Artikulate: Ligeti Style for Any Audio

Yotam Mann
CS & Music, UC Berkeley
yotammann@gmail.com

ABSTRACT

This paper describes a novel audio visualizer that works within Ableton Live called Artikulate. Artikulate extracts audio features using the Zsa.descriptor library[1] and displays them in a style inspired by Rainer Wehinger's aural score to György Ligeti's Artikulation. Unlike conventional visualizers, Artikulate visualizes some of the spectral features of the audio file such as brightness, noisiness, spectral centroid, etc. Artikulate also focuses on features that are perceptually significant.

1. INTRODUCTION

Traditional music notation gives the familiar viewer a good understanding of the music that it represents in a static way. In the later half of the 20th century, and certainly so far in the 21st century, there has been a trend away from traditional music scores and notation and a trend towards working entirely within Digital Audio Workstations (DAW). The issue with DAWs is that they do not provide a sufficient static visualization for many tasks. A composer might have a hard time, for example, positioning a flute to line up metrically with a drum. Or, an editor could have a hard time remembering which track has the synthesizer and which has the room noise.

Other than a spectrograph, currently there are no good audio visualizers available for Digital Audio Workstations. The most widely used and integrated visualization for digital audio is the waveform, where amplitude is mapped to y-axis and time to x. Waveforms have a number of issues for users: firstly, they all look similar to each other. Getting lost in a sea of waveforms may cause errors such as deleting the wrong clip, or playing the wrong sample. This problem is especially bad when working with large libraries or many tracks of audio. Secondly, attacks are not clear. Though onset detection is still an unsolved problem in audio analysis, waveforms make no attempt to mark any of the available heuristics for determining a note onset. Waveforms also show no spectral data, so determining the frequencies within an audio clip is impossible by looking at a waveform.

One optimal solution would be a system that is as descriptive as music notation, but that can be applied to any audio. Then composers and arrangers can work with music the way they have for centuries, cutting audio and lining up instrument tracks using static visualizations. Though, at the moment, it is impossible to turn any an audio clip back into music notation, Artikulate aims to approximate the functionality of music notation and the precision of a waveform representation.

2. INSPIRATION

In the 1970's, a graphic designer, Rainer Wehinger, created a “score” for Ligeti's *Artikulation* (1958) (Illustration 1). This was a reversal of the purpose of a traditional score; instead of the score preceding the music and informing the players, the score came after the music, and informed the listener. Wehinger's score has a legend that displays the shapes along with their encodings. It has a unique and appealing aesthetic, and is easy to follow when viewed aligned with the music.

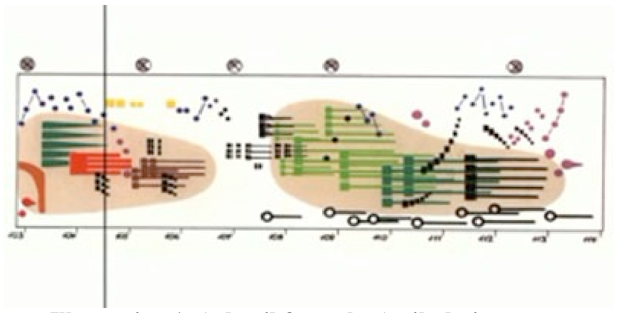
Illustration 1: A detail from the Artikulation score.

The *Artkulation* score occupies a space in between music notation and the waveform; it was created after the audio like a waveform, but displays the content of the audio file like a score. For this reason it underlies the design of the Artikulate software. The advantages it has over music notation is that by getting away from traditional note heads and rests, it is able to display more general audio, especially electronic sounds. Artikulate seeks to capture the hand-drawn look of Wehingers score and the descriptive power of audio analysis tools.

