[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL SCOTT ASATO CUTHBERT: Today, we're going to do two really important things that I really want to get through today, and the first one involves the final project for the class. So what I'm going to ask you to do is to take a minute, and after I tell you what you're going to be doing, find a partner. If we're not and even number, you can do one group of three. But I prefer groups of two have tended to work a lot better, that you're going to be willing to work on a project starting next week and start thinking about what you would like to do.

It can be anything related to music theory and analysis in the symbolic domain, that is to say, working with notes and scores, not primarily working with audio or things like that. Would you want to do that, take Eran Egozy's amazing classes where you can do great projects on audio. And it can be a research type problem. You can try to solve a research problem. It can be more of a business development and entrepreneurship type problem, trying to build a product that does something interesting with music theory, or it can be a creative project.

I mentioned at the beginning of the semester, but I want to say it again, the only restriction besides primarily symbolically based is that, if you are doing a creative project, I don't want it to be based on deep learning AI. And the reason for that is as a safety for y'all. I've allowed it in the past, and it hasn't worked very well. And we will do a lecture on AI later on, or a topic on AI, and then I'll be able to explain a little bit more on what the constraint that tends to make those projects not work very well.

So has anybody already been thinking over the course of the semester of something that you're interested in? I'm not going to ask you to say it, but I'm just curious. Who's been thinking a little bit, a little bit, a little bit? OK, so some over here. So let's take 10 minutes, five minutes to find somebody that you'd like to work with. So let's all stand up.

OK we'll get to-- no, actually, your computers are mostly closed. So let me just remind us a little bit of the partitive topic for the week and talk about some of the things. Yes, yes, it's down. Thank you. Just shout when you get that. I did remember to turn the signal back on, but yep. Thank you. So we're going to talk about some of the other things that we were talking about last class with algorithmic composition. So first off, who now has been able to access the ILLIAC suite and recording? OK, great, super. So we're in a better position there.

We talked about whether or not the ILLIAC suite was more written to create music that sounded like a human, or whether it was created to do something very new. And what did we come to this consensus? Anyone want to throw that out? Or what did you--

AUDIENCE:

30 towards seeing something new?

MICHAEL SCOTT ASATO CUTHBERT: What's that? 70/30 toward doing something new. Yeah, yeah, something like that. Some percentage, some mixture of some parts of what they were trying to do really was about, hey, this is something humans do. Let's see if it can computer can do it. And then there was some moments, like the one with the algorithmic pudding of claps and hitting the violins and stuff like that at different points, which was a little bit more computery.

We also talked about Markov chains in the last class, and we'll get a little bit later on into some of the more sophisticated statistical learning that people are doing today with AI learning. But a lot of it is very, very similar. It's just a lot faster, a lot bigger than things. But I want to talk about some things, and we'll to get to that later. Pretend I didn't show that.

So there's two ways that we can distinguish algorithmic composition. One of them is whether it's deterministic or if it has a random chance in it, whether or not every time you play the piece or run the piece, it will be a different piece or it will be the same.

So I want to start by giving-- I'm going to go pretty egotistically with two pieces that I've written. The first one is not even really a piece, but it's kind of a little demonstration. It starts with a demo that you might have seen in a science class or physics class, where all the balls are moving at different lengths, but they do some interesting optical illusion effects at various times. ASMR.

AUDIENCE: Is there supposed to be music in the background?

MICHAEL SCOTT ASATO CUTHBERT: No, no music yet. OK, so I thought, what if there was a musical equivalent of this? What if we could write a computer program that had 12 lines, or eight lines, or something like that, that all started in phase, and then slowly got out of phase, and came back together like the pendulum? And so this is a little bit what I came up with.

[MUSICAL NOTES]

OK, I can't even follow my own score properly. So what's that?

AUDIENCE: With the double dotted 16th--

MICHAEL SCOTT ASATO CUTHBERT: 16th rest, yeah, yeah. So sometimes when you're making an algorithmic composition, maybe you want to think a little bit about the notation, or maybe even better just to hear it without seeing the notation at all. So one of the questions is, if the computer is just setting it up so that each note comes in at a slightly different time from the previous one, what kinds of things does the composer have for her, or his or their creativity? Yeah, go ahead Jason.

