A Processing Interface for Time-Frequency Analysis of Music Signal

Hongxi Mou

Final Year
Project - 2021
B.Sc. in Computer Science and Software
Engineering

Department of Computer Science
National University of Ireland,
MaynoothCo. Kildare
Ireland

Supervisor: Dr. Tom
Lysaght
Date: March 22, 2021

Contents

Abstract

P5.js could be understand as a combination of javaScript and Processing language with simple grammar and powerful libraries to help user build creative sketches with drawing or sound functionality, which has not been used widely yet. As the core library, p5.js sound library offers a lot functionalities which can affect the output of sound or analysis sound with frequency, amplitude, delay effect and so on information, and all of these functionalities are highly helpful to visualizing music signals. Music signal is an abstract word for most public because it usually can only be noticed by feeling or hearing, however in fact its frequency and amplitude information can be saw in a regular wave and people are supposed to comprehend the wave to be better understand and control sounds. The goal of this project is to realize music visualization through creative sketches by p5.js and organize them inside of one web interface with some buttons allow user to manipulate the effect on music, thus can give the user a clear and intuitive experience of visualization. As introduced above, p5.js is the major technology I used for this project with numbers of inbuilt libraries and functionalities which related with web audio and graph drawing can be used directly. React is another focus point that be used to establish the front-end web interface that supposed to contain all sketches.

Chapter 1

Introduction

Analysis music signal through visualization has been focused by wide files, not only including programmer in computer science but also creative designer work for art or media, particularly since more open source languages have been launched out as p5.js.

1.1 Motivation

A lot of people always think music is an important part of their lives, they may only listen to it every day then hardly thinking about the relationship between sounds vision, however, that is exactly what music visualization tries to work on, to let people see the music synchronized when they play it. To be honest, I am fascinated with music but also just listen to it without any thoughts. Music as a topic I interested in is the reason why I chose this project to research on without hesitate and I am glad for my choice I was obsessed and amazed every time when I saw the vision of music. Sound is a vibration that propagates as an acoustic wave, through a transmission medium such as a gas, liquid or solid [3] is the fundamental physics knowledge that we learned since high school and the acoustic wave is exactly the thing that influences frequencies and amplitudes of sound. By display the wave of sound or transform some features of music into a visual animation as a movement of a ball or variation in the size of a circle could be a helpful way for everyone to comprehends music, particularly for those who want to work in music production or visual design.

In aim of achieve my project goals to have an outline of what kind of visualization I want to do and which certain features of sound I want to affect and how to affect them are the issues I need to concern myself with the most, as it can be the alpha of background color of animation changed by the frequency of music, or rotation angle of a graphic altered by amplitude of sound.

At the same time, self-studying several programming languages is another challenge I should deal with. I've learned processing language before, a language for visual arts, and it open the door for me to focus on visualization, especially after I realized there is a JavaScript library called p5.js which absorbed almost all features of processing language. Although p5.js is highly similar with processing language, it is still essentially a JavaScript language that I've learned about 2 years ago but still not familiar with now, React is a totally new web-building technology for me.

1.2 Problem Statement

An important technical problem I meet here is that to self-learning p5.js as an almost new coding language. The first step was to find an appropriate study platform. Large chunks of tutorial materials as instruction articles or videos can easily be accessed through search engines, however, if I check them one by one could be an absolutely time-waste as most of them are obviously copied from each other, even only part of them. This makes my self-learning very fragmented and does not allow me to develop a systematic understanding in the very began.

While working on visualization I was confused for quite a while as there has various approaches to implement the sound, so I thought about start by manipulating the size of a circle with the amplitude of a song [3] and how to map the variables got from music into the goal shape is the problem I should concentrate. After getting familiar with these mapping problem I wanted to add more options of the music into my sketches, including a song playlist, as for every element each song has its own iconic information. Change sound input inside a same sound feature visualization sketch allows users to have a more immediate and profound sense of how a certain element of sound can be influenced in different sounds, and to achieve this, I had to find a way to write a playlist embedded in sketch that already contained a certain number of music files ready to play.

Before this project I haven't thought about integrate p5.js with React and even React is a new technology for me. It would be the last step of my project to develop so I built an react web project framework only with Heading and Navigation parts without any thought of how to corporate with sketches at very beginning. With the ending of creative sketches part coming closer I commenced to consider how to put sketches at the exact location of the web as I designed in mock-up, because by using p5.js web editor I merely saw and knew how the sketch will be displayed as the first block at the top-left place in a HTML page.

1.3 Approach

Firstly, about how I found the suitable way of myself to study p5.js my supervisor offered me a great amount of help by suggesting some specific websites professional with processing programming or p5.js sketches, such as Openprocessing.org and HappyCoding.io as I tried the Learn page of official website of p5.js before but turned out it is too fundamental and not detailed enough. With one semester's study of Processing program I could pick up the basic structure of this new coding way easily as the core two functions call setup and draw are the same as in processing, and almost all usages of functionality of p5.js sound library is the same with minim library in Processing, so when I firstly approach a new web I would only choose one chapter of its tutorial involved the topic I haven't understood yet. Under the topic whether it accompany with simple examples with relative comments or not is one of the key matters for me to judge the usefulness of a website. And I realized I prefer the simplicity of the descriptions through visual examples rather than textbook style explanations within my self-learning process as I grasp a usage of one function through put it into examples.