3. GOALS

At the moment, extracting features from general audio is a difficult problem. A computer, unlike Wehinger, has a hard time distinguishing instruments, finding note onsets and endings, and deciding what the significant features in an audio clip are. However, this is a burgeoning field, and there is starting to be more readily available tools for analyzing audio. Given the limitations of the existing software analysis tools, the purpose of Artikulate is not to turn any audio into music notation, but to occupy a similar area as the *Artikulation* score in between notation and the waveform.

The goal of Artikulate is firstly, to make different tracks in the DAW more differentiable from one another. This would allow the editor to keep track of clips more easily and not get lost in a stacks of gray waveforms. Secondly, it seeks to encode attributes of the audio file in the visualization so that whoever is working with the audio has some understanding of the sound of the clip without constantly having to listen to it, which can be very time consuming given many clips. Thirdly, Artikulate aims to bring a more aesthetically pleasing experience to working with digital audio by basing its visuals on Wehinger's.

4. RELATED WORK

There has been some research into extracting features from audio [3] [4], but not as much development on turning this research into meaningful visualizations. Current audio visualizers, other than spectrographs do not allow the user to create static visualizations. Instead, most visualizers create abstract, temporal representations of the audio. Commercial products like the iTunes and Windows Media Player visualizers are mostly focused on the graphics component; they use the amplitude data to create non-deterministic landscapes that can accompany the audio, but rarely inform the viewer to the content of the audio.

Some other visualizers go beyond just the amplitude data. Kai Siedenberg's software described in his paper, *An Exploration of Real-Time Visualizations of Musical Timbre* [2] attempts to display as many of the Zsa descriptors as possible; the result can be difficult to read and find what is the significant components of the visualization.

Another visualizer described by Ondřej Kubelka displays a few perceptual characteristics of digital audio [6]. It dynamically encodes an approximation of the mood, tempo, and balance (left and right) of a stereo audio file. The final display resembles a fountain with multicolored streams.

Artikulate attempts to bridge the gap between fairly meaningless, visually interesting representations like the iTunes visualizer, and dense, but meaningful encoding like the waveform, spectrograph, and Siedenberg's visualization.

5. METHODS

Artikulate was created entirely in the high-level visual programming language Max/MSP/Jitter.

The visuals are created using Jitter's OpenGL rendering.

5.1 Extracting Data

Getting significant data from the audio was the first step in making a powerful visualization. Zsa.descriptors is one of the first readily available tools for extracting high-level features from audio in real-time. The Zsa descriptors used in Artikulate are the spectral centroid and spread, flux, fundamental estimation and Bark coefficients (illustration 2). Alongside these descriptors is another powerful tool created at MIT and CNMAT, the Analyzer~ object in Max/MSP [5]. Analyzer~ provides data on noisiness, brightness, loudness, and attack detection.

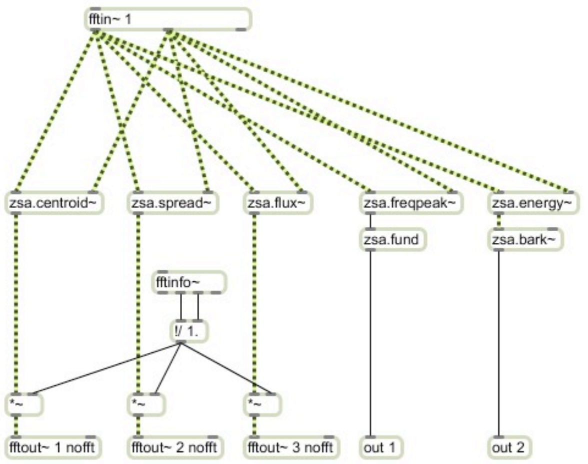

Illustration 2: Zsa Descriptors analysis on the Fast Fourier Transform

The power of these tools is that they are perceptually significant. Spectral spread and centroid deeply affect our perception of timbre. Bark coefficients are a model that breaks up the frequency spectrum into perceptually significant components. Onset detection is another powerful measurement that allows the software to create note heads, a feature that no other visualization software currently has. Though the onset detection occasionally missed attacks, it rarely gave false positives in the testing.

