you changed what because obviously this could lead to horrible messages. If you go looking in the right place, this is not a good example, but if I find a good example you'll find delays in here.

Nope. Nope. OK. 21. Here we go. Oh, bad, bad, bad. Wrong section. Two. Yeah. Here we go. So why don't we, on event five, do all of this good stuff, but after a delay of five seconds we'll start at the other sequencer, whatever that is. OK. So now we have the ability to wait until event number five comes in, never mind his figuring out what queue we're on, but you can think there might be a queue number five, so there's an incrementer in there somewhere.

And event number five means do this, I shouldn't do that, and then after five seconds do this. But what if someone did event number six before those five seconds had lapsed? Then you would have event five then part of event six then part of event five happening. What if this thing started something that this thing was supposed to stop? Then instead of starting and stopping it would stop and start and then you would have the thing playing for the rest of the piece, which you didn't want, right?

So you'd better stop this delay when the next puppy comes in or else it's going to go off. And then it gets worse. OK. So this is making sequencers using message boxes. You can do it, but not pleasant. Also, you will not be able to download a text file from the Internet and have it be that, right? OK. So that was Pluton. I'm not going to show you more about that just now. So there's a better way, which is to make a queue list.

A queue list is read from a text file, I should say, and the text file has a bunch of messages in it that are separated by numbers that are event numbers, or actually that are times I should say. So queue list acts like this. First off, you have to be able to read files in. So I'm going to make a message box that says read sequence one and I'm going to give it an extent.txt because on some kinds of machines if your file isn't named something.txt it doesn't know it's a text file.

Now I'm going to make a text file named sequence1. You all have Macintoshes and you will get out the text editor and it will not make a text file by default. It'll make a rich text file. You've got to make a real text file and there's something in the text editor on the mac that lets you do this, but I've forgotten what it is and it takes some finding. OK. I don't need this anymore. This thing is just an orphan, so let's get rid of it. OK. Now what I'm going to do is make a nice file.

Are we in the right directory? Yes. I'll net it. OK. Text editor. Now I'm going to say message one. OK. So name1 45, name2 67 and then let's wait a second and then let's do name1 back to zero just to make it look like it's turning off. OK? Oops, I don't want that space there. And notice I'm putting semicolons after every line because semicolon is the delimiter here just like in message boxes and if you forget then it will not do the right thing for you.

Did I say sequence or sequence one? I thought I said sequence one. OK. Then let's go back and get the patch. I've got all this junk open that I don't want. Yeah, of course I want to save. Yeah. Now we say read and meanwhile let's just check, make sure we didn't get any errors. We didn't get any errors, good. Oh, and of course if I had said the wrong thing here then I would get some horrible, come here. Then it would say message file.

Yeah, right. OK. So this is good. We believe that this is working right. OK. Now what can we do? We can just say bang and then it says boing, right, and there's the sequence. So do it again. Idiot's delight now. OK so name1 and name2 are the receives and now we hit the queue list and there's the first message and there's the second and those are the two messages or the two pairs of messages I've put in this file.

Where's the file now? I think I must have closed it. There. So that is the easiest way to sequence stuff in PD. And you can combine this with, oh, these can be any kind of messages you want, so you can combine this with line tilde objects to do things that ramp or whatnot. I mean, anything you can do with messages. You can connect things that have messages with bunches of arguments and then use unpack to get them out.

Yeah. Which one would do a lot of, actually ,when you really are doing this. Yeah. What else should I say about this? Other messages queue list takes. I don't know if, well, go away. Print's a good message. When you tell the thing to print itself out on the PD window comes everything the things has, sorry about the back slashes. That's a good way of checking whether the thing is what you wanted to do. And finally, and now we are going to be living dangerously.

If you want to make a nice loop out of this you could do something like this. Let's go back. Where's the text file again? Here. Let's say we want to do this and then we want it to loop. So I'm going to say for another second, by the way I could either have the semicolon or not here for technical reasons, but I'm going to put it just to be simple. And then I'm going to say restart bang. Semicolon just to be complete. Oh, this is not a really good idea is it? I'm not going to ever be able to stop this.

Man 2: Is there a stop message?

Man 1: There is actually. Yeah, so that'll work. So we'll do this. So stop. OK. Let's just check that. So I should be able to say bing, bing, and then the third thing is going to be an error message because there's nobody named restart. Oh, I didn't get an error. Oh, I didn't save. You know, what this is the sort of thing that happens. You've got to save this then you've got to tell it to read the sequence and then you can tell it to do this stuff. Bing, bing, and then error. There. Restart no such object. And now I can say receive restart and that's just going to bang the queue list, which I'll make it flash by hooking through the button. I hope this works. Yeah. All right. So now you have another way of making the step sequencer if you wanted to. There's one more thing about this, one more message that you might like, which is that you can set the tempo.

Let's see. Let's do it this way. Dollar one, recall, is in a message box and the message box context dollar one is just take the value and stick it in the message. So here if I say one it's going to be tempo one, which should be the original tempo, and if I double it it does that for me. And then stop doesn't stop it because it doesn't have a method for stop. OK. I think it's called rewind, actually. Good. Why is it called rewind?

Because there's more that you can do that I'm not telling you about. You can, if you want to, single step through the messages instead of having it sequence through them. So to do that you send it rewind and then next and then you can control your own timing instead of using queue lists' own timing and that would be useful if you wanted to make something that had random variations in the timing or some algorithmic way of controlling timing other than just the numbers in the queue list itself.

Given the fact that there's this tempo message here we could set the tempo to 1/1000 and then just have this thing be in beats like one beat instead of 1,000 milliseconds and that would work fine, although when I'm starting out I just always do it in milliseconds because it's easier to think about that, I think.

You don't have to agree with that. Oh, since everything else in PD's in milliseconds it might just be easier to have them be coherent as opposed to having them be different in the queue list from everywhere else. So that's the queue list object, which is key to making sequences. Yeah?

Man 2: [01:07:31] What did you call it, sending it messages?

Man 1: [01:07:37] Oh yeah. This is a very confusing thing. So what I'll do is I'll print these out and give it a couple of [inaudible 1:07:47] like that and then it makes messages and the messages are what you put in there except that dollar one has substituted for it the value of the first argument that went in. So this is the way that you can make messages that vary inside a single message box. And furthermore, if you have a packed message with several numbers in it they can be addressed as dollar1, dollar2, dollar3 and so on so you can have multi-dimensional variability. You haven't had to do this much because you haven't seen very many objects which are complicated enough where they take a bunch of messages like this.

For the most part, messages are always just numbers because usually simple objects only do one kind of thing, or at least one kind of thing per inlet, so numbers suffice as a message passing language, but an object like queue list it has a bunch of state, there are a bunch of things you might wish to ask it to do like rewind and go to the next thing and change its tempo and so on.

And then you need a bunch of different kinds of messages like this and then you need message boxes that can put together messages that have both words and number sin them, symbols and numbers, and have them still be variable. So I've been avoiding doing this for reasons of sanity, but that's there and ready to get used. Any questions about this? Yeah?

Man 3: [01:09:44] What are the outlets useful for?

Man 1: [01:09:47] The outlets are useful if you want to make your own sequencer. This one gets a bang whenever queue list finishes, if you want to know that. Actually, I could have made this loop in a different way by just doing that. This one doesn't get anything when queue list is being used as sequencer by itself, but if you single step it, if you say 'next,' it goes up to the next number and then outputs the number here. So, instead of having it interpret that number at time, you grab the number or numbers and interpret them to be whatever you want them to be. That's how you would make your own sequencer out of queue list, but I do something more manual.

And there's too much information in the help window for queue list that will tell you all this. The easy way to get confused with queue list is to change the sequence in this text file and then forget to tell it to reread the new sequence. Also, if you read the new sequence, I believe it will insist on rewinding itself, so it won't be able to continue playing the sequence if you change the sequence files.

With one exception, this is able to do all of your sequencing needs. The one exception is that all these names are global, name1 and name2 and so on. If you wanted to have a bunch of instances of Apache, which was using a queue list to control different variables that were local or dependent on the patch, then the queue list wouldn't suffice to do this. You would have to reach for a lower-level tool that allowed you to get those things and play with them yourself. There are ways of doing that.

In fact, if you get the queue list help window, it will send you on to the thing called text file, which gives you less automation and lets you build more general things. He asked what the queue list can do for you.

So, just to review where we're at, because we only have five minutes - that's not enough time to go start doing wave packets or whatever it's going to be next. Next time is going to be frequency modulation in from more of a 'how to do it' standpoint. How to make sounds out of it as opposed to simply with the theory is. I've tried to fit that into this tale about wave shaping, which is what the last couple of things have been about, and ring modulation.

Where we are in the book is now chapter five-ish/chapter six. Chapter six, I think, is what I'm going to squeeze down to a day or two, and that will be Thursday of next week if it's only one day. The topic there is going to be how to go back and use that combination of wave shaping and ring modulation in a way that would allow you to move the energy around from peak to peak without having this problem not having things be harmonic when they're between two multiples of the same frequency. That's an important thing to be able to do, and there are several techniques for doing it. I think I want to show you two of them, although I have to think very carefully about whether I can fit it in a reasonable amount of time. So, that would be next Thursday.

Meanwhile, you see, I think, what you need to be able to see in order to do things that have sequences. That's sort of where we are. And in book land, I've taken you pretty much through modulation because this thing about frequency and phase modulation is that rampage I went on about FM earlier.

Then I'm going to give short shrift to this [inaudible 1:13:30] designer, as I call it, designer's specter; that's just my own fanciful title. But that's going to be about how to make peaks in the energy spectrum independently of whether the sound that you're making is harmonic or enharmonic. That's what's coming up next. I've been looking in there, trying to make a plan as to how to do it but I haven't succeeded.

Then, starting the week after next is going to be time shifts and delays. That's going to be how you make the standard delay effects, but also how you design reverberators and also how you do delayed tricks like pitch shifting, phasing, chorus effects and what not. That will probably take a week. Then, by week 10, it will be time to look at Jim and that will probably be the rest of the quarter.

Questions about today again? Is this going completely too slow or completely too fast? Or is this a decent speed at which to be going over this material? Do you feel like you're actually understanding this?

[laughter]

Man 4: [01:14:43] I think it's a little bit confusing for me. I don't know about everybody else.

Man 1: [01:14:48] OK. Of course, it's hard to formulate questions when it's just too fast. [laughter] Although, that is the ideal way to slow me down is to think of a question. Actually, you can just ask a random question. [laughter]

Yeah, so the yoga is sinusoid control the amplitude nonlinear function, and then you get timbered variations. Then, the rest is just a lot of detail but detail is very hard to systematize. Then ring modulation takes things and shifts them around in the frequency. Those are the two basic building blocks.

MUS171_02_22

Speaker: [0:01] All right. So, last time we ended up doing sequencing. But before we showed the sequencer, there was talk about wave shaping and frequency modulation and, in particular, there was a patch that tried to show the equivalence of wave shaping and frequency modulation synthesis. [0:24] Actually, what I'll do is I'll get that patch out just to remind you that it was there. That would have been 217 FM. Here is the fixed-up patch with a smaller font size, but you can see that this is making the point that over here there is wave shaping and ring modulation which was making sounds. Do we have sound?

[0:51] I didn't check the sound yet. That was really smart of me. Sound. Carrier frequency.

[1:02] And that, hopefully, you're willing to believe sounds something like what happens when I do frequency modulation, which is this thing where you take an oscillator like this, and start changing its frequency, but then start changing its frequency fast enough to get this sort of thing.

[1:31] So that is the well-known sound of frequency modulation, and this patch was a description of why you could think about frequency modulation as wave shaping.

[1:40] And if you want to know what the spectrum of a frequency modulated sound is, that's to say if you want to know what the principle of the partials are, that it can make in what their frequencies are, and then you can analyze it by thinking about what this patch would do.

[1:53] That leads you into engineering mathematic checks to know vessel functions, which I don't want to tell you right now. So vessel functions aside, and in fact all of this aside, what I'll do this time is just go back to the basic frequency modulation patch, build it from scratch, and show you what it's good for, just in terms of sound making.

[2:11] The punch line is going to be that it's a thing which combines two oscillators. But, of course, you can combine 50 oscillators and all sorts of frequency modulating networks for each one, modulate the frequency of some other one, and find where you listen to some oscillator at the bottom. So frequency modulation is this very extensible, very complicatable thing. So I'm going to get rid of this and start from scratch. Just to be nice and pedagogical again.

[2:41] So let's see. We have an output device. I'm just going to start with the oscillator, right? So the oscillator is, it's got a frequency going, which is just a number. And then we're going to say that's going to be the frequency. Just to be pedantic, what I'm going to do is make the oscillator out of cosine sort of phaser.

[3:10] Phaser is the actual oscillator; it's the thing that remembers what the phase was previously and gives you the next phase every time that time moves forward. Whereas, cosine is the thing which is the wave form.

[3:23] It is actually done internally with a table lookup, so you can think of this as being generalizable by table lookup objects. Anyway, here is an oscillator divided up into the oscillator part and the wave table part. And it sounds like it always used to sound, and this is a good check to make sure our computer is still working. 440, OK.

[3:48] Now, we're going to take this and we're going to start messing with its frequency. And, again, just to be as pedagogical and as didactic as I possibly can, I'm going to do this in two different ways and make a claim about how it can be thought of as equivalent, if you ignore a couple of minor problems.

[4:10] So the first thing is, we're going to take this in order to take us ... oh you know what? Before I do that, even before I duplicate it. Of course anytime you have an oscillator, you're likely to want to control its amplitude, and I'm going to want to control the amplitudes of all these oscillators.

[4:26] So we're going to get a multiplier and a line tilde. Line tilde because I want to make everything to sound nice and the line tilde is going to get messages from thinking back up from the tree now so the messages are going to be packed.

[4:43] Some amount of time that I'm going to make the line tilde ramp at, and that's going to now be in control with a number box. Let me get this number box over here. So now what we have... sorry, this is all very repetitious... so now what we have is the 440 hertz tone.

[5:08] I can listen to it at my favorite listening level but I can also turn it on and off that way. OK. Now, we have an amplitude control oscillator, and now I'm going to make frequency modulation out of it in two different ways.

[5:21] So the first way, is going to be by doing the real frequency modulation thing, which is to say I'll take this oscillator, but the input of the oscillator is going to be another oscillator. So it's going to be something plus 440 so I need to now have an adder.

[6:03] And what I'm going to add to it is going to be a whole other oscillator. So I'm going to take this oscillator and make it nice and compact because we're going to have several of these up on the screen very soon, unfortunately.

[6:06] OK. So let's get another one of these puppies, put it up here. I didn't mean to do that scrolling thing. And now this one is going to be added to the frequency of the other oscillator, all right? So, this is going to be boring.

[6:21] So now we're going to say this one is going to talk to us in the control's amplitude. Oh and we need this 440, it's not in the plus yet. And now, this oscillator and I forgot, of course to make a nice number box to sit its frequency. This is the modulating oscillator, which now can get a frequency and an amplitude. And now I don't know if it's audible but that's now making vibrato, or with different parameters, it's making frequency modulation.

[6:56] OK? So that's the thing. And now you can think of this as being two different oscillators. If I put this close to this oscillator, so that you can see them as being essentially the same thing, this could have been plus 120, but I'm just not going to be that diductive.

[7:18] So this oscillator now, if you think of it as an oscillator, it has a constant and a variable part to its frequency. And it has a thing for controlling its amplitude, and then it multiplies and that's it, right?

[7:35] Oh yes, before I forget, I have to tell you one other bit of PD lore that I probably haven't said before, which is it would be natural if we tried to do this. We'll take this thing and disconnect it and just run the signal and the message into the phaser, expecting PD to add these things automatically.

[8:01] This will badly confuse PD because PD is seeing messages here and seeing a signal here. The phaser has to decide whether the signal is a signal or a message, and what it will do, in this case, is it will actually decide, "Gee, I'm a signal because there's a signal connected to me, and these messages are superfluous."

[8:20] If these messages were not superfluous -- that's to say I could just remember these messages -- then you could get them some awfully bad problems because someone could send a number five into this phaser and you could forget it and have this phaser around with a number five and hook stuff into it, and wonder why everything is five or it's off.

[8:38] So instead of allowing that to happen, when you have a signal and a message connection to the same thing, these message connections are simply ignored. So, now we have this rather embarrassing fact that this thing is now playing at 576 hertz if I do this.

[8:56] But if I connect this to it, it forgets the 576 hertz because that got overridden by the incoming signal. So signals override messages. And that is why if you want these two things to add, it's not OK or it just won't work just to put them in the same little inlet as if they were both signals. If they're both signals, they'll be added for you automatically. But if one of them is a signal and the other is control, PD will not know what to do, and so you have to add explicitly, like that, which is how I had it before.

[9:32] So, that's a detail about PD that can very easily be confusing so I wanted to be very overcautious about that. Now, the whole point of this exercise is to take this and to show you how to do it another way. And the reason for this is because the other way is the way it's actually done in practice, which is, you take this oscillator; but instead of having it in oscillator with extra input for frequency, you make it an oscillator with an extra input for phase.

[10:04] What that means is you have your phaser first, which is making phase, but meanwhile you can add your own phase to it. Actually, it's going to be better to work it this way.

[10:24] So now what we have is ... how do I make this... It seems unavoidable to add an extra object to make the oscillator uglier but I guess that's just what it is.

[10:38] Now what we have is an oscillator with two inputs once again. Well one of them is just a number box, but this is now controlling the amplitude, and this is controlling the frequency, all right? And in fact, if I just tell this 440 and this something reasonable, I should hear the sound again.

[10:55] Good. But, now the addition is not being added to the frequency, but it is being added to the phase, which is the stuff that's between the phaser and the cosine. And this is the point of splitting oscillator up into a phaser and a cosine 10 minutes ago, which was that I wanted to be able to get in there and add something in between the phaser and the cosine in which the OSC tilde doesn't have an inlet to do. And therefore, I have to rewrite it in this more elementary form or in terms of these more elementary objects. OK?

[11:24] Now, so this is an oscillator with a frequency control signal input, this is an oscillator with a phase control frequency input. And again, I can just do all the same stuff. This appears to be an oscillator without any controls at all. It's just being an oscillator. And now I can do this. Let's see, just to try and get equivalent results, let's turn this on. Let's see.

[11:53] So now we hear that one. You know what? I have to get a little louder.

[12:02] OK? Oscillator. And now I'll make this do the same thing. 0.3, oscillator.

[12:13] Oops, sorry. That's on. Let's turn that off. And we have the same thing. Now, we can make vibrato out of either of these two oscillators. This one, it's obvious how to do it. We'll take this thing and make it go six times per second, five times per second.

[12:28] Oh, sorry. Five times per second is the right one. And then there's a depth here. And for instance, just so that we can... all right, let's just do it by ear.

[12:40] So we'll do 10 hertz, which is a... No, five hertz, vibrato. Not totally unreasonable setting. Let's do that over here. OK. Can you make vibrato by adding a sinusoid to the phase of the thing, instead of adding a sinusoid to the frequency?

[12:59] The answer is, unless the answer changes, the answer is you can do exactly the same thing because what a phaser really does is integrate the input over time.

[13:15] What that means is that I'm putting signals in here. And if I put a signal here, for instance, a constant signal on my 440, what the phaser does is it fixes it so that its slope is proportional to 440 hertz.

[13:29] Or to put it in another way, what the phaser does, really is, at every point it simply adds its phase. It adds to its previous output, which is its phase, a phase increment which is proportional to the frequency.

[13:42] So what it's doing is adding in values of frequency sample by sample, accumulating them. And of course, there's a little detail that when it hits one, it wraps back around at zero and that's really from numerical accuracy more than any other thing.

[13:54] If we have infinite numerical accuracy, the figure could simply be a straight line going off into infinity. So, if a phaser's an integrator, integration is linear. So we're integrating 440 like a nice ramp but we also can integrate a sinusoid and integral of a sinusoid as you all learned in calculus is another sinusoid so that the indefinite integral of the cosine function is sine and indefinite integral of sine minus cosine. So either way integration just changes the phase and actually the amplitude.

[14:30] But why? Because depending on the frequency, there will be a different constant in there when you do the integration or differentiation. That's calculus; I'm not supposed to use calculus here. Why don't you forget I just said that?

[14:43] OK, so, anyway, what that is saying is that if I am adding, if I want to simulate adding a cosine to the frequency of the oscillator, I could do it by adding a cosine of a different phase and amplitude to the phase of the oscillator. In other words, I could add the thing here or I could add its incremental sum here. And in fact if you don't believe it, I'll play it for you and then you'll have to believe it.

[15:15] So now I'll make this thing be five hertz. It will be in the same frequency, but I'll have to give it a different value here. I don't know what it's going to be, yet, but it's going to be much smaller. Like that. So now I claim this signal - what's the right word -- is similar to this signal. Oops. Give me that signal.

[15:40] OK. In fact, we can even make them be the same by ear. Now this one had to have an amplitude of five because we're going to range from 440 plus five hertz down to 440 minus five hertz. This one had to have a much smaller amplitude because all we had to change this phase by was how much that five hertz could get you in the one-fifth of a second it takes this oscillator to cycle.

[16:16] That's hand waving, but in fact this has to be in the order of a fifth as big as this because this frequency is five. This number is actually going to be five over two pie (2Ï€). And now is it really true that one / 2Ï€ is about 0.13. Now we have to find this out because otherwise, we won't ever know. Free open-source mathematics package. 0.159, 0.16 roughly.

[17:05] So this number I claim, this number here is this number divided by five because the faster this thing goes, the less it accumulates; so, the faster this is going, the more you have to divide by.

[17:21] I'm arguing by proportion and not by actual equations, right? So what we have over here is going to be inversely proportional to the frequency. Proportional to this number because we're trying to get the same sound and I'm just going to tell you that the factor that you have to throw in is 2Ï€, which you will get out of calculus class if you go there. All right?

[17:42] So to try to see if this still holds, I'll try some other number here.

[17:52] I'm sorry. Yes, this is a frequency. Now 30, and now I'll choose some horrendous value here. Now, I'll see if I can get that same sound over here and see if it's still true.

[18:05] OK. So we're going to thirty, and that... oh my! That was too easy. Is that the same sound?

[18:20] No, it has to go up higher.

[18:27] All right? Does that sound similar? Maybe. So let's take this thing and divide it by 30 and divide it by 2Ï€ and see if we get that, right? 70 divided by 30 divided by 2Ï€. 0.37. 0.38.

[18:54] Ears are wonderful things. Your ears can do better mathematics than your eyes. Oh, yeah. That's actually true. I don't know how accurate your eyes are for seeing things spatially, or seeing colors, or seeing frequencies, but your ear can hear three sense difference of frequency which, let's see, three senses of 30th of an octave and you have roughly ten octave range of hearing. So that's a part of 3,000. That's the most accurate sense that you have in your body. That's not bad. OK. It's fast, too. It's faster than vision.

[19:29] So anyway, let's go back to where we are here. So, what I'm claiming, although I'm just giving this to your ears, I'm not giving the mathematics out, is that we can change the frequency or we can change the phase; and as long as it is true that the thing that we're modulating by happens to be a sinusoid...

[19:50] Why does it have to be a sinusoid for this to be true is because I made this hand waving argument about you have to put the integral of this thing and here they get the same affect.

[19:58] In other words, whatever you have here, you have to accumulate it here. And it turns out that if you accumulate a sinusoid by adding up values cumulatively, you get another sinusoid and that's a wonderful property of sinusoids that, in this case, makes it possible for us to rearrange this thing from this form to this form. But that only works for sinusoids. It does not work for the waveforms.

[20:23] For other waveforms it turns out that this is a better thing to compute than this. And I don't know how to explain why very well. But this is more likely to give you what you want than this. I'm not going to try to explain that.

[20:49] Too complicated to get into. All right. Anyway, here's another good thing about this form. I've probably already let the cat out of the bag. This is the way it's always done in hardware. Why? For the unobvious reason and the very interesting reason that good values of amplitude of oscillators that you use to modulate are the same in the same range as good values for listening to this stuff.

[21:25] In other words, if you take these numbers that you have to choose in order to make this thing sound right are much larger than these numbers that you have to choose to make this thing sound right, the amplitudes have to be down here, but the widths of frequency deviation have to be on the order of the frequency itself to have a reasonable effect, which is again the same why you have to divide this thing by this thing to get how strong and effective it is.

[21:53] Here, the proportionality to the frequency is already built-in. So, for instance here, what I could say is, this is the same thing as deviating this thing not by 30, not by 0.38 hertz. This is deviating this frequency by 70 hertz. This is going from 440 plus 70 down to 440 minus 70.

[22:16] This one is going from 440 to what? Well, we heard the same thing. So in fact, it's going approximately from 440 plus 70 to 440 minus 70. But, this is a better way of saying it. It's going 440 plus or minus 0.38; 38 percent of itself divided by 2Ï€. So forgetting the 2Ï€ because 2Ï€ is close to one, this is the proportional depth of frequency modulation, whereas this is the absolute depth, and the proportion depth is a better unit to be talking about frequency modulation in.

[laughs]

Speaker: [23:09] Stony dead silence. Exactly what we want. Every professor wants stony dead silence when they talk. [23:18] Now, with that as an excuse, now I can actually take the entire left-hand side of the patch and erase it and do something else instead, which is to take the right hand side of the patch and populate it with other stuff.

Before I do that, I'm going to do something else, which is this: [23:35] I'm going to show you the spectrum of this again. This is going to be another proof that these things are sort of similar. In fact, this is going to be a test of whether our ears or our eyes are better at finding the similarity.

So what I'm going to do is find sort of do a little data set mock it again. This is a... oops sorry. We have to go back to ignore the fact that you saw the future and that opened [inaudible 24: [23:53] 08] up. Live spectrum.PD, here we are. OK, this is a nice patch which I developed for totally different reasons and which I do not want to explain. This patch, which you can get if you download the patches for the day, but you do the good stuff like "Hi I'm a spectrum and..."

[24:31] Here is now ... OK I will not dwell on this because I will end up talking for hours about the wonderful properties of spectral voices, but what you see here... Anyhow, I'm not going to not be able to stop myself from doing this.

[24:50] There is a spectrum. This is the like the spectral analyzer I had out for talking about wave shaping a couple of sessions ago, but this is a real one which doesn't care that I use exactly frequencies which are aligned to the filter bank that I used to measure the spectrum. Never mind what all that was.

[25:10] This is a general spectrum estimator in which peaks just look like peaks that can move up and down continuously without getting messed up. OK?

[25:25] Good enough. All right, I'll stop it now. What's really happening is that every twentieth of a second, this thing is making a new nice picture and showing us a new spectrum. OK?

[25:34] Now the reason I'm holding this out is not to show off my voice, the spectrum of my voice too much as it is to show off the spectrum of frequency modulations. So what I'm going to do is I'm going to take this patch and use a wonderful feature of PD -which is you that can send and receive signals from one patch to another -to make it talk to this patch or actually listen to this patch. Send. Spectgraph.

[26:12] And I'm doing this so that we can look at the spectrum. So now here's.

[26:35] The wonderful frequency modulating sound that I just made, and here is its spectrum, if I can find that window again. Ta-da.

[26:49] OK. The choice in frequencies I made was good for our ears to be able to find the same bunch of junk, but it's not so great for looking at. So I could change the parameters later to show you how this is all affected.

[27:05] What I'm going to do now is check that this spectrum that we're looking at right now is actually kind of the same thing as these two techniques of doing frequency modulation. So there's the one and here's the other. So, remember what this graph looks like, and now we do this one, and we get approximately the same thing.

[27:33] So that's another non-proof, another sort of demonstration without proof, that in fact what we're doing here is in some sense equivalent to what we're doing here. You heard it, now you see it in some sense.

[27:48] Now I'm going to get rid of this and we're just going to be just looking at the spectra of frequency modulation networks. And I'm making room because this is going to grow as always, stuff never shrinks. Save.

[28:12] OK. So now we listen to this and we're looking at the spectrum. And so now, as we remember, and if we turn this down to say zero, we're looking at a sinusoid which looks like a peak in the frequency land.

[28:29] And now as we turn this back on, so one thing you see is that the thing is getting fatter and fatter and fatter, the bigger I push the deviations in the frequency. All right?

[28:54] Now, the other thing that we sort of already know is - let's get a bigger frequency, like a hundred, so you can see it. The spacing of the peaks that we have here is set by the modulating frequency. So, terminology. I've been sloppy about not defining my terms as I use them, which is unfortunately normal for me.

[29:22] So this is an oscillator. This, in common speech is called the carrier oscillator. And this oscillator is called the modulating oscillator. And I think, although I can't swear to this, that this goes back to the days of radio where FM was a way that you got signals from a radio station to a radio receiver, and this signal, the carrier signal -would be whatever it will be... 92.5, right?

[29:50] Whatever radio station you listen to, you dial in the carrier frequency and this modulating frequency would be the frequency or frequencies that would be present and the signal which you're listening to on the radio.

[30:04] So that's why this is called a modulating frequency. And so this, your FM radio, doesn't sound like this when the announcer is silent, only because this frequency is 92 megahertz saying it's too high to hear.

[30:21] These frequencies would be audio. In this case they're both audio, they're both in the audio range, 20 to 20k and again. As we change the strength of the modulating oscillator, what we see is that the spectrum that we started with grows limbs.

[30:48] And, furthermore, what we see is that those limbs don't move around; that is to say, they don't shift left or right. They stay in the same place, but they change amplitude. Oh yeah, negative does something similar to positive, as usual.

[31:02] And where are these things? While everyone knows, I'll tell you anyway. So this frequency here is 440, and these are 440 plus and minus a hundred, plus and minus 200, and so or to put it on the way all of these peaks are separated by 100 hertz from each other.

[31:21] And the other thing about that is if one of these things lands negative, an oscillator oscillating in a negative frequency is the same thing as an oscillator possibly with a different phase oscillating in a positive frequency because we can only hear the little part of these things.

[31:36] And so, if I push this amplitude so that peaks further and further out from the center, get energy, at a certain point I'll start to see funny stuff in the low frequencies because the peaks will start... new peaks will start appearing. You can't see it with these choices of frequencies. Sorry, I'm going to now increase all these frequencies. Let's go to E and this will be 200 hertz.

[32:18] Try it again. Yes. So now you see there's the carrier frequency. Here ise side bands which are these. I don't know what or why they're called side bands.

[32:30] These are peaks which describe frequencies that are present in the signal that are 660 plus and minus 200 hertz. So this is 460 (660 minus 200). This is 260, this is 60, and furthermore, there is going to be one that is minus 140, which is 60 minus 200. And minus 140 is the same thing as positive 140, if it's the frequency of an oscillator.

[32:57] You see, there's a peak trying to grow here and that peak is 140 hertz. Furthermore, I can put more and more energy into it, and it's not just that we're going to get energy at 140, but we'll get energy at 340 and 540. Those are actually minus, well, morally speaking; these are minus 140, minus 340, and minus 540 hertz. So we see them as positive.