Understanding how to map a certain element of sound into animation for me is a processing of doing research of existing visualization examples. Using the example I mentioned above of alter the size of a circle by the change of the amplitude of one song, how to get a variable of amplitude off the song and integrate it into the constructor of circle is the matter need be focused. The best way I found to see effect of one certain variable is find an example includes it in p5.js web editor examples, then change the value of it step by step (like increase or decrease the number of it) or directly commented that line before running the program again. Comparing the animation details helps me comprehend how parameters are passed inside this example.

I've gained some experience of create playlist button from the latest Processing project I did as a music player, however in that time I got the inbuilt Processing library called controlP5 to help me draw a list and put data by using "set" straightway. P5.js doesn't have an extra button-drawing library like controlP5 so I could only use "selection" as a replacement because this element has a rectangular selection box that looks like playlist chooser. I always thought about define an array variable before setup function, at the same time load song files to another independent variables. The problem I met when I firstly try my array plan is after I load all song files in function "preload" I couldn't add them as independent song file item into selection in function "setup".

1.4 Metrics

My work can be separated into two parts as follows clearly: music visualization sketches made by p5.js and one front-end web page realized through React, and for each part I got specific measurements to test. The visualization I made for sound displayed as interface that also allow users to manipulate several effects by moving mouse or using keyboard, thus let my friends who doesn’t study relative files to use my sketches without my explain would allow me to get the feedback directly. Also, observing others trying out these sketches without me explaining it would quickly help me to identify which areas of my program that are not clear enough for user. For example, if someone still doesn't understand what the program does or how to use it after using it because I haven't built in any prompting statements, then I should make improvements to the more detailed user interface.

Without no doubt, running sketches and observing console information are the most simpleness ways to test a program. As the developer of the program, I understand what I want to achieve before I write the program. Programming and running programs alternately to see which parts of the program achieve what I envisage, which parts fall short of the design, and which parts are not implemented at all. The console is the first place to give details of any logical errors in the program, including the number of lines of code and the correct way I supposed to use for the functions involved.

The website built as a container of sketches also be tested by the usage of my roommates. Let them freely check and roll that web as users who ever see that page before can find more significant bugs than myself.

1.5 Project

The achievements in my project include:

- A React project with 3 pages.
- Changing delay effect of music by moving mouse and pressing up and down keyboard.
- Visualizing a waveform of one song.
- Changing rate of music by follow mouse track.
- Integrating frequency into linear movement.

Chapter 2

Technical Background

Although the first music visualizer was introduced in 1976 [26], there are still numbers of related areas can be involved have not been reach so far.

2.1 Topic Material

2.1.1 Real-time music visualization using responsive imagery [4]

This is the first key research paper I read and it offers an initial understanding and conception of how I should put visional imagery into my project. One section of this research is visualize vocal timbre and piano chord data [4] into color interest me because Robyn and his colleges chose to extract the feature of vocal timbre which is more complicate to analysis than frequency or amplitude. Vocal timbre is not only determined by frequency spectrum but also the envelope [3] and they built a system which can successfully identify each vowel choices with an established vocalized spectrum database.

To extract frequency and amplitude elements from sound they also have another tool as build a Max/MSP object.[4] However, these processes of extracting sound elements, whatever tools are used, is essentially a mathematical analysis of the frequency and amplitude of the sound waves, particularly contributed by Fourier analysis. Since the principle of Fourier analysis is understood to mean that any piece of music can be assembled by striking different keys with different intensities and at different points in time, the process of extracting sound elements is also a matter of breaking up the piece and transforming it into a sine wave of different amplitudes and superimposed phases.

Mapping piano chord into a standard color wheel is another critical point of visualization method I paid attention with. There are three common color modes I've used before: RGB, CYMK, and HSB, and by studying color theory and color modes we will have more ways of mapping sound elements to color. For example, when we use RGB mode, we can map sound elements to red values only, so when we change the color mode and map the same element of sound into a different variable of this mode, the color vision of sound will also be changed.

2.1.2 Sound Visualization for Hearing Impaired

Using sound visualization to help the deaf or hearing disabled people is the topic I admired. In this research paper they also cared about the mapping of sounds to images, however, more special on the sounds made by surroundings in aim to enhance the deaf's awareness[5], and also includes certain studies situation of hearing disabled people in researchers' counties from a sociological way. The research in the Deaf section of this report not only links the problems of such an unrelated social discipline to the problems of computer technology, but more importantly, makes the technology truly relevant to the purpose of serving humanity as after testing, the average grade of this entire visualization program given by hearing disabled people is 88%.[7]

This paper works on a pitch extraction algorithm with mathematic formulas and waveform figures. They thought there are secondary peaks reduce the accuracy of pitch and use the Autocorrelation function:

$$Ry[k] = < y[n]\ y[n - k] > \qquad \text{(where <.> corresponds to the mean)}$$

to clip these peaks would resolve this problem.[8] And they also considered color spectrogram is powerful for vision so using Fourier analysis to prove the frequency components mapping process is unavoidable, particularly focus on the Short Time Fourier Transform.