AUDIENCE: Which notes, which pitches show up?

MICHAEL SCOTT ASATO CUTHBERT: So pitching which pitches show up. Good, good. Cause that could, in fact-- let me do it. Here is with, instead of using-- I think I used eight notes out of the octave. Here is all 12 notes.

[MUSICAL NOTES]

And that sounds a little bit more familiar and maybe a little bit closer. Why might I not have wanted to use this as my main piece?

[MUSICAL NOTES]

It has something to do with the duration of the piece for me, that it ended up being a little bit too long. So yeah, definitely what notes we use.

[MUSICAL NOTES]

We can even go to weird things that a human can't usually play on a piano.

[MUSICAL NOTES]

So this is 19 tone division of the octave. Let's you use a few more notes per octave.

[MUSICAL NOTES]

Oh, blanking. It stops the music. Good. Yeah.

AUDIENCE:

How do you go to 19? I would think it would be 24 or something?

MICHAEL SCOTT ASATO CUTHBERT: Yeah. Sorry, it's a little digression, but it's kind of cool if anybody's interested in it. A 19 tone division of the octave has two pretty cool properties. One is that it has major and minor triads that are very, very close to acoustically-- or major and minor thirds very close to acoustically correct, while also having perfect fourths and perfect fifths are pretty good. So that's one reason.

The other reason for it is a lot of 19 tone piano-- a lot of 19 tone music is written for two piano duets where you keep the five black keys the same between each of the pianos, and then you retune the white keys so that you have a separate note for each of them.

Is that right? 5, 7, 7, 7, 14, and 5? Yeah. I can't remember. It's been a long time since I wrote for 19 tone piano. They don't like tuning our pianos for that very much. But yeah, so it makes it a little bit more human there. Other choices you could do? Instruments. You could do how long it goes, whether it's going to be slower, faster. So these are some things we can do. So then we'll get into non-deterministic things. And so let's code one up first. So go and grab it before I get to particular pieces that are like that.

So let's try some of the things that we might do and see how aesthetically interesting it is. Oops. Where'd that go? Is that over here? OK, gotta look, move this over as much as I can. OK, good. Great, so what's the module you hope I've unlocked? It's not part of Music21. It's part of Python, in order to do non-deterministic composition.

AUDIENCE:

Random.

MICHAEL SCOTT ASATO CUTHBERT: Random. Great, so let's import random. And for now, let's just import everything from Music21. Yep, that's good. You can see that. Good. Who has worked with the random module before? No? There's basically everybody. You know that random dot random gives you--

AUDIENCE: Between 0 and 1.

MICHAEL SCOTT ASATO CUTHBERT: Between 0 and 1, yep. Oh, shoot. No, that's not the same as-- I might have mistyped. I have it here, and it says it should be 0.23. Good, good, good. We all know that random-- so in a lot of languages, of course, you'll do, in order to get between 0 and 100, say, you would just multiply by 100. Oh, we did pretty well on that. Played the lottery today. You just multiply by 100. Python also has this randint from 1 to 100.

And this is one of the older modules in Python. So it has a little bit of inconsistency with the rest of Python, in that, well, if I do this three times on average, I should be able to see the inconsistency that it includes the final number. So if you do range 1 to 100 in Python, your last one will be 99 in most languages. But this one will include that. Really useful-- this might be all review for somebody-- is that you can do random choice. So if you have a list of things or any other algorithm, you can randomly choose one from that.

And you can do a weighed choice in this, simply by putting in something-- oops, I hit the wrong key-- putting in something a lot of times. Probably, if you're going to do this 10,000 by 10,000, there's other more efficient algorithms you can do for this. Questions on randomness outside of music? By the way, there's a way that you can set it up so that it's random at first, but then you can reconstruct your randomness by doing what's called a deterministic seed, which I'll often do when I'm writing compositions, so that if there's something I really, really like, I can reconstruct up until the point I didn't like it.