[33:25] So the amplitudes depend on this amplitude here. And the frequencies are all fixed forever, immutable.

[33:36] All right. Now, one other piece of terminology before I forget to say it, which is that these things all have names. This is the carrier oscillators, this is the carrier frequency, this is the modulating oscillators, and this is the modulation frequency.

[33:55] This is the amplitude of the modulating oscillator, which is also known as the index of modulation. People who were talking about wave shaping back in the 70's stole the word index, I think, from FM and started using it to describe wave shaping. So in a wave shaping setup, where you have oscillator, multiplier, and non-linear table lookup, that multiplier there could also be called the index of wave shaping.

[34:26] So index sort of means either the amplitude of an oscillator before you do something non-linear to it, like this or like wave shaping; or it could mean how much you're messing the sound up by doing something non-linear to it, which is the same thing by what amplitude you throw it through this non-linear sort of coasting over details there.

[34:53] So what does it sound like? The original tone and then you get this kind of stuff.

[35:05] You can get all the partials you want. Notice that the sort of characteristic pattern of frequency modulation, which is that after a certain point the partials start appearing in pairs because negative frequency and positive frequencies ones are both going in an arithmetic sequence with the same separation. So you should get this sort of one-two-one-two-one-two kind of pattern.

[35:35] The other thing about it is the amplitudes of these things... OK, let me drop the frequency. So I made the modulation frequency large then to show you all the reflection about zero, which happens in frequency. But the other thing to wonder about is how do the amplitudes of act? OK they're vessel functions, you've all heard that, but how do they act empirically?

[36:08] OK first off, the thing gets fatter and fatter as you push up the index of modulation, which is the amplitude of the modulation oscillator. The energy starts at the center frequency, and it goes out so the signal picks up bandwidth. The FCC gets very excited about that because, of course, if you have two radio stations, the bandwidth of the sum of their bandwidths ought not to be more than the distance between the two frequencies, or else they'll be cross-talking.

[36:37] So, now talking about amplitudes, then, yes, the amplitudes arrange themselves so that more and more energy appears further and further out from the center frequency. But without these frequencies changing, it's just that the amplitudes are changing in such a way to make the frequency appear to be spreading.

[36:56] And the other thing is that, OK, to start with, you get nice, normal reasonable stuff like this. And it even sounds reasonable, I'm not sure but ...

[37:12] Now, that sounds horrible, but that's because I chose bad frequencies for the nice picture, right? So the other odd thing that starts happening is that as you push the frequency harder, the carrier frequency, the center peak, which is the peak at the carrier frequency, loses energy, but it looks like it's giving energy off to its side bands, which let's sort of pretend is happening.

[37:38] These are side bands. But it actually ends up giving all of its energy off into the side bands. So as we push the index further, we actually lose the center frequency altogether. It happens at an index of about 0.38. I don't actually know what that number is. I've tried to figure it out once but I think it's just a number. OK?

[37:59] There's a number of which you just don't have any fundamental, sorry, you don't have any carrier frequency left at all. But you have nothing but side bands. Furthermore, if you push it further, that amplitude which was going down keeps going down and goes negative. But, of course, negative amplitudes are the same thing as positive amplitudes.

[38:23] So we're going back to zero. So the evolution of the amplitude of the first one is that it goes from large to zero to negative to zero to up to zero, to down and so on like that.

[38:35] Meanwhile, watch this partial. Actually, these two will have the same amplitudes so watch every one of them. And they start from zero, and they start going up. That's all right, but at a certain point, they hit their maximum and start dropping, too. And, furthermore, they will eventually go through zero, as well.

[38:55] Furthermore, the next ones will go through zero, as well, and so on like that so that you'll actually even have this sort of a wave or even a sequence of waves of energy going out from the frequency of the original fundamental.

[39:09] So this is the original carrier frequency. Now we have one lump here and another lump there. And if we start pushing the modulation index - index of modulation - up higher, you'll see more and more of these waves and you'll hear these partials appearing and disappearing in amplitude.

[39:28] And that gives you a characteristic sound that you can sort of describe as a rolling sound, which, you know, you can think that sounds cool or you could just sort of think that sounds like the bad side of FM, depending what kinds of sounds you like.

[39:57] But I will say that if you listen to John Chowning's music, which is worth doing, John Chowning being the person who invented frequency modulation as a synthesis technique for music, you'll find that his indices can tell you a lot.

You'll find that the beginning of his first [inaudible 40: [40:15] 18] of modulation all very nice and small but then he sort of starts feeling his oats and the indices start pulling up. So, never mind I said that.

[40:28] Anyhow, these are the nice ones that are classical sounds that have nice, smooth spectra that just have a peak. And then you can get the funny sounds that are just complicated; fraught full of energy all over the place, which sound like this.

[40:46] OK. So the easy way of describing it, describing what's going on, is this sets the center of the energy. This is the carrier frequency, which says where the energy is going to be centered. This talks about the bandwidth, and not the extent to which the energy is spread out over other partials besides the carrier frequency.

[41:19] And this is the space limit partials. And, of course, now good things happen when you ask for the carrier frequency. 220. The modulating frequency to be multiples of each other. Now we've set up a situation where the carrier frequency and the modulating frequency are the same.

[41:48] And so to start with, we have this and now as we push the index up, the first sideband over here is going to be DC zero frequency, and we won't hear it. The next one will be twice the fundamental, which will be up here, and, in fact, no matter how often used add and subtract integer multiples of 220 to 220, you get another integer multiple of 220. And so what we have here, no matter what, is going to be periodic.

[42:25] And its period is going to be consistent with the frequency of 220, that is to say this period will be 1/220th of a second. And this is the sound that a 1973 sounded to those computer musicians like a trumpet. So if you say make a computer music trumpet, that's the sound.

[42:51] Now, how do you make the computer music clarinet? We'll just make this one be 440 like I did start with, and now the first peak is 220. But then you get 220 plus 400, which is 660, 220 minus 440 is minus 220. So the reflection is of this peak lands right where it was and furthermore every multiple, let's say every 220 plus or minus any integer times 440 is 220, or 660 or, whatever that number is, five times 220, which is 1100, and so on like that.

[43:29] And so now we have a sound that has only odd harmonics. And that, ladies and gentlemen, is the computer music clarinet from 1973.

[43:43] So you got your trumpet, 220. A trumpet is modulated frequency equals carrier frequency, and the clarinet is modulated frequency is twice the carrier frequency.

[44:00] Here's another thing about that. I set the carrier frequency to be 220, and then we saw that all the possible peaks that could arise in the side band would be odd number multiples of 220. But that could also be true if that number were 660, or 1100.

[44:29] All I'm doing is I'm taking the carrier frequency and I'm placing it either here or here or here or here, and the result is always the same collection of possible harmonics.

[44:43] The timbre changes. What's the next one? We'll add 440 to 154. So what I'm doing is I'm moving the carrier frequency to occupy different peaks in the spectrum, but the spacing has always given as 440 and so the spectrum itself stays the same.

[45:12] In general, this is true but actually no matter what this number is or no matter what these two numbers are, you could then add this number into that one or subtract it from it, and you would get different amplitudes, but you would get the same frequency so the partial is present.

[45:28] So now for instance and maybe this is kind of obvious, you can play additive synthesis game to the frequency modulation by choosing carrier frequencies which are chosen to be lying on a desired spectrum. The spectrum, though, of course, has to be a spectrum that you can get from FM in the first place.

[45:49] Then you can through if voices of FM along any of these possible center frequencies and add them and you will get more complicated FM instruments, but still obey that same spacing of frequency.

[46:04] And there are two simple ones that work well. This is the odd harmonic one and then there is the normal one, the trumpet one, as I called it.

[46:19] So now, the carrier and modulator frequencies are the same. But again, now I can take the frequency of the modulating oscillator and add it to the carrier oscillator any number of times, and I get these kinds of sounds. Notice, the fundamentals is the same as 220, but they're twice as many peaks in the spectrum; the other spectrum only had the odd peaks, and this one now has all the integer multiples of 220.

[46:54] And also enjoy how the, since I chose a reasonable index of modulations below that wonderful number 0.37 something. The peak is always going to lie where I put the carrier oscillator. Not really, but sort of. The peak is sort of here. I'll try you another one. Let me drop this back a little further.

[47:21] Oh, sorry. 0.37 is where this peak actually disappeared but there's also some point at which this peak it's being at the top, the tallest, and I don't know where that is. That's some smaller number.

[47:32] So now we have 220, 440, 660, and now you can see that you can make spectra that variously have their energies centered in different places and you can superpose them.

[47:45] What about the phases? The story about phases is ugly and you should look in the book if you want to know how the phases operate. But the short answer is if you work it out so that the phases below the carrier frequency are all in phase, are all like cosine. Then the ones up on the other side are like cosine, too, except that they are alternating a sine.

[48:12] I don't know a good, simpler explanation for why such a thing would happen, so phase is a mess. Just try not to think about phase. Pretend that's on the other side or something.

[48:22] And if you do care about phase, don't do frequency modulation but do something that has a simpler spectrum than this. So the complexity of the spectrum here comes from two things. One is that you get that rolling effect that the index of modulation gives out, that sort of chaotic in-and-out of various frequencies and the other thing that is odd about it is you don't see it but the phase is a bit simple or not terrible or ill-behaved.

[48:54] The good thing is that the amplitude of what comes out is really, really well-behaved because it's just no matter what you do to the frequency or phase of this oscillator, you can modulate this oscillator until kingdom come, it's never going to get outside of the range from minus 0.3 to positive 0.3. And so it's going to be good to your Fender Dual showman Amp, right? In a way that some other algorithms that were chosen to have phase coherence here might not do you for.

[49:20] An example would be the phase or line synthesis technique which is described in chapter six, which shows you how to make these spectra with very nicely controllable amplitudes and phases of partials but which has another Achilles' heel, which is that it gets very spiky and very bad for amplifiers.

[49:42] I don't know of any way of getting both good amplitude and phase behavior and getting a signal whose behavior just in terms of what it ranges from and what percentage of the time it's actually giving you good energy are both controllable simultaneously. I don't know how you would do that.

All right. Now, next thing about this, picturing yourself at Stanford University back in '73. This only cost us two oscillators, that's to say it only took Chowning an hour of computation to hear about five seconds of two operator FM back in the day on his [Inaudible 50: [50:02] 38] F4 computer, if I remembered it correctly.

[50:26] So why don't we make the thing take an hour and a half to get our five seconds of sound and add another oscillator. Where are we going to add it? Well, let's add it.

[50:41] OK, well we know what would happen if we add one down here. We could figure out what would happen pretty quickly if we add another one down here. OK, so we'll need an adder now. So we put these puppies down here. And then I'm going to just add them because I might be adding other stuff in, too.

[51:04] So here's the thing. OK, I'm not going to look at it, too. Good. OK. So if I had another one of these things with other parameters, but I'll reuse the modulating oscillator and just give myself two carrier oscillators and add them, what then would happen?

[51:27] Well, we know what spectrum this thing is going to make. Well, at least we know how to talk about what kind of spectrum this thing makes. And this one is just another of the same thing so it does the same thing, too, right?

[51:53] And so now, we just superpose the two spectra, and now we have more control over the timbre of the sound. The light -- the wonderful motion controlled light, either concluding that we're not moving enough or that we're moving too much. We will never know.

[52:24] So what happened there was, maybe this is just too obvious for words but I'm reusing the signal but in fact I would have gotten the same thing that had two of these oscillators with the same parameters, more or less.

[52:38] What's happening here is I'm just adding, I'm just using two carrier oscillators and what's coming out is just the sound of what would have happened if I'd done the two carrier oscillators separately and that's kind of obvious now that you look at it, right?

[52:52] Except that if I were one of those crazy people who like to do frequency modulation with waveforms that weren't pure sinusoids, one way that I could think about what that would be doing is I could think of the non-sinusoidal waveform as being a sum of sinusoids in different frequencies.

[53:15] And then you could think of the oscillator itself as being an additive synthesis, an equivalent additive synthesis patch that might have an infinitude of oscillators being added into it but anyway you could simulate any waveform you want with additive synthesis. I'm not going to prove that right now.

[53:31] And so this would be a good description of what happens when you have a non-sinusoidal waveform as a carrier oscillator, which, by the way, the FCC cannot be happy about. That's OK, we're not radiating too much here other than acoustically.

[53:59] So that is adding another carrier oscillator. Now, what we can think about is... Oh in terminology, by the time you do this, then you're starting to get the idea that these oscillators and these controllable phases are building blocks, in a sense. And for some reason, the Yamaha Corporation got to calling these things operators.

[54:26] So this is three-operator FM. And of course, three-operator FM, you could invent other technologies. And, in fact, I'll show you another couple that could also be three-operator FM. The famous DX-7 synthesizer that was FM for the masses for the first time was six-operator FM. So imagine all the good cool things you could do with six of these piling up together in different ways.

[54:55] And the thing that made that all possible was the fact that it was phase modulation, as opposed to frequency modulation, so that the units that you describe the amplitudes in were all compatible. In other words, you didn't have to go choosing crazy different ranges of numbers for different oscillators. So you could manage them very easily, even in old fashioned 1980's architectures, '70's even. OK, so go back.

[55:23] So that was putting the oscillator there - the extra oscillator. I could take this operator, if you want. I could take the extra operator and put it up here but you still have three-operator FM. But now what we have is the two other oscillators. Let's see how we get it so we can see all this. There's not much hope anymore.

So I can make this thing take less vertical space. You're just not going to be able to do very well. So let me turn all of these things off and let's see what we get. The carrier frequency is 440 and I'm going to choose two modulating frequencies: [55:47] 220, and here I'm going to choose a different one. What's a good choice? 550.

[56:33] So now, both of these oscillators are turned off in the sense that their amplitudes are zero. So now we just hear the carrier frequency, and now we know what this oscillator will do to it.

[56:46] I'm sorry. I forgot the shift key.

[56:53] There. That's two-operator FM. Here's what the other one does. The other one is 550, so I'm going to make it 550. Sorry. And then it does this.

[57:09] Isn't that sweet? Maybe it's sweet, maybe not, depending on how you feel about it. So just to be painfully slow about this. There's 440. There's 440 minus 550 which is plus 110; and then this is 440 plus 550 which is 990. This is this one plus 550, so it's 660.

[57:37] So we got 110, 440, 660, 990, and so on. It's 1, 4, 6, 9, 11 and so on times the fundamental frequency of 110. That's kind of cool. And it sounds like this.

[57:54] All right? And now, how do you think about what happens when you do both of these together? It turns out to be strikingly easy. Let's do it this way. Let's start with the 550. OK?

[58:28] So if someone gave you a two-operator FM network that was a sinusoid modulating but was a carrier that had a complex waveform, you could think of that as a sum of simpler two-operator FM modules, where modulator was always a sinusoid and the carrier is always a sinusoid because those imaginary carriers would just arrange them to add up to a complex waveform.

[58:53] This thing you can think of and analyze in exactly the same way, because you can combine these two. These are the ones we're listening to right now, you can think of this thing, whatever it is, it's periodic with 110 hertz period.

[59:08] And so it itself is a thing that at least in your mind you could describe as a sum of sinusoidal oscillators and here are the frequencies that they're at. And we could even, if you wanted to, write down a bad formula for what their amplitudes were.

[59:27] And now when we start taking this complex waveform and modulating it with this sinusoid, what will happen is that it will act like each of these peaks was independently getting modulated. And then we wouldl get this extraordinary complicated thing, which is each of these peaks sprouting at some side bands.

[59:48] And of course, the sidebands are all mixed up in each other because the first two sidebands of this peak are here and here, but the first two sidebands of this peak are here and I'm not sure where to say the other one is, probably here again. I'm not sure.

[60:05] And so now, we get all of the frequencies, which are... OK, so to go back. Now what we have is this is two-operator FM. If I turn this one off, so now it's just these two, right? So if 440 plus or minus an integer multiple of 550.

[60:26] Now each one of those you can think of is being the carrier frequencies for a new two-operator FM setup and whatever these frequencies are, like this 440 again will spout 440 plus or minus the neutral multiples of this. So in sum, what you get out of the whole thing is 440 plus or minus any integer times this plus or minus any integer times that, which is potentially a very nice thick spectrum.

[60:56] Whoops. Sorry. So we have that sound and now we add this sound and we get this kind of stuff.

[61:06] And now, it's idiot's delight. We have basically all the spectra that we can possibly wish for and the only thing that we can wish for that we don't have is some way of actually getting from one desired spectrum to another in a continuous and ergonomic way, or thinkable way, understandable way, comprehensible way. All right?

[61:35] So you throw numbers in here and you get spectra. You can say where the frequencies are, and that's all right. But if someone tells you, "I got this spectrum. Can you make FM do it for me?" And then you just say, "Hmm." [laughs]

[61:51] There are papers out about that because people in the early '70's got really excited about that, "Oh FM, that's this very powerful synthesis technique. Let's take the sound of this real trumpet, analyze it and then figure out how we're going to stuff parameters into this FM networks to sound like a real trumpet that wasn't analyzed."

[62:07] And the answer is you can get this waveform when you get that well, not even but you can imitate. You can get a bad imitation or a vague imitation of one set of partial strengths and of another.

[62:22] And then you can try to get to continuously from the one to the other, but then you will find that the parameters that you had to do for this aren't actually in the neighborhood of the parameter that you had to get for that. They're somewhere else, in some mountain range of horrible parameter choices. And as a result, you can't actually make continuous pass and get between spectra in a desirable way that you could wish as a general rule.

So FM turns out to not be nirvana in terms of synthesizing sounds for the simple reason that if someone gives you a sound and asked you synthesize it with FM, it is usually just not possible. So what would you do? Well, you can go back do a [inaudible 1: [62:45] 02:59] synthesis, probably the easiest way to do that.

[63:03] OK, so this now is three-operator FM, with two of the operators... Sorry these are oscillators but we call them operators now because that's what they call them, operating on the third one. And now, of course, you can also say, "Oh cool."

[63:21] Let me explain quickly again the asymmetry of this design or of this picture. This oscillator is like these others, except that I haven't thrown this adder in. I'm just saving real estate because I'm going to... some, if it's going to be a tree structure, it's going to be a tree structure today.

[63:38] And so, somebody's going to be on top, someone's going to be on the leaf because you can't have a tree without leaves, as far as I know. And so this is going to be leaf. So now what we're going to do is we're going to say, "No. I don't want this thing to modulate this oscillator down here. I want it to modulate this oscillator over here."

[63:54] And then what do you get? Now you get yet another spectrum.

[64:04] And even less of any reasonable way of describing what that spectrum is, except to say this. No, you can't even think about this. Don't even try to put this in your brains. This is another thing and you can do it because it's easy to do. But trying to analyze what this does and I think it's just going to be hopeless. So, just don't try it.

[64:31] Al right, let me tell you how hopeless it is. Analyze these two and you'll get an infinite number of frequencies here, which are this frequency plus or minus multiples of this frequency.

[64:44] Now, this thing, then you could regard as an equivalent to an additive synthesis network with an infinite number of oscillators in it. Each of those is modulating this thing. And I told you how to think about two oscillators modulating this thing, which gave you a doubly infinite set of peaks.

[65:04] Well this thing gives you, then, an infinitely infinite set of peaks because each of these infinite number of peaks is independently separately modulating this one. And their indices of modulation, furthermore, are given by the amplitudes of the components of this pair, which themselves are moving in this horribly complicated way.

[65:24] So the whole thing is just completely beyond any rational analysis at this point. But you can dial right up on your DX-7 and say you can enjoy all day and people have gotten good intuitively at making sounds out of this thing, even though it's impossible to understand what's going up.

[65:44] So there is FM operators and all that stuff. Now, just to go back quickly to the question of... Maybe I don't need this one anymore. OK. Going back now to this thing, I said rather a simple thing about this which is that the frequencies present are this thing plus or minus integer multiple of this plus or minus whole number multiples of this, I can even say.

[66:24] Now, it turned out that all of those things were multiples of one number, which is 110, and you can tell from by looking at that this plus N times that always ends with integers that are all going to be multiples of 110. That's cool, right?

[66:37] If I gave you any two numbers here, and said "What is the number of that this plus or minus N times this is always an integer multiple of?" The answer is going to be, I think the answer is always going to be, and the best thing that you can get is going to be the greatest common factor of these two numbers. Greatest common divisor of these two, the GCD of them, which is in this case, 110. But in this case, the greatest common factor of these two numbers is one.

[67:23] And so even though it looks nice now, this thing, if you want to wait until this thing repeats, you have to wait an entire second because after a second this thing will have moved, this thing will have jumped 551 cycles and this thing will have gone 440 cycles and it would be back to where we were.

[67:47] But there's no other point at which these two phases both will be equal to what they were before. And just pushing this index a little bit, now we can do the usual cheap thrill FM thing which is walk through all of the wonderful frequencies.

[68:20] Should I make this a little louder.

[68:26] So if you've ever heard sounds like these, they're not atypical of frequency modulation synthesis. And of course there's a special case where this one and this one happen to be close to integer multiples of something else. So in this case, we have positive and negative things that we didn't quite line up and so they're beating but this number, I believe, is about three hands of this number, is that true? 290's, so half of that is a hundred forty five. So three hands of this is I think 330... is this possible? 435?

Man 2: [69:09] 435.

Speaker: [69:12] Yeah, so now what we have is 150 hertz coming out. Actually we got odd harmonics again, oddly enough. So every once in a while, if we sweep either of these two, you will hit a situation that sort of [inaudible 1:09:32] you where these peaks whack into each other and do something nice and sparse in terms of spectrum, and then you get those sounds. And between those sounds you get all the very juicy, creamy sounds in between. [69:58] So as you're looking at the spectrum, you can imagine these peaks actually just moving through each other like ghosts through a wall.

[70:16] So that, in a nutshell, is the story of frequency modulation. Let's see what I have to tell you other than what I've just said that's important. That's what it is.

[70:33] So things to take home about this are, first off, this is real easy to do and it's a cheap thrill, it's in yourself, that's how cheap a thrill it is. What do I mean by that? It's so cheap that you can do it in silicon using very few watts or microwatts, even. So it becomes a very easy thing to build circuits to do, which is why you hear a whole lot of it.

[71:03] It's very well behaved in terms of the amplitudes that you get out because of the fact that finally what you're looking at always ends in this cosine function. So the behavior is good. You control the amplitude because in the end it's really just an oscillator with a changing frequency.

[71:20] The gotcha is you don't know how to do anything with it, other than the very simplest things with any sort of actual predictability. The only way of finding things out about FM is to develop an intuition on how to get cool things out of FM, which people have spent many years doing because there are lots of people who spend all their time programming.

[71:44] Yeah, literally there are people who spend all their time programming, myself and sounds sound like trumpets and pianos and bells and all that good stuff. And you can enjoy their work and millions of people can enjoy their work, too, but if you want to do, you can count on spending many years messing around with these things yourself. Maybe that would be good, maybe it wouldn't. Depending on what you think do you want to do with your life?

[72:16] That's it. That is the entire story of frequency modulation, unless I've forgotten something important. I don't think I have. We're done.

MUS171_02_24

Professor: [00:01] Homework assignment. This is a... I will come back and present this slide. OK. You have to do this with sinusoids before. That is the way of learning abstractions. So, this is the same, essentially the same thing except that, the synthesis technique is one that uses delay lines and in fact, some of you already know this.

[0:26] If you want to make tones like this, you just play a little white noise or something into a recirculating delay line. For those of you who do not know it, we will see it today. So, if you do not know how to make this tamber, you will know soon how to make it.

[0:44] This idea is attributed to people working at Stanford Karplus and Strong that is called Karplus-Strong synthesis and it turns out that it is great for making harpsichordy kinds of sounds like that and you can push in a couple of ways but then, pretty much, it will give you that very recognizable tamber and if you ever hear that, you just say; "Oh, that is Karplus-Strong," and that is kind of good although this is a good thing as a source for filtering.

[1:12] What I did was I just fixed it so that it has a controllable duration and has a controllable based pitch just as before, and this is a random melody but you can elaborate on that all you want.

[1:28] So, that is... now without a mirror in your heads, that is a thing that you do with delay lines. But, what I am going to do is show you more in general what delay lines are, like what is the range of experience that you can create using a delay line.

[1:45] Delay line simply takes something in and it puts it out at some amount of time later or at the present time, it is putting out what it got at some previous moment and to do it, the simplest possible example might be, let us just use the microphone to start with just because it will be upsetting.

[2:08] OK. So, what you do to make a delay is two things. First of, you make a delay line and to do that, the thing is del write and you have to give it a name because delays like arrays and like send and receive pairs are things that other things refer to and they have to be able to find them by name.

[2:28] So, I will say Del right and then give it a name and you also have to tell it since it has to make space; you have to tell it how much space to make. Unlike arrays, delays really are just arrays. But, unlike arrays, delays are things that have signal running continuously through them so that they actually have a notion of sample, right?

[2:48] And, as a result, in a delay line, you specify not the number of samples but in the number of milliseconds which is the usual time in it. So, I am going to ask it for a five-second long delay just because I cannot imagine to running more than that.

[3:02] And then, you just say delay read, give it a delay name that matches the writing delay line and then give it any amount of delay which you wished to have the thing delay by and furthermore, that is a thing that you can control using numbers. I should make a number box.

[3:25] So, here is the whole patch. I will just take the delay read and throw it to the output. Here. OK. It is complaining to me. Why? That is me. That is this window generating problems. OK. And now maybe, if my mic is on, "Hello," yep. OK. So now, what you hear is me in second previously. This is a good way to really reduce the intelligibility of speech by one.

[4:03] OK. So, if you want to change that, you do not even have to specify the initial value. There is no story to associate with reading the delay lines. It is just getting stories that were made by the delay write and so, almost as in the signal versions of send and receive, the delay write defines the delay line and then you may have as many as delay reads as you want reading from it.

[4:27] So, for instance here, I can dial up the amount of delay that I want in milliseconds. So, here is 132 millisecond delay and so on like that. Well, actually, if you say negative, it makes it zero. It cannot get less than that and I will make that a little bit better by actually giving it a range.

[4:50] So, zero is the shortest delay to make which do not, well, which is basically the delay of getting through the audio system of the computer in plus PD, alright? And, then you get various things.

[05:04] So, certain delay times are just enough to make you queasy but not enough that you can actually hear the delay and then along right here, you get problems with speech intelligibility because there are a lot of phonics of speech particularly the consonants that typically are over in less than 50 milliseconds and so, if you present a delayed copy of speech, you are squashing those phonics out and mixing them with there neighbors which ruins the intelligibility of the speech.

[5:35] And, that becomes near total when you pick up to 100 milliseconds in which you can give a nice echo. Well, for thinking about it, 100 milliseconds sound goes about a click in millisecond. So, 100 milliseconds is 100 feet; so, that is the echo from a wall that is 50 feet away from you. OK?

[6:04] OK, I am just going to save this as it is right now. It is very simple, but this is the basic deal about delay. Yes. I am going to save this except I am going to save a slightly modified form of it which is going to be this. Just to emphasize how delay reads can reuse the same delay line.

[6:26] Now, we have the two speakers. Each of which can have a different delay and then you have this. All right.

[Clicking sound]

Professor: [6:41] Alright. And now, you play an instrument of that then you get to rock and roll.

[Sound]

Professor: [6:49] I am checking here in [inaudible 6:50] .

[6:58] Yeah. Alright, do not worry about that. Of course, delays are great effects and of course you are going to want to turn it up so that you can really hear it and then it will going to feedback. Alright, so delay networks are very feedback prone. OK. That would be a good thing to worry about.

[7:15] OK. So, this is a very simple delay, patriotologists say and then we move on to the next. Yeah?

Audience: [7:24] What are the five [inaudible 7:25] .

Professor: [7:26] Thank you. Yes. So, the 5000; that is the amount of delay line that delay write created in milliseconds. So, in order to make a delay line, Del write has to allocate memory because it has to continually remembering the last five seconds of whatever came in to it and so, you tell it how much memory you want to grab and you can ask it for hours, all right? But, I do not know any situation which you will need to. Yeah?

Audience: [7:52] So, [inaudible 7:53] the tail one.

Professor: [7:56] It is not really a tail. What is a tail? OK. So, I think. I do not want to answer that. So, tails are things that happen after a sound. So, yes because it is creating space for making something come back after it is gone. But, you could make tails in other ways as far as you could make purely... you could make an oscillator base synthesize with the head tail and then you would make a delay line to do it.

[8:29] So, really what the delay line is a way that you could think of it as a circular buffer or as a loop of tape. So, what you are doing is you are writing. The memory is arranged in a circle and not in a segment and it was writing around on the circle continuously so that five seconds later, you rewrite the same thing that you had written before and so on like that.

[8:50] But, in a given moment in time, you can look back up to five seconds in the past and it will still be there. Alright. Now, next thing about that is this. People immediately think of the idea of making recirculating delays and what I will do for pedagogical reasons is I will just make a very stupid design for recirculating delay first and then I will start making it a little bit smarter.

[9:30] So, the stupid thing that I could do is this. Let us make this be a thousand and let us test the delay line if they are working. OK. Then we will turn it up a little bit. Now, what I am going to do is connect the Del read back to the Del write so then I will say something like, "you will never forget this."

[Echo]

[laughter]

Professor: [10:07] OK. Let us keep that out there. OK. So, it is still there, right? I just turned... [Echo] [laughter]

Professor: [10:16] ...OK. Actually, this is making a perfect digital copy of the thing so that it really literally will be the same thing tomorrow or next year until I of course destroy the patch which I will want to do for the same. Notice that I disconnected the ADC object from it. That is because, in this design, I have a little bit of a disadvantage because, well, there is...

[Echo]

Professor: [10:41] There is another thing that is going to happen which is that I can add other stuff into it. [Echo]

Professor: [10:46] This either. [Echo]

Professor: [10:50] And furthermore, if I keep letting that happen, then eventually, I will just get salad, right? And then, it will just be too much and so, what you would really want to patch and be able to do is not just...

[Echo]

Professor: [11:05] ...not just that, but maybe first of, it might be interesting to be able to have subsequent echoes to be quieter than the original ones and/or it might be a very good idea to be able to have your patch actually arranged in such a way that you could control whether you are sending a signal to it or not, alright? That would be the send d delay loop. Alright. So, now...

[Echo]

Professor: [11:29] So, you got this... [Echo]

Professor: [11:33] So, let us do this. OK. Now, a couple of things; one thing, this is programming style and this is little personal. I have a tendency to try to put del right's higher on the screen than del reads, so that the delay reads downward and then this line that went from the output of del read back up to the del write is then feedback and it looks like feedback when you connect the output to something lower in a patch to the input of something higher in the patch.

[12:01] You do not have to do it that way because of course you could jumble around with it anyway you want. But, if you do it that way, it is easier to remember what you are doing which you could think as a good thing. OK.

