[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL CUTHBELT: Hello everybody, and welcome to our last video, at least for a while, on music representation. And I want to talk about a remarkable tool created by a researcher at Stanford, Craig Stewart Sapp, that he calls the Rosetta stone.

I'm going to be skipping a few slides from his presentation, and I've added a few other things, but in this module, I will also present his entirety of slides that he has given. And my hat's off to Craig Sapp for all he's done and his generosity.

So the Rosetta Stone was a stone discovered in, I think, 1799 and deciphered over the next-- I don't know-- 10 years or so. That was a remarkable discovery, because up until that point, Egyptian hieroglyphics such as those highlighted here were not legible by anybody living at that point. But the Rosetta Stone had two forms of Egyptian hieroglyphics in the top two sections and also at the bottom, the exact same text in Greek.

And so since people knew how to read ancient Greek, they were able to go through each of the signs of the ancient Greek and figure out what the signs of the hieroglyphics and the two forms of writing meant. Craig Sapp's Rosetta Stone allows us to look at how different digital representations of music work by encoding the exact same melody in, I think, almost two dozen different formats. These are all the formats that have ever been used, but it's a good chunk of them.

So here's the melody, and it has some interesting things. Here it has different octaves. It has key signature, time signature. It has beaming information. The only thing it really-- well, some of the things it doesn't have are chords and triplets and things like that, but once you see how this melody is encoded in a lot of formats, you can figure out how all the other things might be encoded in those formats.

So here's an example of a code that's not used as much today but was once very important, playing an easy code. And you can see that we have something like-- I can decipher, OK, maybe that's a G on clef on line two, and then we have a flat on b 2/4. And then we have an eighth note with a dot on C. And then I guess the three means 32nd notes DC. And there's a little bracket around it, and that probably means they're beamed together.

Then we have two eighth notes on C beamed together. We have slash means new measure and so on. Throughout the whole thing, we can see that, oh, maybe octave isn't encoded here or something like that. And we can figure out how those go.

Here is another format. This is primarily used for folk songs called ABC. And we can see that-- well, example. Maybe that's the title of the piece. 2/4 key is F. L means the smallest possible. Encoding is 16th note. And then we see that C is worth three of those 16th. D is worth, I guess, 1/2. C is 1/2. C is worth two of those. That's the eighth note. C is worth two of those. We have a bar line and so on. And we can see that the high C is encoded with C with a apostrophe after it.

This is what's called DARMS code. It was once highly used, but it hasn't really been used very much recently. And well, I don't know how to read DARMS code, but I think that the numbers represent pitch. Here we have an eighth with a dot. And T means 32nd, I'm guessing, and so on. We have some RSs, which might mean rest 16th note.

This is GUIDO Music Notation, which uses backslashes to indicate certain kinds of musical elements. We can see clefs keys, meters are encoded in here. Barlines are explicitly encoded. And it looks like you put a slash before a duration. And there's some notes that don't have durations, but they seem to be the same duration as before and so on. I think it looks like underscores our rest.

This is MuseData, which Michael Good said was one of the inspirations behind MusicXML. It's a little bit bigger format, takes more space. It was one of what we used to use a lot, these fixed position encodings, so that if you're writing in a monospaced font, every single place has a particular meaning.

So you can look at column eight or something like that and see the duration. And then, this was a major positive step because it encodes duration in terms of the number of some things, the number of 32nd notes and also separately the look of the duration. And so you can still see that in MusicXML and formats that are still in use today. I really like the use of these left and right square brackets to indicate the start and end of beams.

Humdrum is a code that we have encountered a couple of times, and here it is. And maybe it's easier with humdrum to rotate the musical examples so we can see how things go, maybe, maybe not. But we can see that there's Ls and Js that represent the starts and ends of beams, and different octaves are represented by doubling or tripling the letter.

LilyPond is actually a set of macros for the scheme programming language, which is a kind of variant on lisp. And so you can actually write computer code in LilyPond. And here we have the same representation. You can see again backslashes, indicating starts of larger musical elements, time signatures, barlines that are explicitly encoded with what they look like, and so on.

Melisma was an interesting code format that's a little bit similar to what we might have with MIDI. And what we're doing here is we're encoding only the numeric duration. So we can say that, I guess, 125 is a 32nd note. And we can figure out from there that 500 is an eighth note and so on. And on the far right column, we have the pitch encoded. So we're losing here beams and some other things. We're actually even losing the clef and the time signature, but we do have an info tag that tells us the key signature.

Allegro is a more compact version of the same format. It looks like we have MIDI numbers, 72, 74, 72, for C, D, C, and so on. And I believe 64 is encoding the velocity. That is how loud each note is. And since we haven't encoded any dynamics here, they're all 64s. And we have toward the end start and end offsets of each note. I think if we can go through note 1, 2, 3, 4, 5, 6, 7, 8, nine would be a rest, and at 1, 2, 3, 4, 5, 6, 7, 8, we have-- sorry.