So let's write, first off, what we call a random walk composition. So we're just going to start by doing something pretty stupid and try to make it better. So we're going to randomly walk around the scale. We'll start. Let's create something that's going to be a useful, helpful function, which I'll call it Multishow. You can skip the annotations while you're doing this. So what this does is just going to take in a stream and show it as both an image and MIDI. Bach. We'll just do our favorite Bach piece just to demo. BWV 66.6, multishow, Bach.

This I find helpful for anytime I'm working with something. It always takes a lot longer when I'm projecting and the first time, but you get the score and then below. That never looks right on the video. Oops.

[MUSIC PLAYING]

Yep, that's the piece we know. So that's just a useful, helpful function. So one of the parts of pitches that I find really helpful for not being completely random here is this thing that would have been really helpful on that problem set where you had to determine the number of lines, the generic intervals between notes, if you remember that. That's something called diatonic note number, or diatonic note num. And this is a concept that I was really, really surprised. I was like, somebody must have named this, and it didn't seem to have a name before.

It's simply, just like MIDI has-- I don't think I went over this in class, right? No, I don't think so-- that C4, middle C, is arbitrarily assigned to MIDI number 60, arbitrarily assigned middle C to diatonic note number-- that's MIDI 60-- 29. Neither of these are completely arbitrary. There's a reason for them. And the notion is that all. Cs, C sharps, et cetera, have diatonic note num 29. So C sharp, and that just tells you-- C double sharp, yep, so on, C flat. Great.

And so this is helpful for writing diatonic music, that you can just manipulate the diatonic note number of a note and you don't have to worry making it sound a little bit better or something. We should have a pentatonic note number, also, because that makes it even better to do something. So what I'm going to do is-- and then we can just do n dot pitch, dot diatonic note num, plus equals 4, and move things around that way.

If you haven't seen it yet, one of the things in Music21 that's not so good is that you'll often need to copy things, so we'll get to that in just a little bit. But I'm going to import, copy. And if you've never used this copy n-- oops. Copy is a module. Copy dot copy. So I did turn n into a g, and so now it's a g. There's two things. Who has used the copy module in Python before? No, not too many. OK, good. So because for the most part, you might just want to just create another note if you wanted to do that. Don't need to type that. But you can also copy an existing one.

And what you'll use much more is something called deep copy. And the difference, you can see that copying the note gives you a note that's g. Deep copying gives it a note that's g. The difference is that, well, here. Easier to show than tell. And c is going to be the copy of n, and we'll say dc is the d-- my computer just froze. There we go. dc is the deep copy of n.

The difference is, so here, we'll just make sure that they're all here, they're all, yep, three notes. Difference is if we take n's pitch diatonic note num and go up 4 more, that the copy of n also changes, whereas the deep copy doesn't change. And so what some people might think of this as, you end up working with arrays in a lot of other languages and stuff that copying an array, you still have references to the internal pointers. So anything that is itself an object, like n.pitch and nc.pitch will still be the same under a shallow or normal copy.

But with a deep copy, they're different. Deep copies, pitch. So you'll find when you're manipulating things and you get to a particular point in the score, and now you want to add something else that's a manipulation of what came before, you'll want to make a deep copy of it beforehand. Does that make sense? Cool? Any questions? Yeah.

AUDIENCE: What did you show in the last part? It's like the pitch is going to be different by definition?

MICHAEL SCOTT ASATO CUTHBERT: Oh, sure, sure. I think it's a little bit hard because this is this part here. So when you make a shallow copy and you change some nested structure within there, like a pitch or a duration, it will not be-- they're all shared. So what you do is think of it as-- I don't know. What's something else that's not so? So n is a note and n has-- I wish I had set this before. ID equals note, and dot pitch equals some of a pitch dot pitch object. And dot duration, let's say, dur is some kind of duration object.

When you do a shallow copy and c will be nc's ID equals n's ID, and nc's pitch equals nc, n's pitch. And so since ID is just a string or something like that, once it's copied over, they have no connection to each other. But since we've copied the pitch over, and it's a complex object when you change something on this, when this manipulates, this manipulates. So with a deep copy, what you do is dc dot pitch. Hold that. First let's do a deep copy of the pitch. And so you keep going down and you make sure that nothing is connected between the two.