[12:10] Now that I have done that, what I really wanted to do is make this controllable in sort of obvious ways and I will go as far as to do that and then, I will save that and go on to make another one with gain, alright.

[12:25] So, the important thing here is that, we want to be able to turn the input on and off so let us multiply it by...I could be brutal and multiply each by a toggle switch. By the way, just to be pedagogical again, I am going to explicitly add these two signals. Oh, I cannot without destroying my beautiful delay loop.

[12:48] So, that is not yet. We are going to turn DSP off. Now the delay line is just sitting there and now, I can disconnect this. If you can't follow this, this is just, it is silly. Do not worry about it.

[laughter]

Professor: [13:05] Now, I am going to put a little plus in there. The patch is turned off while I am doing all of this editing, right? And now, I am going to hook this up. Oh, except I want to be able to control the feedback path to. So, let us do another one of this. I am going to be sloppy with this one. OK. So, the Del read will go to a multiplier.

[13:40] The ADC will also go to a multiplier. Alright. I am going just to be sloppy and I am not going to use line until this. I am just going to use toggles. If you are doing this for serious, we would use line until this. OK? So, now there is a delay time which was set and two toggles and right now, with this one, I want to be recirculating this one up and I will turn it back on.

[Echo]

Professor: [14:08] And now, I got something where I can just say whatever I want.

[Whistling]

[Echo] [14:11]

Professor: [14:15] And finally, I can do this and get the thing shut up so, I can start over. This is another one. [Echo]

Professor: [14:25] Alright? OK. I should have done this with lines instead of the toggles; so, I will leave you to think about all these little details. Why? Because these are just amplitude controls like any other. Is it clear what this patch is doing? OK. It does need a couple of comments. This control should have names and they should be on the patch. So, this is going to be recirculation.

[15:02] And, this one is going to be, I think the right thing to call this is just send. In other words, this is a control. This is a send to the delay line. Alright? Which is not to be confused with the return.

[15:21] This is a sound engineering language. You call a send the gain by which you send some incoming signal to some kind of effect and then you would call a return, the gain by which you would take the output effect and putting whatever speakers you have. Alright.

So, I do not have any return control. Well, maybe this is a return controlled [inaudible 15: [15:36] 40] , but here is a send control for sure and here is the recirculation which is a quality of [inaudible 15:46] OK?

[15:48] So, this is the basic recirculating delay line. All right. Now, I am going to save this. Sorry. Can I add one more thing to this patch? This has an ADC. If you are using laptops and the built-in microphones, you have a disadvantage because your mic is real close to your speaker.

[16:16] So, I am going to introduce new object just to be able to apply the send to the noise test signal and this is going to be noise tilda. I am going to be a little careful about this one and I am going to multiply it by some small number.

[16:42] OK. So, what is noise? Noise is this. Now you all know. All right. This is a very 1960 pseudorandom white noise, all right? It is not truly randomized, but it is essentially what could be if you had true randomize for normal purposes and the strict definition of it is a stream of samples. Each sample of which is a new random number completely regardless of every sample that has proceeded it.

[17:19] So, it is a memory less, noise-generator and if I am not lying to you, the range is from minus one to one. It does not have any DC in the long term. All right? Now, that is the good thing to be using with this for a very pedagogically sound reason which is that I want to talk a little bit about frequency responses of these things and you will be able to hear frequency responses if I use noise as the input signal.

[18:00] OK. So, what I am going to do is, in fact, when I simplify the patch, well, I just make as such you can hear the noise if you want...this is another sample which consist the noise sound, I guess you call it. OK, I have noise.

[noise]

Professor: [18:22] Now, what I am going to do is compare that noise to this noise that we have here. Now, put the noise and the delayed noise both being heard in the same speaker. And now, I am going to turn the delay time down to something like 10 and then you get something cool.

[noise]

Professor: [18:53] All right. So, in went noise and out came something that had an audible pitch. It is the first example that you have seen at how you would make a filter. I have actually held out filters before because I have needed a high pass filter for a couple of reasons in various patches in the past, but this is actually a filter with how the filter works.

[19:21] And, what the filter does, well OK. I will say what the filter does in two different ways. One thing that the filter does is it takes the incoming sound and lets you hear it but also lets you hear it with a delay. So, you hear two copies of signal with seven-ish milliseconds between them.

[19:43] Another thing that you hear is that certain frequencies are accentuated in the output of other frequencies or not and that is the aspect of filtering that makes us call it filters; the idea that different frequencies come in or pass through more willingly than other frequencies. All right.

[20:03] Now, why...yeah?

Audience: [0:20:01.4] For the noise, what does the [inaudible 20:08] [20:05] ?

Professor: [20:10] Nothing. I was too lazy to...I just forgot when I was writing it to tell PD the suppressed drawing the inlet, so it just has the inlet. ADC also has an inlet but has nothing.

[laughter]

Professor: [20:21] There might be one or two others and everyone know all that. A question comes up on the PD list. So, the moral is, doing nothing is often not quite nothing enough. All right. Yeah. So, all right. So, Just to analyze what this thing does...OK. So, let us make it for ease of thinking about it, let us make this thing 10 milliseconds.

[20:47] So, now we hear a pitch and, what would that pitch be? Well, OK. To think about that, what would happen when you put certain sinus? "Oh, rats." I am skipping some theoretical stuff. So, I am not explaining that you can think of incoming sound as consisting of sinusoidal components. When and in what sense you can do that is something that the audio engineers just sort of assume and which I will not tell you more about than just to assume it, all right?

[21:24] Because the mathematics is hairy. So, assuming that you think that some very complicated signal like noise tilda might actually consist or be describable as a sum of different sinusoid or different frequencies.

[21:40] And, of course, it'd folk knowledge white noise really is every single frequency with equal amplitude just like white light could be every single optical frequency at the same amplitude although it is not because it depends what temperature is.

[21:53] But, that is another thing that we do not have to worry about. White noise in audio land really is a signal which contains every frequency that digital signal can represent it and contains them all with equal amplitudes and we will just sort of forget about DC and the nyquist for now because this might be special cases.

[22:12] All right. So, if you think of that that way, then what about some possible component frequency of the noise signal? For instance, what if there were 100 hertz sinusoids sitting in there? Well, you would hear the 100-hertz sinusoid here and you would hear the 100-hertz sinusoid one period later because the period of 100-hertz sinusoid is 10 milliseconds, all right?

[22:47] And so, you may hear if you will get the same signal coming out of here for 100-hertz sinusoid as you get coming out of here and so it would sound, it would be doubled in amplitude and sound somewhat louder.

[23:00] If I put in 50 hertz, then something different happens because 10 milliseconds in a 50 hertz signals. So, the 50-hertz signal that has a period of 20 milliseconds. So, 10 milliseconds is long after wait for the sinusoid change sign. All right? No matter what phase it had at the out set. So, it is going to be minus what it was by a half period later.

[23:28] So, what that means is that, at 50 hertz, this signal has the amplitude which is exactly the negative event through the signal and they can see each other out and since we are adding, since we are putting it out, we will hear and hear at the same amplitude. We are not putting anything out at 50 hertz.

[23:55] Now, if you want me to really prove that, I will prove it as in a laboratory. Let us make noise oscillator and let us ask you to play 50 hertz for us and then I will turn it up here how to make it not all the way to 92. 50 hertz is kind of low, all right. And then I will turn it up here too. It went away. OK.

[24:26] I hope this is making on the tape. Oh, you know what, let us do the same experiment, but let us do it at a lower delay and a higher frequency. So, now, what I am going to do is make the delay a mere two milliseconds. Now, we got it back, but I am going to make the oscillator to be 250 hertz. So, now you can hear the oscillator just fine if I just play at...

[noise]

Professor: [25:13] ...but if I add the delayed copy, then it goes away. Alright? If on the other hand I had the oscillator going at 500 hertz, then, if I add the delayed copy, it just makes it louder, 60B louder to be and so on.

[noise]

Professor: [25:22] So, just proving these things with oscillators which are a perfectly respectful way to find out what a filter does by the way. What happened is, oh, a DC, I did not tell what happened to DC but of course, if you put a constant signal on delay line, we will get the same thing out after any delay that you want and so they will add the delay property the same as the original. So, very little frequencies will come out.

[noise]

Professor: [25:48] But, by the time I hit 250, it will be gone. And then, by the time I go to 500...

[noise]

Professor: [25:58] ...It comes back at double strength and then at 750, it goes away again. Now, that is worth stopping and worrying about for a second. Why did that happened for 750? OK. So, that is the easiest way to do the math here.

[26:21] So, 250 hertz, this one, the period of this is four milliseconds and the delay line is two milliseconds which is one-half of a period; all right? If I make this 750, then the period of that is three-quarters of a millisecond. No way, I am doing it wrong. 750, right? So, through 1000 it would be in millisecond, so it is 750, so it is four-thirds of a millisecond; a millisecond and a third.

[27:06] And then, if you make a delayed copy two milliseconds later, a period, yeah, right. So, the periods in millisecond and a third. So, how many periods then fitting two milliseconds. I was trying to make this easier doing once ahead, but it is not. I am sorry.

[27:32] It is one-and-a-half period. So, we have one and a third millisecond, that is four-thirds and then a half of that again is two-thirds and four-thirds plus two-thirds is six-thirds of two and so, this is one-and-a-half of these periods.

So, what that means is that we are hearing and hearing and hearing it there not a half period later but one-and-a-half period later which [inaudible 27: [27:49] 55] purposes in the same thing. Similarly, if I go to a thousand, now it appears one millisecond and two is then two periods.

[28:12] So, two, that number is constant or is fixed for now is a half period of this one. It is one period of this one. It is one-and-a-half periods of this one. It is two periods of this one. It is two and a half periods of this one and so on.

[28:28] Oops! What happen? Why do I hear that? The reason I heard it truncation error. This is two milliseconds, but we are running at a rate of 44K1, so the delay is not exactly two milliseconds and so, I did not succeed in notching it out exactly and see here a very quiet little tone likely 40 DB down but you are hearing the error in the allowable length of the delay line. So, there is a thing about delay line that I have to tell you about.

[noise]

Professor: [29:13] Which is that, well, sorry. A thing about Del read tilda that I have to tell you about which is this. It will read in all samples of the sound that is going or the signal is going down the Del right. But, it is limited to integer number of samples of delay. It will not interpolate before you try to guess what thing would be at the 5-1/2 sample to go.

[29:36] It will leave you do five samples ago or six samples ago. So, going back to this example. Truncation error is sort to give me the live. Well, I will continue the experiment anyway. So, 250, yet; 500, got it; 750, no; 1000, yes; 1250, almost no; 1500, yes; 1750, almost no, almost delay; 2000, yes. OK.

[30:07] So, the things that got truly were 500, 1000, 1500 and 2000 and seeing the pattern. It is going to be all the multiples of 500 and all of the numbers in half way between those multiples, all the integers are multiples of 500.

[30:23] So, all the half integer multiples of 500 like 250, 750, 1250 and so on are getting notched out as an audio engineer would say. They are getting canceled out by the delayed copy. All right? Now, going back to the example of noise, should I add that? I should add it.

[30:54] This is the oscillator's inn and then I will just make a separate one for the noise, so that it is all nice and clear. So, I will put add over here and I will make yet another one which is un-noise. I will see if it is going to fit, duplicate, all right. It is getting messy. We are going have to stop real soon.

Of course, there is noise. Now, we are going to add that; I mean we are going to throw that in here. By the way, I am being a little sloppy there. I am using this inlet of odd with another odd. So, this is a stylistic thing for me. You just put a plus tilda, the you throw as much stuff into it as you want, it's making it clear that it is all getting added anyway. Better than just throwing it all [inaudible 31: [31:38] 59] here for some reason. OK.

[32:02] So, now, we are sending the oscillator over this into here and now, we are going to hear the oscillator. We are going to check this. The amplitude is by the way; need to be exactly the same for this to work perfectly. All right, so that was the oscillator example and here is the noise example.

[noise]

Professor: [32:29] And then, you get a thing which is roughly 500 hertz which, you know, it is not too far from the C above middle C here.

[sound]

Professor: [32:44] OK. So, that is the 500 hertz tone, sort of; or you could compare it to what happens when I make the oscillator to be 500 hertz. OK. Observations about this? I am going to save this and now really, it is time for me to start a new patch because it is getting too crappy. This is the end of this patch. OK.

[33:06] So, observation about this. This is a linear process. The linear process meaning is taking a signal and making a delayed copy of it and adding the two or even not. So, any, well. Adding any number of delayed copies of the signal to the original signal is a linear process in the sense that, if you add two signals in, you will get the result of what happened if you put the two signals in separate way.

[33:39] It is furthermore time and variant by which one means that if you put something into that network now or if you put it in a minute later or a second later, you will get the same thing out as if you just put the original thing and just done the whole a second later.

[33:58] You could have things that are not time and variant. A very good example of something that is not time and variant is multiplying by a sinusoid as it is in ring modulation because that one moment it is putting the thing through positive and in one moment you are going to think negative.

[34:12] So, if you put an impulse into it at some point, it will be positive or negative or nothing depending on when you put the impulse in. So, that is not time and variant. This is time and variant. Putting impulse in here at any time or any other time, you will get the same thing out simply at that time, all right.

[34:32] It is a property of linear time and variance systems that you can fully describe them by finding out what they do to sinusoids. The way and the pass that we have generated frequency that was not present in incoming signal.

[0:34:55] So, there have been lots of examples where you have an oscillator and then you make different frequencies come out either by multiplying it by another oscillator which is not time and variant or by running it through a nonlinear so called transfer function which is wave shaping.

[35:07] Each of those things is capable of generating frequencies that are not present in the original signal, but it is in general truth above linear time and variant transformation that whatever frequency you get out. What am I saying? If you put a sinusoid in, you will get a sinusoid out of that same frequency. Yeah?

Audience: [35:32] Would you explain time and variance again?

Professor: [35:34] OK. So, time and variance is this. It is that if you...so, it is a process that has an input and output. OK, and so, making input, run the process and get the output. And now, I will make the input delayed by any amount of time you want and apply the same process, you will get the output delayed by the same amount.

[36:03] So, the laws of physics are believed were probably time and variant because if you drop an apple today or drop an apple tomorrow, the same thing happens although it happens a day later.

Audience: [36:12] And then explain why multiplying by an oscillator is not time and variant because it depends on what phase of.

Professor: [36:19] Yeah, right.

Audience: [36:21] Got it. OK.

Professor: [36:25] OK. So, I am not going to try to explain why a linear time and variant system has this wonderful property but, good thing is about this. Well, it would be good if your amplifier was linear in time and variant, all right.

[36:43] It would also be good if your speaker system was, all right. Because if you, well, if you hertz frequencies coming out of your speaker that were not on the recording or a thing that you are generating then you would think that there is something wrong with the speaker like what you call harmonic distortion or something like that.

[37:00] Linear time and variance does not make up frequencies for you. It does however, or can however change the amplitude of sounds at different frequencies. In other words, you can change amplitude and a frequency in dependent way and this is a very simple example that where I am throwing in sinusoids and the happy ones that are multiples of 500 are getting double in amplitude and the ones that are happening between them are getting zero in amplitude. All right.

[37:31] So, that is a thing which we would call a filter or I do not know what historical reasons. But everyone I think has this folk knowledge where the filter should be undo and this is basically how you make filters. You make amount of delays.

[37:51] So, delays now, just too...see, I do not want to change the patch, but I will...let us go back to doing the send again. So, now, can you see, we are sending and oh yes; so, I do not like to make the patch even more readable than it was by doing same thing to all the three sources which is to say adding them up like that.

[38:21] I did not make the patch very much more readable, alright. Then I can lower the stick. That is good. OK. So now, here is the original patch maybe, let's see. I have to make it a little louder before this is good for everyone.

So, now this is my voice being filtered in the same way and you, might be able to tell that there is some kind of tamber variation. Actually, you will hear it [inaudible 38: [38:39] 48] original voice, there is decreased in proximity effect [inaudible 38:54] to my voice, it becomes too low. But, it is basically what is going on, right.

[38:59] And here is now the filtered result which is, you know, 5000, 1500 and 2500 accentuated everything else. Other frequencies now, alright. OK. Yeah, you could now say, let me have this as a continuously variable process. So, "Aaaahhhh," I do not think a tone or anything is going on there.

[39:29] But, what you are hearing is a varying filter. I am doing this in a rather sloppy way. In fact, I can do it in a more easy to understand way by using noise again with less amplitude.

[noise]

Professor: [39:55] And now, we have...OK. And, that is changing the frequency response of the filter. OK. What is a frequency response? Frequency response is a name for, at any given frequency, what is the gain of a filter?

[40:13] So, the frequency response is a curve or it is a function of frequency which in this case, has a shape like that, has pitch in multiples of a fix frequency and you can change that for its frequency which is therefore changing the frequency response to the filter in its very audible way.

[40:35] Now, that is what the filter is doing for you when you have very short delays. Let us go back to voice now. This is a thing which either can give you this kind of effect, "Hello." I don't have it on so I am not hearing anything, because I am doing the wrong thing. "Hello." So, now, what I am doing is something that you do like a time domain.

[41:03] So, that now, the delay time now is more than the magic 30 milliseconds which is something like the threshold of which you will hear as a time interval. So, if it is more than 30 milliseconds, you get this and if it is less than 30-ish milliseconds, you will start getting things that are described to as filters.

[41:23] And, by the way, notice, I have been sloppy about this without explaining to you that this is a problem but, of course, if you test this thing carefully, you will hear that it does that when you change the delay time, all right?

[41:43] That is exactly the same effect as if you are using an array as a sample, as a recorded sound and suddenly jump from one spot and then record sound to another without controlling it by enveloping it somehow.

[41:58] And, it is that for the same reason almost which is that if you change the delay, you are changing essentially, you are stopping playing the thing at one delay.You are starting playing it discontinuously another delay. It is almost exactly the same thing if you pick the needle up on a recording and drop it somewhere else spontaneously will cause a discontinuity in the signal, all right.

And, that was not a problem before because I was changing it by small enough amounts and I was doing this with a shift key. You can still hear that is a problem but I was able to start talk over then hide the [inaudible 42: [42:25] 40] effect.

But, now that I have sensitized it you should hear it. All right. Well, that is this patch in gory detail. Yeah. I have not turn the recirculation back on since I have been doing this for a good reason which is that the recirculation made since when I had the delay time up to some large [inaudible 43: [42:41] 04] .

[43:05] So, the recirculation example was I have a thousand here and then I could do something like turn on the recirculation and then give it a couple of, say an oscillate. OK. So, now I can just say, "Ooops". OK.

[43:36] This is now back to what I did at the very beginning of the class which is just the recirculating delay network where you do not want to be putting things through in a continuous way which is what I have been doing, what I have been describing the way this thing access a filter.

[43:56] You can actually think of this is a filter, but it would be a really good filter because its frequency response would be infinite at any frequency except one that was very carefully chosen to be notched out.

[44:14] So, we put that in another way. For instance, if I just put DC into it, if I just put a signal that had a constant value of one. We just add on to itself then it would continue doing that forever and eventually, we will have an arbitrarily large number coming out. All right.

[44:35] So, this is how to do range of operation where you really will think of it as a filter. This is using it in as something else. I am not sure what. OK. So, delay lines are usable as filters but they are delay effects which would be unstable if you left them in place for any amount of time and so you cannot regard them as a thing which you can do in a continuous way and time and therefore, you probably should not use them as filters but as other things. Questions about this? All right.

Audience: [45:09] How would you remove the [inaudible 45:11] noise from the...

Professor: [45:12] How would you remove the zipper noise? I have been saving that for a little later because there are couple of ways you can do it any way that you want. Yeah?

Audience: [inaudible 45:26] [45:25] .

Professor: [45:39] You can do it. In fact, you have to do it to be able to do the homework.

[laughter]

Professor: [45:47] So, that's the formula. Yeah. So, for instance, if I wanted to do middle C...

Audience: [inaudible 45:54] [45:53]

Professor: [45:59] Well, yeah, and I have been resisting calling up the expert tilda object which lies just a typed out formulas which is of course a great thing to be able to do but there is syntax which I have not yet found a thing that is really been unavoidable. But, how would you compute...OK, so one hertz should correspond to a thousand. Alright?

[46:20] Two hertz should correspond to 500 and so on like that. So, the general formula is this thing should be a thousand divided by the frequency in hertz; and to divide a thousand by something, so it is...this is worth knowing how to do.

[46:39] So, here is a number. We are going to convert it to frequency, MIDI to frequency; and then, that's going to give us a nice number again which I will look at. But, now what I want to do is take a thousand divided by that.

[46:54] So, to do that, OK. So, I needed a thousand down here which is a message because I do not want that value to change; and then I want to divide that by this number. Oh wait! I am sorry. I need an object. I am sorry.

[47:15] I will take a thousand divided by this, but we have to then bang the thousand after we put this number in and so, we need a trigger, yup, trigger, bang, float and I usually space this like this when I am doing this.

[47:36] I think that is decently readable and now, we do this! Destroy the coherence of our patch; and now, I say, "middle C, please" and it says, yeah you want 3.8 to two milliseconds and theoretically, now, if we play the noise to this with these two gains the same. Yeah. That should be this pitch. Middle C.

[sound]

Professor: [48:10] All right. One of the things I should warn you bad about this; this is abstruse and weird. There is a minimum amount of delay that you can get in through this circulating network for a technical reason which is the PD does everything in blocks of 64 audio samples just to save computation.

[48:28] As a result of which, since the delay write needs the delay reads output to be able to write into it, that output is a block and so, the minimum amount of delay which you could have in the slope is one block worth.

[48:43] And, without trying to explain that and you better than I have already, what I am going to do is to show you that, oops everything is hunky-dory until you get up to a certain delay here.

[noise]

Professor: [49:01] And then, it stops. Stops to write it. I think, it stops to write at 1.45 and that 1.45 is the number of milliseconds in 64 samples of 44,100 sample, right? If you compute that. You do not have to compute that now. I hate that number a lot because, well, it is the link of the block of 64 samples of time.

That also is the numerical accuracy of the line object if you ask it to do something in a specific time. The time actually will be the nearest one of these which can be a limitation and if that is a limitation, go look up the V line [inaudible 49: [49:35] 50] which corrects for that. OK.

[49:57] OK. So, I did promise you I was not going to add up to this patch so I am going to do... because I am going to save this. What can I do? OK. So, I am going to use a send here. I do not want to hang. I don't know how I am going to deal with making this to be able a readable patch when I try to put it up on the web.

[laughter]

Professor: [50:45] All right. So, there is that. Now, the next topic is what if you want to have a recirculating delay and not have it last forever the way I had it last in this example and of course, it is easy.

[51:06] You just take it and multiply it by some game that is less than one each time around. But, even though that is easy, I want to show it you because it has interesting ramifications and there are actually things that you can think about it, all right?

[51:20] So, now I am going to save that, have this three delay re-circulate so, this is going to be four delay gain re-circulate. Sorry, long name. Here, what I am going to do...see, I am not sure what incoming signals we need. So, now, what I am going to do is I am going to make one where the assumption is that we are always recirculating.

[51:53] So, instead of having the recirculating be on and off, I will make it be a nice number box and in fact, I think what I want to do is have the number to be in one-hundredths for sanity sake. All right.

[52:13] And now that I have done that, so, I am going to increase the...well, actually, let us be in middle C still. And now, I am going to say "noise please." It is going in there. So, we can listen to it.

[noise]

Professor: [52:28] OK, now all you hear there is unfiltered white noise because the noise is going in. I am going to make an improvement to the patch and you know, listen to the output of this adder. Yeah. OK. And, now, we still hear it. Yeah.

[noise]

Professor: [52:59] So, now, we are putting noise and we are not putting the oscillator or the ADC just noise and the noise is getting multiplied by zero as it gets red and then re-circulate it, so we hear nothing but the original noise. As I turn this value up, I get more and more recirculation which gives me a more and more nearly pure tone.

[53:31] OK. I am going to stay away from a hundred for now. OK. All right. So, now, to go back to, not a thousand at this time. How about 150? OK. Now, I am going to go back to talking into it. So, now, I am talking into thing and you hear just my voice with no delay and now, I turn out the recirculation.

[sound]

Professor: [53:55] And now, everything that you hear has several copies. OK. And now, this is the thing that you heard [inaudible 54:00] push my [inaudible 54:05] .

[sound]

[sound] [54:06]

Audience: [inaudible 54:08] [54:28]

Professor: [54:30] Is that it? Yeah. I did not just turn the thing up.

[sound]

Professor: [54:36] But, I am afraid to turn it up, so that is no longer true because in some way, we are going to get feedback and then it's going to be bad because I will be over there. Actually, I have a switch, but I am not going to [inaudible 54:45] , OK.

[54:48] So, hear these delays and now, what is happening right now is that each delay is 63 percent as loud as the previous one. Oh, this is this. "Hey, you guys! Do not listen to that."

[laughter]

Professor: [55:03] Yeah. OK, so the bigger I make this, the longer the sound stays around until I foolishly push this pass to a hundred.

[sound]

Professor: [55:15] And, you do not want to leave it like that. That is an unstable filter; alright? Filters can be unstable and what happens when you have an unstable filters? Of course, in everything, it is just, "waakrraak," OK.

[55:30] So, you can do it. That is mathematically possible, but it might not be want you want really. All right. So, this now is making a recirculating delay because the thing is about it is very feedback prone.

[55:47] It is used it with caution. But, also it is a bit of a special effect and it is way overused. I do not think about it is. This is close to what is...what you do it from an artificial reverberation.

[56:03] So, if you wanted to make me sound like I was speaking in a church room, musicals, or something like that, you would do something like this except that you would want the echoes to be a lot closer to each other in time like this. "Hello."

[sound]

Professor: [56:20] And then, you have to turn the recirculation off. Then you get something a little bit unexpected [inaudible 56:28] If you do this, theoretically, you are not getting any direct signal out of the speaker. Doesn't really work, you could at least hook it up. OK. So, now, it is not a nice cathedral reverb sound at all, it's got a pitch.

[56:48] So, the reason I dropped the delay time into something short. Right now, it is 13 milliseconds. The reason I dropped that was so that the echoes would be close to each other because when this was a larger value in milliseconds. I am going to drop this a little bit now. Then, you just heard...

[echo]

Professor: [57:08] ...then you heard a bunch of echo like this. All right? So, this is not a nice reverb because you know, you play a [inaudible 57:15] for that. OK. Now they'll come together, all right? OK. But, if I decide to try to make that echo denser, then I can get...

[sound]

Professor: [57:29] ...down below the magic value of 30 and I start getting pitches again.

[sound]

Professor: [57:36] All right. In fact, at this point, I can just dial this thing up like this, "Hello, this is [inaudible 57:41] "

[laughter]

Professor: [57:49] Now, all I am doing is filtering. All right. OK. So, there is something that I have not told you about this. OK. So, it was filtering. It was perfectly filtering before I put the recirculation in. But, you noticed that when I put the recirculation, then the filtering got a lot stronger. All right.

[58:07] That is the loose way of saying it, right? What really happened was I replaced the non-recirculating filter with a different filter which is recirculating which has the property that can have a very, very sharp resonance that is which you will perceive as pitches. And, this, this is the Karplus-Strong technique that I referenced in the homework at the beginning of the class. This is a thing which no matter what you put into it, outcomes...

[sound]

Professor: [58:38] A thing that is happening at the pitch that you dialed up here. All right. Oh, and furthermore, what you do that is reflected in a tamber chamber what comes out is a different kind of impulse that is going in and you get different tambers going out..

[sound]

Professor: [58:55] So, a little bit like what happens on a stringed instrument when you pluck it. All right. Those of you who play guitar, when you pluck the string, the sound of the tone is pretty much the sound of the pluck dying out. Of course, the higher frequency is [inaudible 59:10] lower, so that is not quite true but, the moment that you get to really control the tamber of a guitar or piano string that just anything gets hit because after that it's just ringing.

[59:21] It is just doing it's own thing after that. And, this is a very, well; it is a fairly close imitation of that process. It is at least a conceptual imitation of it where you put the signal on the string. It went down the string. It comes back up and get it again and every time it goes by, you get it again, right?

So, a guitar, [inaudible 59: [59:38] 39] or a stringed instrument in some sense, you could think out as doing a recirculating delay line; and that turns out only to be in approximation. But, it is good enough for approximation to capture certain aspects. Here, try as you might it is going to be hard to make this actually sound like a guitar...

[sound]

Professor: [60:01] ...and it sounds like...It does not sound like a computer trying to be a guitar. If you give it something that has more high frequencies...then, you can almost get, make yourself believe that you are hearing a struck string of some sort like a string being hit without mallet of some sort.

[60:22] And, furthermore, if you could...I am not going to build a patch to do this right now; but, if you could turn this noise on and off very, very rapidly so that you make a burst of noise, then you would get a very sharp signal that would sound in the same way that that FM tones sounds like a clarinet.

[60:46] That would sound like a harpsichord. OK. And in fact, that is exactly what I did in my patch here which is the homework demonstration patch which I had still have up. Let us see unless I could get rid of it. So let's go get it again. So, this thing.

[harpsichord music]

Professor: [61:20] All right. So, the homework is simply to take this principle and build it into nice polyphonic instruments so that it can do something reminiscent of homework six, two weeks ago. And of course, you will be tired of that stupid random melody, so you can make something more fun. Yeah?

Audience: [61:38] With this process, is this kind of like a basis for vocorder?

Professor: [61:45] It sounds a lot like a vocorder. In particular, well, the thing it makes a sound like a vocorder is when you put your voice into it, alright. It is really, you can almost think of it as an alternative to vocorder. All right. I will not do it anymore.

Audience: [62:02] Is that [inaudible 62:03] .

Professor: [62:04] Sure enough. So, why do I hear anything now? That is weird. OK. Let us turn DSP off. "Hello."

Audience: [62:13] Is there a previous switch on your port?

Professor: [62:19] There might be. I am not going to look for it now. I think there is probably a monitor switch down there just turned on, all right. Sorry about that. All right. So let us ignore that for now and what I will do is just start sending the ADC to this and now we get, "Hello."

Yeah. So now, [inaudible 62: [62:36] 39] . Well, OK. First of, a vocorder usually lets you put the sound in that this thing is forcing you just to be pitch of 60, all right. So, vocorders send more powerful thing than this in some ways in a lot of important ways.

Another thing is that if you compare this to some of the standard vocorder sound, the vocorder sound [inaudible 63: [62:58] 03] more local than time. So, this does not respond real fast. In particular, it does shut up real fast when you stop talking.

[63:16] So everything gets squashed out although there is a period of time which a good vocorder would not do to. So, you could, you know, this is almost, so this is a cheap wannabe vocorder. Any questions? About that? Yeah.

Audience: [63:35] Right now, you have [inaudible 63:37] how would you.