2.1.3 Audio-Based Music Visualization for Music Structure Analysis

This paper firstly bias on visualize music by arc diagrams and list a clear analysis system steps as extract, project and generate. [9] The interesting feature I found after reading this paper and compare it with the two above is that they all have a part concentrated on color---- this research "use chroma features to represent harmonic content in the music signal." [10] Recurrence Plot is another research they have done in this paper that aims to nonlinear dynamic system with the study of time-delay embedding process. Mathematic file is important in this part special the capability of Euclidean space and Matrix, and these are also core math abilities required in computer graph theory so they always directly created analysis model by certain computer graph applications.

2.1.4 Creating Access to Music through Visualization

While the focus of this study is also on how music visualization can enhance the music appreciation experience and help people with hearing loss learn music, I am more concerned with the 3 different visualization tool options it offers which have already done for this topic: Music Animation Machine, iTunes and Motion Pixels of Music. [11]

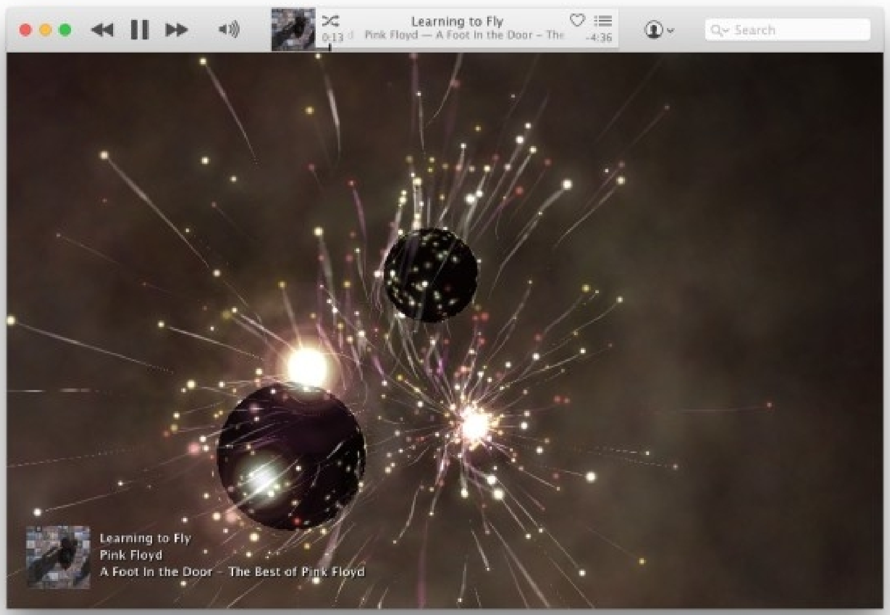
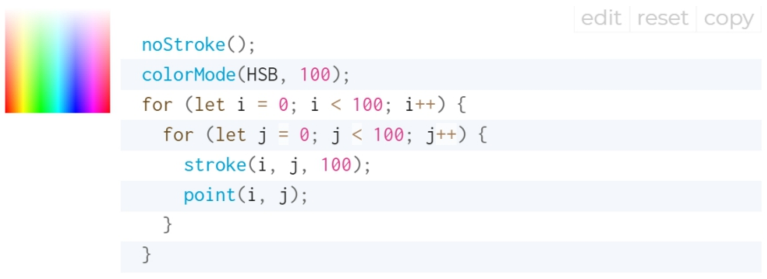

Learning to Fly
0:13
Pink Floyd — A Foot in the Door – The
-4:36
Search
Learning to Fly
Pink Floyd
A Foot in the Door – The Best of Pink Floyd

Figure 1. An imagery of Learning to Fry generated by Apple iTunes [5]

2.2 Technical Material

2.2.1 P5.js library, reference and examples

This is the website that helped most to do programming with p5.js. Sound library is the core library of p5.js and the fundamental libraries as Color, Events, Math and etc involve in my project too. Both websites have paired all functions and methods with one or more suitable samples where possible, and include the ability to modify the source code while displaying the sample effects, so that I can make direct changes to confusing lines of code while reading these sample codes, and then click the Run button to see how the changes look when displayed. ColorMode() function in the reference is one of the functions given me a deep impression with two different color models as variables. I never learn HSB color model before so this function allows me to do various attempts. ColorMode() has to setup before project color data into p5.js and should be putted inside function setup() if the program has.



```
noStroke();
colorMode(HSB, 100);
for (let i = 0; i < 100; i++) {
  for (let j = 0; j < 100; j++) {
    stroke(i, j, 100);
    point(i, j);
  }
}
```

Figure 2. Source code and effect of chose HSB color model

Sound library is the core library of p5.js with detailed explanations with each variables of it. I learned to include sound files to sketch by using loadSound() method [12] in function preload() and this function works properly only when it executed before function setup(). Class p5.amplitude [2] need to instantiate by creating an object for it and could extract the level of sound amplitude. Through relative example of p5.amplitude I learned how to influent size of an eclipse with synchronous amplitude value of music. Calling the mapping function is a step in this process that should not be overlooked. This is because it maps the target value up or down to a given range. Since the default size of the sound amplitude is between 0.0 and 1.0, when I need to use this value to make a more pronounced impact on the visuals, I have to map it to a larger range.

```
amplitude = new p5.Amplitude();
```

Figure 3. Instantiating p5.Amplitude class

```
let level = amplitude.getLevel();
let size = map(level, 0, 1, 0, 200);
ellipse(width/2, height/2, size, size);
```

Figure 4. Mapping volume amplitude into circular size

P5.FFT is another powerful class has been used in music visualization widely. It offers two critical methods, an .analyze() method and a .waveform() method, widely used for plotting bar graphs of frequency spectrum and sound waveforms respectively. Both visualization situations data are a changing persistently array values so read their values sequentially in for loop is the optimum approach.

