Jaiden Atterbury
jatter41@uw.edu
Section AA
02/05/2024

A4: Creative Data Visualization in p5js

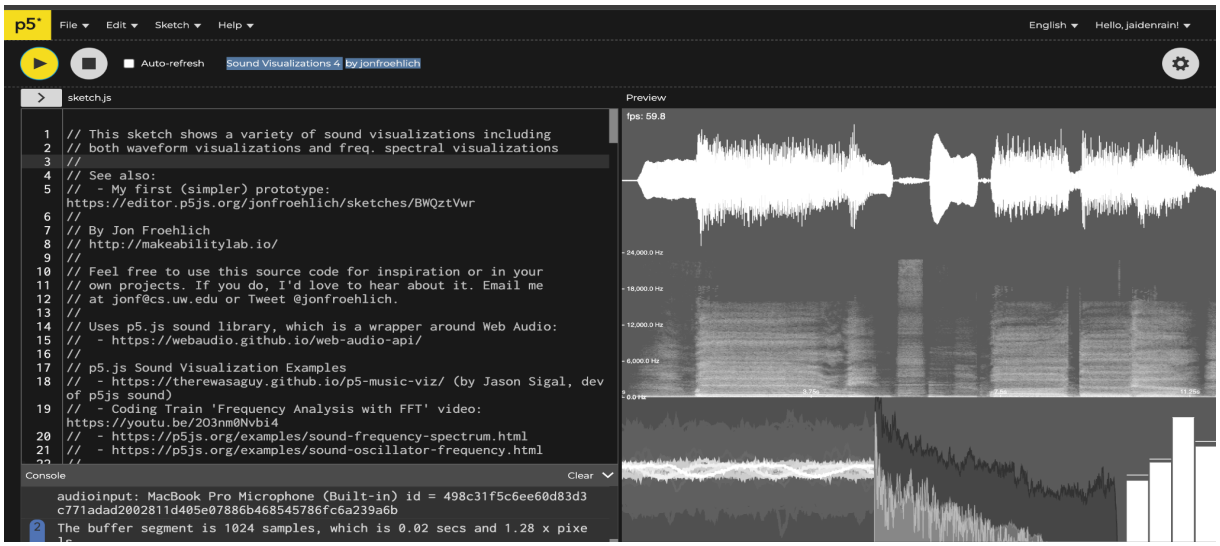
Link to assignment specification
https://docs.google.com/document/d/1DpbYsOpuNmRnm-6CwQJ0Hy2aUPhs5osnY0DJZS-bca4/edit

Link to my p5.js sound visualization from Step 3
https://editor.p5js.org/jaidenrain/sketches/CmtbQBhu2

Step 1: Complete the Intro to p5.js Tutorial
In this step, to begin the assignment, I went through Johannes Preis' Introduction to p5.js, which, through an evolving example with a circle, was an excellent introduction to p5.js, the coding editor, basic graphic rendering, and interactivity. In this part, I also looked at the official p5.js Getting Started guide, which paralleled some of Preis' content but in much less detail.