Professor: [63:46] Oh, it will be easy. You will just...

Audience: [63:49] It is just like [inaudible 63:49] ADC.

Professor: [63:51] You get another level control, yeah and then you get the ADC and just send it straight out like that and then, this would be the so called dry signal in audio process be the white signal.

Audience: [64:04] Can you put them together [inaudible 64:05]

Professor: [64:08] You could; so you could add the ADC into this and then, yeah.

Audience: [inaudible 64:13] .

Professor: [64:17] Yeah. You could do that. So, it is more a question of just how you wanted to define or how you want the interphase to be, all right. Because basically, the voice is down to two different levels or another. All right.

So, the thing that I want to tell you about the...I am realizing, 15 minutes. Do I really want to explain why the frequency responses like this or do I want to not touch that with a [inaudible 65: [64:33] 55] .

[64:58] OK. I am going to tell you how you would find out and I am going to avoid going into the gory details for the simple reason that it might be useful to come back to this in two weeks' time and so it would be nice to have been got started and then, anyway, if I never managed to follow up on it, then you could at least see me to where find more about it.

[65:23] So, this is now our two or three theory. So, let us go over here. Book. So, we are now on chapter seven of the book which is time, shifts and delays. Chapter eight of the book is filters and filters are really, as I have explained, they really just a psychological...they are point of view on time shifts.

[66:00] In other words, filters are made out of combining signals with time delay, copies of themselves including the possibility of recirculation which as you have seeing simply has the effect adding more and more copies on it. Yeah.

Basically, [inaudible 66: [66:16] 15] single copy to make the thing re-circulate. So, if you learn all about time delays, time shifting which is time delaying [inaudible] real time process. You know all about that then one of the things you can ask, what is the frequency response of the network and a good definition of a filter I think is, it is a delay network that was designed in order to give us particular frequency or phase response, usually frequency response.

[66:50] So, one thing that you want to know about in delays is how did you predict the frequency response and then when you are doing filtering, it is not just how you predict the frequency response but, I want a frequency response that acts like this. How would I set about designing a filter that has that?

[67:09] So, a filter design in some sense turns the question around. Whereas, in a delay, if you are just making delay network, you might just sort of ask, "What would the response be?" In the filter thing, you possibly response and look backwards to get a delay network that does not work. At least, that is the way I think of it. All right.

[67:31] So, delay networks have other things that you can talk about often besides the frequency response but I just want to talk about that for now in order to prepare for whatever little bits of filtering we are able to get into in last week class before we do Gem which is of course what everyone really wants to see.

[67:47] And, what I want to do is just talk about, I mean take this slightly out of order because I am going to motivate the jive about complex numbers by showing you first how you think about time shifts and how they change the phase of a thing. OK.

[68:04] So, just do the hand waving thing. Of course, shifting a signal in time such as delaying it, changes the phases of all the component sinusoids; and of course, I should not say of course they are because that is presupposing this. Do you think that you could think about a signal as being the sum of sinusoids in the first place?

[68:28] But, if you could, and if you have something linear time and variant which reduce although they were not making frequencies that we did not have before and so on like that, then you can save yes. I will just describe what this thing could do to a sinusoid and that we will describe what it does to anything. So, then we say.

[68:40] All right, we are just going to send sinusoids down our delay network and ask what happens then; and the answer is, you put it down delay and you get out a sinusoid of the same amplitude in a different phase.

[68:52] And, then if you, for instance, add that to a non-delay copy of the signal, then you will get phase cancellation or not depending on the relationship between the two phases you got.

[69:04] So, if you can predict the phase, then you can go to the math and work out what that response is going to be. So, how do you predict the phase? Well, it is easy. The frequency of a sinusoid is just how much the phase changes from one sample to the next.

[69:22] So, and if you do it in appropriate units, the phase change associated with the delay of say, D samples which is going to be D times the frequency, actually, minus D times the frequency of the sinusoid that you put in. Why minus? Because if you delay it, then that gives it delay and you listen to what it was earlier in time, so it is actually played backwards. But, if you think too hard about that, then you will get mixed up.

So, it is best to just sort of remember it's minus D times the frequency if you express the frequency as degrees or radiance per sample which is the good way to describe frequencies [inaudible 70: [70:00] 13] . OK.

[70:17] So, time shifts and phase changes. What is a time shift? A time shift is just, I give you a signal X and you give me a signal Y which is XD samples ago and N is just math. N is the number of one, two, three, four and so on. OK.

And then, what happens when you time shift a...oh! Now, I have to go back. OK. So, I am motivating complex numbers. If you have a real valued sinusoid, if it is ugly, it is cosine of omega N plus five. So, we call it cosine of omega N [inaudible 70: [70:36] 57] .

If I give you a sinusoid that contains only a positive frequency, it would get complex by its sinusoid with a simple formula so that if in that sample, so X of the N that is the N sample my sinusoid is some constant amplitude times a complex number of ways [inaudible 71: [71:07] 26] .

So, a sinusoid is an exponential an exponential sequence; and then, it is really easy to say what happened in the delay because if you take an exponential sequence and delay, you will have to do this at exponential sequences in high school, geometrics [inaudible 71: [71:28] 47] .

So, adding geometrics here, you consider what happens when you just delay it and then subtract it off from the original and just get a multiple of the original and [inaudible 71: [71:49] 56] .

[71:59] And so, this thing, clearly, if you substitute it, if you said Y of N is X of N minus D, in other words, you played it three times, all you would be doing is you would just be dividing by Z with the D. OK.

Now, this could be a possible misconception that you would have here is the Z is not going to be an ordinary number like a half because of course, a half is being not a good sinusoid. That is a [inaudible 72: [72:15] 27] . A good number to raise against to do that [inaudible 72:31] complex number which lives on the unit circle. So, now, I am going to go back and show you complex numbers. So, that comply to sinusoids. All right.

So, now, we can go back. So, what I did is I sneak preview the time shows in phase changes to show you that I wanted to have a nice exponential sequence so that I would know what time [inaudible 72: [72:42] 51] into it.

And, I was trying to compare that with what happens if you could just look with real numbers and then you say cosine of omega N and then when you say cosine of omega N minus D, then you have to use the cosine in some angle formula [inaudible 73: [72:54] 07] much faster.

[73:10] So, you will be much happier with what it is in complex land. Now, what I want to do is show you without going into the everything else. Here is what Z is, alright. So, this is the complex plane. The complex plane is the thing which you could brought up if you got out of pre-cal.

[73:38] This is the real axis. This is the imaginary axis. Complex numbers in general are had a real part and imaginary part which is put somewhere on the plane because you can see coordinates and so, instead of a number line in real line, we had a number of plane for the complex number.

And everything that is good about the real numbers [inaudible 74: [73:58] 02] . You can multiply and add and all the usual [inaudible 74:08] . One, of course has a real part of one and imaginary part is zero and in general, if you make a number which lives on the unit circle, that is to say the real part and imaginary part, you have their squares [inaudible 74:24].

[74:27] OK. So, that is the circle which is maybe A2, B2 of one. OK. This is called a unit complex number and it turns out that if you take one of this unit complex numbers in square root, all you do is you get the angle further on the circle. So, numbers on the real line, here, if you start multiplying in and you start going in and out.

Numbers that are on the unit circle on the complex plane if you start multiplying them, they stay on the unit circle and all you do is to change their angle. So, complex numbers know all about trigonometry. They just do trig for you. Alright. That is how you [inaudible 75: [74:51] 04] .

[75:10] So, in particular, if I for instance, consider the sequence one, Z, Z2, Z3. I did not tell you this, but actually, this angle is the same as this angle or this arch is the same as that arch is the same of this arch and so on.

[75:23] So, if I gave you the sequence of numbers one, Z, Z2 and so on, you will be going at a constant rate around the unit circle and that if you only look at how that projected on the real axis, you would see a nice sinusoid. It will look like cosine omega T or cosine omega N if omega was this angle from here up to Z.

That is called an argument of Z if you like. But, you could just call it angle of Z, and that would be the frequencies in radiance for sample of your complex sinusoid. So, this picture says that one one, Z, Z2 [inaudible 76: [75:53] 07] is a sinusoid which happens to have zero phase at the beginning of time which is here.

[76:18] And, it also happens to have unit amplitude and now, if you take that and multiply it by some arbitrary complex number with capital A as an amplitude, the complex amplitude has not just a size but also a direction.

Then you would get a sequence of numbers, A, AZ, A Z2, that will be advancing at the same angle but they will have a different amplitude and different phase. So, these numbers here, A, AZ, A Z2 and so on. That is the most general form for a sinusoid. That is the output of a oscillator [inaudible 76: [76:34] 54] . And furthermore, it has this very simple mathematical form which you can do stuff with.

In particular, you can delay it because delays have been rotated by some multiple times the angle omega. OK. Furthermore, you can add to that. So, if you have one other amplitude, if the frequency of the two are the same, the one other amplitude [inaudible 77: [77:03] 21] you just add in D as complex numbers [inaudible 77:27] get and receive graphically what the amplitude phase will be at the sum goes to sinusoid.

So, that is all delay networks do. They delay things and they superimpose numbers, so you can add them up, and you know what delays do now? I just rotate and then [inaudible 77: [77:37] 46] vectors and now you have the two that you need to predict the frequency response and [inaudible 77:55] response without another any kind of delay network [inaudible 77:57] .

So, that is the F [inaudible 78: [78:03] 04] . If there is some time actually especially when you get into filters as such I want to try to show you how you would translate that into like a nice band pass but it is all here. This is what all the engineers all do when the design a [inaudible 78:22] . Yeah?

Audience: [78:30] So, it is like a low pass filter. I do not fully understand this description, but filters that you showed us before. This is kind of like a notch filter, notch one frequency and it is like the other one. How would a low pass filter [inaudible 78:51] delay for every sign with frequency in it?

Professor: [78:57] OK. So first of, it is [inaudible 78:59] whatever you see that repeats itself every so often depending on the period.

Audience: [79:10] Yeah, got it.

Professor: [79:11] The smaller the period of the delay line, the smaller the length of the delay line, the more [inaudible 79:15] tilda to make the delay line to run the sample. You would only get one hump and it reaches all the way from zero to [inaudible 79:21] .

Audience: [79:25] OK.

Professor: [79:26] And so, that is how you get a notch to repeat, but then, how do you get to move around? And that is a little bit harder; but you just think you have to shift [inaudible 79:34] frequency.

Audience: [79:35] And how do you get like a low pass and a high pass?

Professor: [79:38] Low pass [inaudible 79:39] . A low pas is nothing but [inaudible 79:40] had a response to zero at the first notch. So, that first notch of the [inaudible 79:48] goes down from two to zero [inaudible 79:53] low pass.

Audience: [79:55] And then you just suggest that [inaudible 79:56] .

Professor: [79:58] Then we change the recirculation to get a [inaudible 80:01] .

Audience: [80:04] Oh.

Professor: [inaudible 80:06] maybe... That is basically what happens. And then, when you want to move the peak off of zeros, then it has a resonance [inaudible 80:14] [80:06] .

MUS171_03_01

Man 1: [0:34] So, this is where we got last time, with the exception that I added some nice comments to try to make it clear what was going on. We got as far as to make a research relating delay network. And, I demonstrated that you could consider this either as a thing that does things in time or a thing that does filtering, which is to say changing the frequency content of a sound. In other words, making some frequencies loud and others less loud. [0:42] So, to be overly pedantic, I'll just go ahead and re-demonstrate that quickly. So, here's noise.

[noise]

Man 1: [1:02] And, here is noise recirculating, it's a recirculating delay. I'm going to recirculate the delay. Oh, the delay will be 10 milliseconds long, and I'll recirculate it, multiplying by some number that's less than one. [louder noise]

Man 1: [1:08] So, we're going to multiply now by 88 percent. And, now what we're going to hear is a tone. [loud buzzing noise]

Man 1: [1:36] And, that tone, if I checked it on a piano, which I don't have one of right here, it would be at 100 hertz because 10 millisecond period corresponds to 100 cycles per second. Or, to put it another way, anything that comes in to this thing is going to come out, and then come out again at 10 milliseconds later, and almost as loud. And then, again, 10 milliseconds yet later. [1:59] So, everything that comes in, it's going to come out, almost repeating itself, every 10 milliseconds, which will therefore sound like a thing that's at 100 hertz. And, that would be true for delayed times that go up to some 30 milliseconds, or terms that go down to 33 hertz.

[loud buzzing noise in background]

Man 1: [2:08] You can almost say [hums] down an octave there. Then, below that you can't really hear a pitch any more. [changing noise]

Man 1: [2:25] You just hear some kind of thing in time. So, now, getting rid of the noise and turning on the famous microphone. Is this going to work now? I didn't test this. Hello. [feedback from microphone]

Man 1: [2:46] Oh, yeah. So, now this is a nice research relating delay with voice coming in. And what you hear is echoes every 84 milliseconds, which is what? 12 times a second or something like that, which you hear as an amount of time [snaps fingers] and not as a frequency. [2:59] If I make those echoes be close to each other in time, like 20 milliseconds away from each other, then you no longer hear that as [snaps] as a series in time, but instead you hear it as a nice pitch. And now we get...

[noises from microphone]

Man 1: [indecipherable 03:04] [3:40] which is just me talking through a cone filter. A cone filter is just another word for a recirculating or, anyway, for a delay network that likes frequencies that are multiples of a certain fundamental frequency. In this case, this cone filter likes frequencies that are multiples of 50 because 50 hertz corresponds to 20 milliseconds here. It's called a cone filter because if you look at the frequency response of this thing, considered as a filter, there's a peak every 50 hertz, regularly. That looks like a cone of some sort. [4:14] Looking forward, this and better ways, or rather, elaborating this idea in the future, we'll be able to design arbitrary filters with desired frequency responses. But, right now all I'm doing is reinforcing the notion that this thing can be considered as a filter, which is a thing which will take any sinusoid and then give you a sinusoid out at the same frequency, but perhaps at a different amplitude. That's a filter. Or, you can consider it as just a delay network, which is a thing which makes echoes.

[4:34] Those two things are really the same thing, except that they're psychologically different and the parameters that you put into it might make it act more like the one thing or like the other, or seem like more like the one thing or the other. So, there's that. Let's not even worry about that any more.

[5:01] Now what I want to do... Oh, yes, I do want to say one thing before I leave here, which is that this is a linear time and variant network, as the engineers will call it. And, what that implies is that if I put a nice sinusoid in, at any given frequency, like 50 hertz, nah, 100 hertz, out will come a nice sinusoid of the same frequency that went in.

[tone plays]

Man 1: [5:25] There's no way that a recirculating delay network, or any kind of delay network that doesn't have time variant stuff in it, can take a sinusoid of any given frequency and put out a sinusoid of some other frequency. If you put a color of light into a filter, or into a prism, or any other optics like that, you get the same color light out, if it's monochromatic. [5:54] That's a thing which we count on because it makes it possible for... Well, your ears seem to like to segregate sounds that come in by frequency, and so, if something sort of leaves frequency to where they are, it doesn't make different frequencies out of incoming frequencies, then you can say that your ear might hear a very clear relationship between what goes in and what goes out. Maybe. I'm hopeful that that's true anyway.

[6:21] Now, what I want to do is get into some practical stuff that you can do with delay networks. The first thing that I want to comment on is this. Let's see. What I'm going to do is do a save as, and start all over again, because for this next example, I actually want to make a non-recirculating delay.

[6:42] Instead, what I want it to do is be something where I can change the delay time. So, we'll call the delay time change dot PD. There it is, and now we're going to make in non-recirculating which means I don't care about the gain anymore.

[6:58] As it turns out I don't care about listening to the original sound. I just want to take the original sound and throw it into the delay line. Let's rename the delay line. Just so you can have both patches open at once if you want.

[7:15] We don't need this pitch calculation anymore. Go away. Whatever that is down there get rid of it. OK, and then we're going to read with the delay time that's given. And then we're just going to listen to it. Like that.

[7:19] So am I doing anything stupid? Find out soon.

[8:09] And so I'll do the brutal thing of putting a sinusoid in. In fact I'll give it a nice 440 hertz sinusoid. And now we hear [tone starts] a 440. Very good. Actually... [tone fades out] Make that a little bit louder at the mixer so that I don't have to do anything funny. [tone starts] OK, and now we're going to change the delay time. OK, so a 440 hertz sinusoid comes in and you hear it 20 milliseconds late. You don't hear the fact that you heard it 20 milliseconds late. It's coming out as a perfectly clean 440 hertz sinusoid except that the phase is different. And that's just what that is.

[8:45] Now we start changing the delay time though and we get something else. [tone begins to stutter] Which is ugliness. And the reason for that ugliness is very simple. [tone stops] It's just that if you change the delay time, it's the same thing as if you were reading from a wave table and you suddenly changed the location of the wave table that you were reading from. That would make discontinuous change in the amplitude or in the signal. And you will hear that discontinuous change as a [tone starts] click. [tone stutters] All right. [tone stops]

[9:15] Just to throw out a warning, of course if I was listening to noise. [white noise starts] I could change this delay time and you can't hear the click. Because there's no correlation between one sample of white noise and the next sample anyway. So the fact that you changed the place that you listen to it from doesn't sound like anything different from how the noise sounds to start with.

[9:40] I'm telling you this because you're going to make networks. And they're going to sound great because you're going to make them with some noisy signal like an overdriven electric guitar. Something like that. And you're not going to know that you're actually doing this to your signal. [tone starts] [tone begins to stutter] Until [tone stops] someone with better ears or experienced ears than you points it out to you or something horrible like that.

[9:55] So test your stuff with sinusoids, which are the most punishing signal that you could possibly put through a thing. Even though it's the simplest signal too. So that you can tell whether your patch is clean or dirty. Oh yeah let me get rid of this comment which is superfluous and this one too.

Now. So what if you wanted to be able to make an application? What I'm going to do is take my incoming sound and I'm going to listen to it... oh, great I'll do it here for clarity's sake. I'll do this with a microphone now just to annoy you all. Come here. What we're going to do is take the incoming sound which is the microphone. I'm going to talk into the microphone and see if it's [voice begins being amplified] amplifying my voice correctly. OK great. Slight delay but it's OK. And now I'm going to give you a delay. A delayed copy of it. [amplified voice begins to be delayed] Hello. And now it's obviously far too late. So here's my voice and here's my voice [indecipherable 0: [10:53] 10:51] . [voice amplification and delay stops]

But now I'm going to say can I change this delay time continuously, sorry change this delay time with a mouse and get away with it. [voice amplification starts] So now I'm on a different delay. And well [indecipherable 0: [11:29] 11:06] . [voice amplification stops] Never mind I thought it was going to be more forgiving than that. But my voice is already low enough and dull enough that changing the delay time is very, very bad and nasty. So I didn't succeed in demonstrating something where you can change the delay time and think it was clean and in fact wasn't. [voice amplification starts] It must be this shhhhh. [voice amplifications stops] I can do that and you don't hear the problem.

[11:57] If you want to have a delay line where you can change the time, then you have to work. And the work that you have to do is really better described as PD lore than described as anything theoretical. So now we're going to enter into PD lore. Actually this is computer music lore, because if we were using some other software from PD this would also be necessary and you would do almost exactly the same thing. And the same thing is this.

[12:42] Before I do it right let me do it wrong another way. It's always fun to do things wrong. Let's make it change smoothly. I'm afraid I'm being repetitious here because I think I already showed you this for samples. But I'm going to take this thing and just line it. So we'll say pack the thing with 100 milliseconds of packing. And then I'll make a nice line. And then I'll put the del read on. Right. Sorry I'm really belaboring points here today but this is... [voice amplification starts] Belaboring point. There's a delay and now I'm going to say 'ahh' and change the delay. So 'ahhhhhh'. [voice amplification stops] Did not help.

[13:00] OK the reason that didn't help is because the line is not an audio signal it's only updating every 20 milliseconds. And so you're just hearing 50 problems a second instead of however many problems a second I was generating with the mouse. No better.

[13:48] Watch PD do something wrong just while I'm thinking of it. Really at this point, I got a nice error here. There's a signal coming out of this line, but del read doesn't want a signal, and so I got an error message here which you don't see very often. "Signal outlet connected to non-signal inlet, Ignored." What that means is this thing was lined without a tilda, which is a control object, and I was able to connect it, and then I changed it to something else that it wasn't able to connect, but it didn't have the heart to disconnect it because I might want to change it back. But, nonetheless, it's not working right now. This thing really should turn red and blink or something like that.

[14:07] Also, by the way, you don't see this all the time because usually the order in which you do this is you get the object built, and then you try to connect it, and then PD just won't let you connect it. That's probably the correct thing to do in a situation. Anyway, you saw that happen. This is not going to work either.

[14:59] Is there a version of delay read that I could plunk a signal into? The answer is yes. There's one called variable delay. This started out in a different language where this wasn't such an ugly acronym. Variable delay is a delay object whose input expects a signal instead of a control message and, as a consequence of that, it does two things that del read tilda does not do. The first thing is it's willing to change its delay time every single sample.

[15:45] Then there's another thing that immediately comes up which is that if you're going to be changing delay times from one to another in a continuous way, actually one sample of accuracy isn't enough to make the result clean. So variable delay has to interpolate the incoming samples to possibly simulate delays that are not integer number of samples. So del read tilda, the non-interpolating control message delay, will always give you a delay that's actually an integer number of samples, and you can get pretty close to whatever delay time you want, like within 20-ish microseconds, better than that maybe.

[16:18] But variable delay, VD tilda, will actually make a four point interpolation of the stuff that's in the delay line. The delay line's actually a storage area. It will go and find four points and make a four point interpolation among those points to try to guess what the sample ought to have been that is in between the samples and the delay line that corresponds to exactly the delay that you asked for.

[17:09] The advantage, of course, is that the delay time is exact or as exact as split point allows you to be. The disadvantage is that it costs more. There's more computation involved. Also, if you do that, that interpolation has its own frequency response which is not perfectly flat. So you will not get a signal whose spectrum is exactly the same. This thing, because of the interpolation, will drop off in high frequencies somewhat, and that's a bad thing which you can control by raising your sample rate, but which is always going to be there whenever you interpolate.

[17:32] However, going back to the good stuff, now what happens when I change the delay time is it does the right stuff. [speaking inflected through software] We have a delay going, and now we're going to start to change the delay. In fact, I'm going to say "ahh" and change the delay. Ahh. [inflected] Ahhh. Good.

[18:01] Now we have a wonderful production that can generate other pitches than the pitch that you put in. That contradicts what I said before except what I said before was you couldn't go changing anything in the network if you want sinusoids to come out of the same frequency. Now, I'm changing something -- it's no longer time invariant -- and as a result, that nice property of stuff that doesn't change in time is no longer there, and now we're making other frequencies.

[18:19] Well, what about that? Now that I've done that, I'm going to stop irritating everyone and go back to the sinusoid. Let's turn this down. Turn on the sinusoid.

[19:00] Here's the sinusoid. [tone sounds] It's 440 hertz for you, again. Then if I listen to the delayed copy of it, [tone] same pitch, but of course if I change the delay time, [tone] I'm changing the pitch. Well, that's cool. That should make you immediately think, "I can do all kinds of things with this. I can take someone who sings in a monotone and make them sing a melody or vice versa. I could take something that came out of melody and turn it into a monotone, things like that." Well, you sort of can. In fact, I'm going to work toward that.

[19:17] However, first off, it might be nice to have an idea about what that frequency is. In other words, how would you predict what that frequency should be? Why would you want to do that? So that you can get whatever frequency you want...

[19:51] Oh, frequency. What comes out is a transposition of what goes in, so as in a sampler, it's probably appropriate to talk about what kind of transposition you're getting. That's to say what kind of change in the frequency or relative change in the frequency. What is the frequency multiplied by? That's a transposition. To make that painfully obvious, I'm going to throw in a pair of oscillators, and you'll hear a nice interval which will be a fifth because I'm going to tune this one up to 660.

[20:23] Let's see. Do we hear a fifth? Yeah. And now when I start playing it with a delay when I start messing the pitches up you will still hear that the pitches are related to each other. It's always a fifth moving up and down in parallel. That's the same thing as saying, "Yeah, the frequency's all got multiplied by some constant or multiplied by some number rather than maybe added to some number like a ring modulation might have done to it or something like that.

[20:46] OK. Multiply it by what number is the next question. To answer that I have to do something a little bit more... What's the right word? ...a little bit more controlled than this. Right now I'm just sort of mousing willy nilly at this number box. But in fact what I should consider doing.

[21:11] Oh, you know what? Let's save this. Oh no. It's OK. We'll set delay time. And now we have a way of making delay times up here. You know what I'll call the delay time down here. And then I'm going to make some message boxes so I can do stuff. So I'm going to save for instance this jump to 100 and go up to...

[21:58] We'll go to delay time of zero. Well, I can't really get down to zero really. Am I going to tell you all this? I'm going to... All right. I'm just going to cheat. I'm just going to ignore the problem. I'm going to go down the delay time to zero and I'm going to gradually go up to a delay of one second. How long am I going to do it? Take a second to do it. Oops sorry. That needs to be a pair. All right. And now I will put in the nice sinusoid. So this is an alternative to that. So now I'm going to... This is the sinusoid. Oops sorry. Let's get rid of this one again.

[22:34] And now I'm going to start the delay line changing and you all know what you're going to hear, right? Uh, bad! A beautiful bad example. [laughs] That was not my plan. What did I just do? I just made the delay line grow at exactly the same length. Sorry shrink at exactly the same length that time was passing in such a way as I slow the thing down to stop it entirely. OK. This is wonderful but this is not what I really want to do.

[23:07] Let's go up to one half second and take one second to do it. All right. And now I'm going say. Now what you hear is for the period of one second it drops by an octave. And what if I wanted to... OK So now you've seen two examples. One is I was able to stop it altogether and the other is I was able to slow it down by a factor of two.

[23:50] Why did it slow down? Well, there are two moments in time here that might be appropriate to think of. One is the moment where the thing starts and one is the moment where we're in. I'm thinking about time in... I'm thinking about so-called "real time" that is the time at which the sound is coming out of the delay line. That's real time for us. A second real time passes while the delay time starts at nothing and goes up to 500. But this line is being asked to jump to zero and then to ramp to 500.

[24:55] In that second of time how much of that sinusoid do we hear? Well, at the beginning we hear the sinusoid when it comes out. Sorry what am I saying? At the outset you hear the sinusoid that is coming in at the same moment as you're listening to it. A second later you're listening to the sinusoid as it had been one half second earlier than that, which is to say only one half second after you started listening to it. So you succeeded in slowing the sinusoid down by a factor of two because you only heard the one half second of it that went into the delay line between the original time minus zero and the time a second later minus 500.

[25:26] Rather than trying to explain that better, I'll make another example which is this. Let's go to 100. Wait. What's a good number? Let's go to 333 milliseconds--one third of a second. The delay starts at zero and then it ends up at one-third of a second.

[25:56] So how much of the sinusoid do you hear? You hear the other two-thirds of the second of the sinusoid, which is to say there's one third of a second we haven't heard yet because at the end of the process the delay line is a third of a second long so we didn't hear the last third of a second of the sinusoid. You've only got the other two-thirds. So we all know what that is as an interval. It means going down to fifth.

[26:18] So now when I whack this it goes down to fifth. So now if keep on whacking this I could get it down to a fifth and I could get it to stay there, right? Sort of. While we're at it. Now we have a nice tool for changing pitches.

[27:02] So now we can listen. Let's see. I'll shut this up and I'll be me. Hello I'm talking. You'll hear me talking in fifth below the frequency that I'm speaking now. So I just made a nice wonderful object that transposes my voice down a musical fifth. Oh, yeah, I played you already what it sounded like when I ring-modulated my voice, which made it an inharmonic sound, usually, because there might be some weird interval between the voice I was speaking in and the frequency of the ring modulator, back when I was ring-modulating my voice. We saw similar things when we were doing frequency modulation two lectures ago.

[27:23] And now what you're getting is a thing which is different from that because it takes the voice and maintains the relationships between the partials that were in the voice, but moves them all down proportionally by the same amount, so that their relationship stays the same.

[27:35] So that the partials going in have frequencies with ratios of one to two to three to four to five and so on, and those ratios are fixed, even though the frequencies of the partials are being multiplied each by two-thirds.

[27:50] This is a favorite trick of Laurie Anderson's, if you've seen her perform. However, there's a little bit of a problem here, [sound] because if you listen, try to understand what I'm saying here.

[28:25] First off there are clicks all the time because [sound] I'm having to change the delay time constantly. Oh, why don't I just never stop? I want the thing to go down a musical fifth, and I want it to last forever. So, we'll say, I don't know, we'll go on for a million milliseconds, which is 20 minutes. And we'll do that over three million milliseconds, which is an hour.

[28:41] And now I can talk for the ... better make the delay time really long now, right? Like instead of five seconds, maybe ... I don't know how long to make this before I run out of memory, but let's just live dangerously.

[29:01] So, now we have five million samples of delay. Oh, it hates me. It's reaching for five million samples within memory right now. Is it going to succeed, or am I going to have to give up? I hear my disk drive. [laughs] It did it.

[29:56] OK, so we now have a delay line that has five million samples in it, [sound - echo] and you hear me only on the second delay line, because I'm only using whatever I asked for. But now, I can start giving you a lecture wouldn't last 80 minutes, but wouldn't last four-thirds of 80 minutes, if I'm computing right, which is longer than we have in this room.

[laughs]

[30:07] Or, to put it another way, well duh, the delay time is getting longer and longer and longer, and now just for fun, hello. We'll just give ourselves a bomb that will wake us up in a few seconds.

[30:45] This is not a good way to do fifth-shift if you want to regard that as a real-time process. Right. How would you ... OK, so maybe we should go back to the other thing and be continually resetting the delay time to smaller values, [sound] but then of course the smaller values ... but then you couldn't do it without clicking. Right. So what would you do. The answer could be, you change it, but you shut it up while you're changing it. And then you let it start off again.

And now, let me try to explain this better. OK, so, going back to ... oh yes, and I'm going to do this two different ways, too, as I do a little bit too much of. Oh, you know what, I can get rid of this [indecipherable 0: [31:07] 30:59] . I don't like what this is doing to my disk drive. So, I'm going to go back to using a reasonable amount of memory here.

[31:38] OK. And you didn't see this, so I'm just going to erase this from the record. Right. I'm going to lose this idea. What we're going to do instead is make the delay line get quiet, change to zero, and then get louder again and then start changing. Let's see, let me even make a simpler example than this. OK. So, what I'm going to do is I'm going to ... OK. So, I'm going to show you two different things.

[32:05] OK, so first off, this was cool, and we're going to save it, and then we're going to do a Save As, and we're going to go back to regular old ... How about three delayed time change, and then I'm going to make more ... going to look ridiculous, file name. I'm going to use an envelope generator to change the delay time cleanly.