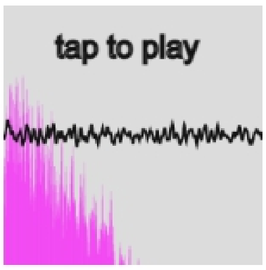
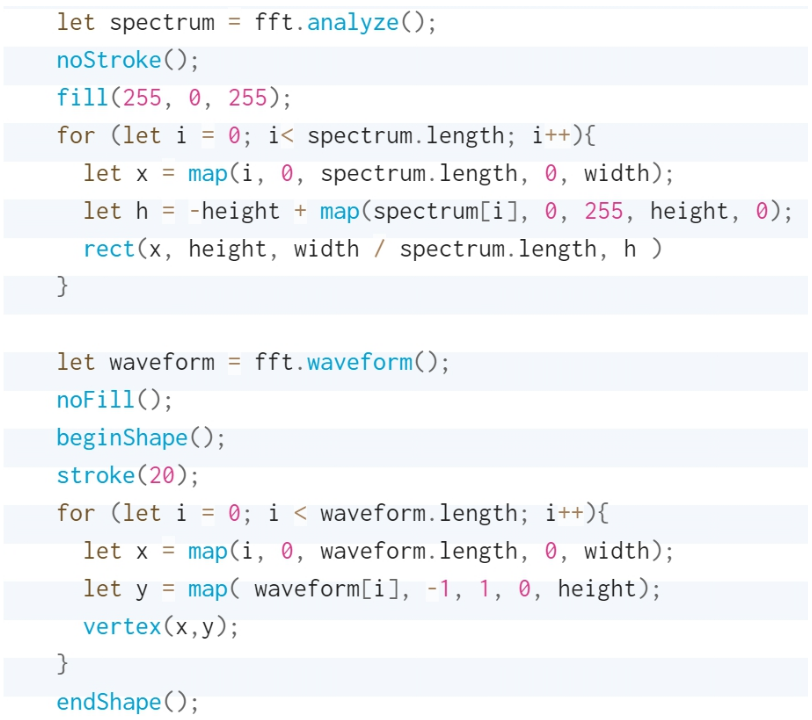


Figure 5. Visional frequency



```
let spectrum = fft.analyze();
noStroke();
fill(255, 0, 255);
for (let i = 0; i< spectrum.length; i++){
  let x = map(i, 0, spectrum.length, 0, width);
  let h = -height + map(spectrum[i], 0, 255, height, 0);
  rect(x, height, width / spectrum.length, h )
}

let waveform = fft.waveform();
noFill();
beginShape();
stroke(20);
for (let i = 0; i < waveform.length; i++){
  let x = map(i, 0, waveform.length, 0, width);
  let y = map( waveform[i], -1, 1, 0, height);
  vertex(x,y);
}
endShape();
```

Figure 6. For loop for spectrum and waveform

2.2.2 Tutorials of p5.js in Happy Coding

To be hones, the tutorial series on p5.js on the Happy Coding website is the most beginner-friendly and easy to follow tutorial I have found based on my self-learning experience. It has 18 chapters covering all the features of p5.js and how to use the basic functions of it. I quickly grasped the first 10 chapters of the tutorial as they were very similar to Processing language I had learnt so I was able to focus on the more advanced parts in this series, like arrays with array functions, using object and creating classes. Although I had learned arrays in other programming languages before, about I didn't have to specify the size of the array in p5.js still confused me for quite a time and took me longer to learn it than other part by going over the examples and reading the notes. I also learned how to impose array functions to achieve mouse movement track ---- with cooperating array with p5.Vector. [14]

2.2.3 Components by Ant Design

React is the technical used to build the goal website for this project, which is basically could consider as made by components. Antd Design of React refers to a React UI library built by themselves that has abundant components for user to add into their project, not only including general components as Button and Icon but also height-quality components as Menu, Layout and Carousel [22]. If user want to use antd design in create-react-app, they must install yarn it firstly through npm like the following code [23]:

```
yarn create react-app antd-demo
yarn add antd
```

Chapter 3
Visualization Approach

The approach I chosen to use in my project can breaks down to 3 steps, referring my comprehend of the methods given in Audio-Based Music Visualization for Music Structure Analysis [9]: first import or create sound signal, secondly extract sound feature and replace these features into visional effect in the end.

