Matthew Fong

IGME 330 - Audio Visualizer Documentation

**Usability and overall UX:** In terms of usability and user experience, I feel my audio visualizer controls are both clear and intuitive. All checkboxes, buttons, and sliders are clearly labeled. I have also grouped related features with each other and separated unrelated features with a thematic break, <hr> tag, in the HTML/CSS. In addition, the controls are evenly spaced, with checkboxes and sliders being vertically aligned with its respective labels.

**Interactive Design:** Although my audio visualizer does not have mouse interaction, it does however, give the user plenty of opportunities to manipulate the visual presentation of the audio visualizer. I have radio buttons that allow the user to switch between effects being displayed, as well as being able to switch between two versions of the audio visualizer. Furthermore, I have six checkboxes that allow the user to toggle on/off extra visual features and manipulate the color of the pixels on the canvas screen. For sliders, the user can adjust the amount of bass boost and apply a delay to the currently playing song. The user can also enter fullscreen mode through the simple click of a button. My audio visualizer features eight really good songs for the user to listen to as well.

**Canvas API:** For bitmap info fetching with ctx.getImageData() and ctx.putImageData(), the user can manipulate the pixels on the canvas screen with three checkboxes that do the following: apply a greyscale, threshold, and/or invert effect. The audio data (frequency or waveform) is displayed on the canvas through lines, rectangles, arcs (ovals and circles), and quadratic bezier curves. I also implemented a gradient that is used to stroke and fill some lines and rectangles in "The REAL VISUALIZER" version. The use of context state variables are also used properly when drawing shapes and visual effects. Additionally, canvas transforms, specifically ctx.translate() and ctx.scale() are used appropriately, specifically scaling an arc to create the "Horizon Effect". Pushing and popping the canvas drawing states are also used appropriately; ctx.save() and ctx.restore() are used before and after transforming any drawing states. They are also used before and after applying shadow blurs to canvas text.

**Web Audio API:** With the checkbox feature, the user can toggle between viewing frequency data and waveform data. Aside from the analyzer node, I have implemented a second audio node called delayNode. The user can manipulate the amount of delay applied to the currently playing song.

**Media and Presentation and CSS/HTML:** The eight song files are all in mp3 format and are more than 30 seconds in length. I have embed a web font and applied it to both the HTML portion of the project and the canvas text. In terms of an attractive visual design, I made the background and controls a dark color so the canvas' colors are brighter, has more contrast, and catches the user's eyes more.

**Code:**  In terms of code, I feel I have met all the requirements stated in the project document. I have included "use strict"; all my JavaScript code is in an IIFE. Naming conventions of variables and functions are proper; code is clean, properly indented, and well-commented. I have accounted for the principal of D.R.Y - Don't Repeat Yourself. There are also no console.log() calls or debugger; statements. Furthermore, I have not used any JQuery or JS libraries without prior approval. Any sources and reference has been included in both my code and documentation.

**Above and Beyond:** For the above and beyond, I feel I have made my project very visually pleasing. I have implemented a great handful of visual effects. I also implemented a display feature to the canvas that displays the currently playing song and duration of the song. I also implemented a particle system that adds to the animation along with the frequency data being displayed through shapes. I feel implementing two versions of the audio visualizer is also very neat in the sense that I initially show the user a very simple version of the audio visualizer. Then, after absorbing this in, they can toggle-view a much more visually pleasing and "cooler" audio visualizer and begin to absorb that. I feel the project as a whole comes together very nicely.

**What went Wrong:** Although I am mostly satisfied with my overall project, there are a few things I would have liked to add/change. Most importantly, I would have liked to optimize some effects more. Specifically, I would have liked to further optimize the horizon effect and bezier effect. Additionally, I would have liked to changed my bezier effect quite a bit. Instead of drawing four bezier curves with sort of "fixed" control points, I would have liked to altered the controls points so that they correspond to frequency data from the song files, expanding along the outline of the image icon, similar to what my "Surrounding Lines Effect" is doing. For a clearer description, I have included a youtube link exemplifying my inspiration. From about 0:30 seconds into the video is when the effect can be clearly seen.

https://www.youtube.com/watch?v=s8XIgR5OGJc

If more time permitted (and more coffee), I would have also liked to make the speed of which the particles move also correspond to the frequency data. Therefore, the higher the frequency data, the faster the particles would move; the lower, the slower they would move. I would also like to implement some sort of mouse interaction as well.

Finally, I would have also liked to add more effects, whether it be drawing more shapes or allowing the user to apply more filters, without them heavily impacting performance.

**What went Right:** Overall, I am very pleased with how my audio visualizer project came out. Throughout the whole process of building the audio visualizer, I feel I learned and experienced a great amount with not only JavaScript coding, but also some design aspects; I feel I have a stronger understanding of canvas as a whole. I am also very pleased with the visual effects that I was able to implement and be able to put everything together; from the background image I borrowed online, to the image icon of one of Justin Timberlake's albums and myself, to the various shapes and colors I used to graphically represent incoming frequency/waveform data.

**Self-Grade:** On a scale of 0-100%, I would grade myself a 97%.

**References/Sources:**

* Font used: https://fonts.googleapis.com/css?family=Maven+Pro
* Vertical-Aligning Checkboxes: http://stackoverflow.com/questions/15230820/vertical-align-checkbox-label
* Background-Image: https://s-media-cache-ak0.pinimg.com/originals/5d/6c/31/5d6c3144809459f595aabef653bdedb8.jpg
* Greyscale and Threshold Tutorial: https://www.html5rocks.com/en/tutorials/canvas/imagefilters/
* Particle System Tutorial:

http://www.howtosolutions.net/2016/09/javascript-canvas-simple-particle-system/

**Inspirations:**

* https://www.youtube.com/watch?v=JhCEXRgbc\_M
* https://www.youtube.com/watch?v=28GpKacWLWI
* https://www.youtube.com/watch?v=gkime9M4z34