[32:37] So, now we're going to go back to del read, for del read. For, let's see. For simplicity's sake. So, we no longer have the right to put this line tilde in here, and it's complaining to us, but I'm going to get in with this. Now, I'm going to try to ... Yeah. OK, that's good. Let's get the numbered box out of here.

[33:00] OK, now we're back at the situation where ... Oh, go away. [tone] We can't change the delay time without making the noises. So, what if I wanted to change the delay time, but not have the bad noise?

[33:22] The answer is we mute the sound and then once the sound is good and muted we changed the delay time and then we unmute the sound. OK. So to do that we're going to have to multiply it by a ramp generator in order to mute it.

[34:14] My disk drive is still is still churning after that five million point delay line. My computer has indigestion right now. Oh right. And now this line of course we know how to turn it off. We throw it a message that says go to zero and take some amount of time. I don't know how long--maybe 10 milliseconds. That again is a value that you're going to have to find depending on what kind of signal you're throwing in. And then we have a nice thing to turn it on. OK. This is an on/off switch. See if this works so far. So there is sound. [sound] OK. It is a delight right now.

[34:43] So what we're going to do is every time this thing changes, we want to first say "zero" and then change it and then when we change it at the same time after its quiet we can then ramp the amplitude back up, all right? So what will happen is after a delay of a second... So let's get a nice delay object out. Sorry after a delay of...

[35:18] OK. So what I want to do is I want to make does this. [sound] It mutes the thing, changes the delay time to whatever you want and then turns it back on, right? So whenever the number comes in we're not going to change the delay time at all right away. What we're going to do is we're going to...let's see... we're going to send a...just... Sorry this is... I'm going to be a little bit pedantic here.

[35:55] I'm going to send a bang off to this nice shut-up button. Now I have this wonderful network which has a property that when I try to change the delay time it just mutes it and it stays off forever, right? We don't have everything built yet. And then after a delay of ten... So we'll bang this delay of ten and then we'll turn it back on. And now we have a wonderful patch that whenever we change the delay time it just turns the thing off and turns it back on which you can hear a little bit.

[sound]

[36:40] The only problem with this is that first off I can't really ramp very nicely with it because it has to mute and unmute it very, very quickly. It sounds ugly. But I can still change it like that. It's yea OK. As long as I'm putting something complicated through we get away with that. But it didn't actually change the delay time. Nobody's talking to the del read and of course I can't just send the delay time in right away because at the time I'm sending the delay in, it hasn't succeeded in muting yet. Sent this message to start muting but I really need this thing to come in ten milliseconds later after this thing has shut up.

[37:13] So let's take the signal and store it and then when the delay ten is done is when we send the new value of the signal delay. All right this is going to need a little bit of cleaning up before it is really palatable. But now we have something where we can change the delay time and it's smooth.

[37:54] So now for instance and to prove you that the delay time is actually changing I can now use it as real time thing. [sound] So now you hear this nice delay. And now the delay goes away. And now the delay becomes a second. Oops. And now the delay is a second. And now it's very short. So now I can change delay times on my voice, and it won't make that ugly sound. But at the same time. So this is a good thing. This is a good way if you want to make yourself a delay effect to be able to change a delay time and not have people complain at you because it sounded ugly.

[38:18] Now to go back, the previous patch actually got the variable delay object out and it's showing off Doppler shift. And then I was saying, "Oh it would be cool if you could use that Doppler shift thing to make a pitch shifter. That's to say a thing where I could sing in at some pitch now it becomes a continuous singing at other pitch without having a delay time that was gradually either growing or shrinking.

[38:40] So let's apply this principle to a variable delay line. So the variable delay is changing all the time. You can change the delay just fine without having badness. But the thing that causes badness is when you cause the delay time to change discontinuously that you had to do periodically.

[39:18] So let's see. Now what I need next is actually closer to this patch from the previous one. So I'll start with this one. OK now we're going to make a prototype of proto pitch shifter. That's number two, is that number three? That's number four now. We're being productive today. We might actually get up to five or six patches.

[39:42] OK so now what we're going to do is go back to variable delay land. That means we need to drive it with a nice line tilda object. So this is no longer just going to bash discontinuously changing value into a float. Instead we have a nice line tilda. And now what we're going to do...

[40:12] The patch that I'm now going to make is for pedagogical purposes. I would not be likely to use this. I'm going to show you how to do this better. So I have to tell you is I'm going to do something deliberately sort of OK, but this is going to be replaced by something substantially better in a few minutes, OK. So, the not quite so great thing is this, we have this nice message box, for instance I had zero and I grew up to 333 in a second.

[40:38] And this had the property that... so I'm just going to check this and make sure I'm still where I was. So here we are. Now we have a thing that takes my words dashes it down a fifth, musical fifth. Oh, yeah. I have to remember by the way to tell you how to compute these numbers to do your own intervals because not every note is a fifth, right?

[41:27] And now what we're going do is, well everything was cool except that going back to the sinusoid again to demonstrate this. You hear a click every time you... well you might hear a click depending on the bass every time you reset this thing. So that's bad, but of course we now know what to do about that which is we just... OK shut this off while we're... We just send off a bang to the... that's bad because that's going to make two bangs. Let's give ourselves a nice button. Yeah.

[42:12] So the button is going to mute the thing and it's going to set this new... No! It's not going to do that. It's going to happen after a delay, isn't it? Then the button is going to set the delay off. So now what's happening is... let's see if I can make this readable. It's not great. OK. Whenever I press the button what happens immediately is the line until it gets muted and then what happens after 10 milliseconds is I restart this process of changing the delay line continuously and I unmute the output.

[42:48] Now I have a bad but partly serviceable pitch shifter, pitch can shift us down as it goes into the fifth. Except of course if I forget to keep whacking the button eventually it goes back up. Either way, no matter what I put in here if I keep going in this way I will never do anything other than... well, is this true? I can transpose down in a clear way and get all sorts of intervals transposed downward.

[43:20] Oh yeah, I want two more down. There's a musical perfect third for you. How would I make the thing transpose up? You can't go down from zero, so rather than do this you would start at some value like 200 and wrap down to zero over time, so that the length of the delay line is decreasing instead of increasing.

[44:07] And then we get, when it gets going real old then we're going to get down a perfect minor third up. OK. So, why did it go? The delay time decreased, it decreased from 200 down to zero, so over the period of one second we quit hearing something that was 200 milliseconds old and gradually got to where we were hearing real time than at delay zero. So we heard 1.2 seconds worth of sinusoid in a mere one second of time which means we heard it at five... no, at six fifths at one and a fifth times the great.

[44:59] That's to say it's 1.2 seconds divided by one second worth of sinusoid that we heard over one second of time. We heard it 1.2 over one times too fast, 1.2 times normal speed. In general, over this amount of time, if this amount of time is one second, then the amount that we hear is this minus this plus one... oh, plus one second. So this is actually a fifth of a second. So in units, what we get is... the transposition is one plus, OK. One, because time is always moving forward, so if you do nothing at all then you get as much out as you put in.

[45:24] So it's one plus this minus this is the ratio by which the frequency went up. And if you give these three things name then you can make a formula. So, if the delay time is increasing, the pitch is going down if the delay time is decreasing the pitch is going up.

[45:50] By the way everyone will immediately use the word Doppler to describe this. This is a sort of Doppler shift. This is the Doppler shift that corresponds to, not the one that you'd normally hear which is you're sitting on a park bench and an ambulance goes by. That's the source moving and you are the listener and the delay time is the air. Simply the air carrying the sound from the signal source to you.

[46:14] A better metaphor here is there's something emitting a sound that's fixed and you're moving because you're changing the delay of which you're listening to it. But at the same time that's Doppler shift. You can hear it if you're running around on a bicycle or something like that and listen to someone who's stationary blowing a car horn, however that's not as often... that doesn't happen as often as you're hearing the horn stationary instead.

[46:30] OK. But it's Doppler shift anyways and this is the formula for Doppler shift too if you want. Should I tell you this? You can even have a Doppler shift that is so intense that it turns the sound around backwards.

[46:40] So imagine that someone was sitting here talking and what they were saying was so unpleasant that you were running away from that person at twice the speed of sound.

[47:11] You all know the speed of sound, right. It's well, it's a foot a millisecond. 1,000 feet per second, roughly speaking. So, you're high-tailing it 2,000 feet per second away from your professor, and as a result, you're hearing everything that the professor is saying, backwards. Because, you're actually ... the sound is sitting there in the air waiting for you to hear it, but you're traveling at twice the speed of the sound, so you're hearing the sound as it's getting further and further away, further and further down the delay line. Oh, we can even do that. Watch.

[48:02] I can make this delay line do that, by saying, we're going to start with no delay at all, and then we're going to run two seconds of time away in one second. Let's see if I can do this. And then I say anything at all, like fruitcake. And nothing comes out. [sound] Oh, and anyway, that. [sound] Fruitcake, and it didn't work, oh, I didn't put the comma in. Also, this is terrible, I'm making all sorts of distortion, because I'm being sloppy about the sound OK now let's see. I hear something. Test. Test. What? [Echo] Oh, I have speak before you start running away from me.

[49:09] So, you're going to hear it forward then backward, like this. Jelly beans. Oh, it didn't work. What's going on. Jelly beans. Jelly beans. [sound] Oh, and that was feedback. [echos] You remember that. So, that is a transposition factor of minus one. In other words, I'm running the sound a lot backward by changing ... by making the delay, I'm getting bigger, faster, than time is even ... So, that was a slight digression. And so anyway, let's go back to this thing. So to make a very cheap pitch shifter, we would say metronome, I don't know, chose some number of times a second we're going to do this. Maybe 10 times a second.

[50:06] And now, everything that I do will be transposed. [sound] OK. Transposer, pitch shifter. Very nice. This is kind of a bad pitch shifter, although it's been working. Let's make a slightly better pitch shifter by ... OK. So, it's a little bad that the thing is actually dropping out completely. But what you might wish to do is have two little lines and be cross-fading them, so that the sound is continuously working, even when you wish to change the delay line.

[50:22] And you can do that, but maybe it would be better before we do that to prepare the example by changing the way we're doing this anyway, in the following way. So, let's do a Save As.

[51:02] I'm going to switch now, instead of using a line tilda, to using a phaser to drive the delay time. Oh, yeah. And why, to drive the delay time ... oh yeah. And why, well, it's going to be ... you're going to be able to figure out why immediately when I show it to you. So, now, go away. We don't need that anymore. I'm going to quit doing this for now, because this is not really going to work for us.

[51:15] So, for right now, I'm just going to cheat and say multiply by one. That's to remind me that later on I'm going want to control the amplitude again. But meanwhile, I'm just going to drive the line tilde with a nice phaser. And it's going to have some nice frequency going in.

[51:33] And, oh yeah. And if I just throw this right in, I didn't even have a line tilde. If we just throw this right in, it's going to vary between zero and one, which is going to be interpreted as milliseconds down here, which is not so great. So, we're going to have to change the range to something reasonable.

And I will make that be a message box, too. And at this point, I should say that I don't actually know what order this delay right, and this delay read are occurring in. [indecipherable 0: [52:11] 51:54] name again. Just so, here. So, it would be appropriate at this point to add a couple of milliseconds because there could be a 64 sample delay engendered by the fact that this del write might happen before this del read.

[52:27] There's other PD lore that I'm going to avoid telling you about, how to force that into happen write. But now, I'm just going to add a nice delay, which ideally should be at least a couple of milliseconds. Maybe I'll take that away later and see if it hurts us.

[53:03] And now, let's see, we'll go once a second, and we'll have it vary by, let's say, 200 milliseconds, and then we will throw a nice sinusoid in there, which we listen to, and out comes [sound] you all know it. And out comes, you all know it. So, it's going to be a if I got it right, [sound] down a minor, a major third. Ooh. Bad example. This example, this was too good.

[53:36] I can change the delay discontinuously by one second, and because there are exactly 440 cycles in this thing in a second I got away without any discontinuities at all. Don't try this at home, or let me show you what could go wrong if you did. Let's try 440 and a half. Everything's going great. We're transposing, but there's a discontinuity every second when this phaser resets.

[54:26] OK, so I'll go back to 440 to pretend it's working nice and now I have a nice continuous control over the pitch shift and, in fact, the shift of the pitch I can compute, I think. OK, so this number is really a fifth if it's in seconds. It's in 200 milliseconds. Oh, this is interpreting as simply as milliseconds. So what's happening now is, so the phaser is happening in an amount of time, which is one over this, and it's happening and it's changing by this amount every time.

[54:50] So the transposition that comes out, the transposition factor is one if there's nothing happening at all. That's to say, when the phaser isn't moving at all. But if the phaser is moving the thing is being increased or decreased, well, it's being increased by one. Sorry, that's the one which is just oneness because time is passing.

[55:22] One minus the product of the phaser frequency and the phaser amplitude. So in this case it's one minus one fifth because this is in hertz and this is in milliseconds, so this is really 0.2, so this is one times 0.2, which is one fifth, and one minus one fifth is four fifths, which is down a major third. OK. Now you know how to compute no matter what this is what it should do. Let's see. [sound]

[56:10] So this is now one minus two fifths, which is three fifths, which is a major sixth. Oh yeah. Let's play the original here. OK. Alright. Yeah. So, transposition equals one minus product of phaser frequency and phaser amplitude and then you can work that backward any way you want. Now, let's get rid of the clicks. Oh, you don't hear the clicks because I fudged it here, but now if I change either this frequency or this.

[56:49] Oh, yeah. Oh, minus three means the phaser's phasing backwards, of course. If I change this, I might be changing the delay time by an amount that's not an integer number of cycles and as a result I'm getting clicks, which is kind of bad. So I do have a thing that can continuously change frequency, but it's clicking like all get out and that's not something that I want.

[57:15] Another thing about this is, of course, since the delay is ranging between zero and 121 or whatever this range is, there's going to be a delay between when you do something and when you hear it, which could be a problem. For instance, when I start loading my voice in or actually click into it. There's a delay there. So I've got the transposition OK, but I've got delays as well.

[57:46] The delay's actually varying between zero and 120 milliseconds. So that's OK. We can make this number be as small as you want. Let's make it 10 milliseconds. So here's the sinusoid again to test this. So now by the time I give it enough frequency to give it a decent transposition I've had to drive this value up high because this value is small and the transposition is controlled by this product.

[58:15] As a result, I have more and more problems, more and more discontinuities per second. So there's going to be a trade off in pitch shifting between the size of the delay I'm willing to tolerate and the speed of changing it that I'm willing to tolerate. In fact, it's going to become even clearer that this is a trade off when I fix it so that it doesn't click anymore.

[58:39] So to fix it so it doesn't click anymore, and this I think has already happened, what's a good way to take a nice phaser and turn it into a signal that will shut up right when the phaser jumps from one back down to zero, or in this case jumps from zero to one because I'm running it backwards? There are a lot of possible answers to that.

[59:12] One thing is you can design a parabola that goes from zero up and then back down to zero as you go from zero to one. The thing that people do most often that I see, anyway, is they use just the best quadrant they can find of the nice cosine function. Quadrant's the wrong word. So cosine, if you feed it zero you get one out. This is cosine of two pi of its input, or cosine of its input in cycles.

[59:32] So from minus a quarter to positive a quarter the cosine goes from zero back down to zero. That's one half cycle of the cosine and it's the one half cycle that's positive. There's another half cycle that's negative that comes right after that or before it. So how do I get that nice cycle out of this phaser, or half cycle out of this phaser?

[60:28] This is going from zero to one and I want to go from minus a quarter to a quarter. So, in general, if you want to change the range of something first you decide how big you want the range to be and multiply it by that. So I want the range to be a half big because it has to reach from minus a quarter to plus a quarter and then I have to subtract a quarter to get it in the right place. Let me say that more clearly. So, this varies from zero to one. Now it varies from zero to a half. And, now it varies from zero minus a quarter to a half minus a quarter. So, it goes from minus a quarter to plus a quarter. Now we just run this thing through it, and then we just multiply. Ooh. And, it doesn't let us connect, because PD is cool and that times ones said, I want control inputs there, which I don't any more. OK.

[60:55] Let's go back to something reasonable. 10 to the second hertz. Here's the sound going in. [tone plays] And, here's the sound going out. [different tone plays] Pretty good. Hmm, would be pretty good if it weren't changing its amplitude all the time.

Well, there are ways of dealing with this. Certainly the way that's easiest to describe to deal with this is the following: [61:14] Let's see. I'm going to take these things and get them out here. You'll see why in a second.

[61:27] Let's make another one of these things, and let's make it run out of phase from this one. So that, whenever this one is quiet, the other one is loud and vice versa.

[tones playing]

Man 1: [62:11] So, how do you do that? OK, so this is all good review stuff. How do you make a phaser that's a half cycle off from this phaser? This goes from zero to one, so we could always say, add a half. That means we go from a half up to one and a half. And, then if we wrap that... OK. Control. Then if we say wrap, then... OK. This warrants explanation. [62:45] So, this goes from zero to one. This goes from a half to one and a half. This wrap leaves the part that goes from a half up to one, but then the part that goes from one up to one and a half becomes a straight line segment goes from zero to a half. Draw this out on a piece of paper if you don't believe me, but the result is just line segments, the same as this. But, this thing changes value discontinuously whenever this thing crosses a half, because that's when this crosses one, and that's when the wrap changes its mind about what integer to subtract.

[63:14] So now, we've got ourselves a nice out of phase phaser. And, we can use our out of phase phaser. Let's see. I'll need to be compact here. Maybe I just don't have to be so compact. Let's move this stuff out of the way somehow. Don't need that any more, move this whole thing over. Now do I have room to have another one of these? Not quite yet.

[63:53] So, I'm just going to take this whole thing, including the multiplier, and make another copy of it running out of phase. And, I'm going to reuse these number boxes, like this, so that I'm multiplying and adding by the same numbers as before. I could clean this up, but don't know how. The right thing to do would be to do this and move things around, but maybe this is clearer for now.

[64:15] OK. So, now we have two transposers and one of them is jumping when the other one is being stable. The jumping one, of course, has been faded out in order to allow it to jump. So, one is always fading in while the other is fading out. Oh, yes. In listening to just the first one to start with.

[tone sounds]

[64:18] It's doing that for us. Let's try to make them faster. On the other one, if we listen to it, along with doing something similar... [tones playing] Whoops, except I have to repeat these things because I didn't put them in enough little objects here. And now, if you add them together...

[65:21] We get something that's not quite as variable in time. It's not perfect, but it's a little bit better. And now, again, we can continuously vary the pitch that's coming out. We can also drop the amount that it's changing the delay by. Of course, in that case, we have to move the phaser faster in order to get a fixed transition. OK, so let's go back to listening to the voice.

[65:52] So, this is close to the classic pitch shifting algorithm. [distorted voice] So, now we're shifting pitch and we have decently small delay and a reasonable transposition. And, we can go up, like this, and so on, like that. And we can make silly sounds by transposing up an octave or two, which I won't get into. It will sound like a chipmunk.

[66:07] So, this is a classic kind of a patch you could call a pitch shifter. People frequently call these harmonizers, but the word harmonizer is a brand name, so call it a pitch shifter if you want to be generic.

[66:59] Let me just show you where this shows up in the help browser because you might care how to compute appropriate numbers to stick in the phase and the delay line yourselves. Of course, I did all that work, and if I were a good didactic person I would think. You all do this work too but instead I'm just going to show you how you find the answer. You go down to the delay examples and you find the... there's a delay geo nine pitch shift, and just get this patch out. And this is a fabulous patch which... well anyways, I think it's fabulous. Which plays... [sound plays] plays a nice bell.

[sound]

[67:01] That's Johnathan Harveys bells on there and lets you transpose it. Any number of well OK... its repeated in half tones. OK now, let's turn this thing off so I can talk about it. OK. So here, the only thing that I've added to what I just showed you, here's the phaser and and the rap and all the good stuff that you just saw. The only... Yeah, I did exactly the same thing.

[68:20] Oh, I did this in the opposite order, sorry. Work it out, it's the same deal. The thing that I changed here was that I actually went to the trouble of figuring out what frequency you would give this phaser in order to get a transposition that you would specify in half tones, which is the western unit for pitch shift. So here, for instance if I say I want to go up seven half tones, that means I want to play it 1.5 times, well almost 1.5 times the normal speed.

[68:59] So a ratio of a fifth, a musical fifth is seven half tones. So A, A sharp, C, C#, D, D sharp, E, F, F sharp, G, seven. Seven half tones, that's these seven right here and that is a factor of roughly one and a half and how do you figure out what you should feed the phaser? Well, you're going to multiply the phaser by sum number which is here called the window, that's this number here. Let's see if I can show it to you in the old patch.

[69:36] Here's a help browser, go over here. So, I multiplied the phaser by this number which is the range of delay change and the product of this and that once the units were fixed, well one plus the product of this and this... one minus the product of this and this was the transposition as a factor. So, if I give you the transposition as a factor then you can do that algebra backward and that will tell you what you should feed the phaser as a frequency if you already also know this delay change, which here I'm calling the window size in milliseconds.

[70:30] So, if I set this to a 100 milliseconds say, that's a tenth of a second, then this number here is a tenth. Here I'm correcting to seconds from milliseconds, so here's the delay time in seconds... this is the delay change in seconds and if I wanted to change by this factor that means I have to add in frequency .498 to it, so I subtract one to get that by .498. And then because rising delay times transfers down I have to multiply by minus one or to put new way I have to subtract what I'm going to get here to get the transposition, so I'm going to need to run the phaser backwards to transpose up.

[71:09] So, in fact I have to transpose it by this number times 10 because this number is divided by this number in seconds. And what's this? This is taking half tones and changing it into a factor and this number is the logarithm to the base 12, no, logarithm to the base two of one twelfth. You can compute that. You can pull out your pocket calculator pretty much if your one of those to find that out for sure. I didn't do this in my head, I pulled out a calculator to get that number, OK?

[71:56] So this now is one of those that I just showed but packaged and engineered in such a way that you can get any desired transposition that you specify in half tones and as before, there is this wonderful trade off. Let's listen to it. [sounds begin] If I want a fixed... let's see, you can't change. I'm sorry. I'm going to just turn DSP on and off here to control this because I can't have the volume control on these things up simultaneously. But now, if I want to do the same thing... Oh right, when you hear this you hear a sort of... [plays sound] Well, you don't hear the real problem here.

[72:39] The problem here is going to be that since there are two delay lines that are different by one half of this window size, 50 milliseconds. Everything that happens and it happens twice 50 milliseconds apart which if we were putting voice through this it would be an annoying thing. However if you try to fix that by making this delay time itself smaller, then you can watch here it has to make the thing run faster to get the same transposition. I'll make a big 20, say and then it has to multiply this by five. And now we're getting the same transposition... [plays sound] That was interesting.

[73:06] That sounds modulated because this thing is changing so quickly that this envelope is raising the amplitude of it this many times a second. That's causing amplitude modulation of sound which we hear as frequency elisions. Which I guess in the bells since the bells in harmonic anyway that's not such a clear example.

[73:47] I can make out of my voice. Let's do that. Since after all it is computer I can tell it do anything I want to. Let's just do me instead of the bell now for a minute or two. And now let's see. We turn it on. And now you're listening to your [sound modification] professor transpose. And now if I make a nice big window you can get a decent clean sound. [sound] "Ahhh".... Not great. But anyway it's only changing one and a half hertz now. So it's not terribly messed-up. [sound] "Dohhh". Still pretty messed-up.

[74:16] But anyway also if I make a speech into it. Hello this is speech. You can hear that everything is replicated twice--the twentieth of a second apart. actually clicks is even better than speech. That's the clicks coming through because there are two copies of every click because I had to make two of these delay lines because they are crossfading in and out in order to cover for the fact that they had to be changed this continuously while this is going on.

[75:05] So then I make this delay time smaller but then this number has to get higher and then you get another problem. This is transposition but... I don't know why I'm getting away with this. Well this contradicts what I'm trying to tell you. I was able to get outrageously small delays here. [sound] "Ohhh"..... Yeah. But it's just in harmonic now although you can't really hear it. You can't really hear the inharmonious. I have to replay it with an instrumental sound and I don't have an instrument handy. But this would be a problematic setting too because this number is too small which is pushing this is pushing this frequency too high. It would be good.

[75:50] So either this number is too large or this number is too large and you will never get both of them simultaneously small with such a crazy transposition. People usually use pitch shifters with small transpositions like a half turn or a quarter turn or over a half turn or a whole turn like this. And then you can get both this number and this number decently small simultaneously like maybe this-ish. Those are almost kind of reasonable numbers to be feeding in to both of these inputs. But for larger transpositions you have to either get a ridiculous window size which is the maximum size of the delay or a ridiculous frequency of interchange and you will get gradually more and more eeriest sounds as you do this.

[76:36] All right. So in the interest of time I'm going to skip over the rest of the lore of making cool things out of delay lines although there are other cool things that you can make out of delay lines. I will just mention the existence of one of them because it's a good thing. You can make artificial reverberators out of delay lines. I do want to save this and I'm going to close that to get it out of here. I'm going to go back and get my help browser and just go get something that shows off a nice reverberator just to let you know it can be done.

[77:20] There are library reverberators even in PD Vanilla which you can get, but this one is the pedagogical reverberator which just shows how you make reverberations. Here's the test input again. Let's see. This patch is designed in such a way that you make this pitch move around and it shuts up when you stop moving the thing because... well you'll see why. Now we're going to reverberate it and we're going to hear reverberation. And reverberations sound like this.

You can guess how I might have done this. It's recirculating delay lines. But the standard recirculating delay line has a limitation in that you either make a delay line real short and you get frequency response [indecipherable 1: [77:44] 17:40] or you make a delay line really long and then you hear individual echoes.

[78:32] Here... Let's use me again. Here you don't have that trouble so much. So now let's see. Now you have me reverberating. But you don't so much hear... Sorry that's not going to be good. You don't so much hear individual delays although this is not a perfect one. As you just hear reverberation as to say sound that's sticking around after the sound like it would in a real room. And the way you do it, not to put too fine a point on it, is you have bunches of delay lines reading and writing in a complicated network which you have to think hard about.

[79:09] This is all explained in gory wonderful detail in the book why this thing works and why it's stable and how you would design it. All I'm going to do is just sort of say this exists. Go find out how to do it yourself, or if you just want reverb just say reverb two for instance and give it a nice big... I'll give it a nonsensical argument just to make the box big. Now you get a nice reverberator with inputs to control various things about it and there is a nice help thing on there.

[79:43] This is nice abstraction which I built just for making a reverberation in case you just want a reverberation. It's there for you to use. Revo one is experimental and strange. Rev three is higher quality than rev 2. There's a whole collection of well there is a collection of free reverberators that you can choose from and get the help window and check them out if you want to find out how to use it and the theory is in the book. I'm going to skip it because we have many other things to find out.

[80:02] Let's see. What do I want to hear? We have many other things to deal with than this and it's now trying to stop. Next time I have to start talking about filters which are the other point of view on delay lines where you in fact might find yourself designing delay lines with a specific frequency response in mind.

MUS171_03_08

Man: [0:08] Someone asked a really cogent question, which is grading weighting, what's the weight of the final versus all the homeworks. And the answer is the final is worth two homeworks. So out of 100%, the homeworks are all 10% and the final is 20%. [0:27] The grading, whatever you call it, is the way it will be graded is the same as your homeworks, except that of course you will only have something like three minutes a person to look at the finals, assuming that there are 53 of you, which is how many people signed up last I saw. Although all I see now 10? [laughs]

[0:49] But of course it's only three minutes after the beginning of the hour, so people will wander in.

[0:56] So I think what we're going to do is set up some kind of a tag team thing where we'll set up a couple of tables and allow someone to be presenting while someone else is fumbling with their laptop.

[1:12] And I'll ask this again later, but if you have a patch but don't have a laptop, then you'll want to use a laptop that I'll provide. But all of my laptops are running screwyOSs, so you're going to have to work on that, I'm not sure. Maybe Joe's got a laptop that's got a normal OS that you guys know about. Oh you do. That Joe didn't show up. The other Joe. Grown-up Joe.

[1:43] All right. I want to do one thing to finish off filters, although it being one thing doesn't make it necessarily short. I'm not sure how it's going to work. The first thing is an observation, which is that you can put delays on things. So filters are made out of delays.

[2:04] So I'm going to make observation number one just using delays, so you can see and hear it with long delays. That's to say audible ones. And then I'll go back and show you how it's done with filters proper. And then I will drag all of you to complex plane again one more time just to show you how this spins out in calculations, which you can actually calculate the frequency response of any filter in the world. Maybe.

OK. So the main observation I want to make is this: [2:30] I'll make the re-circulating delay loop, which is cool, and then I'll show you how to make the delays go away again. I didn't actually realize this worked until last year. Wait, I don't want to do this.

[2:48] So what we're going to do is we're going to take a signal to add a delay to it. OK, signal. Let's see. Let's just do the microphone for now.

So do we have a microphone going? Microphone, so that we can do things like have the delay's time. Is that amplifying, still? Because it shouldn't be. OK, that's me not knowing how to do it with my mixer. So anyway, I can drown it out by doing this.

[3:19] So microphone. Now, what I'm going to do is make a nice delay just like you've seen before. So what that is, is you say "delay," right? I'm just going to call it delay 1, and we can give this a nice long time because we'll maybe want to set the time that we read it from. And then we'll have a delay read, and we'll give it a nice delay time that we can hear, like a fifth of a second.

[3:55] And I want this thing to have a gain maybe of less than one, so I'm going to say... Sorry. All right.

[4:08] OK, so you're going to multiply it by some gain, and that's going to be your control. So we'll put down a nice number box. It might be good to have it in hundreds. So I'm going to say "divide by a hundred." That will be the gain of that, and we'll listen to it to see how it all works. OK?

[4:28] So now if this network does what I think it does, we'll hear me and then --it doesn't do anything. Here's the echo, and the echo has a volume that I can control. Let's do something reasonable. That's all right. Let's set this to zero. Hello? OK. So delay line.

[4:49] Now, how would you make this delay go away? Now why would you want this delay go away? To make the point. The point is some delay networks have inverses. That means that some filters have inverses because filters are delay networks. So let me show you how to invert the filter.

[5:09] And this is good because - and there's a reason this is good, but I can't... Oh, it turns out that I can tell you how to figure out the frequency response of this network real fast. Basically you already sort of know what it is because the resident frequency, it's going to be a comb filter, and so it will have peaks at multiples of five hertz in this particular example.

[5:34] And if you want to do the math, you can even figure out what shapes the peaks have as a function of this gain here. So if it's zero, it's flat. If it's a hundred, then it's notching completely up 2.5 hertz, and doubling five hertz, and it's doing something in between. And you can compute that. And then if you can compute that, you can compute the one of its inverse because it's just going to be one over it.

