MUSIC VISUALIZATION USING WEBGL

Graeme Clarke
University Of Victoria
(CSC - Undergrad)
gdclarke@uvic.ca

Spencer Vatrt-Watts
University Of Victoria
(CSC - Undergrad)
asvw@uvic.ca

George Tzanetakis
University of Victoria
(Instructor)
gtzan@cs.uvic.ca

ABSTRACT

Music and audio visualizers have a multitude of uses through music information retrieval. There is no singular method of extracting music information for the purposes of visualization, as audio can be visualized different ways for different purposes. Audio visualizers can be used in audio production/editing for viewing EQ, audio balance, relativity to Equal-loudness curves, and much more. Audio visualizers may also be used simply to create interesting images to accompany music. In any case, it is important that music visualizers are accurate, versatile, and usable across multiple platforms. For this reason, we will be providing a number of audio extraction techniques for use in various visualization techniques, all implemented using the Javascript rendering API, WebGL.

1. INTRODUCTION

Because WebGL is a browser-based API that requires no plug-ins and works on every major mobile/desktop browser, we believe it is an ideal choice for this project. At the time of this project's conception, neither of the contributing researchers (Graeme Clarke and Spencer Vatrt-Watts) have experience using WebGL. We believe our shared experience of using C++ will help our understanding of GLSL (WebGL's shader language which shares similarities to C/C++ [1]), and we also acknowledge that the process of learning Javascript and WebGL could potentially serve both of us practically in the future. This creates an uncertainty in the development of the project, one that may affect our ability to accurately estimate how long it will take to achieve certain goals (see section 4). That being said, the primary focus of this project is to create relatively simple platform-versatile audio visualizers - as supposed to using advanced WebGL features. Furthermore, our initial goals are to only utilize 2D features, so we can focus on delivering accurate representations of audio information. If we are able to complete any of our initially proposed goals ahead of schedule, we will experiment with the more advanced features of WebGL (3D graphics), and more advanced audio analysis techniques.

A major focus of this project is to not only create visualizations using the previously mentioned technologies, but also to create visualizations that use a variety of music information retrieval techniques. Initially, we intend to utilize more simple extraction techniques such as the Fourier transform [2], and its variations. Following this, we intend to utilize a variety of audio information extraction techniques from academic papers, and other academic sources (depending on their accessibility, and our ability to implement them in Javascript using WebGL).

For each of the techniques used, we plan to visually represent the retrieved information in multiple ways. We believe that this could increase viewers' understanding of a given technique (in a visualized context), regardless of what background they have in MIR. In other words, an individual who may have use for a certain technique outlined in an academic paper would be more likely to understand it when presented with a visual representation of it. For example, if we implement a MIR technique that classifies the genre breakup of a song, we could potentially visualize this with color, shape or more advanced visual techniques. Section 2 provides a more detailed summary of how we plan to utilize audio information extraction techniques and the research we performed on them, while Section 3 provides a detailed account of how WebGL works and why it is the ideal API for this project.

2. AUDIO VISUALIZATION

We were first inspired to research audio visualization after using sndpeek [3], a real-time 3D animated audio visualizer. During our research, we discovered several examples of other existing visualizers, on GitHub [4], Chrome Experiments [5], and through Google Play Music [6].

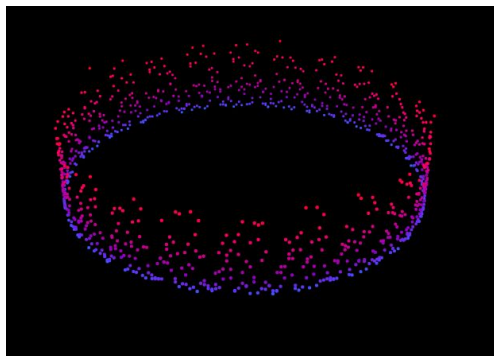

Figure 1 - A WebGL-based Audio Visualizer, created by

Sonia Boller [7]

These programs visualized various things using a variety of methods, from scanning the frequencies in an audio file, to creating a visualization of code in a file. Since we are strictly considering audio visualization, our focus needed to be narrowed a bit. The majority of the audio visualizers encountered during research functioned the same way we intend to start ours: by using some version of the Fourier transform to split an audio signal into different frequency bins. However, there were outliers as well. One article discusses the concept of visualizing audio based on self-similarity [8], by displaying the acoustic similarity between any two instants of the audio as a 2D grid-like image. Each cell in the grid corresponds to the similarity between two specific points in time in the audio. Another paper [9] details how researchers were able to model songs based upon the relationship between their changing tempos and loudness levels. For example, if a song became louder and sped up at the same time, this relationship would be visible in the generated visualization.