We're not doing start and end. We're doing start, offset, and duration. And there we have a start offset note eight of 2.5, duration of one. And then the next element does not start at 3.5, but it just starts at 3.75. So we're not even encoding rests. We're saying that they are gaps in the musical format.

Director musices is also a lisp based format, I believe. That's why we have all these extra parentheses where we're loading things into a stack, and we can see that we're encoding the meter after the first note, which is kind of interesting. And 32nd notes are one over 32. That's kind of nice. The dotted eighth note is three 16th notes long. So we see 3/16 can be an interesting format.

SCORE has come up at least once in class. It is perhaps the best, most beautiful music editor ever. This is how a user might input SCORE, and it makes it a little bit easier to enter things. You're putting slashes between notes and so on, and then it generates an internal syntax that specifies exactly where on the page every note is.

And so musical engravers really like working with SCORE because you can precisely move things around. Probably a professional engraver would have moved the C at the beginning of the second measure a little bit to the right because it's a little bit close to that bar line, and you might confuse the bar line for a stem. So SCORE is sometimes still used, even though it hasn't been updated in decades.

This is MusicXML. As we've seen from Michael Good's presentation, it is a much more verbose format. Things are displayed-- you know-- it takes a lot more space to do it. But MusicXML was invented around the year 2000. Some of these other formats were around since the '80s. So there's a lot more space on the hard drive-- well, there are hard drives now-- to be able to store all this information. So here, Craig Sapp has simply highlighted the part 1 measure in order to see how this would work in MusicXML.

MEI is another format. You can't really see things here, but it's very similar to MusicXML. As we said, it's the format-- or as Michael Good said, is a format mostly used by professional musicologists and almost nobody else for encoding common Western music notation. I don't see it as having advantages over MusicXML, and in fact has many disadvantages because it's not supported by many formats. But MEI can also encode music such as Medieval music, which uses different representation formats. So it has a particular niche.

This is a standard MIDI file represented in binary. And it's a little bit unfair to compare the two because it's a binary format, so we can't really see anything. But look at what's being highlighted. Here we begin with 00 90, which is the indication for a note off. So we're ending the previous note. If we go to the beginning of the next line, we can see 80, which is the beginning of a note on, and then 4d, which-- let's see if I can still wear my hexadecimal.

So four times 16 is 64. Add d to that, that is-- what is that? I don't know, 14, 13. And so you end up somewhere around 78 or so, which is the representation of the note f. And then the 40 after that is 64 in decimal. And so that would be the velocity, how hard we're hitting. Since velocity hasn't been encoded in this and there's a number between 0 and 127, 64 seems a pretty good number.

And then, we date that the note is turned off with the 90, and we say it's turned off after a certain amount of time. So how much time has passed. And you'll see this 51 seems to be a representation of the length of a 16th note. If you really want to look at MIDI. you want to use a translator. There's one in music21. There's one other places that will translate this format into a series of events like no ons, no offs, pitch bends, time deltas, which move us forward in time. That will allow you to actually give this particular-- a fair shake.

You can see also this is NIFF format, which Craig Sapp has translated some of the hexadecimals into particular letters so that we can see what they are. Somewhere in the middle of the page I see clef, I see fing, fingering, I see a font change, glissando harp. I'm not sure that this is-- oh, these are just all the things that could be in here.

So we haven't even gotten to encoding the piece yet, but basically NIFF encodes where each shape is on the page. It is a format used by music scanners. It's not used very much anymore, but the idea is that it's much more important to get an exact representation of what the page looks like rather than what it might encode.

The last format, which in this presentation collects outputs first, isn't really a digital format at all, but it is the Braille music code, which is an encoding of music notation. That was created by Louis Braille at about the same time that he created the TeX Braille. So he was really interested in music encoding also. And we can see some parts of Braille that does encode a clef, even though it's not really used in the reading, because we're going to encode specific notes.

Why encode a clef? Because blind and visually impaired musicians often need to talk to musicians who have sight. So it's easy to say, well, you know, it's in treble clef and there's one flat key signature and so on. And so there's a time signature and then-- that's not October 5. It's octave five. We set the octave, and we see that there's a C8th. We encode an augmentation dot D 32nd, C 32nd, C 16th, F 16th, and so on.

Interestingly, we can usually get both the note and the duration into a six dot segment, which would represent six bits or less than a byte. So it's a very efficient encoding. That makes reading music with your fingers very fast.

If you want to know more about the Rosetta Stone and these various musical encodings, I have also included Craig Sapp's watch all video. Thanks for watching.