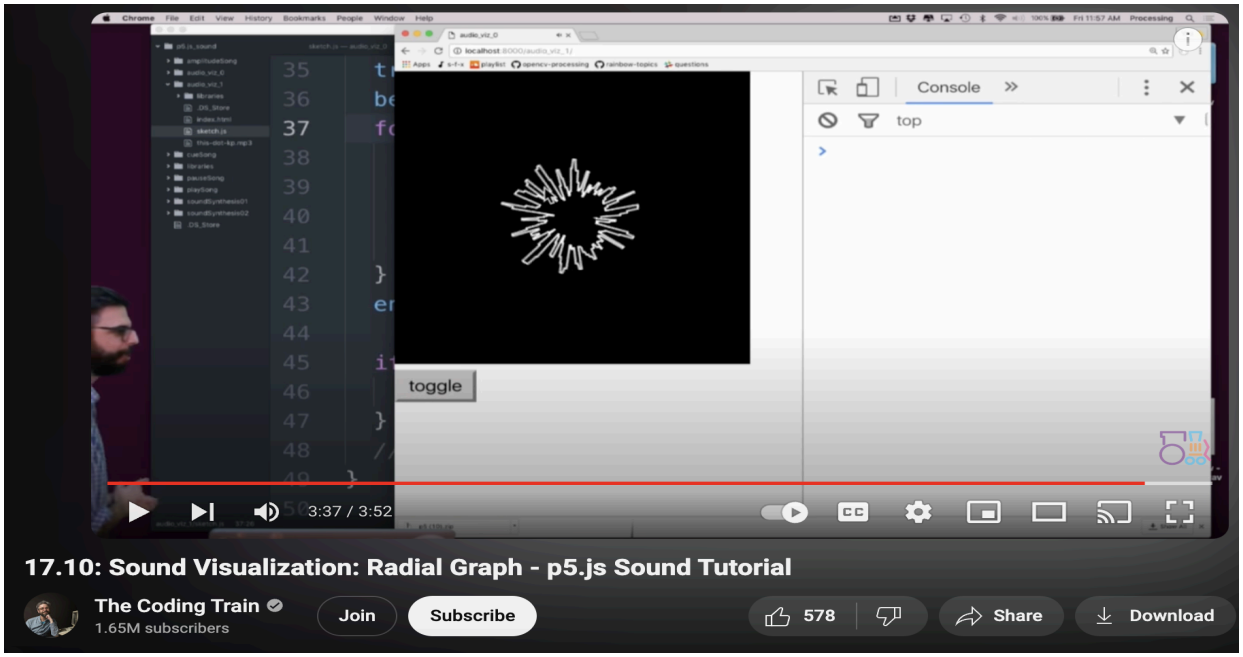
Step 2: Design Research
In this step, the goal was to search online for inspiration. In particular, I browsed the internet to see how others have used p5.js to visualize sound. Specifically I have collected five examples to showcase. In each example, a screenshot and link to the visualization, as well as a brief description of the visualization itself is included.

1. Sound Visualizations 4 - By Jon Froehlich



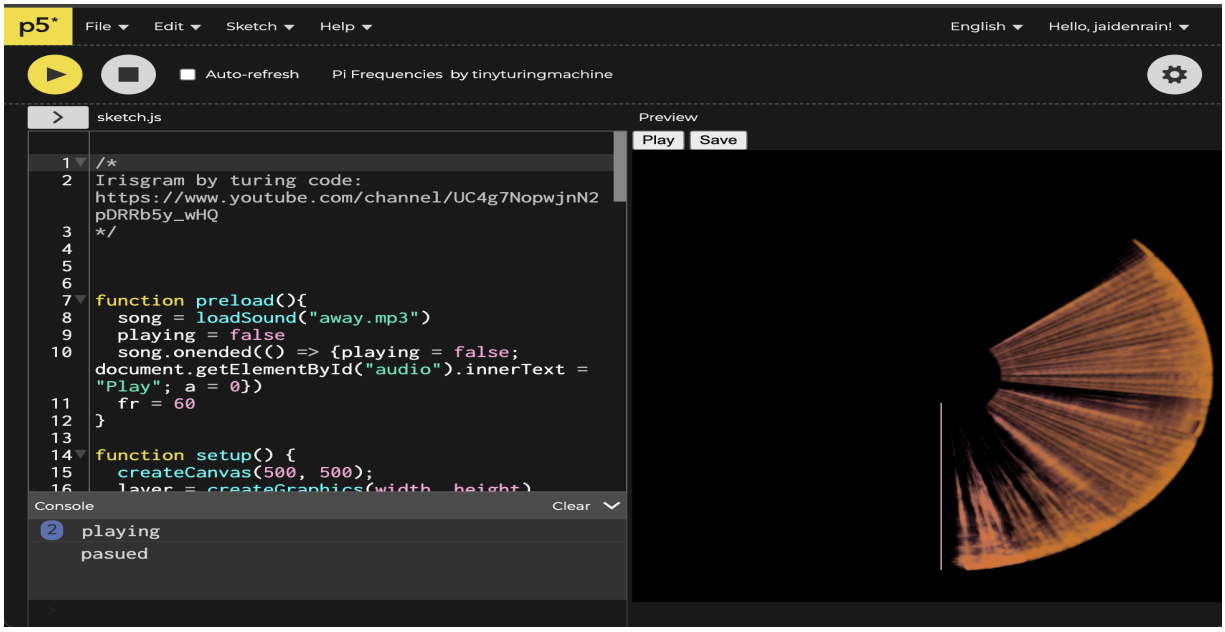
This visualization, which was made and posted on Reddit by a previous instructor of this course (Jon Froehlich), is the fourth iteration of the “Sound Visualizations” series made by the author. Here is the original, and here is the second iteration (I could not find the third version). In this visualization, the sketch shows a variety of sound visualizations including waveform visualizations and frequency spectral visualizations. In particular, the visualization contains a “real-time waveform of the current fast fourier transform buffer,” a “real-time frequency spectrum of the current fast fourier transform buffer,” a “scrolling spectrogram,” a “scrolling waveform view,” and lastly, a “spectrum bar graph.” All of these graphs rely on user audio input.