AUDIENCE:

So this also is slightly misleading, right? Because there's nothing that actually says nc.pitch pitch is equal to n.pitch. In fact, that's the whole point. If it did, that would be an example of deep copy, right?

MICHAEL SCOTT ASATO CUTHBERT: No, because you're referencing the same object in point. So when you say n dot pitch, it's actually thinking in the computer, oh, that is the complex object being stored at some random memory location somewhere.

AUDIENCE:

OK, so it's still pointing to the label.

MICHAEL SCOTT ASATO CUTHBERT: So it's still pointing to the same label. Yeah, so exactly. For people who know that kind of terminology, it's pointing to the same label, whereas this one goes to the label, and copies everything and makes a new pointer. So that ends up being pretty important. And if anybody wants to make their final project making Music21 faster, you could find places that we deep copy more than we have to because that's the slowest thing in what's happening.

Too much of an aside. Back to some music. Great. So we'll get back to our random walk composition once I find where this went. Yeah, here we go. Great, so we're going to be copying things pretty soon. But first, let's just create. We'll create a generic stream, which we rarely do nowadays. Maybe we want to make it a part or something. But generic stream will work pretty well. And let's just say for i in range 30, let's just append 30 random notes. Or no, let's do this. No, let's start on c, current. Well, we'll just call it cur. That's the current diatonic note number.

When we're coding fast, we don't need my obsession with long variables. So we'll say note equals note dot note, and pitch diatonic note num equals the current one. And that works pretty well. Stream, append.

AUDIENCE:

It's called cur now.

MICHAEL SCOTT ASATO CUTHBERT: Sorry, thank you. That's why I shouldn't do that. Thank you. Please, always just do that. I appreciate that. Big help. And we'll just say we're going to choose between going down a step, going up a step, and so on. So our first note will always be--

AUDIENCE:

C.

MICHAEL SCOTT ASATO CUTHBERT: C, C4. And now let's multi-show that. Pretty boring piece, but here we go. Oh, this one went down this time.

[MUSICAL NOTES]

Et cetera. Anybody's more interesting? Anyone's go up? Anyone's just hit Play? Yeah. So not a great piece. The easiest thing I could do to make it a little bit better, by the way, would be to make my number of notes I attach be-- what do you call it? A multiple of 4. But OK. So let's move on to do some things that are a little bit more interesting. Maybe we could do something like this. I'm going to copy and paste a lot on this, so we can go ahead and just tweak what we're doing.

You, in fact, probably don't even need to copy and paste. You can just keep tweaking the same cell forever. So what's something we should do to make this a little bit more interesting? Rhythm. Great. So let's have each note and dot duration, dot something. What do we want to do?

AUDIENCE: I'm just thinking about something for-- it wasn't for a duration.

MICHAEL SCOTT ASATO CUTHBERT: OK, then hold it when we get-- let's finish the duration, and then yeah, then we'll get to it. John?

AUDIENCE: A choice for 4s to choose a quarter, a half, eighth, n for the duration.

MICHAEL SCOTT ASATO CUTHBERT: Great. So random choice. While I start typing this out, what goes after the n dot duration dot quarter, half, eighth? Anyone remember what this is called? It's unhelpful. It's called the type. The type quarter length, we would have to say we could put it in as one, two, half. Yep. Let's see if that makes it a little bit nicer.

[MUSICAL NOTES]

It's actually already nicer. Usually, this doesn't help as much.

[MUSICAL NOTES]

Next improvement I'm just going to keep going on this. Yeah.

AUDIENCE: You can make it [INAUDIBLE].

MICHAEL SCOTT ASATO CUTHBERT: What's that?

AUDIENCE: You can make it go up a third.

MICHAEL SCOTT ASATO CUTHBERT: Cool. Yep, it can go up a third or down a third. Now, do we want to make it a weighed probability at all? So how much more do we want it? Up a third is minus 2, because we're thinking in computer think, not in music theory think. So that would be unweighed. And how do we want to weigh this to make it feel more musical?

AUDIENCE: Higher weight on stepwise motion?

MICHAEL SCOTT ASATO CUTHBERT: Higher weight on stepwise. Negative 1, negative 1, 1, 1. OK, let's try that.