[5:56] First off, let me make the add explicit so we can talk about this as a single thing. So either it's a delayed network or it's a filter, depending on how you think of it. And now, we're going to run very quickly out of space, so I'm going to just sort of squeeze it. There.

All right. So what's the opposite of this? The opposite of this is the following thing: [6:22] We're going to take the filter again.

[6:30] Oh, yeah, so how would you get rid of that echo? Well, all you have to do is you would make the same echo --you'd have to use a different delay line to make the echo, of course. So name it two. And then we're just going to subtract them. So we'll take this thing and multiply it by minus one, and have that control of this.

[6:52] And then if I take this signal... So here's a signal with a delay on it. Let's see if this works.

So we're still talking and we're now making a delay. So now if I subtract that, that would make it go away, right? The answer is: [7:02] try again. So what happens is, we'll take this thing and we'll make an echo of it that is minus the same multiple of the original. And then we'll try it.

[7:39] And then we get... What I forgot was, there are two signals coming out of it, or two delayed copies coming out of here and they're separated by a fifth of a second. And then I subtract that off. But of course it subtracts not only the original but it subtracts the delayed copies off. So what I really get is signal minus the signal that is 400 milliseconds late. Oops, right?

[8:13] So how do I really get rid of it? The answer is I'll make it re-circulate. So I'll take the signal. The signal now has a delayed copy, and I'll subtract the delayed copy, but that will subtract another twice delayed copy, so we'll subtract it out and that will make it three times delayed copy, and I'll subtract that out and it will make it a four times delayed copy, and so on. And eventually I could get them all subtracted out and there'll be nothing there.

[8:37] And the easy way to do this is to make this delay line re-circulating. So what I'm going to do is, rather than just add the delayed signal into the original signal, I'll take the delayed signal, add it to the original signal, and feed it back. This is now a re-circulating delay.

[9:06] How can I prove that? Let's see if this is actually working as a re-circulating delay by listening to it alone.

[9:16] OK, we have clarity problems here. So there's the non-re-circulating delay, here's the re-circulating delay. If I did it right. And I'm going to listen to it and make sure it really is a re-circulating delay. By the way, it's got a negative feedback coefficient.

[9:35] All right. Re-circulating delay. Now if I take that re-circulating delay and apply it to this delay, so I took out the original signal and now I'm putting in a signal and itself delayed times 86%, now we're listening to both delays and we've got rid of the delay.

[10:05] One caveat about this that will immediately have occurred to you, of course, if I made this gain more than one, then to get rid of it I would need a re-circulating gain delay with a gain more than one, and that would be unstable. As a result, if I made this gain more than one, at least this approach to finding the inverse filter is going to fail. All right? But maybe we should not worry so much about that.

[10:38] The next thing is, so now I have a delayed network that is just giving us the same thing, and slight observation is we could make this gain negative, and it's the same signals. Now what we got is the re-circulating delay that has a gain of 74% and the non-re-circulating delay is subtracting a copy of it. This is actually easier to think about than the other case. And again, as you hear it, I got rid of the delay that we had before.

[11:17] Now I've told you this, although I don't think I've really emphasized this. Linear time in variant things like this can mute. So let's get this up here. What I'm going to do now is switch the order of the two delay lines. So now what we're going to have is, the original signal will go into this delay line, and it has 74% feedback, and now we have the re-circulated version, OK?

[12:00] And now I'm going to take that and throw that into this network up here - the non-re-circulating one. In fact, to save our sanity, maybe I should make them agree spatially with what I'm doing. So let's get this thing.

[12:30] So now we have a re-circulating delay, like you just heard. And then we have a non-re-circulating delay that we will put this into and then we will listen together. And now --hello? Why is it delaying? What am I doing wrong? Interesting.

[13:10] I have no explanation as to why this is canceling out. Just to be sure that I'm not going crazy... So I'm adding - I'm sending this thing into this delay line. Oh, that's not the output of the delay. Sorry, I did this wrong. I wanted this thing here. This is the output of the re-circulating delay. Now, we have nothing again. No delay. OK?

[13:53] Now a quick analysis of the situation. Sorry, I don't know how to make this neat. This is maybe easier to understand than the other one. So now what I've done is almost miraculous. I have this messy re-circulating delay network, which you put instantaneous sound and out comes a thing that lasts forever, right? And here is a nice thing that cuts that infinite delay, of trying to delay this out, entirely canceling it out.

[14:27] Why did that happen? That happened because, one way of thinking about it is when you study geometric series in high school, they taught you that all you have to do to figure out some of the geometric series is you take the thing and multiply it by that number and that makes all the terms match up to all the terms but the first one, and then you subtract it and they all cancel out.

[14:49] The exact same thing happens here. The result of this network is a train of delays, each one is 77% as loud as the previous one, or is having 77% of the amplitude of the previous one. And then, of course, if you apply a delay that takes the original sequence and subtracts 77% of it, the data on the first one cancels out the second one. And that delay of a second one cancels out the third one, and the third one cancels out the fourth one, and so on.

[15:20] Or to put it another way, this re-circulating delay makes a train of echoes, each of which is dying out exponentially. Take that whole thing and delay it, the same delay time and multiply it by minus 77%, then you cancel out perfectly the infinite train of efforts. And the only reason you should believe this is because you just saw me pulling off in a patch.

[15:48] You might think that this means that now you whenever you have a recording and it's done in a bad space, all you have to do is make the inverse of that space. Right? So rooms are basically delay lines in a sense. So whenever you talk in a room, when you put a microphone somewhere and you hear just a bunch of delayed copies of the sound in a very hand-wavy way of describing acoustics, right?

[16:13] So all you have to do is make the network cancel out all those delays and you could zero out the effect of any kind of room acoustic that you wanted to and start over. So you made a nice recording of someone, but they were playing in a horrible acoustic space, just take the space out, and then you can apply the other kind of treatment to it that you want from the original raw signal.

[16:37] Two things wrong with that. One is notice that this only worked... for re-circulating delays, I think it's very safe to say for any stable re-circulating delay, you can make the inverse of it and get rid of it. For non re-circulating delays, you can only invert the thing if the echo is softer than the original sound. When that spins out into a real acoustic situation, it turns into a statement that you don't actually know, or you don't know in advance that the inverse filter, whatever the room is, is a stable filter.

[17:19] A stable filter is, for instance, a re-circulating delay that has a gain of less than one. An unstable one is going to have a gain bigger than one. So there might not be a stable inverse to a real live situation.

[17:34] The other thing is that you can't do it because, first off, you can never measure perfectly what the response of the room is. Second off, the sound source is always moving. You almost can't get a sound source to stay completely still. And even if the sound source were perfectly still, the air temperature and air density in the room is constantly changing because there are air currents.

[17:58] And as a result, the reverberation, the response of the ring is always changing enough that if you canceled out the reverberation over any given instance of time, it wouldn't be good for any other given instant in time. So you'd be out cold.

[18:13] So people have been chasing this fool's dream of trying to get rid of -- or trying to inverse filter reverberant patterns for decades. Conveniently forgetting, or inconveniently forgetting, that it's actually pretty well-nigh impossible to do. However, in this nice completely artificial, simple setup, you can do it completely.

[18:41] I told you that the reason that this was going to be interesting was because now we can make inverse filters. That's OK. But we can analyze the frequency response of a re-circulating filter if we can analyze the frequency response of the non-re-circulating filter.

[19:01] So to make that audible, let me get rid of the microphone, put in some noise -- but I now have to drop the delay time of the filter so this will be cleared. So I'll put a number into the del-read read objects to change their delayed times to something tiny like two milliseconds. Then we put noise in.

[19:36] And if I do it right, I get 92% re-circulation. Oh, I'm changing the wrong one, but so now we got a nice comb filter. And now I can take that filter and invert it and reconstruct the original sound. But what that also means is that this filter and this filter - I'm play now the original noise going into this filter like this.

[20:11] So here's the inverse filter, if you like. This filter is the inverse of this filter. This filter's kind of ugly-sounding, but it's the inverse of that one. Here's the inverse of this thing. All right?

[20:28] And I told you that that was going to be easy because we can analyze this rather easily, and it will take more work to analyze this one. Now, when we take engineering courses or other kinds of electrical and signal processing or signal analysis kind of courses - signals and systems - they will make you do algebra to analyze this signal or this system. And what I'm doing is trying to avoid the algebra by just sort of analyzing that out of thin air by making this claim about these two systems being inverses. All right.

[21:04] I should tell you a thing about this. This is a perfectly good filter. Here is another one that has the... OK, so I could make one that has the exact same frequency response as that one simply by exchanging the two delayed copies. So right now it's the original signal, and then it's minus 92% of the delayed copy. But I could take 92% of the original copy and then full blast the delayed copy and that would sound exactly the same.

[21:41] But that filter would not be invertible because the second echo is louder than the first one. So if I try to make a re-circulating filter to cancel the echo out, it would be unstable.

[21:53] So there are two different forms of this filter. One of which, the one I'm showing you here, it is invertible. And the other way that you could put it together, which is backwards sometimes, is not invertible. Or at least it's not invertible with a stable causal of a filter on it.

[22:12] Now how would you analyze the frequency response of this thing? If we say the gain is flat out one. Yeah, if the gain here is either one or its minus one, then we can make a claim about what the frequency response should be pretty easily, because you're simply adding a - think of putting a cosine wave and then what comes out is a cosine wave and a self-delay, and the sum of the cosine wave and a delayed copy of that string which you can handle, not quite in your head, but it's pretty easy to do.

[22:54] However, when this gain is not one, it takes a little bit more work, so I'm going to draw you a picture to show you how you might think about that.

[23:04] As usual, as soon as the trig gets hard, the right thing to do is avoid dealing with the trig by jumping into the complex plane. So we will spend ten minutes looking at the complex plane, and we then will forget the complex plane forever, unless I can end up showing you the 48 transform next time. But maybe we won't have time for that.

[23:26] So here is the thing that I just showed you. Now, this is how chapter eight of the book. We've set all the delay times equal to one now because we're making filters. So the deal there is comb filters are filters that are perfectly all right, but if you make comb filter whose first resonant frequency is the full sample rate, then it doesn't act like a comb filter at all; it acts like something that does its thing once in the entire audible frequency range. Doesn't do the combing thing.

[24:04] So this is the way you do a filter. And here what I'm doing is making a number of capital Q for reasons that you don't see in the book. I had to name the variables carefully. And then you subtract that from the incoming signal. The reason that we're subtracting here is because as shown in the patch, it's by convention...

[24:26] Yeah, it's convention. By convention, you think of the re-circulating filter, this one, as having the positive coefficient. It doesn't have to, but I make its coefficient negative if I wanted to. Like that. All the same reason happen, we have that and this and those two things are still inverse filters.

[24:48] But by convention, when one usese the re-circulating coefficient as the variable that one names capital P or capital Q, depending on whether we're re-circulating or not. So as a result, since we're going to be talking about inverse filters, the inverse filter can be subtracting, that's to say taking some negative --anyway, subtracting some coefficient times its del-read.

[25:16] To tie this in with last time, of course this is a real value-to-comb filter, and one of the things that I showed you last time is how to make a re-circulating comb filter use feedback coefficients was a complex number.

[25:34] And there is, in fact, no reason that all that this number has to be real number; it could be complex number. I just made it real number so that I could make the network easily and show you this inverse property. But if we were using complex numbers, which means delays and pairs of delay lines, and then doing complex arithmetic on them, then all of the steps would still hold. And then we could give you things like making band pass and stop band filters.

[25:58] So now, in fact over here, if you're reading through chapter eight while I'm doing this, these are already complex numbers. The hypothetical reader of the book is completely bathed in complex analysis by this time.

[26:14] Now, here then is how you analyze that. I'm going to skip the equations and just show you what you get. So this cube, here is a complex number, and this is the re-circulation of a -- if you think of a pair of filters that are inverse, this was either going to be the re-circulating coefficient of the inverse of the re-circulating filter or it's going to be just the coefficient of the non-re-circulating filter except it's going to be with the minus sign. So the minus sign is still up here.

[26:58] So what happens up here? So again, we imagine that we are going to put a sinusoid into the system. And sinusoid and the complex number land into the following thing. You choose a good number Z, a capital Z, which is on the unit circle. So here's the number Z.

[27:21] And the number Z encodes the frequency of the complex exponential. Or it encodes the frequency of a sinusoid. So now you think that the sinusoid is the points 1Z, Z2, Z3, Z4, Z5, and so on, forever.

[27:39] And of course, when we listen to this, we're just going to take the real parts. So we're going to project that onto the real axis and it's going to look like a cosine. But in truth, the real signal, the thing that's happening underneath is that there is complex number spinning around the unit circle. And the trick is that each new sample is simply Z times the previous one.

[28:02] So now we know how to talk about delaying that sample. Delaying a sample is simply dividing the signal by Z. Why dividing? Because if you have (1/Z) Z2, and if you delay it, you get (1/Z)Z2, and that's 1/Z times the signal that you have delayed. That's confusing, but it's important to remember that one over.

[28:26] 1/Z, by the way, is just this number down here, which is what you get when you take Z and reflect it around the unit circle, Z to the minus one power. So it's down here. All right?

[28:41] So now what we have is the original signal, which is the signal times one, and then we have the signal times Z-1 because we delayed it, and times Q because we multiplied it by Q and the network. So let's go back to the network and just check.

[29:04] So here is QZ-1. Here's Z-1, which is the delay, and here's Q. And we're going to subtract. Now this isn't even figurative, this is figurative when you're into electrical engineering, but this is real for us because we're thinking that we're putting this complex exponential signal just as sine, a complex sinusoid into this. So this really is multiplication by 1/Z or Z-1. And this really is multiplication by Q, and this is just subtraction.

[29:41] So here is the left side of the thing, and here is the right side of the thing, which is the signal multiplied by QZ-1. In other words, the signal is multiplied by one, and it's multiplied by QZ-1, then you subtract the two. So it's one minus QZ-1, which is to say it is a complex number which gets from here to here.

[30:10] So if you think of it as a vector, it's the short vector which starts here and points there. The way I drew it makes it look like that's a very small number. But the number, in fact, is in the range from zero to two. And this amplitude, this thing, 1-QZ-1, its size and its absolute value is the gain - the frequency response of the filter at the frequencies. So Q is a parameter of the filter. You can control it but you can treat it like a constant.

[30:42] Z is the thing which depends on the frequency that's going in. In fact, what we think of the signal perhaps is it consists of many or even infinitude of sinusoidal components all going in with different value of the Z.

[30:55] So Q is fixed and Z is everything in process. And the easy way to think about that is not to think about how QZ-1 changes, but to multiply the whole thing by Z to get this picture. So now we have Z, which encodes the frequency of the sinusoid, and here's Q, which is the coefficient.

[31:18] And now as we imagine looking at values of Z that could range all the way around the unit circle, you see there's an area of the unit circle where the value that is close to Q, and those are areas where the gain of the filter was low. It's small. And around here on the other side, the gains for the filter are large.

[31:41] And the gain of the filter in fact is nothing but the absolute value of this complex number, that's to say the length to that signal. Furthermore, if you want to think about it, you can also get the phase response of it, it's the angle of this thing. So if you care about phases -- which people will eventually make you care about, but you don't have to worry about that yet -- do you know how to get the phase out of this diagram?

Audience Member: [32:07] So is Q fixed on there, even if it's fixed in space there? Even as Z travels around and Z-Q will...

Man: [32:16] That's right. So Q is the filter parameter, it's the knob. Z is the frequency which you're thinking about going through the filter. And this is a way of thinking about for all possible values of Z, What does the filter do? And the answer is it multiplies it by that.

Audience Member: [32:30] So what is (Q)Z-1.

Man: [32:35] This. This part of the diagram explaining why this part of the diagram makes sense. So this is the real response of the filter. But this is harder to think about because this point is moving around. This point is not fixed. It's easier to think about the thing just rotating the whole thing about the Z so that this point is fixed and this point is moving, instead of having this point fixed and this point moving. Although you could think about it differently. This is the way engineers think about it. [33:07] Now, the punch line to this is... Wwell, there are two punch lines. First off, if I took a bunch of filters and put them in a series, only if I put them in series. Parallel would be a mess. But series. And if all of them had this form, then I could tell you what the frequency response of the whole mess was, because you would simply multiply all the frequency responses for each stage in turn.

[33:34] So now we can analyze the behavior of complicated filters using the behavior of simple filters. Or to put it another way, that could give us way of of designing complicated filters. Because we might want to design that filter with several stages in order to accomplish something or other. Of course, we're not really going to get there, but that's the thing that people think about.

The other cool thing is this: [33:55] going back to the beginning of the lecture, this is non-re-circulating filters. But of course I've told you and showed you that re-circulating filters can be thought of as the inverse of non-re-circulating filters.

[34:11] So here's the non-re-circulating filter. And the re-circulating filter, which is the inverse of that, is this filter. I have to go forward one. I have to go forward two, sorry. Sorry, I didn't have a diagram. Never mind that.

[34:37] You know what it should look like. It should look like this, except that you're re-circulating the output of the delay line to the input and we're multiplying by Q and adding at the input of the delay lines, instead of subtracting at the output.

[34:53] Or to put it another way, we can go back to the patch. It realizes that this is the re-circulating filter where we are delaying, we're multiplying by something, and then we're adding. This is the non-re-circulating one where we're delaying or multiplying by a thing minus the thing and adding. In other words we're multiplying by something and subtracting.

[35:17] And those two things are inverses. And what that implies is that for the re-circulating filter, the frequency response is the length of this segment. And for a non-re-circulating filter, the frequency response is the length of this segment, which changes as Z changes, which includes the frequency. And for the re-circulating one, the inverse of it, the frequency response one over this.

[35:54] So the non-re-circulating filter has a notch right where Q points towards Z, and the re-circulating one has a peak where that happens. And what's the smallest value that this thing can give us? The shortest the segment can be when Q lies right after the Z, and so that is one minus the absolute value of Q. And that's the amount that Q fails to be right on the inner circle.

[36:23] And so what that shows us is that the re-circulating filter has a greater and greater gain; the closer Q gets to the unit circle, the greater the gain of the re-circulating filter is right at the choice of Z which lines up with Q.

[36:41] So the angle of Q chooses either the notch frequency of the non-re-circulating filter, or the resonant frequency of the re-circulating filter. And the absolute value of Q, this radius here called R, is going to control where the resonant frequency or notch frequency is.

[37:08] So with those two rules, and with a lot of messing around, you can design filters to leave any kind of specifications you want just by making assemblies of re-circulating and non-re-circulating filters with various coefficients.

[37:25] But in particular, you can do the thing that I showed you last time, which is you can make a re-circulating filter that resonates at any given frequency. And the way I described that last time was you know what the impulse response should be, it should look like a damped sinusoid.

[37:43] And I know how to make a damped sinusoid because I can just make this funny re-circulating complex filter. Just by pure thought, you can think about what the impulse response of that should be, which is ringing. And therefore that re-circulating delay line would act as a resonant filter.

[38:04] Now what I'm doing is showing you analytically is why that same re-circulating delay line acts as resonant filter. It's the same filter that I made last time, but I'm coming at it with a completely different line of reasoning now. This is the correct line of reasoning in the sense that now you can take this and can actually compute things, whereas what I showed you last time was just sort of a phenomenological explanation of what's going on. Like picking shells up on the beach or something.

[38:33] All right, questions about this?

[38:40] This is pretty much it for filter theory, I think, because I don't know how to show you much more without... After that, it's just details, but the details go on and on. For instance, how would you design now peaking filter, which is a filter that, around an area here has a little bump that either bumps up or bumps down, but around the rest of the circle it's basically one-ish?

[39:12] Well, the answer is you put a re-circulating filter and a non-re-circulating filter with their coefficients at the same angle, put them rather close to each other, and therefore anyone out here is about the same distance from both of them, and therefore the gain there is about one, because the two distances are almost the same and you're dividing, because one's re-circulating and one's not.

[39:36] But in this neighborhood, if you put one of them closer than the other, then you can make the thing have a positive or negative gain, depending on the ratio of the two distances. And that's your peaking filter.

[39:48] And reasoning like that, you can make it as complicated as you want but it's thoughts like that that make you get through all the elementary filters that people use everyday and compute with.

[40:02] So this is a much better way of thinking about it than is this sort of phenomenological, you know, make something that rings. Because you can actually doe reasoning on this complex plane and in a way that you can actually figure out that would make the filter do something that you want.

[40:22] Oh, and furthermore, I just told you how to make a peaking filter. It's a filter and it's inverse --it's a re-circulating and a non-re-circulating filter. Now if you set the two coefficients to be exactly the same, those filters are exact inverses. And so that's how you get away with putting all these filters in series in an audio chain.

[40:43] It's actually true that when you set the gain of a peaking filter to zero db, neither up nor down so that they pull on zero in exactly the same place -- sorry, the re-circulating and the non-re-circulating coefficients are exactly the same, the filter is immunitive. It is as if there were no filter there at all.

[41:01] And you can put a hundred of these things in series and it won't change the original signal. And so that's why you can actually have things like graphic equalizers that don't completely destroy your signals, because each one of those filters is actually nothing if we can zero it out.

[41:19] Question about this?

[41:24] So that's filter design. And study with Tom or Shlomo and learn all the deep stuff if you want to go further on this. Now what I'm going to do drop this entire line of inquiry and start talking about graphics.

Audience Member: [41:42] I'm sorry, directly related to this. [Inaudible question 41:43]

Man: [42:03] Yes, this is called the Z plan, just because one uses the letter Z to describe the complex number on the inner circle. Yeah, indeed. This is something that I wake up every morning wondering about. [42:21] And the good news is you didn't have to know any calculus to do this. You do have to do complex arithmetic, that's to say you have to understand about angles and magnitude of complex numbers, but you don't need calculus. Although, sometimes calculus helps very well. This is all high school mathematics. In fact, I've probably already said this, but this is the reason high school mathematics is interesting; so that you can do computer music.

[laughter]

Man: [42:52] You cannot do banking, even. Bankers don't do algebra or geometry, but computer musicians use that all the time, and this is exactly how it comes up. So people should teach you computer music in high school because that would make the mathematics interesting and make it stick with people. [43:10] But that will only matter if you become a high school mathematics teacher in your futures. So I'm going to save this. And I've pretty much finished making all the point that I want to make about this, so how we're going to go work with numbers.

[43:26] Now I have to tell you something about graphics programming, which is that I don't do a lot of it. So I'm not exactly the right person to teach you this. You'll see. I can't make great examples, because I'm not a person that does this kind of thing, and there are a lot of people who do.

[43:49] So I'm going to quit here. And I'm actually going to change directories. And furthermore, I'm going to run GEM, and not PD.

[44:04] First thing about GEM, you'd probably already see this when you start up, if you downloaded PD-extended as opposed to PD, then you have GEM. And in fact, when PD-extended starts up, you see all this kind of stuff. This is PD loading GEM. GEM is a library that is larger than PD, I believe, that PD reads and uses to define a whole collection of objects.

[44:37] OK. Describing that depends on... I haven't said something I didn't say, which is that PD, most of the objects that I've shown you have been built in on this. I think all of them have been. But if you type it, the name is something in PD that PD doesn't know about. And if it's not an abstraction that is to say that it's not going to patch.

[45:00] Now the other thing that it could be is a name of a file which is an object file which PD will load then to define a new object so you can make objects in PD that are C-code that fit inside boxes and do things that you want them to do instead of the things that I thought you might want to do with the built-in objects.

[45:19] GEM is the... So the name stands for the Graphics Environment for the Multimedia. This is by Mark Danks and is now managed by IOhannes ZmÃ¶lnig. And all these names are also people who worked on this.

[45:38] What GEM is, is a collection of something like 200 objects that pertain to graphics in some way or another. Graphics in GEM land is the thing which is called OpenGL. OpenGL it's the name of either a worldview or of an API, which regards computer graphics as being drawn polygons in space.

[46:12] That sounds really stupid until you find out that we can actually do with it, which is not drawing polygons in space, but it is drawing one polygon in space and pasting images onto it. So I'll show you how all this works.

[46:32] There are two points of view on computer graphics. One is the 3-D point of view which is everything has a model, the thing that you do if you make a dinosaur or something like that. A dinosaur is a bunch of triangles which are a bunch of vertices, a few points in space, and a bunch of segments between them that describe polygons.

[46:54] And then, if you want to render a nice dinosaur on your screens - it's called rendering --for each one of those triangles that is described, you ask the computer to paint a picture of that triangle, except of course if it's occluded by or wholly or partially occluded by some triangle that's part of it, then you would act out in front of it in that place instead of the one behind. So you divide it and all that kind of stuff.

[47:20] And this is exceedingly popular as a way of thinking of computer graphics, basically because of the influx of two industries.

One is Hollywood, which started making high-budget, three-dimensional rendered animated films like "Toy Story" or "Up."

[47:45] And the other thing is computer games. Which there are the interesting computer games and the stupid ones. The interesting ones are the ones that have a thought that could monitor them and the stupid ones are the ones where you're chasing things around and shooting at them.

[48:00] And if you want to make a computer game where you chase something around and shoot right at, it turns out that 3-D rendering is a very good way of making that appear realistic. And so basically the first-person shooter games, FPS games, and Hollywood movies are the basic reason to have three-dimensional graphics.

[48:24] Now, personally, turn your ears off, because I'm going to express anesthetic opinion here. I think both of those things are aesthetically bankrupt. But they're there and, furthermore, both of those things aree multi-billion dollar industries. So if you know how to do this kind of stuff, you can actually make money at it.

[48:41] Now GEM is OpenGL, which is the Open Graphics Library, which was originally called the Graphics Library. It's one of two or three competing APIs which the world uses for for doing 3-D rendering. However all of the 3-D rendering things ended up having to work with images as well.

[49:11] So why isn't all that stuff images? Well, because, as I've told you but then got lost trying to explain, there really are two points of view on making pictures with computers. One is draw a 3-D model, which is what OpenGL and these other things are designed to do.

[49:27] And the other is to think of it as painting on a screen. In other words, the screen consists of a bunch of pixels. It's flat. And what you're really doing is you're concerned with the color of each of the pixels. And then the other tools that are of interest are video cameras, because video cameras make images, they don't make 3-D models.

[49:46] And tools like compositing, which is to say taking one image and painting onto it with another image, which could by, the way be the image of a blob made by a paintbrush, so that you could regard painting as a compositing operation where you put bits of one image, which is actually just paint, onto another image, which is the actual canvas that you're painting.

[50:14] So OpenGL has a huge facility for doing image processing in this second sense, which is called texture mapping. Why? Because the only reason for doing images if you're back in chasing dinosaurs around in some computer game, the reason for having images is so that you could paint the images onto the skins of the dinosaurs to make them look scaly. That's called texture mapping.

[50:42] Say you don't really want to have your dinosaur to look like a bunch of green polygons running around. You want it to look like dinosaurs. So what you do is you hire artist to make a picture of what a dinosaurs might look like and you take that picture and you tile it onto the body of the dinosaur, all over it, in a way that doesn't make it obvious that it's repeating. And that's called texture mapping.

[51:05] So to make a 3-D picture... Is this stuff that you all know? Sort of? Maybe I'm repeating things here. So to make a 3-D image, really, you make a model but then you paint a texture on the model and texture on the image. And so there are image processing things which are basically there's 3-D texture mapping but they allow you also to do things like taking in parts of images and compositing them onto the other images and stuff like that.

[51:35] And those are the cool things that you can do with GEM, as it turns out, at least from my point of view.

[51:41] So now what I'm going to attempt to do, and this is dangerous because I don't really know what I'm doing, is show you the basic tools in GEM for making shapes and for mapping textures onto them.

[51:53] Just to maximize the embarrassment, I'm going to do this from nothing so that you can see everything that you have to do in order to make a copy, and then I'll fall back on simply paired ones because -

you'll see. Things don't just work the first time in GEM because there are more details to keep track of than there are in audio land.

[52:12] So what we're going to do is make a new window and before I do anything, I'm going to save it, and it's going to be 0-GEM-Try-This. And I'm going to call .PD even though maybe it should be called .GEM. It's going to be a PD patch but there are going to be a bunch of new objects which are GEM objects.

[52:34] So the first one is going to be this thing. Oops, sorry. The first thing, you'll need a window to put the graphics output in. And there's an object called GEMwin, whose purpose in life is to maintain a window that will be the image that PD makes.

[52:58] And it takes messages, and messages are something like dimension, which allows you to say how big you want the thing to be. So maybe 300x by 200y. I'm going to make it really small because my screen doesn't have much resolution. And that is just going to tell the GEM window how big a thing it should make, and there's a create message that makes it and there will be a destroy message that gets rid of it.

[53:34] There's other stuff here that I'm not telling you about. So tell it what dimensions and create a window. And ta-da, we have now a nice window, which is just being a window that doesn't know what it's doing. It's just sitting there being a space on the screen. It isn't even being managed by the window manager right now.

[53:56] Now, next thing to do is to be able to throw it messages to start it and stop it. And the obvious thing to do would be to put a toggle. So you can send it the number one, and the number one starts at rendering if you've said so.

[54:17] So now it's rendering, which means that it can redraw itself, and it's just redrawing black every time it gets redrawn now. And then if I turn it off, then it's not rendering anymore and that it's just being catatonic like it was before.

[54:34] Now me, when I'm making patches, I don't do this, I do this. It turns out that as soon as you create a window, it's time start rendering. And if you want to stop rendering you'd probably want to get rid of the window, too. So I alwasys just do that, creating on or off the string.

[54:57] Now, making objects. So I told everything's a polygon. So I'm going to get out of the mess and show you as an easy-to-manage polygon that won't do much for you, and then I'll show you a complicated polygon that will do more stuff for you, but it will take a lot of work. The easy one is a rectangle, and the complicated one is a triangle.

[55:19] Why? I'll show you. Rectangle is this. Say rectangle, and then we'll give it dimensions.

[55:34] And then we've to say -- and here's the thing that I always get in trouble trying to explain, but I always try to explain.

[55:43] Now what you need is to have, in some sense, the equivalent of the ACD-tilde object, the thing which just starts things rolling. So rectangle is an object which doesn't -- it looks like it has an output, but it doesn't.

[56:01] It's at the bottom of a chain of things that we will do, which is we will first off make us a source of messages that will go down this chain of objects. And eventually the chain will terminate in the rectangle, which is the command to draw something.

[56:17] The top of the thing is an artificial object which is called GEMhead. So we will have a GEMhead at the top everything, which let's call a GEMchain.

Now we have a rectangle. This is insulting your intelligence, but rectangles have dimensions. And so now we've got graphics. So now you can immediately tell that you could make patches that have abstractions that have thousands and millions of these things and start making [inaudible 56: [56:32] 54] and stuff like that. In fact, you can do anything with this now. Almost.

[56:58] OK. Now this is, however, a little bit stupid because the rectangle is white and here's some other things to do with that. Let me tell you something good to know. The dimensions in GEM roughly speaking, everything is in three dimensions, although we don't see it yet. And this rectangle is on the Z=0 plane. Z is this direction, Z is positive towards you and negative away from you. I'll find out if I'm wrong. This is Y and this is X.