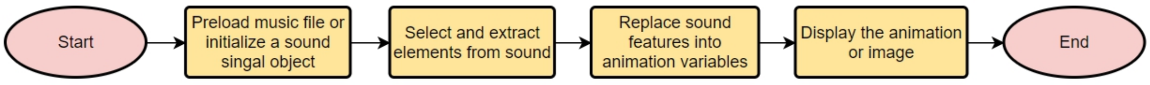

Start
Preload music file or initialize a sound singal object
Select and extract elements from sound
Replace sound features into animation variables
Display the animation or image
End

Figure 7. Flowchart of imagery steps

So according to this approach, the most problematic areas are steps two and three: in step two the problem is obtain the mathematical analysis of the music signal, and in step three it is how to logically inject the analyzed music data into the correct place in the code and implement control over it.

3.1 Introduction

3.1.1 Sound signal analysis in mathematics

An audio digital signal is a series of numbers representing a continuously changing sample in the time domain, often referred to as a "waveform". To analysis a digital signal means to analysis its frequency information inside waveform. Time-frequency analysis and Fourier analysis are the major mathematical1methods. (Figure 8)

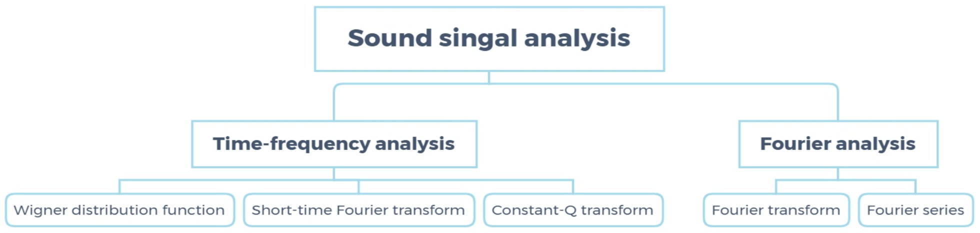


Figure 8. Mathematics methods can used in analysis

3.1.2 Design the execution effect of visualization

Visualization refers to making the abstract tangible and the imagery based on music data could be showed in various shapes and animations. [17] After the preceding section questioned how analysis the information from audio theoretically, the way of how using these information of visualization become the matter. Like before painting, the painter will choose a topic firstly and seem for design imagery, to select a major object or theme also matters at here.

3.1.3 Parameter injection in the programming process

There are detail examples of the programming problem in step three happened to me:

A. I have designed a circle whose diameter size changes in real time according to the amplitude of the volume of the music being selected, and the position of this circle can in turn affect the frequency or rate of the music. In this specific case, at least 3 different elements of one sound would change continuously in real time hence only make movement of mouse as the input to manipulate them is not enough as well as could create misunderstanding. How to line keyboard into the sketch is another matter.

B. The problem of creating a music list for a sketch that allows the user to select and change the input source in real time is applicable to all sketches about visualization where audio input is achieved by pre-loading a file. In Processing, it has a "scrollable list" component in an extra library called controlP5 that has the same structure as a playlist, at the same time in p5.js there is a "select" class. How to merge several song files into one single object and put it into these classes, with premise of loading all song files as another independent objects. After changing music by check playlist, to stop previous music and start the chose one successfully means achieve functionalities of playlist fully.

C. Building React website framework could be recognized as an independent system compared to visualization sketches, however I need to make it cooperate with p5.js sketches. The animation and interaction display on sketches should not be influenced by React and same for React' functionality because the front-end website barely consider as a container for sketches. Thus, how to combine these two technologies is one of the core challenges.

3.2 Modelling

3.2.1 Mathematics Formulas

Mathematics is not a simple file that anyone who has not focus on it before would have a deep understanding of quickly, thus I hardly selected two mathematical models, short-time Fourier transform and Wigner distribution, to allow me have basic awareness about how the analysis functions in processing minim library and p5.js sound library work.

The short-time Fourier transform is performed by first dividing the signal into frames and then Fourier transforming each frame. Each frame can be considered as an intercept from a different smooth signal waveform, and the short-time spectrum of each frame is an approximation of the spectrum of each smooth signal waveform. The formula is written as [15]:

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-i\omega t}\,dt$$

Figure 9. STFT formula

About Wigner Distribute function, the formula is written as [16]:

$$W_x(t,f) = \int_{-\infty}^{\infty} x\left(t+\frac{\tau}{2}\right) x^*\left(t-\frac{\tau}{2}\right) e^{-2\pi i\tau f}\,d\tau.$$

Figure 10. WDF formula

3.2.2 Structure design for programming

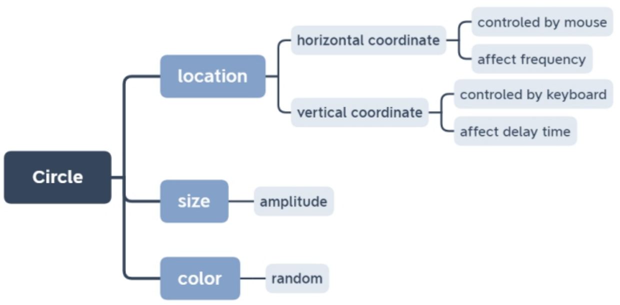


Figure 11. Key features of an imagery circle with sound

3.3 Analysis

In this part I will only put detail problem description of programming as mathematics is not my research emphases with.