5.2 Encodings

I analyzed Wehinger's score and legend and extracted the shapes and colors that I would use for Artikulate, then mapped these to the descriptors, or groups of descriptors that were extracted in the analysis portion of the software.

5.2.1 Circles

The purpose of the software was to make different tracks differentiable, not to display exact values of the descriptors for analysis purposes. The circles encode the value of the noisiness and the minima and maxima of the loudness of the audio file. The amount of noisiness alters the hue of the circle. I chose to use hue even though it is not as discernible as value because it added more color and variation to the end product. The amount of loudness is redundantly encoded as the position along the y-axis starting from the top and the size of the circle. Circles are only drawn when the amount of flux is sufficiently high. This makes the circles appear most often when there are small note attacks (illustration 3).

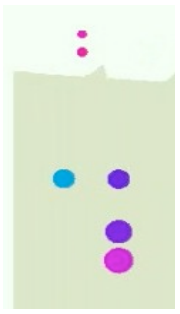

Illustration 3: Circles encoding noisiness, loudness, and spectral flux.

5.2.2 Background Blobs

The beige blobs in the background encode spectral centroid and spread (illustration 4). These are modeled after the beige globules of Wehinger's score which denote phrases. The centroid is at the center of the blob, and the amount of spread is how far from the center the blob spans. To make these shapes correspond more closely to the phrase marks, they stop whenever there is silence in the recording. The values are all on a logarithmic scale so that the blobs hover roughly around the center of the screen, and span most of the screen. Once again, the reasoning there is that the goal is not to show exact values, but rather to style the visuals after the score and differentiate the tracks.

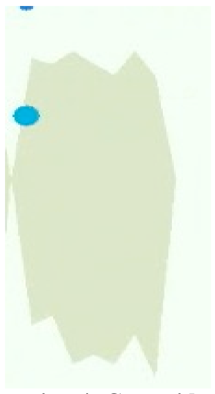
Illustration 4: Centroid at the center of the blob, and spectral spread is the distance from the center

5.2.3 Notes

The notes encode attack, frequency and brightness. They are only created when the Analysis~ object reports and attack in the audio. At that moment, the brightness is determined and encoded in three ordinal variables: the least bright attacks are circles with tails, then squares with tails, and finally triangles for the brightest attacks (illustration 5, 6, & 7).

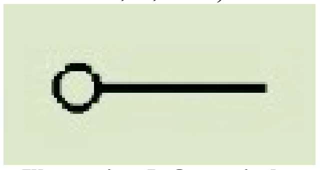
Illustration 5: Open circles have the dullest attack.

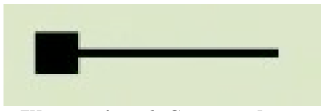
Illustration 6: Squares have the next brightest attack

Illustration 7: Triangle encodes very bright attacks such as percussion.

The y-axis position of the shapes encodes the frequency of the note and the length of the tail depends on how long the note lasts for. These are both determined by the Bark Coefficients. Bark Coefficients show the energy in a small number of frequency bands that are the most differentiable to the ear. The highest value Bark Coefficients are monitored until the values drop off. If the harmonic lasted sufficiently long, then it is displayed as the tail of a note. A note can have many strong harmonics; those are shown as stacked notes (illustration 7).

5.3 RENDERING

All of the images are created using OpenGL commands and rendered using Jitter. All of the text commands are pumped into a queue as they are generated from the audio signal. Once the audio file has finished processing, GL rendering is started.

5.4 ABLETON LIVE INTEGRATION

After completing the code in Max/MSP, it was fairly simple to make Artikulate a software plugin in Live. Live's new "Max for Live" integration allows Max patches to be easily converted into plugins. Once the plugin is added to a track, the visualization is generated. Then, every time that track is selected, Artikulate shows the visualization at the bottom of the screen, so the user is able to see both the plugin visualization and the waveform in the arrange window. Where ever the user places the playhead along the audio, Artikulate shows a visualization of the surrounding two seconds of audio. Artikulate will also scroll along with the audio file as it plays.