2. Sound Visualizations: Radial Graph - By The Coding Train



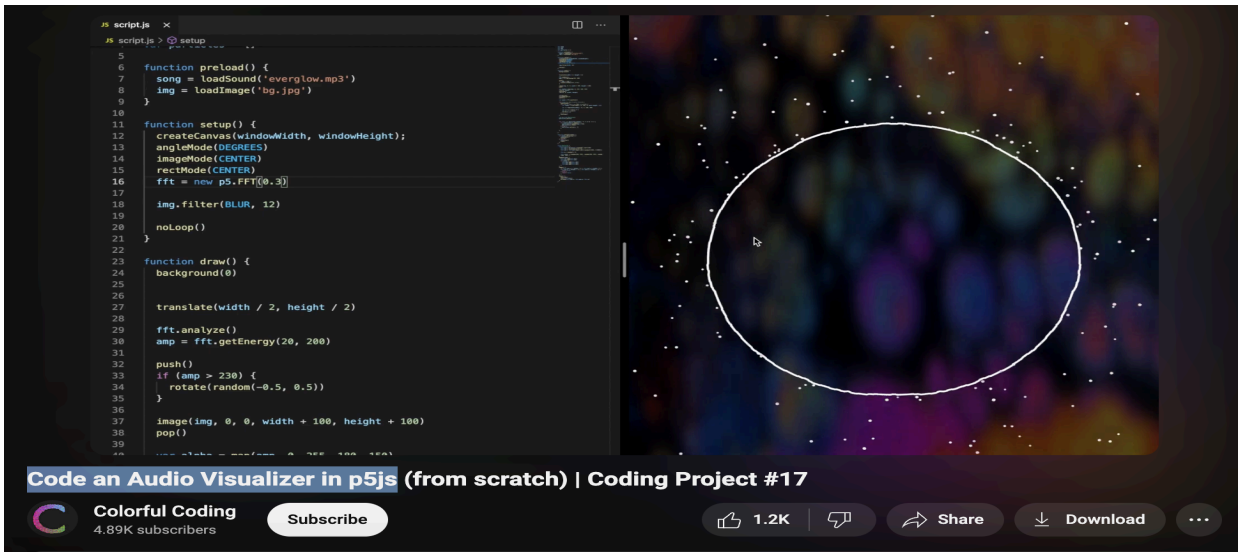
This visualization, which was made and posted on YouTube by Daniel Shiffman of The Coding Train, plots and translates the radius of a circle based on the loudness of audio input. In particular, this audio input comes in the form of an mp3 file called “this-dot-kp.mp3.” As mentioned in the previous sentences, as the mp3 is played, the loudness of the audio file is plotted as the radius of a circle, with the length of the radius being dependent on the amplitude/volume level.

3. [Sound Visualization - 'Irisgram' (Spectrogram)](#) - By turing code



This visualization, which was made and posted on [YouTube](#) by turing code, produces a circular spectrogram, which he describes as "nothing but a neat way to visualize audio information." In particular, the spectrogram uses information on the frequency in hertz of an mp3 file in order to sequentially build up the spectrogram. The length of each line represents the frequency in hertz of the audio file at a given time, with time being represented by the angle of the circle.