To make the design more fleshed out, it would include the question of the theme, the color, the shape, whether to choose a single object to control, or to do a monolithic animation that changes on the background at the same time. Visualization of sound could even also be a certain game.

Continue with imagery circle matters talked in 3.1.2 A, the highest frequency of the music that can be heard will be reduced when the circle is moved to the left, and the delay time of the music will be extended when circle is moved downwards, and vice versa. Therefore, in this instance, it is a question of which function body in the program should do the substitution of the music frequency information and the delay-time information with the circle position data, as well as whether the construction of the circle should be placed in the draw function.

Chapter 4

Devising, analyzing and reasoning exist examples

As mentioned in the previous section, my confusion falls into two main categories: understanding the mathematical principles of music analysis, and understanding the logical relationships between the various parameters in the coding process. Since the main purpose of my research is not to study mathematically the various types of time-frequency analysis formulas, the learning process of them is only to enable me to understand exactly which parameter is responsible for which feature of the sound signal, therefor I can give more precise and fit-for-purpose values when calling the analysis functions from library. Hence I will not go into deeper and more detailed analysis of the mathematical issues here.

For logical relationship problems in coding, the part requires more attention, I have summarized 3 steps as a template for solutions: firstly, visualize the problem, e.g. make a design draft, give a flowchart of the code, draw the structure of classes and functions; secondly, refer to and reason from existing sample code; thirdly, adapt the sample, and replace some parameters into it by use the logic of the parameters in the sample.

4.1 General Analysis

Reading and running lot different p5.js projects not only these embed in p5.js editor and Processing minim library officially, but also abundant creative sketches contributed by p5.js and Openprocessing communities, is the source of inspiration for me to thought about all problems. First those various sketches much clearly show how visualization could be looks like and at the same time the source code relates to each sketches are the most efficient approach to study and be familiar with certain function.

4.2 Architecture

4.2.1 Diagrams for Programs

Before start write the code for Delay Circle sound visualization in p5.js editor, there are at least 5 functions have to be covered as follows: function setup, preload, draw, mouse pressed, mouse released and key pressed. I drew an abstract structure diagram for it to make me capable to inject parameters in further work. Figure 12 is the diagram.

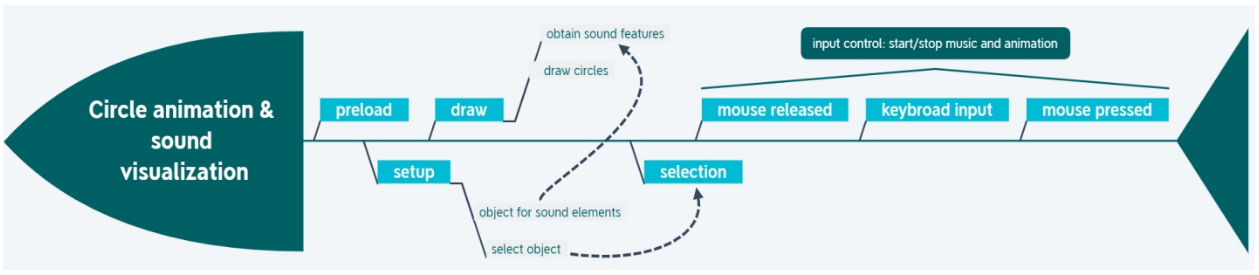


Figure 12. Abstract diagram of an circle visualization sketch, showing structure of its functionality.

By creating analysis diagrams like figure 12, I could go ahead with all visualization source code I collected and found they can all be modeled as fundamental functions (setup and draw), controller functions (if needed) and specific functions (for the classes they include).

`Setup()` is the function that runs before `draw()` and will only be run once in a sketch [17] hence we use it to initialize and create every class or object we need to pictured later, and the step of set up canvas should definitely put here---- `createCanvas(width, height)` is the function to do it. The `width` and `height` parameters is the width and height of the canvas will be drew.

Processing language has a core feature that the function `draw` will continually called by the program at a rate of 60 times per second in default. [17] Therefore if put constructor of drawing the circle inside of function `draw`, with constant position (x and y coordinate) parameters and a changing value as diameter information, a circle whose size is constantly being changed will be drawn in the same position is the animation that is supposed to display.

Same for all elements which need alter their value in serial. For specific instance, analyzer to obtains amplitude value from the music in real time, the value of it need to change continuously, thus I planned to project related operation with it into function `draw`.

In purpose of increase more interaction into this sketch, for example in Delay Circle case, the position of the circle supposed to be altered by user through input devices and that's why I need controller functions. If choose mouse to control the position, set the x or y coordinates with `mouseX` or `mouseY` will be enough, although for key broad control needs to be the appointed function `keyPressed`.

4.2.2 Design Draft for Goal Website

There are 4 major parts in website building: Heading, Body, Container and Foot, so in my design is to put all visualizations in container part, displaying in an ordered, organized layout accompany with succinct relative description under each sketch. Figure 13 is the design draft I made.