[MUSICAL NOTES]

Hey, there's our first step.

[MUSICAL NOTES]

Not so good. OK, so we still need some things. That's definite improvement. Thank you. Another thing to improve. Yeah, Jason.

AUDIENCE: Decrease the weight on staying on the same note?

MICHAEL SCOTT ASATO CUTHBERT: Decrease the weight on staying on the same note. OK, so what we can do is the easiest way when you're programming really fast in front of a class-- do that-- is cut and paste. Don't do this in your actual things you turn in. But cut and paste and just delete that 0. Is that good? Yeah, it already looks a little bit better. Next, let's do another improvement on rhythm. Yeah.

AUDIENCE: It's more likely you have a note starting on beat one.

MICHAEL SCOTT ASATO CUTHBERT: Good. So we want to try to weigh it somehow so that we have notes that start on the downbeat more. Talk with your neighbor on how you might implement that.

[INTERPOSING VOICES]

MICHAEL SCOTT ASATO CUTHBERT: OK, let's come back together. Anybody?

[INTERPOSING VOICES]

MICHAEL SCOTT ASATO CUTHBERT: OK. Let's jump in, well, and we'll try to program it together because we'll do this in class. Somebody I haven't heard from yet want to give a suggestion who hasn't unspoken? Yeah?

AUDIENCE: We can try to keep track of an offset, and then based on that offset, have a different--

MICHAEL SCOTT ASATO CUTHBERT: So have a different probability based on the offset. What offsets in currently with the default time signature 4/4? Why do we make that default very hegemonic. But what offsets represent the downbeat?

AUDIENCE: Zero.

MICHAEL SCOTT ASATO CUTHBERT: Zero definitely represents a downbeat. What other offsets? So have we been creating measures, by the way. First off, let's look at that. So we've kind of taking advantage of Music21 or something creating measures for us. So our offsets, we start with zero, and then we're just appending. So what's our next downbeat?

AUDIENCE: Four.

MICHAEL SCOTT ASATO CUTHBERT: Good. And somebody else, what's our general formula we're going to find for downbeat?

AUDIENCE: 0 mod 4?

MICHAEL SCOTT ASATO CUTHBERT: Yeah, we could check if the offset is mod 4 is 0. So we'll get to, before we do that, so let's say, what's our current offset. So one of the things we can do is we can keep track of our current offset. Now, we're going to wish we had-- 0.0. Now we're going to wish we had done this as quarter lengths. So I'm going to rewrite this as quarter length. Why? Because it's easier to add quarter lengths than it is types. So 1, 2, 0.5. And we'll just say ql equals random.choice.

So what we can do is, well, now, I'm just going to make sure that I still have a current thing and cur_offset plus equals ql. So I haven't done anything to adjust our probability yet. Just making sure that this works. Yep, it it's still generating. So how can we use this to implement the great suggestion that we want to have things that fit on the downbeat more often do something else? So we'll say, is on downbeat equals cur_offset present zero. You got to say, bools on. Here we say true. This is the easiest way. If cur offset 0 equals 0, else false.

You can do it. You can realize that Boolean expression will be reduce itself, but I like to be explicit when I'm thinking. So now we know if something is on the downbeat. How do we want to do it differently? Do you need to brainstorm? Thanks, Adam. I want to hear from some more voices, even though you're doing great. Vanessa, yeah.

AUDIENCE: On the downbeat, you can use the formally chosen borderline pitch, and then--

MICHAEL SCOTT ASATO CUTHBERT: And if it's not on the downbeat?

AUDIENCE: You leave it as what it was before.

MICHAEL SCOTT ASATO CUTHBERT: You leave it as what it was before. Let's try that. Oh, yeah. So we have a last ql, so we'll define ql outside of here, will be pretty Pythonic, equals-- we'll say our last one is 1. We're actually not going to. Because we will always start on the downbeat, ql will be defined. But let's do good programming techniques and define it outside. Good. So if is on, if not is on downbeat. Let's see if that-- otherwise, we're just going to stay on whatever we were on before. Let's see. Think it through first. Did I get something wrong?

AUDIENCE: ql will have a not defined error. Oh, sorry.