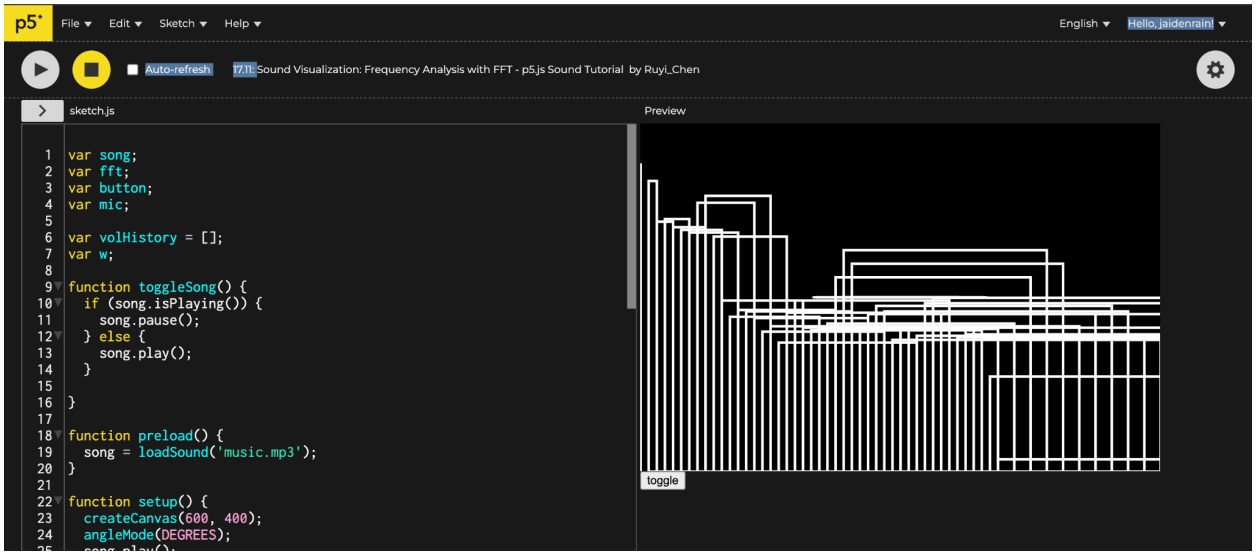
4. [Code an Audio Visualizer in p5js](#) - By Colorful Coding



This visualization, which was made and posted on [YouTube](#) by Colorful Coding, produces an "audio visualizer" that represents the intensity of audio in music. In

particular, as explained in the video, the visualizer can be broken up into three components: the waveform in the middle, the particles around the waveform, and the responsive background behind the waveform. Specifically, this program draws a circle that will stretch and vibrate based on aspects of sound in the given mp3 file.

5. Sound Visualization: Frequency Analysis with FFT - By Ruyi Chen



This visualization, created by Ruyi Chen, is an adaptation of the visualization made in this YouTube video by the channel The Coding Train. In this visualization, the user uses the Fast Fourier Transform (FFT) object in p5 to analyze the frequencies/spectrum array of a sound file. The amplitude at different frequencies are mapped onto the sketch through the use of rectangles.

Step 3: Create Your Own Real-time Sound Visualization

In this step, the goal was to create my own real-time sound visualization. In this section, I will include screenshots of my audio visualization, a link, and a multiple paragraph description going over my process and describing the visualization itself.