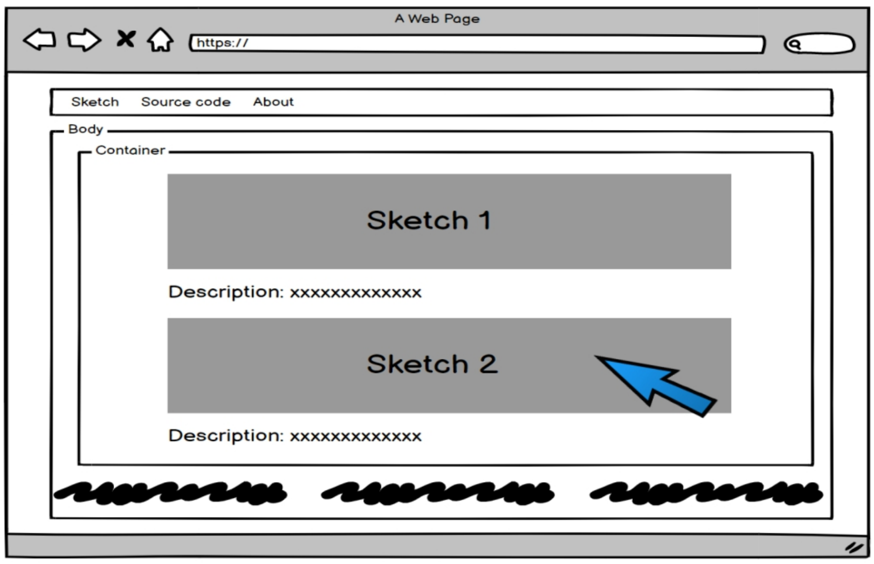


Figure 13. Mockup of goal website, showing the position relationship between web and sketches.

4.3 Low-Level Design

4.3.1 In p5.js

I started the coding by make a selection of which sound function in sound library offers I wanted to use. There are part of the functionalities in sound library: Amplitude class is for getting the current volume of a sound; FFT analyze the frequency of sound; Delay class delay the parameters for feedback, delay time and low-pass filter. [13] Continue with the circle case, I chose delay as the major effect for it so the parameters need to identify should include an object for delay as seam as analyzer for amplitude, sound file for load file with the following code:

```
let delay, analyzer, soundFile;
```

In Delay Circle example, except the essential create canvas function called in `setup()`, generating new object for `Delay` and `Amplitude` classes should also be done in `setup()` as the following code:

```
delay = new p5.Delay(); analyzer = new p5.Amplitude();
```

Then `delay` has to set process and delay type. The 4 parameters in `delay.process(sound, 0.12, 0.7, 2300);` means the aim audio signal, delay time, feedback which means on each repeat the decrease rate of volume and a filer value that will cut off all frequency higher than that value. [1] Because in this case we wanted to add a playlist by using `select` class, the initialize of a selector also program in `setup().` When user changes the selection, there is a changed function of `select` to deal with it and we wrote a specific `mySelectEvent` function to do it. In the `setup()`, `maySelectEvent` liked through the code as follow:

```
sel = createSelect();sel.changed(mySelectEvent);
```

Pressing mouse is the input approach in this sketch and whenever it pressed there suppose to happen similar effort and mouse pressed also is one of the functionality belong to canvas.Therefore `setup()` and a `mousePressed()` class named `start` will be connect by the following code:

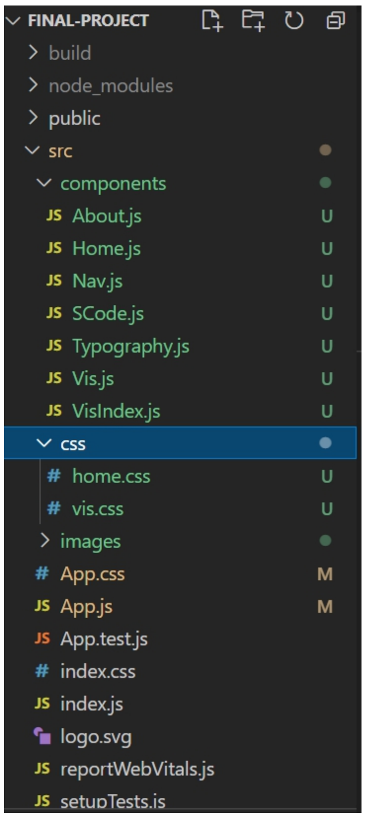
```
cnv.mousePressed(start);
```

Function `mouseReleased` comes after mouse pressed so it doesn't need connect with `setup()` too. `Background` let user change the background color of canvas, it also could be allocated at `setup()` when the sketch only need to set background color once. As mentioned before, `draw()` is the function that will continue be called so when I allocate `background()` here as seam as `ellipse,` which draws ellipse, on canvas there will only one ellipse displays. I designed to alter the vertical coordinate of circle and delay time by key and the vertical coordinate has to write when construct the circle in `ellipse`, thus there need a parameter as the Y coordinate in both `draw()` and `keyPressed().KeyPressed` be called when any key pressed, it will check the pressed key with `keycode.`

The detail code of the structure be discussed above could be find in Appendix C 3.