6. DISCUSSION

Artikulate was tested with a six differing audio clips, the resulting visualization was assessed for correlation to the audio file, expectation for the visualizations, and closeness in asthetic to the Artikuation score.

Visualizations that Artikulate produces are comparable in looks to Wehinger's *Artikulation* score. They are similar in terms of colors, shapes, and density of symbols with the audio files tested. For the most part, they parallel the audio file well. The system analyzes the audio file at a high sample rate, around a tenth of a second, which makes note attacks and tails line up well to the sounds.

Artikulate works best when notes and especially attacks in the audio are loud and clearly defined. There is poor results when the sounds are too complex or polyphonic, like an entire band playing on a single track. Luckily, many people working in DAWs generally split tracks into individual instruments or sounds.

The encoding that works the best is the notes. These generally correspond the best to the audio source, though, in densely rhythmic or quiet audio, they will miss many attacks. the ordinal encoding of brightness also works well for visually describing the audio. Frequencies and note length are also in accord with the sound of the audio file. Highly harmonic audio will generally have many stacked notes and notes will often start and end at the same time as they do in the audio file.

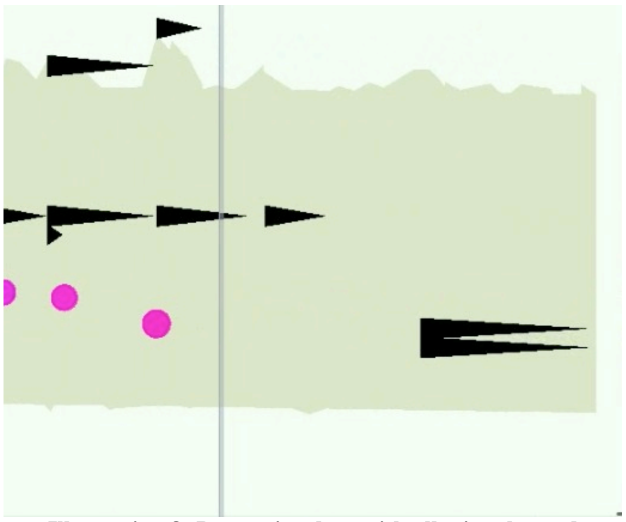
Illustration 8: Percussive data with all triangles and harmonics accurately encoded

The circles add a lot of color and differentiability between tracks and sections within a track, and also encode a lot of attacks that the notes miss since they are only generated during moments of high spectral flux. The placement on the y-axis of the circles does not correspond well to the audio. Louder section do not always have lower, larger circles than quiet sections. The color encoding of noisiness does not seem to coincide well to the audio either. Noisiness may not be as perceptually significant a variable as some of the things like flux and loudness.

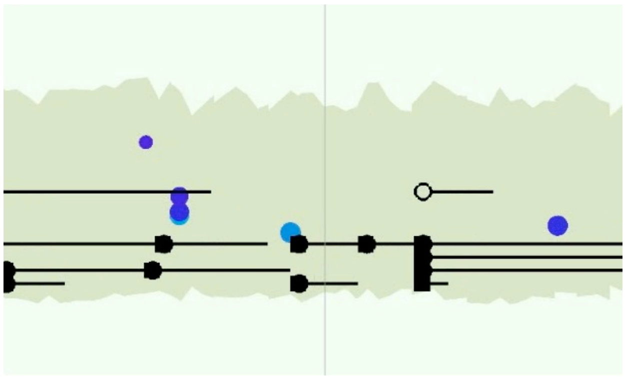
Illustration 9: This harmonically dense recording of piano has lots of notes and circles.

The background blobs encode the data accurately, but do not add significantly to the understandability or differentiability of the audio file. They help with the aesthetics by making the visualization resemble Wehinger's, but they are too choppy and do not follow phrases breaks very well.