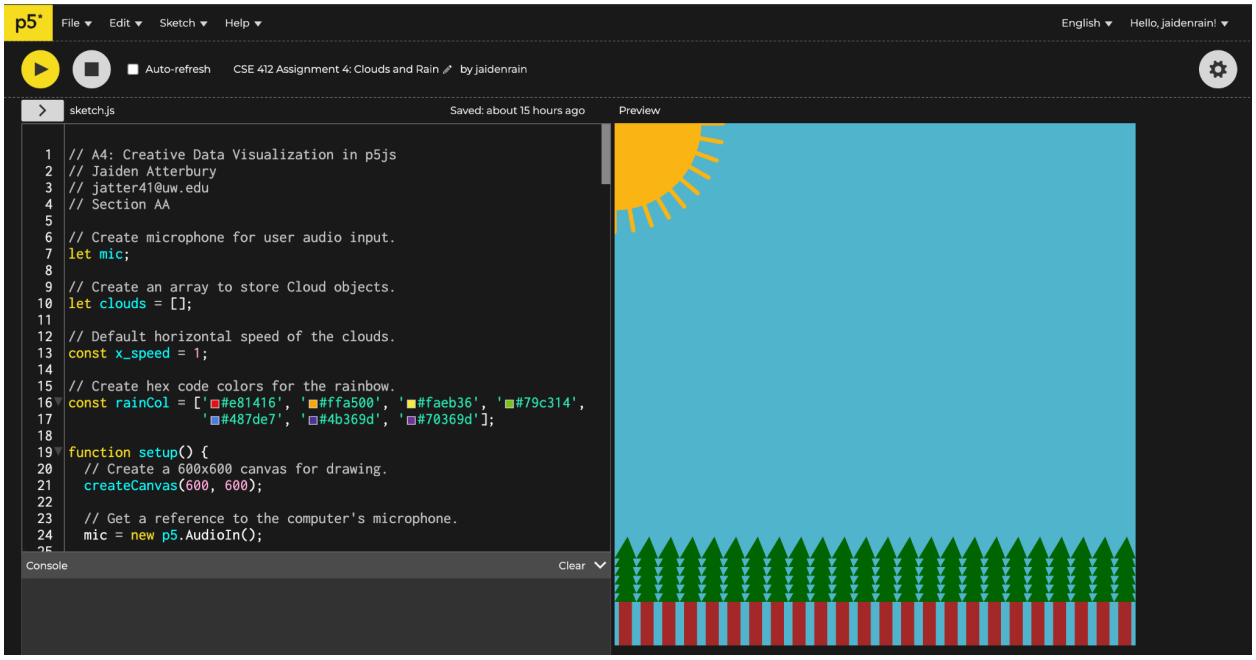
Process:

At first, I wanted to make a sound visualization that mirrored the bouncing DVD logo. However, I felt like this would be too similar to the bouncing circle example from Johannes Preis' article in Step 1, so I was tasked with coming up with another idea. I then had the idea to plot the amplitude/volume of a given mp3 file over time, sort of like this example. However, I felt that this would be too simple and similar to things we've gone over in class, so I again had to look for another idea. With this task at hand I came up with an idea relating to a passion of mine; hiking and the outdoors. With this inspiration in mind, I created a visualization called "Clouds and Rain" that is primarily

focused around raining clouds, with certain aspects of the scenery changing depending on user audio input. The main challenge of making this visualization was finding the correct way to represent clouds and rain. At first I tried using the let cloud = {xpos:...} syntax in JavaScript to create cloud objects, but I figured out very early on that making a cloud object with specific methods would be difficult using this syntax. With that being said, I used my knowledge on classes from Java and Python, as well as a little help from The Coding Train and w3schools.com, to learn and create classes in JavaScript. Once this was done I was able to make a class representing a Cloud object, as well a class representing a Raindrop object. Both of these classes had fields and methods that allowed me to mimic raining in p5.js.

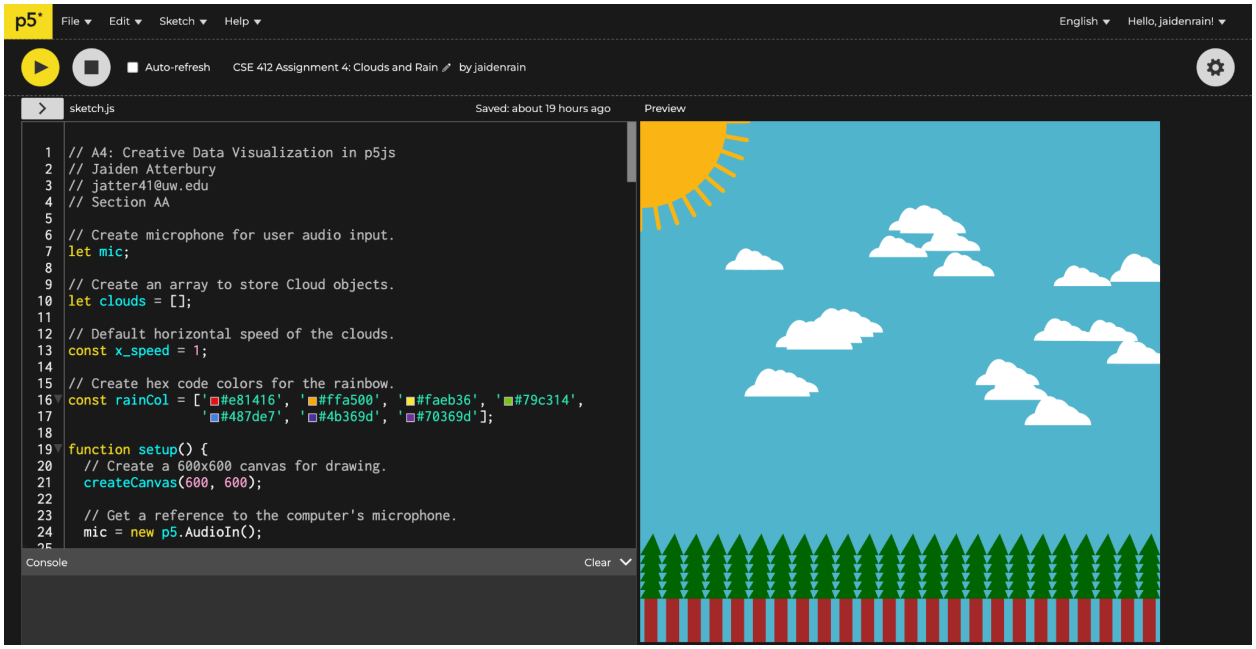
Description:

Now that I've gone over my design process and rationale, I can now describe the contents of the visualization itself in detail. After pressing the play button, the background art is plotted, namely a pristine blue sky, a radiant sun, and a full forest are all displayed without any user actions. As shown below.

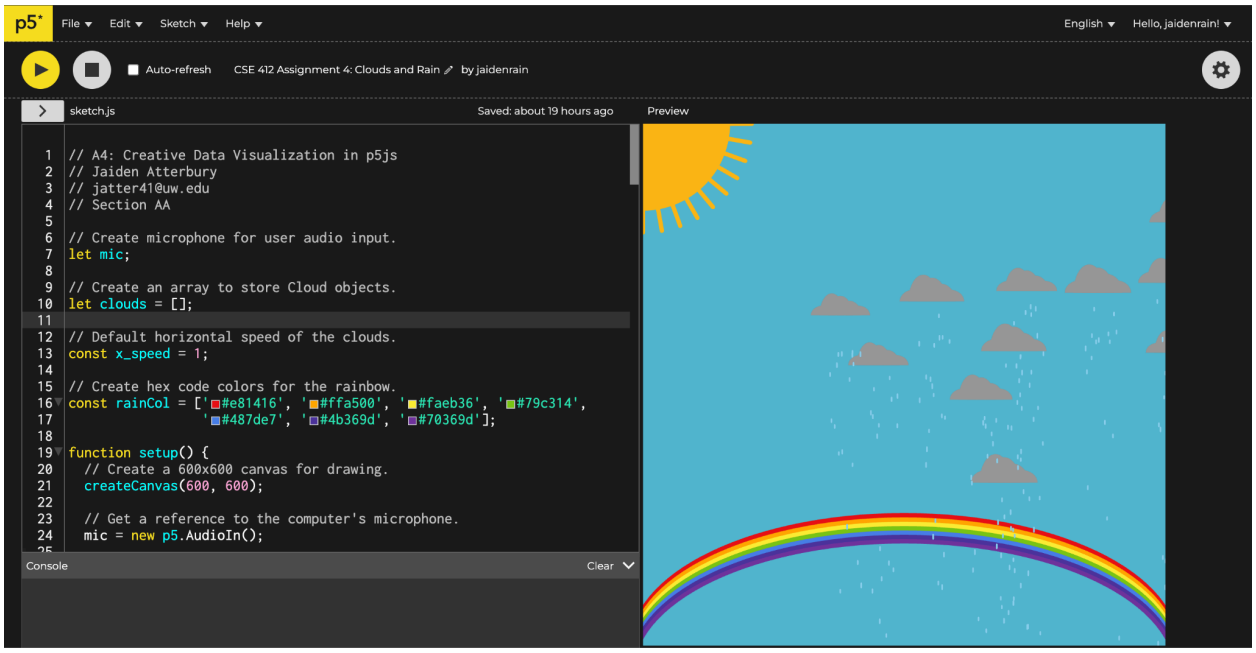


Once this background is created, this is when the user comes into play. First off, and most importantly, everytime the user clicks on the canvas a cloud object is created and appears on the screen. There is no limit to the amount of clouds that can be on the canvas at once. There are only two ways that these cloud objects can be removed from the canvas; the first being that the cloud floats off the screen and is thus removed from the canvas (removed from the cloud array), this happens naturally and isn't influenced

by the user. However, users can remove all clouds from the canvas by pressing the delete or backspace key on the keyboard. This cloud filled environment is shown below.