Other examples of note that came up during our research of audio visualization were visualization using audio tags and features [10], and the extraction of vocals from audio files [11, 12, 13]. Audio features are indicators in numerical form, extracted from an audio file using various MIR techniques. The paper regarding audio tags and features describes a process where the similarities in audio features and tags between different songs was displayed in the form of force-directed graphs [10]. The vocal extraction papers were also very interesting, and varied in focus. The first discusses the use of an SVM that feeds data to a 2D Adaptive PCLA algorithm, which then generates output in the form of background sound and vocals separated. The SVM is used to label the parts of a song that contain vocals and the parts that contain only background music. The labeled song is then passed to the 2D Adaptive PCLA algorithm, which then learns the spectral signature of the background and uses it to learns the spectral signature of the vocals, thus enabling it to accurately separate them [11]. The second vocal extraction paper discusses the separation of voice and music through the assumption that repetition is a key feature of music. It proposed that once the repeating structure of a certain musical piece was found, a repeating segment model could be generated. Then, each time-frequency bin in the piece is scanned, and those that are similar to the model are labelled as the repeating background music, with the remainder labelled as the vocals. Thus, the repeating background sounds and vocals are split [12]. The third and final paper analyzed discusses the extraction of vocals from music using neural nets. It details the training of a deep neural net with around a billion parameters to estimate the ideal binary mask in order to separate vocal sounds from other music [13]. The research we conducted on the topics of audio tags, features and vocal extraction inspired us to plan to add genre classification and instrument identification to our visualizer, if we have time after the main implementation is completed (see section 4).

3. WEBGL DEVELOPMENT

To understand why WebGL is an ideal API for the development of this project, one must first understand its basic workflow. WebGL programs are written in Javascript, and embedded into HTML documents. This code then calls the WebGL API functions, which utilizes OpenGL ES 2.0 or DirectX 9.0 on a given user's machine to render/display graphics within their browser [14]. This is best visualized by Figure 2:

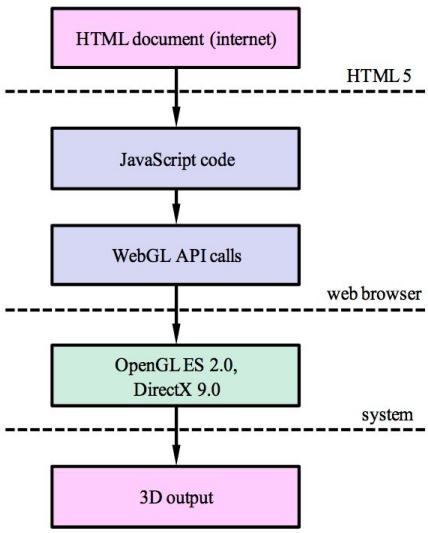

Figure 2 - Workflow of WebGL-based Graphics Output [14]

This allows for similar rendering techniques to standard OpenGL, while not requiring large amounts of complex (system-specific) computation on a given machine. This is the primary reason for our choosing of WebGL, so that we can develop on any platform, for any platform. Furthermore, because it was recently introduced in 2011 [14], and is being widely used for various projects and experiments today, there are a multitude of resources available. We intend to utilize these resources both as visual inspiration, and as means to quickly and effectively learn WebGL.

For similar reasons, we also intend to utilize *three.js,* a Javascript 3D library that "makes WebGL simple" [15]. In preemptively researching for this project, we discovered that almost all of the most impressive graphics experiments and visualizers using WebGL also used three.js. Primarily, *three.js* will save

us time and effort; we won't have to write basic subclasses for audio listening, vector and matrix math, camera functions and morre. A large list of *three.js* projects can be found at *threejs.org*, including various projects made by Google. The apparent technical efficiency of WebGL and three.js validate their potential effectiveness as a modern 3D graphics toolkit. This, in combination with their multitude of relevant educational resources, make them ideal for this project.

4. TIMELINE

By each listed date, the following work will be completed:

- **February 27** - Submit design specification, to be adapted throughout project completion
- **March 7** - Complete a basic implementation that utilizes Fast Fourier Transforms to display frequencies in real time
- **March 20** - Add the ability to display EQ information as well as Fourier transform frequency information in a variety of ways, i.e. bar graphs, line graphs, spheroids etc.
- **April 4** - Submit final report

Stretch goals (To be completed if possible, time permitting):

- *SoundCloud processing* - Perform visualization on audio files from user-provided SoundCloud links
- *Genre classification* - Attempt to classify the genre of a provided audio file by spectral centroid analysis, and represent it visually
- *Genre-based visualization* - Visualize music from varying genres differently, utilizing different techniques depending on the classified genre of the inputted audio file
- *Identification of different instruments* - Split audio visualizations by instrument type, for example, visualize the drums, bass, vocals and guitar in a song separately
- *Advanced 3D graphics* - Represent all of the previously made visualizations in a higher fidelity form, potentially providing additional information from an additional dimension

5. ROLES

For this design specification:

Graeme Clarke - Wrote the Abstract, Introduction and WebGL sections. Edited the other sections.