[57:37] And at Z = 0, you can see X and Y, if I remember correctly, between positive and negative three. Now why positive and negative three? It's because that's the way the coordinate system is set up. It's a thing that you could change, but you think of that as the camera.

[57:59] In other words, there's a virtual camera looking at this scene, and that camera has a particular lens length and all that kind of stuff, but it is such that this ranges from plus to minus three at Z=0.

[58:17] Let's see if that's actually true. So I'll make the thing three. Three means three units Y so we'll make it six units Y before the screen.

[58:30] So there's that. Now, first thing that you might wish to do - well, there are a bunch --would be to change the color. Anyway, it's the thing you'll eventually wish to do. Now we start with all the 200 objects. I've shown you three objects, now we'll start with the other 197.

How about color: [58:58] RGB. I have to say very soon I'm going to have to go consulting the help files, because all these objects have lots of inlets and have lots of complicated things that you can do with them. But right now, I'm just going to fly with this and hope for the best.

[59:25] So if I remember correctly, RGB are actually these three inlets, and their values range from zero to one. I'll find out if this is right or not as soon as I start doing this. I'm wrong. Change that to one. Let me get rid of this. We're going to just not know what the last inlet does, and do this instead.

[59:57] So this is also confusing because the color is white, which is R, G, and B are all equal to one to start with. And so I had to actually send it zero to turn it off. The ends do not necessarily start out at zero in GEM. Although it almost always will in PD.

[60:18] So here's red, here's green - wait, no. Sorry. And of course blue. And now everyone knows this but take red and green and you get blue. Sort of yellow.

[60:43] So this is video colors rules, where colors add and subtract. That isn't yellow, is it?

[60:53] And exactly what color you get depends on your projector or your screen. In fact I have a radically different color here than what you're looking at. So there's all that.

[61:04] What is happening here is the following thing. Maybe you all can understand this, but this is a thing that I find very, very mysterious. The way GL thinks and the way GEM plays into GL. So GL is the API, or OpenGL is the API that GEM is talking to. API is an applications programming interface, it's just a bunch of function calls as far as we're concerned.

[61:32] The way the API thinks about life is, you don't just render things. So this is rendering a rectangle here, so you can render a rectangle. But before you render the rectangle, you're allowed to do all kinds of nonsense which are called transformations.

[61:50] Transformations are things like rotating or translating the object in space, changing its color. Now why is that a transformation? I don't know, but it is. And making it not rotate make it part-transparent. And worse yet, unbelievably wrong, but this is not Mark Danks' fault, this is OpenGL's design that's just weird, but another transformation of an object consists of saying what texture it's going to have mapped on it as it's rendered.

[62:26] So to put it another way, you go waltzing down this GEMchain until you get to the last thing, which is a command to draw something, which is a draw command. But you collect all sources and you try this along the way. And if you try this, this thing's like rotations, translations, color changes, transparency changes, and textures.

[62:53] So this right now is the color thing. And I believe it's true that if you do it again, it would simply replace the color with the new one. I shouldn't be doing this, but we'll see. Now this color is white. This color will simply overwhelm that color.

[63:23] So as I told you, we can translate or rotate. So how about translate XYZ? Sorry, this is painfully stupid of me. Now while we're here, if you want to do animations, the obvious way to think about doing animations is to make an object and then start flying it around. Don't do that. It just is stupid. This is a stupid animation.

[64:05] A good animation is representing motion by drawing things in sequence, which is not the same thing as drawing a thing and make it move around. But the people who have designed OpenGL were convinced that the right way to animate something would just be to make something and make it fly around. And so there are all these wonderful things in OpenGL for making things fly around.

[64:30] Next one, of course, we have translations. But of course, we're also going to have rotations. This one, I think, you can actually say rotate XYZ. No, you can't. Oh wait. Yes we can.

[65:06] So rotation. There are many ways that you could represent a rotation, but one possible way uses a three vector. By the way, it's a complete coincidence that the same number of color dimensions is the same thing as the number of spatial dimensions. That didn't have to be true. That's our eyes didn't actually choose the number three. But that's the number we have, right?

[65:32] So rotating about XYZ is the following thing. You specify a vector and the magnitude of the vector is going to be the amount of rotation. What units do we suppose OpenGL could propose for that? If colors are going to be from zero to one, the obvious thing to do is have the rotation be in degrees, right? No, but that's what they did.

[66:03] So, for instance, if you want to rotate the thing on the plane that you're looking at, what you're doing is you're rotating about the Z-axis. We're going to describe the vector normal to which you're rotating, and furthermore, the link to that vector is going to be how much you rotate.

[66:20] So here, we're rotating about a Z. And notice by the way something incredible in the image. This staircase thing is called aliasing, and that's bad. You can fix that, but I'm not going to show you this right now. The other thing is you can also rotate like other axes and now certainly get to see the fact that you're being three-dimensional.

[66:48] Now, this is a perspective drawing of the nice rectangle whose top edge is at 140 at whose bottom edge is pointing away from you. And compound rotations can be compound things. Now this is interesting for about 30 seconds and then it's profoundly boring.

[67:15] So next I'm going to make two of these, so you can see how GEM thinks that things should superpose.

[67:35] Let's see now what happens. This one, they're the same size. We're going to rotate this one. There you go. I have to tell you something about my computer. A pixel in this direction isn't the same size as the pixel in this direction. Anyway, I didn't tell you to make it a square window. That probably is what's going on right now, is that since my window isn't square -- in fact I want to make my window square because I'm going to be confusing here.

[68:17] So you never actually make a square window, but I'll make a square one so that you still only see the aspect ratio problem. So this shape should be the same exact shape and I think it's not, but I think that's because my aspect ratio is messed up. The aspect ratio is whether the pixels are square or rectangular.

[68:40] So now we have two things. Now, who said that the yellow rectangle is behind the white one? OK, so why is the yellow one behind the red one? Because I did this first. That's not good. It would be better to tell these GEMheads in what order they should be rendered.

[69:17] What's really happening is that the yellow rectangle is being rendered first and then the red rectangle is being rendered so that even though they are at exactly the same location, pretty much space, the red one shows one on top.

[69:31] You can actually specify that by putting numbers here which would be our priorities, which would be the order in which that the things are going to be rendered. But rather than worrying about that right now, I'm going to start translating again in the Z direction and translating the yellow one and move it toward us, and then away from us.

[69:54] You get two options. What is that? That is Z buffer is what you call this, where one thing is in front or behind another. O, to relate to the point probably, let's take this thing and rotate it to about X. Now we have an engineering drawing where we have this piece of metal going through a slot on this piece of metal or something like that.

[70:27] Now this is the way 3-D rendering works. You draw stuff and then this stuff cleans the other stuff. Now comes the risky part. Let's start putting textures on this thing. And the reason for wanting to do this is just so that we can actually start over the images, which is much more interesting than working with shapes in space. At least in my opinion.

[70:55] So to do that, this is really strange. It's not really stupid, but it's just strange. So I'm going to do yet another thing, which is to tell the thing that it has a texture. Now that is called pix--texture.

[71:23] And what's the texture going to be? What this thing does, fix textures that says whatever the image is, that's what you're going to texture map at. But what is the image? There's no way to specify that directly, and so what you do is just say part of the environment that we're having is there is a current image and the image is going to be set by pix--image, this object. And furthermore, this object takes a message. The message is going to be "open test1.jpg."

[72:16] So I'm going to do this and we're going to try to explain what it is. So pix--image is a thing which is memory, which contains an image which we can read the image in. Didn't work. How do I do this?

[72:42] We're going to go look and see. So we say pix--image open pix--texture. There's nothing wrong with this. So why do I get an error message? Oh, right! The error message goes "turn on the standard error."

[73:25] So we're going to set a tempo on that. And then do we get anything? OK, it's complete. There.

[73:46] OK, so let me go back and make this thing be white. So if you can see the image in all its glory. Here's the image. It's a stupid image, but you know, someone found this along the road parked and then I found it on the web somewhere. And now what's happening is this image is being texture-mapped onto that XY rectangle.

So the operation's done in the following way: [74:11] you have a fixed image, which is a buffer which contains the image. And fix--image's job is simply to say, "All right. If anyone needs an image, I've got an image and it's in this buffer." And then fix--texture's job is to say, "Has anyone got an image? If so, let's texture map it."

[74:37] And of course the person that's got the image is this one. So that becomes the image that gets texture-mapped. And that's the basic way that this thing gets back through. OK.

[74:49] Now images are cool, but where do images come from? Images come basically from three possible places that I'm aware of right now. The medium cool one is files, the really cool one is the camera, or your camera, so you can take your camera and make it show up on this thing.

[75:11] And then the uber-cool one is you can take the two images and composite them to the third image, so that you can actually work on various kinds of image synthesis, like you can take an image and punch it into a part of another image and blah, blah like that. I'm not going to have time to describe this in detail today, but I'll try to show you some of these examples next time.

[75:30] And finally, on some OSs, maybe all Oss now, I'm not sure, you can take whatever you just rendered, make than an image and save it, which is a way of being to paint stuff in and call it off screen into an image and then we can use that as raw material to make something else.

[75:49] So, with all that, you have a ready source of a bunch of images that we can use in recursive ways to build up cool visual effects.

Audience Member: [76:03] Can you mess around while you're using IOs and stufrf.

Man: [76:06] Yes. And furthermore, although I'm not ready to talk, well, I don't have time to talk about this because we have two minutes left. But furthermore, the two can be throwing information back and forth to each other. [76:17] So you can. I tried to get this demo working today and couldn't. Maybe on Thursday we can take the camera and point it at something and turn that into sound, like a wave form or a spectrum, or you can have a sound going into a microphone and have that effect and the image that's happening, make an image jump around when you have a signal coming in to the microphone.

[76:47] A lot of people have worked hard on coming up with cool and interesting ways of using this musically and visually. Some people call this visual music when you actually make designs that consist of both images and sound that are changing in time/.

[77:06] Although, of course, that's a music-centric name to give it. But the name "visual music" I think dates back almost a hundred years now. So it's not like GEM thought that term up.

[77:22] Yeah, so now you can all go out into clubs and make money because people will pay to dance to images jumping around the screen, until they suddenly realize that that's not something you really dance effectively to and then it'll go out of fashion and then you won't be able to make money this way any more.

[77:40] So more on this next time.

MUS171_03_10

Professor: [0:00] Finish up with Gem today. This is an example of taking audio and graphics and in some sense, bridging the divide between them. And this is a very good patch for you to load on your laptops while your professor drones away. [laughter] Basically, well, you can tell exactly what's happening. [0:21] So, as I was saying, Gem is all about polynomials, or sorry, polygons. And what you see here is a bunch of polygons, but so far, I've shown you two rectangles so this is a whole thing with a whole mess. So what I want to do is just sort of show you how you put together something very simple like this.

[0:41] And I got to say, this is not going to win you any prizes being able to do this kind of thing but the interesting stuff that I think that you can do with Gem, with geometric kind of modeling like this is things where you algorithmically generate hundreds or thousands or millions of polygons to make shapes or other kinds of collections of stuff in space.

[1:08] And, I don't have any good examples of that sitting here right now. So, I'm not going to try to develop one. Instead, I'm just going to show you this kind of very simple sort of generating example of OK, here's how you would use audio analysis to drive a picture and then, we'll just sort of leave Gem for future years and then I have five other topics in PD that I'm not going to have time to do properly, so I want to mention that they exist and give each one of them maybe 10ish minutes each.

[1:42] That will be plenty for today and then, we're going to be done for the quarter except of course for final projects.

[1:51] So, the plan for today is not just one thing. It's a bunch of things but each of them is only just kind of on the surface. One is, I'm going to finish up with Gem with this example. And then, the example actually, has an example or is an example using audio analysis and there are lots and lots of things that you can do with analyzing sounds.

[2:14] And so, I want to show you the basic tools that PD has available for doing audio analysis and some of the things that you can do with that. And then, there are other things that you might want to know about which I will tell you about as I get to them; but, basically, four other classes of things.

[2:33] Each of, well, I should say, these are just sort of quick topics. Net receive and net send is making network connections between computers. Read SF, write SF is spooling audio to and from the discs so that you can do things like make sound with PD and have the sound file afterward.

And, PD tilde is a trick for doing PD with multiprocessing. It's a way of having PD [indecipherable 02: [2:51] 56] PDs that can run on other processors. So, each of those things takes 15 minutes to describe and then they're done.

For re-analysis and re-synthesis, I'm going to just show you one very simple example of this, but this is a thing which you could easily study for months or years. In fact, it's true that, it's true that most people have to study this and think they learn it and then come back a year later and realize that there's actually another point of view on it that you wanted to know it from and so on for three or four generations before you really have enough points of view down that you really believe that you know [indecipherable 03: [3:02] 37] analysis and synthesis fluently.

[3:39] So, this is a thing that which I want to tell you the existence of, it will not start actually going on because it is just a big thing.

[3:46] OK. But, anyway, back to the Gem example. Alright, so what's happening? Every time I talk, the sound is going to the microphone. So, we're having to do two things. Really, we are, measuring the loudness of a sound.

[4:03] That's a technique which is old and it dates back to the, at least to the analog synthesize days, it's called envelope following and it's one of the three kinds of analysis that I am going to be showing you in some more detail next time, but at least now, what I'll do is show you how it works into this example. OK.

[4:20] So, envelope following. OK. This is kind of a mess. I like actually running sound files because if you have a recording of your favorite politician giving a speech, this is a great way to listen to that. So, what I am going to do is try to figure out where all the stuff is, not there. Alright. Come on. What am I doing wrong? Got that, got that. Audio in. Oh, PD works. Here we are. This is the real thing. Alright, so what does a bird consist of?

[5:00] OK. So, last but not the least is going to be PD sound where we are actually getting the loudness of the sound coming into the microphone. So, I can save that because for continuity sake, I will just talk about the graphics first even though the flow of information is from sound to graphics so, I'm going to be doing, going up stream in information flow here for a few minutes. OK?

[5:21] So, what would you do? Well, these things are all, well, these, I'll do this in some detail. This, these hundred rectangles are going to be the bird's body. So, part of the trick to drawing the bird is making this shape which is a very irregular shape that you can't make very simply out of polygons, so there is a thing for making hundred polygons that makes that shape that I'll show you. OK.

[5:53] Then, there are isolated things which are. Let's see. This thing is a rectangle, this twig that it's sitting on. The legs are I believe trapezoids although I have to go check. And, these eyes are actually hexagons. That's cheesy, but that's what I did and this beak is three triangles. There are two triangles for the top part of it and just one triangle for the bottom, alright.

And, those triangles are the only thing in the whole thing that's moving; and in fact, the only thing that's moving in this, are four points in space which are: [6:21] First off, the two sides of the beak which are [clapping noises 06:33] chosen at random whenever it finds a new attack. So that, basically, the width of the beak is sort of changing word by word except it's not working too great right now.

[6:45] And then the, you can tell that the bottom and the top of the beak, sorry, the two points in front of the beak which if there's no sound going at all are one point there.

[6:56] So, there are two points which lie on a vertical segment and they're some fixed point plus and minus the envelope value that is coming in from the incoming sound, right?

[7:08] So, these two are random, but are set off by attacks and sound and these two are the continuously changing, that's the whole deal, alright? OK.

[7:17] So, how do you do it? What I want to do is find some simple stuff first and then show you the complicated stuff, alright.

[7:29] So, here is simple stuff. This is, so, I told you that everything is triangles and then of course here, I'm making a four vertex polygon which is a quadrilateral. If you make quadrilaterals in Open GL or Gem, make sure that the four points are coplanar because it will do the wrong thing if you give it a skewed quadrilateral, that's a quadrilateral whose vertices are not planar.

[8:02] In this case, almost everywhere, the Z value, so you can all just look at my screen.

[laughter]

[Indecipherable 08:11] [8:08]

[8:09] Alright, so the basic yoga of Gem is you're going to make things to have planar polygons. The easiest way to make a planar polygon is to make it a triangle because there's no way you will ever find to make a triangle not be planar, alright?

[8:26] If you make something four or six points then, it's more of work, but, the easy way then to make a polygon to be planar is if you just set all the Z coefficients to the same value; then you know it's on a plane, which is Z equal constant.

[8:38] Otherwise, you might have to work harder. OK? Now, I'm just going to sort of talk into the air to tell you the hard part which actually is getting the body of the bird not the beak. The hard part is this. I went in and made a table which has two, actually, I made a, let's see.

[8:57] So, you have made arrays before. I might have shown you on one event or at one moment a graph that had two different arrays in it. So, I made a graph with two different arrays in it and I drew the arrays to make the outline of the bird's body. So, it's not, it's, let's see what's the right word, it's not, this wouldn't be true of any possible shape but the bird's body is actually designed in such a way that every cross section of it, every, anytime you intersect a vertical line with it, you just get a segment. You never get two different segments. In other words, the shape never does this.

[9:38] The shape is always just from one point to another vertically. And so, you can describe that fully by just making two functions versus just say the contour of the top of the bird and the contour of the bottom of the bird, alright? And, that was the only reason that I have the patience to stand and do this because I was able to think of a shape that this was true of, right. OK.

[9:58] So, then, what do you do? Well you make a whole bunch of rectangles. So, you saw although I didn't go into it, an abstraction that was called a Rect 10, there were ten of them at the left side of the screen.

[10:09] Each one of those things is making 10 rectangles. Oh, cool! Oh! Even better. Alright. Ladies and gentlemen, Brady Baker.

[applause]

Professor: [10:24] Alright. OK. So, where were we? Oh yes. So I was telling you about waving my hands in the air. OK. The body of the bird goes from here to here and here it is. It is PD tables and this is my own artistry. So, I went to draw these two lines and then how do you get those out? Well, you probably know, but, OK. So, this is stupid. How do you make a hundred things in PD if you don't want to draw? [10:55] But, if you want to make a hundred boxes, you make 10 copies of the abstraction. Each of which has 10 of the things.

[laughter]

Professor: [11:03] And if you want a thousand of them, you know where to go. [laughter]

Professor: [11:06] You know how to do that too now. OK. So, the basic deal here is, we're going to tell this rectangle, oh, dollar, one dollar two dollar, three that means that array 1, array two and 10, 20, 30, 40, blah, blah. OK. So, what is happening here is, this is going to be Rect array 1, array 2, 10, 0. And this will be same thing except 20, 0 and so on like that. [11:30] So, we go in here. So, now, dollar three is tens and dollar four is units and you're going to add the two of them to get a number which will range from 10 to 99 or something like that, I'm not sure how it works. And, what are we going to do? Well, we are going to go reading four points out of the table which are,sorry, let's get back and get the table.

[11:57] So, for each, so there, let's see. I said rectangles but really, the strips that we're drawing are trapezoids because the bird's top and bottom are not necessarily parallel, alright? And so, what we are going to do? We are going to drop four points, a planar polygon having four points in it and the four points are going to be picked up by looking at two adjacent points up here and two adjacent points down there. Alright.

[12:25] So then, how do we do that? First off, we get the data that we need which is tab reading the array one and array 2. Oh, yeah. So, here's, dollar one is array1, dollar two is array 2, dollar three is 10, dollar four is 0. Right?

[12:44] So, here's tab read array one and tab read array two and here is tab read array one and array 2, but there the locations that we're looking at are one point further. One point further than what?

OK. So, we are going to receive from somewhere upstairs a message which is just a sequence of bangs in time called Do It [name spelling 13: [12:56] 04] . Actually, every single time Do It [name spelling 13:05] gets banged the same thing comes out because nothing's changing here but maybe something will be changing later. We want to make the bird to do something funny, right?

[13:13] So, this is a good way to just sort of compute a shape in such a way that, for instance, for every single frame that we want to draw in Gem we might want to re-compute the shape.

[13:23] OK. So, this is, so bangs come in and now we compute what number we are and we do that by adding dollar three and dollar 4. Trigger bang, bang, get out dollar 4, get out dollar three and add them and then we take that number and throw it in five different places.

[13:39] Why do we need...Well, six different places if we count this. OK. So, we need to do six things. We look up four points out of the tables and we need to figure out the...so, that's four Y coordinates and then we need X coordinates and there will be two separate X coordinates which are the X location of the left hand side of the trapezoid and the X location to the right. So, it is X1, X2 and the Y1, Y2, Y3, Y4, right?

[14:03] So, I told you about getting the Y values out of the table. The X's are just, take the X value, whatever it is and fudge. So, in general, when you want to change something's range, you multiply by something and/or add something. In this case, I subtracted 50, so that it goes from minus 40 up to 50 I believe.

[14:26] I'm not sure about that, up to 49 and then, multiply it by something which is, well, divide by 20 which is to say getting it into Gem units which are from minus three to plus three if you're on the Z equals zero plane and that's just sort of knowledge that I have that you have to go from minus three to plus 3. That's the size of things in Gem unless you change it. OK.

[14:47] So here, this thing has a range of about 100, so when we divide by 20, it has a range of about five which is to say, most of the way across the screen which is what you see in the bird. OK? And, so, we just take the number and then we take it plus 2, plus 2, yeah, alright. Whatever.

[15:06] And...that's strange. Why wouldn't that be plus 1? So, the numbers that come in here are all the numbers from 10 to 109 I think actually in increments of one and what this is saying is that the rectangles are actually overlapping slightly because it's going from 10 to 12 and then from 11 to 13 and so on like that. Go figure. That's what I had to do to make it work right. Open GL. Alright. OK.

And then, here are points. Points are three coordinates apiece and [indecipherable 15: [15:37] 45] the Z coordinate is always 0 because we are flat on the Z equals zero plane and X and Y are just being computed, well the Y values are coming out of the tables, 1, 2, 3, four and the X values are coming out of these computations and those are the vertices of this trapezoid which is going to be a polygon with four points.

[16:03] And, the four points are specified using triples, packed triples each and so, here is point 1, 2, three and 4. You can make a polygon with as many points as you want but do make it planar. In fact, make it convex. Why? It says in the Open GL, you can only make convex polygons. OK.

[16:25] And then, meanwhile, now, what you saw last time was simple Gem examples where there was basically a Gem-head object, blah, blah, blah, blah, until finally you get down to a thing which you have to draw which is a polygon of some kind.

[16:40] And here, Gem head set the color to black which is that and then, draw a polygon. So, the Gem chain if you like that's to say the sequence of events which happens in this window every time it renders a frame, is head, set color and draw a polygon. And there are a hundred of these, right?

[17:02] So, that is this body; and now, I am just going to go fishing around and find the beak. I believe this is the beak down here and yeah, let's see. Can I scroll in in such a way as to make it easier to see this? Oh. OK. These are the eyes.

[laughter]

[17:40] I told you they were hexagons. I just did this in my head and figured out the points that are regular, the coordinates of a regular hexagon. If you studied trig in high school and you haven't forgotten it, you can do that. Alright? So these are two hexagons with centers, they should be getting added but...Oh, I see. I am using translate to move the eyes over to where you want them and in fact, just for testing sake, and this is by the way how I actually designed this.

[18:11] I fixed it so that the translate actually had a nice message box on it so I could figure out where I wanted the eyes to be and then when I got the eyes where I wanted them to be, I copied the numbers into it. Alright. Seat of the pants. OK.

[18:32] Now, the beak. That's the only part that's actually doing anything animated, right, and I have lost the beak. These are the two legs. Where is the beak? The beak is probably going to have to be on that sound, alright. Come here. Scroll. Oh, beak, duh. Beak, here's is a beak.

[laughter]

[18:59] Alright. This is where you start to realize that PD makes a lousy programming language, right? This, if you were a programmer would be two lines of code, but since we're in PD land, it's not two lines of code, it's a whole messy page of stream of consciousness, right.

[19:20] The good thing is you can actually program PD by stream of consciousness which you can't really do and see. The bad thing is of course then you have to explain it.

[laughter]

[19:26] That stream of consciousness which is what it is. OK. So, here's the deal. Beaks consist of three polygons and there should be a pair of two of them that are, that share vertices. Where are they? Drat. I've got to make this smaller, so I can start scrolling. There. Good enough. OK. There are the three polygons.

[19:55] Now, what you don't see is this number here is coming in from here and this is the number that goes a up "huh," when you make noise with the mike and goes down when you don't. Alright. So, that we're going to look at later. That is the envelope follower, alright.

[20:14] And notice again, I've messed around and tried to get this thing to have some reasonable, what's the right word, some reasonable range of values, alright. So, now, here what we have is, let's see, this is going to be the bottom triangle of the beak. I can tell because this number is going to be used in two different ways depending on where we're at. OK.

[20:44] So, what's happening here is, there's a floating point number which is going into the second which is the Y coordinate of this polygon. The X coordinate, that's to say the first values is 0 which means the beak is right in the middle of the window. Is that true? It's not true. Oh, there is probably a translate object. Oh, yeah, alright. Here we go. Here is translation again. This is me figuring out where to put the beak.

[laughter]

[21:10] This is great.

[laughter]

[21:13] OK. Let's put it back. Oh yeah, right. OK. So, we translate it. So, since we are using the translation, we can think that everything as being centered around 0 which is what's happening down here. So, all we're doing is we're packing something that has a Y value which, oh, which is the envelope value times minus one, but what's happening here?

Oh, I'm sorry. This is the other thing. This is the width of the beak which is this number, alright. OK. So, that is getting randomized every time there is an attack [clapping 21: [21:38] 50] which I will show you later. So, that is something called W for width and then the other thing is H for height and that's getting computed elsewhere.

[21:57] So, the height is the, OK. So, there are three polygons, sorry, there are three points of the polygon which are the left corner, the right corner and the bottom point of beak. So, here's the left corner which is at 0. Here's the right corner, ooh, sorry, that's the right corner at zero. Here's the left corner at 0 minus 1. So, there's point 1, weird, OK. And then, that number is changing when this changes. Interesting, alright.

[22:34] How can I get a good number there? There. Alright. I got the beak back. Alright. OK. So, I'm going to stop improvising now and just tell you that after a while, you'll make all sorts of mistakes and then at some point, you'll get all those points right and then you'll have a nice looking beak.

[22:58] OK. Now, what I do want to do is go back and show you where these numbers come from because that's the, that's the connection with the audio, and so those are numbers which are H. This is received height and received width and those numbers are being computed as a function of the sound. So, cool object.

[23:18] The envelope follower. This is a thing that I can sort to tell you what it does. It is taking the signal and squaring it and then putting it through a low pass filter. Yeah?

Audience: [23:30] Are they coming in from the mike into the laptop?

Professor: [23:32] Yeah. This is the mike. So, if I turn it off, now you are seeing the noise floor of my audio system and now you are seeing noise floor of the room? And now, you're seeing the noise of me talking which is a little bit louder than that. Alright. OK? [23:51] This is in decibels and reasonable dynamic range of something that's happening depending on what it is. It could be 30 to 60 decibels. So, when you're just sort of talking, it's, you know, it's varying like 20 or 30 decibels.

[24:10] If you have a trained singer singing something, it is going to be more like 60 dB or 60 decibels because they use dynamics a lot harder than regular people talking do. OK. So, what we're going to do is take that and figure out a good way to change its units into something useful.

[24:29] Useful units are things that range maybe from 0 to 1-ish which is what's going on here. How do you get it to range from 0 to 1? Well, you just basically fudge until something good happens; and, in this case, what the fudge consisted of, let's assume that the noise flooris going to be 30 dB.

[24:49] Why? Because it turns out that a wide variety of rooms the noise floor is somewhere in the 20s of dBs when you set a reasonable volume level. It's just a good number.

[25:00] OK. And so, what you do is you take this and subtract out what you think the noise floor is so that you'll get a number that is positive when something's happening above the noise floor and negative when it's just noise.

[25:10] And then, you use this wonderful binary operation max which is maximum. So, if you ask for the maximum of 0 on something, then when it's positive, the maximum is the number and when it's negative, the maximum is 0. So, this is a way of clipping the thing below by 0. Alright.

[25:28] And then, this is where your hand really starts seriously to wave. What should be the transfer function by which the amount of decibels in excess of 30 turns into the height of the beak? The answer might not be just plug the number of decibels straight into the height. I discovered that it's better to have loud noises, have a proportionally larger affect than quiet ones.

[25:59] So, I ended up deciding that the right thing to do is to square the number. So that for instance, 10 dB turns into 100 but 20 dB turns into 400. Alright. And, it should therefore be true that as I get louder, the change in the beak gets more pronounced. It is not really true though.

[26:24] Anyway, if you don't square it, then you get that the mouth is just sort of always sitting kind of open and doing a little bit of this stuff and you really want it to just open and shut and it turns out the way to do that is to square the envelope. Square the number in dB. Go figure.

[26:40] In fact, I'll show you how it can be lame, I'll just not square it. And then of course, I'll have to multiply it by something different. Let's actually get a new one. OK. So, we'll just say dollar F1 times something small.

[26:56] So, 5, ten thousandths, let's try this number. Here we are. That's kind of lame. Right? And, so, if you don't square the envelope, you either have that kind of lameness or you have the beak always shut kind of lameness but you don't get a decent range of openness and shutness. And I don't know how to explain any better than that.

[27:23] And furthermore, I don't believe I even explained the X per object, did I? This is your object for making C-like mathematical expressions where dollar F1 means the floating point number coming in inlet of one and dollar F2 is the floating point number coming in inlet two and so on.

So, if you wanted to do this with just regular objects, you have to do a trigger float, floats so you could multiply the thing by itself and then you'd have a separate object to multiply it by. So, this is replacing three objects with one Expo [name spelling 27: [27:40] 53] . Expo [name spelling 27:54] was written by Sharod Yadigari [name spelling 27:55] who teaches in the Theater Department here. Yeah?

Audience: [28:00] What's that [inaudible 28:03] ?

Professor: [28:03] Oh, yeah. Thank you. OK. So, I pulled it, I ran roughshod over one important detail. So, envelope followers are classically anyway and envelope followers take the signal and rectify it somehow such as for instance, square it and then run it through a low pass filter. [28:22] This particular envelope follower is better than that. What it does is it looks at a certain window of the signal and multiplies it by a smoothing function and then measures the total power within that window.

[28:34] This number that you give it is a power of two which is the size of the window that it analyzes. So, here, I decided that I wanted it to analyze a tenth of a second at once, basically, because the tenth of the second is large enough to hold a syllable or to hold a sort of an utterance of some sort.

[28:59] Why is this a tenth of a second? It's in samples and we're at 44K1 sample rates of 4096 as about a tenth of a second. If you make this smaller, which you can, like let's make it 256.

[29:14] Then it starts moving. It starts outputting stuff real fast and then, "Hello, hello," that's still working just fine. Never mind that.

[29:26] I was expecting you to like for instance, if I make a steady sound with a low pitch it might get to the point that sometimes the envelope falls between peaks of the thing and gives you a smaller number and sometimes it falls on a peak and gives you a bigger number and so if I give it 100 hertz or so, "Aaahhhh," it is not very steady, alright?