These are the basic non-sound oriented capabilities of the program, we will now shift our attention to the audio input aspect of the program. These aspects can be broken into two categories; functionalities dependent on the exact sound level, and functionalities dependent on a sound threshold. The functionalities that are only dependent on the mapped (using map()) user audio level are: a cloud's horizontal speed, the randomly chosen vertical speed of a cloud, and lastly the number of raindrops (or more precisely the probability of attaining a raindrop). All of these actions occur even if the clouds aren't raining. The other type of functionalities depend on if a certain user audio threshold is reached, namely mapLevel/soundLevel > 1.5. If this threshold is reached, the clouds will turn gray, start raining, and a rainbow will replace the forest at the bottom of the canvas. This functionality is shown below. It is important to note that this rainbow will only appear if there are clouds present on the screen (as if there are no clouds there will be no rain, and thus no rainbow).



These are all the functionalities of the program. In summary, the louder the noise, the faster the clouds go, the more sporadically the clouds bounce up and down, and more raindrops are produced.

Step 4: Learning Reflection

In this step, the goal is to critically reflect on the assignment and point towards my primary learnings I gained from this assignment. In the following paragraphs, I will include meaningful reflections, strengths, weaknesses, etc.

<u>Reflection:</u>

Before this assignment, I was not confident in my abilities to use p5.js, as well as JavaScript classes. However, after completing this assignment, I am a lot more assured in my abilities to create things in p5.js, especially those in which classes are necessary to create certain recurring objects. This confidence was gained through the use of the “Resources” tab in the assignment specification. In particular, The Coding Train’s p5.js series had information on almost any topic I wanted to know about, and provided me with the knowledge I needed to know in order to be successful on this assignment.

One of my favorite parts of the program is that it is responsive to multiple kinds of user inputs. In particular, it responds to mouse clicks, keyboard presses, and user audio. Furthermore, the program uses this user audio in multiple ways in order to mimic rain. Specifically, louder noises make the clouds move faster, and bounce up and down more sporadically, this was implemented in order to mimic the wind intensity of a storm.

The biggest weaknesses of my program, in my opinion, are that there are more things I wanted to implement but didn't get around to doing, as well as a lack of domain knowledge required to correctly simulate a rainstorm. First off, I wanted to create lightning bolts once the mapped audio input reached a very large threshold, but I was having troubles doing this due to the fact of my lack of experience with beginShape() and vertex(). Another thing I wanted to implement but didn't have the time to do, was implement thunder at this same volume threshold. Moving onto the next weakness, in terms of a lack of domain knowledge, with most of my usage of the random() function, there was no reason for the bounds I chose. Instead, they were merely arbitrary values that looked good in the output. With more knowledge on how rain falls, I could create a program that more effectively encapsulates this behavior, with more reasoning behind the values I chose to use.

Resources Used

1. Tutorials on classes in JavaScript.

https://www.youtube.com/watch?v=T-HGdc8L-7w&t=116s

https://www.w3schools.com/js/js_classes.asp

2. Information on specific colors (not comprehensive, just the ones I remembered to save).

https://www.colorcombos.com/colors/FDB813#:~:text=%23FDB813%20Hex%20Color%20%7C%20RGB%3A,%2C%2019%20%7C%20SUN%2C%20YELLOW%20ORANGE

https://palettemaker.com/rainbow-colors

https://www.rgbtohex.net/

3. Stackoverflow/Stackexchange for information on polar coordinates and KeyCodes.

https://math.stackexchange.com/questions/260096/find-the-coordinates-of-a-point-on-a-circle

https://stackoverflow.com/questions/34930858/how-to-capturing-delete-key-press-on-macbook

4. Inspiration/starting template for my clouds and their movement.

https://editor.p5js.org/jackiezen/sketches/rJEziNOR

5. P5.js references page.

https://p5js.org/reference/