MICHAEL SCOTT ASATO CUTHBERT: Yep, that's what we do. That's why I was talking about defining it. Good catch, though. Always don't know crashed 787 because of not defined errors. That's a famous story. OK, so let's make sure that we're not going to end up permanently syncopated or something. We start on the downbeat, and then we choose between-- maybe I should put these in some of a logical order 2, 1, 0.5. We randomly choose between half note, quarter note, and eighth note, and we're on the downbeat, and so we're going to stay on that for a bit. We'll always end up back on a downbeat at some point.

We start on a downbeat. We choose 2, 1, 0.5 in this particular piece. Who votes yes? Who votes we might be permanently syncopated? Spend more time reading code than writing code.

AUDIENCE: It looks like we might, yeah, because it always switches the quarter length if we're not on the downbeat.

MICHAEL SCOTT ASATO CUTHBERT: Oh, did I do that wrong? Do we want to keep it-- we wanted to-- do I have a misplaced not? So that's what we wanted a little bit. If we're on the downbeat, we can change to a new rhythm. If we're not, we can't. I think let's run. Let's generate this.

AUDIENCE: [INAUDIBLE].

MICHAEL SCOTT ASATO CUTHBERT: Yeah, so what's the consequence of this? Somebody else.

[MUSICAL NOTES]

MICHAEL SCOTT ASATO CUTHBERT: Yep.

AUDIENCE: It's the same type of notes that are changing.

MICHAEL SCOTT ASATO CUTHBERT: It's the same type of notes every measure. You know what? I'm going to change that over. Oh, no, no, we can't determine the length of the piece. So we might have that. Yeah, every measure will always have the same type of thing. This is kind of nice for-- a lot of my junior high etude books were like this, that you only switched rhythms. But maybe what's one way that we can switch a little bit better, more often? We could just say, we should rename our variables, but maybe this will feel much more unified. We'll just do it on every two notes.

[MUSICAL NOTES]

OK, that's enough. That's already becoming more unified. So what are some things that we could hack into this? We're done for a little bit, but what are some things that we can make it sound a little bit more human?

AUDIENCE: Harmony.

MICHAEL SCOTT ASATO CUTHBERT: We can add harmony. Good. We could add random or appropriate. Probably more appropriate. Yeah, Jason.

AUDIENCE: We could allow it to occasionally break these rules by, well, probably by random chance.

MICHAEL SCOTT ASATO CUTHBERT: So certain random chance that we break the rules. Yeah, one of the things here, I'm not going to do this. But I gave a particular random chance that something might go up an octave in another time when I did this, and there was some danger in allowing that. This is a particular problem with MuseScore, that if you overflow 256, it jumps. It's unsigned, which is kind of cool that the world of notation wraps around. But yeah. So I obviously was allowing this one random-- seven is an octave if you're adding to it to happen a little too often. So that can be a danger, but it happens. Thanks for having that.

Yeah, so those are some of the other things you can do. Maybe we want to sound a little less-- I'm going to start doing this. I just need to not make it full screen. You can make it a little bit less human and make it do these weird microtones, or not less human, but less traditional. And where'd that go? Here we can hear it. These, unfortunately--

[MUSICAL NOTES]

So there are ways that you can do some things that allow you to be more experimental if you want, or less experimental. Great. So we'll keep playing with that. What you'll probably realize, one of the things we're building up to, so you're given a problem set 7.8, as I said. It's not in 7/8 time. That would be too lopsided. But what it is, is it gives you some work with scales and with Roman numerals. And then there's two parts for you to do an open ended thing. One is, we're going to be asking you, in the first half to write a Roman numeral analyzer just for triads in major. And it's just going to be for block chords.

I'll give you three notes. You tell me what Roman numeral it is. If you're a little bit behind-- sorry. If it's been a while since you've done Roman numeral analysis or you feel a little bit weak on it, collaborate. Talk with other people on that. Make sure that you're really clear what a diminished triad and all that things is. So grab somebody before you leave the room. And then there's a little open ended thing where I just ask you, do something a little bit more. Maybe you have something that works in minor. Maybe you have it do it seventh chords.