[laughter]

[29:46] "Aaahhhh," like that. So now, if you tell the envelope to look at more samples at once, do the same thing and it gives you much stabler looking result. "Aaahhhh," I think. "Aaahhhh." Yeah. It's better.

[30:06] So, in general, this is actually easy to understand with envelope calling but it's in general kind of true in audio analysis that the larger a window of data you look at, the stabler the less ragged the output is going to be. Nevertheless, quickly changing the output is going to be. Alright.

So, I just threw at it a large number for that reason. And yes E and B tilde [name spelling 30: [30:30] 34] for technical reasons only likes powers of 2, I'd say. If you give it some other value, it will just change it to a power of two for you but of course, if you just give it the power of 2, then does exactly what you said and you can see what it's doing which is better. Alright.

[30:51] So, I haven't told you a whole lot about E and B tilde, I'm just sort of letting you enjoy what it does and so, now, what I'm going to do actually, E and B tilde, I've told you just about everything you need to know about it. Although I will just pull it out in its own right. Actually, everyone's tired of the bird now, right? Can I get rid of the bird?

[laughter]

Audience: [31:12] Only if you give it a name.

Professor: [31:15] What would its name be?

Audience: [Inaudible 31:18] [31:16] [laughter]

[31:18] OK. Well, you know, keep the bird around. I don't think it will hurt us to have the bird out. The only thing is I should try miniaturize this window.

Audience: [inaudible 31:38] [31:35] .

Professor: [31:37] Do what?

Audience: [inaudible 31:39 to 31:40] [31:39]

Professor: [31:40] Yeah. This is...you know, I made this some years ago. So, this is probably flying around the net already, but I'll stick it up on my site too. This is a very useful tool. [laughter]

[31:50] You might not believe it now.

[laughter]

[31:56] But, when you grow up, you'll realize that you need things like this.

[laughter]

[32:00] Alright. OK. Envelope following. So, it's the usual thing that you can imagine. Take, oh, actually, let's not look at that. Let's look at a nice oscillator. So we'll take an oscillator and I am going to have a number of boxes to say what it is.

[32:21] Oops, no, what am I doing? So now, we will have an oscillator with a controllable frequency and then we are to going say envelope. Whoops, gosh, I'm not using the right key colors today. And I'll, just for consistency's sake, let's say 4096 now.

[32:40] And then, we will look at the result. Oh, we could look at the result using one of these and yeah. I'm not sure this is actually better pedagogy or not.

[32:58] This is a slider. This is a thing which, it's a nice graphical control that lets you do this kind of stuff, right. Well, you all probably found it already. Alright, so, gee, what's is happening? Into this oscillator has a frequency of 0 and out is coming a value which I believe is going to be a hundred. Actually, we should look at that too. Ta-da. Alright.

[33:20] So, the oscillator is putting up one as a signal, right, because it has 0 frequency and one has a 100 dB. Why? Because it's arbitrary how loud 100 dB is chosen to be, decibels are a relative scale, but in PD, there is a convention that 100 dB has an amplitude of 1, alright.

[33:41] Now, we'll set the oscillator doing something. So, let's play A440. And then lo and behold, the thing drops by almost exactly three dB, well, anyway, 3-ish dB. That's because when you start an oscillator going, it, you know, it does hit one and does hit minus one but the average value in root mean square land. In other words, if you took it and averaged it in the way that you get when you square the thing, average it and then take the square root, that's called a root-mean-square, which is the good average for doing things in audio.

[34:20] He root-mean-square average of the sinusoids is the square root is a half, sorry, it's the square root of a half, it's 0.707-ish and, or put it another way, one-half power is almost exactly three decibels. It is 3.02 decibels for people like me and that is what you get here. Super oh, 1ish. Actually, that's not quite right, but good enough. OK.

[34:46] Notice though that this value is exceedingly stable, alright. It's nailing it to four decimal places and not varying at all. In fact, they even make thing fatter. OK. Not that fat. OK. Look at that. Rock solid. That's just too good to be true, right?

[35:11] Now, let's start dropping the frequency. And now, you will see that the slower this thing oscillates, yeah, the less there is a tendency, well, OK. So, what's really happening here? The envelope is running every, you know, a certain number of samples. Actually, it does an overlap of two. So, it is doing an analysis every 2048 points, but it's doing it on a window of 4096, so everybody gets seen twice. Right? OK.

[35:44] As you get fewer and fewer waves of the oscillator in the window, and depending on the phase of the oscillator right at the beginning of the window, you might get slightly different results. And as you slow the oscillator down, so there are fewer and fewer waves so that there's less and less averaging going on from one part of the wave form to another and you'll get more and more variation until at some point, at some horrendously low frequency, at like a tenth say, you actually see the thing, what's happening now is the oscillator itself is taking 10 seconds to do a cycle and the analysis period is only 1/10 of a second and so the analysis thing only see only one tiny portion of the elephant, so to speak, so now what we're seeing is highly variable.

[36:36] In fact, this is not a good representation of the RMS loudness of this oscillator in some sense; or maybe it is because what does that even mean when we're doing this? Questions? OK. At any rate, it will turn out that as soon as you get up to a frequency so that or such that a whole cycle fits in the analysis window, so that will be 10 hertz.

[37:05] By the time you get up there, it is able more or less to give a good answer. So, in this case, with this big fat window, we can actually measure the loudnesses of oscillators all the way down to about 10 hertz; but if we try to get down to something lower, it starts messing up. OK.

[37:28] And that is a trade-off with the size of the...it is another trade-off in fact with the size of this window, this analysis window. So, this is slow. This is only giving us, well, let me put it another way.

[37:43] If you put something in with a sharp attack, you know the thing was off and the thing is now suddenly on. You might actually want to know right when that attack is in time. And so, you might want to have some time resolution in the output of the envelope follower. OK.

[37:59] The time resolution here is terrible. It's a tenth of a second and so, if you gave it an attack, you would see the thing sort of decide there was nothing but then an over entire tenth of the second gradually decide that there actually, yes was a signal there. Alright.

[38:12] So, if you made that window smaller, then it would give you a more and more higher time resolution estimate of when that attack occurred. So, that would be a good reason for wanting to keep this window size small; so that you get good time resolution. But, of course, as the time resolution goes up, the frequency goes up that you have to get to do the thing works.

[38:40] So, again, if you decide that this thing is going to be a hundredth of a second long, then unless you have at least a hundred hertz sinusoid, it sounds like the thing is getting louder and softer in that timeframe. Alright. So, different time scalesl tell different stories. So here, 100-ish times per second is about five twelve samples. Again using hertz.

[39:04] It can still do a 100-ish hertz I think. Yeah. In fact, 512 if I remember correctly, this is 186th of a second. Now, why do I know that? Because I am the sort of sick person who actually stays up all night working on these things instead.

[39:27] That is a constant thing that comes over and over again. So, I should be able to go down to about 86 hertz and still get decently stable results but, when I drop below 86 hertz, yeah, then it starts going nuts. So, this number 86 is sort of the bottom frequency at which this gives me a stable result.

[39:49] But, this is a nice fat envelope follower. It's an 86th of a second, so it will tell me when an attack is to within whatever that is in 86th of a second time resolution; and that is a trade-off.

[40:00] That trade-off isn't but you could sort of hand wavingly call that the Heisenberg Uncertainty Principle. In fact, when you get into chapter nine of the book, you will actually really see the Heisenberg Uncertainty Principle. But, you'll have to do Freudian analysis to do it for real. OK.

[40:19] So, this is envelope following and how to choose this number depending on how low frequency you want to be able to deal. Alright. Questions about this? OK.

[40:35] Oh, outcome decibels. Frequently, you want other units besides decibels like just RMS linear units and then you have all these wonderful conversion objects like dB to RMS that you can use to fix that.

[40:49] And, another little thing is as you have seen in other situations if you put nothing in, it considers nothing to be 0 dB although actually, nothing is minus infinity Db. It refuses to give you a number below 0. I do not know why, for some sort of sanity reasons. OK.

[41:08] Now, envelope following is great; but, another thing that you might want to know about a signal is what is the pitch of the signal. So the sort of basic things that you talk about in music are pitch and loudness. Time of course, but time is just passive.

[41:25] So, how do you get pitch? Well, the answer is you reach for different objects. So, I am just going to tell you what object it is. It is called Sigmund tilde it's named because it does analysis and out comes pitch and envelope and the pitch is the good output or the interesting output for right now. It is in MIDI units and here, this is stupid.

[41:57] This could have just been zero too but for some reason, I end up thinking that you want to be able to deal with MIDI values that were below zero for describing fiber optics feeds and things like that which are below MIDI zero and so, this is really the smallest number that you can reasonably represent in pitch.

[42:15] This is the MIDI number for the smallest possible floating-point number almost exactly. OK. And here, you put your noise oscillator in and out comes in number which should ideally be the MIDI pitch not the frequency in hertz corresponding to this frequency.

[42:37] So, theoretically now, if I converted from MIDI to frequency, then I should be getting out the frequency of this oscillator, sorry the pitch of this oscillator and this is horrendously stable, right? This is changing by 0.005 hertz plus or minus which is probably inaudible, alright?

[43:13] And, in fact, I am torturing it a little bit by giving at this very low frequency. If I give it something more reasonable like A 440. Now, it is down to a part in a million. So, this is, you know, good, you know numerically accurate stuff going on.

[43:34] Unfortunately, it's not true that real signals have periodicity, alright. And so, when you give it a real signal, no one will ever know whether it's saying something accurate or not because what could you measure it against, somebody else's notion of what the pitch should be. So, who knows what the accuracy of this thing is really.

[43:58] But, anyway, to use it, you just for instance, run the analog to digital converter into it; and now you get when nothing is happening, it still gives you that and when you start giving it pitch then you start seeing numbers come out; "hello," and of course, you know, speech is not singing, but speech still has pitch, alright?

[44:21] So, now, we can do the following horrible thing. And by the way, this is another copy of the envelope just because it computes the envelope anyway as a byproduct of what it has to do internally and so it gives it to you.

[44:36] So, now, we will just take that and use it to drive a noise oscillator, why not? OK. So let's say, the simplest way to make a nice sound is probably to take an oscillator and then to add something to it, a, wave-shape it, right, so, we'll add something to it and then just take the cosine.

[45:04] So, here, I am not going to dwell on it, but this is wave shaping as you saw it a few classes ago and then we'll say cosine and then, I want to listen to the output.

[45:18] Oh, wait, I want to multiply it by something to control the amplitude, right. Let's see, I'm not going to need this anymore now. Put this up where we can see it. Don't need that. And now, let's see.

[45:38] So, to control the amplitude, we will take this and multiply it by the output of one, all the good and usual stuff and now, I'll just take these frequencies...oh! I do not want to give it minus...oh, that's, I've gotten rid of that other thing which is in hertz not in MIDI, right?

[46:02] So, I did need that other, go away. OK, let's try this. We are going to convert MIDI to frequency and by the way, I don't like those zero values, so what am I going to do is I'm going to say only give me people who were at least something reasonable like 20 hertz.

[46:35] And, now, what's going to happen is when it actually gets, it is here in my fan and stuff like that right now. But, whenever it gets a pitch, you see it and then, like here, "aaahhhh," and then when it does not have a pitch. It just freezes on whatever the previous pitch was, alright. OK?

[46:52] So, this now will be our noise oscillator frequency; and now, what we are going to do is take this number here, convert it; so it is in dB, so we have to say dB to RMS and I'm going to sneak a look at and make sure we've got something reasonable. Yep. Before I go, playing it; and then, we will...OK.

So, this is, let us see. So, Sigmund by default, it is running at 86 hertz which is every 11ish millisecond, 12ish maybe. So, we are going to pack this with 12. You could measure that by the way, but I am not going to get into it. And, now, we are ready to listen to the result ... alright. This could be good and it could be awful. "Hello," yup, [machine noise [48: [47:19] 06 to 48:07] wow stupid. Why is, oh, I know, I put a, so that the bird would work, I had put a huge delay on the audio.

[48:14] So, I've got to say this bird is computing at ten frames a second. So, I put about 100 millisecond delay on the audio so that the thing can think for a tenth of a second to try to compute these things so now you hear this huge delay and if I...I'll have to stop the bird rendering if I want to take the delay out. So, we will just live with the delay right now because everyone seems to like the bird, right?

[laughter]

Professor: [48:35] So, now, what you got is just me [machine noises 48:43alright. So, this is a nice voice control synthesizer. OK. So, you can...Oh, yeah, yeah, right. [laughter]

[48:48] Yeah. That's actually a trombone with a mute. Yeah, but you could have done it this way. Yeah. OK. So, that is a demonstration just of using pitch and amplitude to control some very simple synthesis voice, alright. OK.

[49:12] About Sigmund. Pitch tracking is a much more complicated thing to do than envelope following. In envelope following, you just square things and add them up and you're happy. Pitch tracking, there are at least a thousand papers on how to determine the pitch of an acoustic signal and Sigmund happens to use one which I just sort of pulled out a hat and works OK.

[49:38] I do not have any proof that it works better than anyone else, alright? I think it works pretty as these things go and in fact, this is the third one I have written and it works better than first two I think for most Sigmunds. You can get to do all sorts of stuff. In particular, it has to find the sinusoidal peaks that are present in the signals so that it can try to figure out the pitch.

[50:02] Because it, you're basically using a frequency domain, and as a result, you can ask it to output not just the pitch but all the sinusoids it found; all the sinusoidal components.

[50:14] And, you can catch those and then you can make a bank of oscillators that plays just sinusoids following the tracks and then you will get a re-synthesis of your sound and sinusoids which can be a very powerful thing to have because in, for instance, you can freeze sound or morph it into some different kind of sound.

[50:31] So, I am not going to show you how to do that because it will take, you know, it will take some time to work it all up, but there is a lot of stuff that Sigmund can do for you. That is worth looking at. OK. And, the help window is scarily detailed. Alright.

So, what I am going to do instead of doing that is move on to another thing. The other main analysis thing that PD comes with which is useful which is called [indecipherable 51: [50:49] 00] . This is the attack detector.

[51:11] Yeah, so I mentioned although I did not dwell on it that this bird example actually has an attack detector in it, ooh, the bird died. Oh, I turned the mike off. "Hello." So, the way the attack detection is happening here is very, very crude and I won't dwell on it.

But basically, it is got the pair of thresholds and whenever the amplitude goes below a low threshold, the thing is off and then whenever it goes back through the high threshold it's on and there's a very simple [indecipherable 51: [51:31] 41] machine patch that you can look up. It's a good thing to learn how to do it if you want to detect beginnings of things. That is the way.

Block [name spelling 51: [51:49] 58] is a thing that is actually for detecting attacks with a very high time resolution. So, higher time resolution and then envelope follower could possibly give you what it is as a filter bag and there is a paper about it and all that kind of nonsense, but what it does is, you just run a signal in and by default, hypersensitive.

[52:12] So, rather than go, mess with its parameters; there are hundreds of parameters you can give it. I am just going to cheat and take my incoming signal and multiply it by 130th so that it is about right. And now, I am going to go get a button to just sort of flash whenever it gets happy and now...

[Snapping 52:41]

[52:39] Oh, it's finding pitches there or something. Don't know why that is. Anyway, now I have something that whenever I give in an attack...

[snapping sound 52:52]

...It puts out a message and this is a good thing if you want to do something like...Well, I won't insult your intelligences by doing this but, for instance, attach this to choose a random number and play a resave bell tone or something like that, and then you have something where you can just do this [snapping noises 53: [52:51] 10] and out come bells or something like that, alright?

[53:13] And, that would be a very simple, you know sort of elementary thing that you could do with it. Here's a slightly less elementary thing. I'm going to see if I can get away with dropping the audio advance. I don't know if this is going to work or not because I want a smaller delay.

So, it's still working OK, "hello." Sort of, and now, I can go down to 50, 50 [machine noises 53: [53:38] 46] it's still good. OK. We're happy. Alright. This is not instantaneous, but it's close, it's ...

[snapping sound 53:52]

[53:52] ...it's fast enough that you can sort of pretend it's instantaneous. OK. Now, what I'm going to do is go here and show you for instance how you can use this to measure time. So, I should have told you this before but there's a wonderful object called Timer which you use in the following way.

[54:13] You click on the left inlet and it sets a stopwatch in a sense and then you click on the right and it reads it out. So, "bing, bing," one second. OK. And, then if you whack it again, it tells you how many seconds still since this last one was done. So now, we just see rising numbers, OK. Timer. OK.

[54:41] Timer is in a sense the opposite of metronome, alright? Metronome you give it a time value and it generates events at desired times. The Timer, you supply the events and it tells you what time was. So, those two things might be a good thing to put in concert.

[54:58] For instance, supposing we wanted only one button, let us do this. What I will do is I will just measure the amount of time between two different hits of the same button. So, every time we get a bang from the button, we will read the time out and then we will reset the timer and then I am going to check with that does for us.

[55:24] So, now we have button and so now, we just have incremental times, alright. Great. And now, we could do something like, oh, this could be a good time for metronome. So, let's, we'll need a bang and float and then, we'll feed it to a metronome. I'm running out of room. I'm going to want this. Alright. Let 's get this out of here. Maybe I should shut this up now. OK. Alright. So, now we have already got a cool thing that will take events coming in...

[snapping noise 56:11]

[56:10] And measure tempo, right?

[snapping sound 56:16 to 56:18]

[56:15] OK. So now, we have a way to measure tempo and have a synchronized drum machine or metronome I guess. We'll have that be the time of the metronome and then we'll start the metronome whenever we get a measurement and then I will make that flash for now although I will do something more interesting in a moment.

[clapping sound 56:51]

[clapping sound 56:58] [56:52]

[56:57] No, it doesn't like that, right.

[laughter]

[clapping sound 57:05] [57:04]

[57:04] OK. Now, what if, for instance, every time we find out that there is an attack from block, we just record what the attack was. Now, you could do this well but I am going to be this sloppily.

[57:21] Sloppy is we make a table. Let us make it 10 seconds worth and then we'll use tab right to record the table and we will just start recording whenever block says bang and what we will record is the audio coming in. OK.

[57:45] And then, every time this metronome goes whack we will just lay the results on the table and again, I will just be a slow tech as I possibly can. Tab, play; and then we will get one of these to listen to it; and now, is this going to work?

[snapping sound 58:11]

[58:09] We'll do this.

Do this. [sounds 58: [58:13] 22 to 58:27] So, this isn't working because my voice is triggering at crazy moments. That's actually not so bad, but what I really want to do is something like...

[58:36] "Hello," "hello." It's not working. Oh, right it is recording a new ...I have to do this.

[sounds 58:54]

[58:47] Hello," "hello." Alright you get the idea. So now we have a, how can I describe this? This is a tempo-driven sequencer. It is a thing that measures the tempo of incoming events and loops at the same tempo as it's picking up.

[59:17] Obviously, you are going to want to do this on someone who actually...well, if you want this thing to work, you might have to practice having the thing not get set off accidentally and figure out very carefully what your thresholds are, and where your mike is.

Otherwise, someone is going to sneeze in the audience and set it off and then, you will be embarrassed. OK. So, usual rules of show business apply but in any rate, this is of simple example of something that uses block for [inaudible 59: [59:31] 48] . OK, any questions about how this is working? Yeah.

Audience: [59:52] Are you sure [inaudible 59:54 to 59:55] in this context?

Professor: [59:59] Oh, yeah, yeah.

Audience: [Inaudible 60:03 to 60:05] [60:00]

Professor: [60:05] Yeah. No. OK. Yeah. So, block does a whole bunch of other stuff that I haven't described here. [laughter]

Audience: [60:11] OK.

Professor: [60:12] So, the original purpose to having block was to be able to actually transcribe a drum set in real time. So, it not only picks out attacks, but it also tries to classify instruments by their tamber. And I'm not going to try to do it here right now because you need drum instruments and sticks to do this well. [60:33] You would need it to do a much better job of miking and playing than I can do right now, but it will actually attempt to, it will give you a number from 0 to N where you train it on different kinds of incoming sounds.

[60:45] And, that is what I was using it for in '97 is someone was playing a drum set in Portland and we are listening to it in New York and what we are doing is transcribing the drum and then sending the transcription over a network.

[61:05] Yeah. That's the whole thing. Yeah, all this stuff has a lot of history. People have been doing computer music for 20 or 30 years now; so, no more than that since '57.

[61:15] So, lots of this stuff goes a lot deeper than what I'm talking about I think. OK. So, basic notions are this is the stupid thing that you just reach for whenever you want to know how loud something is.

[61:30] These are the sophisticated things that take a little bit of computation time, but will do cool things like figure out pitch or rhythms attacks. Yeah?

Audience: [61:40] How did you [inaudible 61:42] make a copy, a recorded copy of that?

Professor: [61:46] What? Say that again.

Audience: [61:50] Like for example when you are triggering, the metro by snapping [inaudible 61:54] , um ...

Professor: [61:58] So what does...Yeah. Oh, right. So, what is happening is when you do this. [snapping sound 62:06]

[62:01] Then, what you get is the last thing. It does not play what happened between the two recordings or it doesn't play what happened between the two clicks but what it played is what happens after the second click. So, that's why to do, to get it to whistle you do...

[whistle 62:26]

[62:21] ... and then always be careful not to do something that would re-trigger it or else it would have done something else.

[62:34] Yeah. And of course, if you don't keep changing it, it gets boring fast. Any questions about this? This is all just kind of...this is the very simplest kinds of, oh here's what you can do with audio analysis.

[62:51] Let me make one more good example or at least an example I like. It's probably not a good example. OK.

[62:56] So, let us go back to Sigmund here which is giving us frequencies and here we're already, what is the right word? We are already filtering out so that we only get the frequencies which where it actually believes there is a pitch there. We're filtering out all the zeros. OK.

[63:12] So now, what would happen if we took our sound? OK. So first off, sound, OK, ring modulation. Everyone knows about this. We will take a sound and multiply it by an oscillator. Oh, we are running out of room. Put this over here. Try to remember about it later. OK.

[63:36] So now, ring modulation take an incoming sound, multiply it by an oscillator and I'll give the oscillator a frequency which I will control with a number box to start with and so I already did this quite a few weeks ago but now, we have a nice "hello." OK. This and then we have a nice ring modulator. OK?

[64:04] Now, ring modulation is a thing which gives you a nice harmonic result if you happen to hit the tone that it is like a I think I can hit 90 hertz, "aaaaahhhh," and then it gives me something good but if I give it a different pitch like "aaahhhh," then it gives me something in harmonic, right, because that's how ring modulation is. OK.

[64:26] Well, what if you just decided to keep the thing harmonic by taking this pitch track signal and making it a modulated sound? In fact, that's not going to be terribly interesting, but let's do something a little destructive and multiply it by 20. Oh, we don't need that tilde.

[64:50] So now, we're going to ring modulate by a frequency which is 20 times the measured frequency of my voice. And now, we have a wonderful...OK, so it's still pitched "aaaaahhhh." Oh, it is bad. Yeah. I am having trouble because this bird is still running. "aaaaahhhh," But, so it's not continuous, but it would be continuous if the bird weren't running right now.

[65:15] That bird is important. So, we are going to keep the bird up.

[laughter]

Actually, let's go back and, I'll tell it you can be a little slower and maybe we would can get away with this for a little while. "Aaaahhhh."

Yeah. It [inaudible 65: [65:17] 34] itself, doesn't it? I'm just overloading the CPU right now, "aaaaahhhh." OK. So, then we can, that's pretty bad.

[65:40] Now, another thing is you can always take any sound if you want and drop it by an octave by ring modulating it by half of its original frequency, alright.

This is the well-known effect, "hello, I'm now going to be down an octave," except of course, you know, if you divide the frequency of the incoming sound a little bit here, fundamental frequency by two and then ring modulate by it, then you will only get the odd harmonics. OK. So, right now, what you hear is odd harmonic sound that [inaudible 66: [65:49] 17 to 66:18] . If you want it more natural, let's say a sound that's both odd and even harmonics, then you would take the ring modulated sound but you would then add in the non, un-modulated sound too. Like this...and now, you have the good [inaudible 66:51] , right?

[66:52] This is, it sounds better in people with higher pitched voices than me, but that's an all purpose trick. In fact, that is plug ins there, you just reach for plug-in when you want that in your sound montage system. But this is basically how this thing works more or less; and now, you can do it in real time. OK.

[67:14] So, there's nothing funny about the chain. The chain is nothing but taking the ADC and multiplying it by an oscillator, but the reason it's interesting now or the thing that's making it interesting now is the fact that you're using the audio analysis to parameterize the thing that we're doing to the audio screen.

[67:31] OK. And you can do this with very small delays. There is a delay associated with trying to figure out the pitch of a signal which is on the order of the window size of the analysis which by default I think for Sigmund is a thousand twenty four samples but I'm not sure.

[67:47] However, this signal chain that's going from the ADC to the output doesn't have much of any delay at all. It's just going in and out and so, the thing is real time and since having a very small delay except that the pitch that it's modulated by is always going to be slightly out of sync with reality because the pitch determination is always going to be 10 or 20 milliseconds late.

Audience: [68:12] Can you get away with a microphone simulator?

Professor: [68:14] You could and then you would maybe get a slightly better sound but you would also hear more delay, so that would be a trade off. Yeah, and that could work better for voice but that wouldn't work as well either for guitar or percussive sounds where delay is not good. OK. [68:33] So, this is the menagerie of audio analysis stuff that's useful. OK. And now, I am going to...oh yes, I am going to save it. Oh, and this is all in one patch. So this is going to be a glorious mess when you try to download it.

[laughter]

Whoops. Alright. OUS, [indecipherable 69: [69:00] 07] OK. And, now, what I want to do is go back and see. We're going to have to do some triage here because we got through this and then everything else is kind of not happening. So, I'm not going to show you net send and net received.

[69:19] This is if you have two computers you can, messages only, this will not work with audio although you can find objects in PD extended that will do this for audio too. If you have two computers and if you know the IP address of computer number two which unfortunately has to be an IPV4 address. It has no IPV6. Alright.

[69:38] You can send messages from the one to the other and they're just PD messages. Now how it works is the messages just get printed as asking and sent in network packets and the object to do that, it's like send and receive, it's called net send and net receive and you have to give it the IP address or name, host name of the machine you're sending it to.

[70:01] Among other things, you can, you know, don't do that.

[laughter]

[70:06] You can, yeah, you can, If you're one of these people that likes to project your patch while you're playing it, you can put a net receive up there and then people in the audience who you know can send you messages and change your sound.

[70:17] Can send you messages and change your sound. OK.

Audience: [70:22] Do you have to give it a port?

Professor: [70:24] Can you what?

Audience: [70:25] Do you have to give it a port?

Professor: [70:25] Yeah. So, net receive takes a port number which is IP language. You give it a number like 3000 which is just the number that it will address by and then the network send which is on a different machine has to know the machine's IP address and port number and that port number can be a way by which you can have net receives that are receiving streams of messages from several different places using different ports. Alright. [70:51] OK. Read SF, write SF. So far, I've only told you how to record things into arrays which of course is then limited by how much memory you feel like giving the array and also, there's a bad that thing which is that when you read and write and array to disc, PD itself grinds to a halt while your computer synchronously reads or writes what might be several megabytes of data, right?

[71:17] So, getting sound to and from disc is not, using arrays of tables is not a real time activity. That's a thing that you would do before the show as you were setting up or before you start making sound anyway.

[71:31] There's another thing, which is a pair of objects which actually spools sound to disc using a separate threads that it can happen in real time and that's the objects called read SF and write SF and they are almost shockingly easy to use.

[71:49] So, here for instance, let's just make...I don't know, there's a nice oscillator. Here's a write SF object and I'm going to give it the name of a sound file, I'm going to throw it in Temp, just to be, no, it's going to be more portable if I just put it right here.

Oh [indecipherable 72: [72:16] 17] nuts. Let's just write SF and then you can tell it how many channels you want but by default it's just one channel. You can make a hundred channels if you want. Now, I'm going to have a message, "message please;" and the message says, to open a sound file, and then you tell it once it's got something opened, there are messages to start and then to stop.

[72:55] Actually, let's make this be the microphone again. OK. This is not going to work perfectly because I have Gem running and so, oh, you're probably reminding me to do something aren't you?

Audience: [Inaudible 73:11 to 73:12] [73:08]

Professor: [73:11] Oh, no. I have a production sub master. I can tell you who has master [indecipherable 73:20] . Gem.

Audience: [73:22] They're all gone.

Professor: [73:22] They're all gone. Pretty much, there's one other, Trevor.

Audience: [73:29] I'm going to try that one.

Professor: [73:31] Good luck. But, that does remind me you have websites that you used to tell UCSD whether I'm a good teacher or not. I don't think whether you think I'm a good, or whether UCSD thinks I'm a good teacher or not because it turns out that my pay doesn't go up if I teach better. [laughter]

[73:49] So, you can say anything you want. But, it's good if you go ahead and participate in the thing because it makes us look accountable and more important is tell UCSD if your TA was good which I think he is because he will be able to use those things later on when he's looking for a job, he can say, "Everybody thought I was a great TA."

[74:09] So, it's actually a very good thing to tune in and give your TA's evaluations. So please, remember to do that and I will try to remember again to tell you this during the final presentation in case you forget which you probably will; but, that's a thing that I really hope you will do. OK. Yeah.

[74:27] So, here it is, we'll just say, "Hi this is a recording." Oh, yeah. And then of course, I didn't connect the start button and I didn't tell it to start which is to say I didn't do anything. And also, I've got this other thing running which is going to get irritating. So, let's get that out of here. OK. And let's save this, it's going to be called "Record," and now maybe I can do this. OK.

So, we open it and then we say, "Start. This is a test. This is only a test." And then, we say, "Stop" And now, we have a sound file. I 'm not going to try and play it for you because I would have to make another patch with the read SF object but [indecipherable 75: [75:02] 17] , don'tworry.

But, if I went and looked now, I would find a nice file called sound F1 dot wave which would be a 16 bit pcm sound file with that thing that I just did. So now, you can save the wonderful sounds you're making as patches. It'll do 16 or 24 bit or 32 in fact [inaudible 75: [75:17] 34] and it'll do wave or AIFF although floating point AIFF is a bit of a stretch I think.

[75:42] Yeah, and it is 10 'til. So, I am going to just not tell you all the other good stuff.

[75:47] PD tilde if you have a multiprocessor will allow you to have all your processes run in separate PDs and send signals back and forth between them in case you run out of gas.

And yeah, show up, yeah, well show up obviously on Tuesday for final presentations and also Tom Erb [name spelling 76: [75:57] 05] will be teaching 172 and it's wonderful. So, everybody show up for 172 and check that out.

[applause]

[Silence 76:20 to 77:05] [76:11]

[76:21]