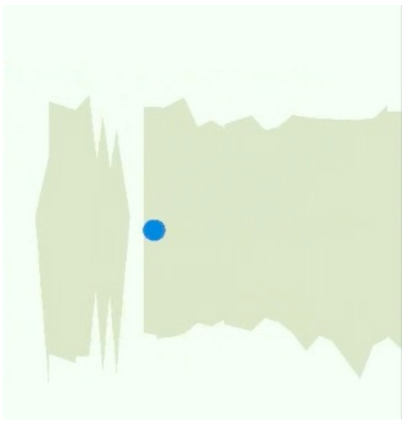
Illustration 10: Background shape with two "phrases"

The performance of Artikulate is fairly slow. Analysis and visualization creation run at about 2-5 times faster than the audio file, which is reasonable for short files, but a long wait for any audio over a minute. The image scrolls along with the audio well if there is only one plugin running; with more than that, rendering all of the OpenGL graphics becomes very processor heavy.

7. RESULTS

Artikulate bridges the gap between deterministic measurements of the audio signal like spectrographs and waveforms, and purely aesthetic, non-deterministic interpretations of the audio such as the iTunes visualizer. In contrast to commercial products like the iTunes visualizer, Artikulate corresponds closely to the audio signal and is not based on any randomness. It produces a static image of the audio file, like a score, which scrolls through as the audio plays. Like iTunes, Artikulate seeks to create a unique aesthetic experience that accompanies that audio.

Similar to more scientific approaches to audio visualizations, like Seidenberg's visualization, Artikulate heavily relies on the FFT of the audio source, not just the amplitude to determine perceptually significant attributes to display. Unlike these visualizers though, Artikulate's purpose is not just to convey information about the audio, but to display the significant components of that data in an interesting way. Artikulate is also unique in that it was designed after an existing audio visualization drawn by hand.

Artikulate's goals were reached in this iteration of the software. It produced visualizations which are easily differentiable from one another and correspond closely to the audio source. These visualizations inform the listener while the clip is playing, and the editor/arranger while the audio is not.

8. FUTURE WORK

Artikulate will require more fine tuning in future iterations. The background shape should be adjusted so that it still shows the centroid and spread, but scaled so that it is a smoother shape and corresponds better to musical “phrases”.

Artikulate could also include layout optimization so that the final product does not have overlaps and will look more like it was drawn by hand, like Wehinger's score. Greater smoothing in the OpenGL would also give a hand-drawn effect.

Future iterations might also move away from OpenGL, and adopt other visualization APIs like Protovis which would provide a higher level of graphics and greater interaction such as selection and zooming which at the moment Artikulate does not support because of the processing load of those operations[7].

9. ACKNOWLEDGMENTS

I would like to thank Maneesh Agrawala for teaching an interesting and informative class on visualizations.

10. REFERENCES

[1] Mikhail Malt, Emmanuel Jourdan. *Zsa.Descriptor: A Library for Real-Time Descriptor Analysis.* Proceedings of 5th Sound and Music Computing Conference, Berlin, 2008.

[2] Kai Siedenberg. *An Exploration of Real-Time Visualizations of Musical Timbre*. CNMAT, Berkeley, 2009.

[3] Juan José Burred and Geoffroy Peeters. *An Adaptive System for Music Classification and Tagging.* Proceedings from the 3rd International Workshop on Learning Semantics of Audio Signals. Graz, Austria, 2009.

[4] Anssi Klapuri. *Extracting meaningful auditory objects from music signals: methods and applications.* Proceedings from the 3rd International Workshop on Learning Semantics of Audio Signals. Graz, Austria, 2009.

[5] Tristan Jehan, Adrian Freed, Matt Wright, and Michael Zbyszynski. *Analyzer~ Object.* Massachusetts Institute of Technology & CNMAT, 2001.

[6] Ondřej Kubelka. *Interactive music visualization.* Czech Technical University.

[7] Michael Bostock and Jeffrey Heer. *Protovis: A Graphical Toolkit for Visualization*. Stanford University.