Maybe you work in jazz and popular music, and you think Roman numerals, oh, that's only for classical music. I'm going to have it do chord symbols, CDM, and things like that instead. So whatever you want, there'll be just a small part of it. I should have it open. In fact, I think I-- no, I only have it with all the answers open. Never mind. And then, the last part is, I want an algorithmic composition that you like. It can be multi-part. It could be single part if it's interesting. It could be with chords, without chords, but just something where you've done some of interesting music theory.

I'll say that what we've given here is the quintessential D minus so far. Try to go better than, maybe it's a D. Actually, but at this point, we're getting a little bit better. Don't spend the entire semester on it. This is not meant to be the gigantic project that you take forever on. The only restriction I'm giving is that I'd like it to be about a minute long, not two minutes long, not 12 minutes long, because I have to-- if everybody turns in a 10 minute composition, then I'll be listening to algorithmic compositions forever. So if you can do that, that's going to be that main part.

Any questions about what we did with playing around with algorithmic compositions today? Then I'm going to be a little egotistical for a second and talk about one of my first algorithmic compositions that, at some point, people were performing. And it's a piece for this group called the Bang on a Can All-Stars, which some of you might recognize the clarinetist is Evan Ziporyn. So another professor who's here, who does a lot of things. And it was based on Markov chains.

So the notion of what I wanted was some kind of a melody that was a little bit boring-- not boring, but just undistinguished. Except it had these weird bumps in it, some things that sounded a little bit weird, like this opening tritone, and then some of these weird gestures. And so some things that you'd hear enough times, you're like, OK, I know that that's what the piece is about. And try to see if this is-- I can't remember when it's from. 2005. It was a little bit pre-deep learning and things, so just probability theory, whether or not the Markov chain would do something interesting things with this.

And then one of the things I really wanted was someplace like an old record player that would get stuck in a loop and need to be jumped out of it. So I have a few places where the same thing happens over, and over, and over again. If you're thinking about what we were talking about at the end of last class about the probability of B, given A, the next note, or we could say n2, given n1, this is-- what order or Markov chain are we talking about here? First order, based on one previous note. I used a bunch of different orders in this piece, depending on how random I wanted it to be.

I wanted that, when you got to that note A, that the probability, given A, was very high that you would do another A. So I weighed the probability. So the probability of an A, given an A, was, I don't know, something like 0.9 or something. So the computer would get stuck in this particular groove at some points. And we'll see. What I would do at some points is just keep the melody, just keep the notes, and let the rhythms be out. But if this is treble clef, that's an opening gesture of B, F, comes back a lot B, F.

[MUSICAL NOTES]

You'll hear that a lot, and then getting stuck in that a groove again. Da, da, da. You'll hear that. And what I did was, back in the old days, I just had it generate a whole bunch of melodies, and I kept the ones I wanted. So I wanted to do my computer type of thing. One last little thing is that I had this sort of progression that begins one beat per second, so 60 BPM, then moves. There was a section at 1.5 beats per second, 2, 3, 4, and then the final note is a dissonance that vibrates five beats per second. So that's kind of the little Easter egg in the piece.

And then I took the 1.5 beats per second and moved it, because that's what composers do. When the piece was premiered-- they'll never be happy with me at the printing office again-- I gave every person in the audience a different set of program notes that began by saying the opening of the piece is rather straightforward. And then I wrote about 10 pages worth of program notes, and everyone got a Markov chain of the program notes. So it has something. Kind of looks like something, and then it ended always with a hint of order, no matter what they had. So I'm just going to be indulgent and play a little bit of something I wrote a while back.

[MUSIC PLAYING]

And here's where I let the computer start taking over.

[MUSIC PLAYING]

I think I told the computer to multiply the intervals by 2 for more leapy.

[MUSIC PLAYING]

This is completely Markov chained here.

[MUSIC PLAYING]

You can kind of hear where it's stuck again.

[MUSIC PLAYING]

Any time you see a two measure repeat, the timing didn't work.

[MUSIC PLAYING]

That's 2002. Well, thanks, everybody. You don't need to write that much. Please don't write that much for your algorithmic composition, but feel free to have a mixture of your own personality and your own desires, and let the computer do some things that kind of inspires you.