Nate Glick

IGME 330

Audio Visualizer Documentation

1. Requirements
   1. Usability and Overall UX
      1. For my visualizer, I created a simple HUD in the upper left corner of the page that has clearly labeled controls for all the aspects of the visualizer the user can control: current song, length of the audio bars, volume, number of audio bars, choice of waveform vs frequency data to visualize, color choices (random color or a gradient), and toggles for bezier curves, noise, and fullscreen mode. At the bottom of the page is the audio control for playing and pausing the music. The HUD will also become hidden when the mouse is inactive for a few seconds to help the user focus on the visualizer.
   2. Interaction Design
      1. In addition to including some default songs to choose from, the user may also specify a local music file using the “choose file” button. There are also dropdowns, checkboxes, and sliders for each of the controls mentioned above. Finally, the user is able to click and drag the visualization around the page in order to pan the audio.
   3. Canvas API
      1. There is an option to enable noise, which takes advantage of bitmap manipulation. There is also an unused function that will fully fade the background, but it lags the page too much so I left it out. Audio data is mainly drawn to the canvas using lines that are rotated around the center of the canvas. The circle in the middle of the page is resized depending on the average of the frequency data, which is useful for showing the beat of the song without complex beat detection. A series of quadratic curves that can be toggled on and off display the frequency data of the audio. And finally, a gradient can be toggled on for the color of the main audio visualization in the middle of the page.
   4. Web Audio API
      1. Both frequency and waveform data could be toggled on and off for the main visualization, while only frequency is used for the bezier curves. A panner node is used to pan audio when the user drags the visualization on the page, and a gain node is used to control the volume.
   5. Media and Presentation and CSS/HTML
      1. All sound clips are at least 30 seconds long (although any the user selects locally might not be 30 seconds long). “Source Sans Pro” is imported from fonts.google.com for use on the page. The page itself has a simple dark theme, which is easy to look at, and the visualization uses many colors to help it “pop” off the page.
   6. Code
      1. jQuery was used in order to hide the HUD after a few seconds, which was an allowed use of jQuery. Strict mode is used. Let is used for all variable declarations. Most if not all of the code is not repeated. All variables and functions are named with a lowercase letter. All lines end in a semicolon. Code is commented. No console.log calls or debugger calls are in the final code.
   7. Above and Beyond
      1. The biggest addition I included was the ability to directly interact with the visualization by allowing the user to click and drag around the page to pan the audio. In addition, I included a “choose file” button to allow the user to select their own music if they so choose.
2. What went right and/or wrong
   1. Overall, I think the project went well. I originally created a version of the AV that was very bare-bones and only had a line of bars to represent audio data across the screen. As I started to implement more of the requirements, I began to make the AV more interesting - I added color and changed the way the audio was visualized - from a single line to a circular pattern, which is much more interesting. Not too much went wrong, but there is a slight bug where there will be a flash of random colors when swapping colors for the visualization, I’m not sure what causes it or how to fix it, but it isn’t very noticeable. Also, I tried to force fit bezier curves into the project, so the final decision on how they were included isn’t the ideal solution, but it still looks nice and functions correctly. In the end I also decided to scrap color inversion as it produced a strange flashing effect which was very had to look at, and I opted for background noise instead.
3. Partner Contribution
   1. Nate Glick did all of the javascript code, which the other partner, Nate Glick, did all of the HTML/CSS
4. Grade
   1. Personally, I would give myself a 95%. Although I think I covered all of the requirements for the project, I feel that I could have really worked out some of my problems, such as the random color flashes or what exactly to do with bezier curves. In addition, I would have liked to add more interesting audio filters to the project if I had more time. I think that overall I could have added more polish to the final result, and because of that I think I should take a few points off myself.