Word count (398)

Waves visualisation

- This visualisation displays the input from the web camera, and has rectangles change in size according to the brightness of that particular pixel. This means that the user moving can affect the size of the rectangles. There is also a waveform that is affected by the current track. Overall, the visualisation is not only affected by the movement of the user, but the changes in the track.
- The uniqueness is what extends the templates initial design. Not only does it add the element of user interaction to the visualisation, but loading pixels from the camera was learnt outside of what UoL provided.
- The code follows OOP techniques, keeping all the relevant variables and methods within one file. A nested loop allows a grid of rectangles to be drawn, where the size of the rectangles is dependent on the variables responsible for pixel brightness and sound waveforms.

Customizable settings

- To make each visualisation versatile and personalized, all visualisations have customizable settings that only apply to that visualisation.
- Certain changes in visualisations weren't enough to classify it as an extension. However, adding dynamically changeable settings, that change in real time extend each visualisation beyond its surface level. It also adds universal settings that can be used throughout the app.
- Each setting is its own constructor function, and all changeable settings are done with parametrized constructors or if statements. All setting related variables are private, and if they are changed, then the visualisation changes accordingly.

UI and playback controls.

- This extension added multiple tracks, and buttons to cycle through each track. It added a scrubber to scrub to a particular part in the track, to have better control over the music that's playing.
- Having multiple tracks allows the user to see all the subtle differences in a visualisation, and truly see how they react to the type of music that is playing.
- The code for each button is a separate constructor function, in its own file. Using prototypes in ES6 I was able to create a generic button constructor, and have the relevant buttons "extend" from that prototype. Each button contains various methods, which include "draw" and "click". All buttons are made, and all methods are called in the controls and input object, to keep with encapsulation and OOP.

All code has been changed to an ES6 syntax, and all libraries have been updated to their most recent versions.