Spencer Vatrt-Watts - Wrote the Audio Visualization, Timeline and Roles sections. Edited the other sections.

Going forward, both team members will work collaboratively on the implementation process. Graeme has more experience with 3D graphics, having studied them both in class and as a hobby, so his skills will be particularly useful for 3D visualizations, shaders, etc.. Spencer has spent time learning and working with JavaScript, so he will be an asset when the team is working with WebGL through three.js.

6. REFERENCES

[1] "WebGL", *En.wikipedia.org*, 2017. [Online]. Available: https://en.wikipedia.org/wiki/WebGL. [Accessed: 27- Feb- 2017].

[2] B. Milewski, "The Fourier Transform", *Relisoft.com*, 2017. [Online]. Available: http://www.relisoft.com/Science/Physics/sound.html . [Accessed: 27- Feb- 2017].

[3] "sndpeek : real-time audio visualization", *Gewang.com*, 2017. [Online]. Available: http://www.gewang.com/software/sndpeek/. [Accessed: 27- Feb- 2017].

[4] "GitHub - willianjusten/awesome-audio-visualization: A curated list about Audio Visualization.", *Github.com*, 2017. [Online]. Available: https://github.com/willianjusten/awesome-audio-visualization. [Accessed: 27- Feb- 2017].

[5] "GitHub - wizgrav/clubber: Application of music theory in audio reactive visualizations", *Github.com*, 2017. [Online]. Available: https://github.com/wizgrav/clubber/. [Accessed: 27- Feb- 2017].

[6] J. Sneddon, "How to Try The New Google Play Music Visualizer", *OMG! Chrome!*, 2017. [Online]. Available: http://www.omgchrome.com/enable-google-music-particles-visualizer/. [Accessed: 27- Feb- 2017].

[7] "Audible Visuals", *Soniaboller.github.io*, 2017. [Online]. Available: https://soniaboller.github.io/audible-visuals/. [Accessed: 27- Feb- 2017].

[8] J. Foote, "Visualizing music and audio using self-similarity", *ACM Digital Library*, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=319463.319472. [Accessed: 27- Feb- 2017].

[9] S. Dixon, W. Goebl and G. Widmer, "The Performance Worm: Real Time Visualisation of Expression based on Langner's Tempo-Loudness Animation", *OFAI*, 2017. [Online]. Available: http://www.ofai.at/cgi-bin/get-tr?paper=oefai-tr-2002-15.pdf. [Accessed: 27- Feb- 2017].

[10] T. Lin, W. Woon and J. Peng, "Music Visualization Using Audio Features and Tags", *ECMLPKDD*, 2017. [Online]. Available: http://www.ecmlpkdd2013.org/wp-content/uploads/2013/09/MLMU_Lin.pdf. [Accessed: 27- Feb- 2017].

[11] D. Mendez, T. Pondicherry and C. Young, "Extracting vocal sources from master audio recordings", *CS 229: Machine Learning*, 2017. [Online]. Available: http://cs229.stanford.edu/proj2012/MendezPondicherryYoung-ExtractingVocalSourcesFromMasterAudioRecordings.pdf. [Accessed: 28- Feb- 2017].

[12] Z. RAFII and B. Pardo, "A SIMPLE MUSIC/VOICE SEPARATION METHOD BASED ON THE EXTRACTION OF THE REPEATING MUSICAL STRUCTURE", *Interactive Audio Lab*, 2017. [Online]. Available: http://music.cs.northwestern.edu/publications/Rafii-Pardo%20-%20A%20Simple%20Music-Voice%20Separation%20Method%20based%20on%20the%20Extraction%20of%20the%20Repeating%20Musical%20Structure%20-%20ICASSP%202011.pdf. [Accessed: 28- Feb- 2017].

[13] A. Simpson, G. Roma and M. Plumbley, "Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network", 2017. [Online]. Available: https://pdfs.semanticscholar.org/8a54/a76f7c2b95bd2514b461253e239f8f1ea427.pdf. [Accessed: 28- Feb- 2017].

[14] N. Baek, "A Standalone WebGL Supporting Architecture", *Waset.org*, 2017. [Online]. Available: http://waset.org/publications/8553/a-standalone-webgl-supporting-architecture. [Accessed: 28- Feb- 2017].

[15] "three.js - Javascript 3D library", *Threejs.org*, 2017. [Online]. Available: https://threejs.org/. [Accessed: 28- Feb- 2017].