4.3.2 In React

Based the design for the web I showed above and in React the class is made by several components, I separated the website into 7 components: Home for home page, Nav for heading which suit for all pages, About for about page, SCode for showing all source code, Typography used as a text container, VisIndex is where displays all sketches and Vis is the class where really organized the sketches.



```
FINAL-PROJECT
  > build
  > node_modules
  > public
  v src
    v components
      About.js            U
      Home.js             U
      Nav.js              U
      SCode.js            U
      Typography.js       U
      Vis.js              U
      VisIndex.js         U
    v css
      home.css            U
      vis.css             U
    > images
    App.css               M
    App.js                M
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
```

Figure 14. The structure of React

4.4 Implementation

4.4.1 Functions in P5.js

Follow the diagram and structure we talked above about Delay Circle, here we start to focus on how amplitude and delay exactly used in this case. As them are both change continuously, all operations relate to alter their value have to put in `draw()`. The information of amplitude can got from `getLevel()` function of Amplitude class, we use a parameter level to obtain these information and because the extract volumes are quite small number [2], we need to map it to a bigger range and store them in another parameter `levelHeight.` The aim of Delay Circle's visualization is show the real-time change of music on the diameter size of a circle, so finally we inject `levelHeight` as the diameter value into circle's constructor. Similar process for delay: delay' filter function need two parameters, one is the highest frequency of the audio signal could pass, another is the highest rate could pass. We generate these two parameters in `draw()` and give them the value of mapped `mouseX` and `mouseY` separately as the following code:

```
let filterFreq = map(mouseX, 0, width, 60, 15000);
filterFreq = constrain(filterFreq, 60, 15000);
let filterRes = map(mouseY, 0, height, 3, 0.01);
filterRes = constrain(filterRes, 0.01, 3);
delay.filter(filterFreq, filterRes);
```

The vertical coordinate of the circle is used in both draw() and selection(), so it is set as a global variable `cY` at the beginning of the program. We use `cY` in keyPressed() to realize change the vertical coordinate of circle by key press and the delay-time at the same time. If the pressed key is the one we wan to use, `cY`' value should be decreased firstly and then map it into 0.01~0.5 to put into `delayTime` function of class delay. The following code show this process.

```
function keyPressed(){
  if(keyCode === UP_ARROW){
    cY = cY - 10;
  }else if(keyCode === DOWN_ARROW) {
    cY = cY + 10;
  }
    let delTime = map(cY, 0, height, 0.01, 0.5);
    delTime = constrain(delTime, 0.01, 0.5);
    delay.delayTime(delTime);
}
```

The full detailed code of Delay Circle could be found in Appendix C 3.

In total there are 4 visualization sketches included in this project and 2 of their animation ideas were come from community contribute examples in Openprocess website. One is anima the mouse track as fire and smoke particles [20], and the other is showing a ball changing its size with left-right movement of line in canvas [21].

Because I already got an idea of alter the sound signal by movement of mouse, this mouse trace recording animation suit my request prefect. By reasoning its source code line by line, this sketch actually is based on two constructor functions called fire and smoke. Smoke class shows how the parameters we give for smoke will be constructed, including position information of current and previous smoke particle, size of smoke particle and how long it will maintain. It has a `display` function which actually draws smoke by function `stroke`. Class `fire` is the same. The detail full source code is on Appendix B 1.

This coding structure can used for another style of mouse trace too, so I wrote code directly on this example code which can switch the color of trace animation into blue and green, also add sound signal and extra function to generate a link between mouse and audio. In my sketch, I add two parameters called `volRate` and `playbackRate.` As their name tell, `volRate` uses to set the rate of the music volume and it is a map of position of `mouseX` from 0~width to -0.5~0.5. `PalybackRate` is the number which will impact the rate of music's playback, its value is the map of `mouseY` from 0.1~height to 0~2. After setup these parameters, we add them into their relate function `setVolume` and `rate` of sound separately. The follow code carry out this process:

```
let volRate = map(mouseX, 0, width, -0.5, 0.5);
soundFile.setVolume(0.5 + volRate);
let playbackRate = map(mouseY, 0.1, height, 2, 0);
playbackRate = constrain(playbackRate, 0.1, 5);
soundFile.rate(playbackRate);
```

The full code and display details of this sketch are in Appendix C 1.

The second example sketch has the animation that give people a feeling of jump and that is the reason why I like it. Thinking about combine the left-right jump movement with increase and decrease of sound frequency maybe a good visualization result, I modified the movement controller parameter with frequency of a sine signal created by oscillator. First new a sine wave oscillator object in `setup(),` then set a parameter in `draw()` called `freq` to store a value as a mapped number of the controller of movement from 0~width into 100~500. Adding the `freq` into oscillator's `freq()` function as following code:

```
freq = constrain(map(x, 0, width, 100, 500), 100, 500);
osc.freq(freq, 0.1);
```

The detail code is in Appendix C 2.

4.4.2 Solution of iframe

Thanks to my supervisor, Dr. Tom Lysaght, as soon as I told him that I didn't know how to link React and p5.js sketches he found an React example, sandboxfiles [24], solved this issue by using iframe and send it to me. In that case, there are 3 blocks in `Index.js` mean to display 3 different sketches, however, even there are 3 source links in iframe on the import `element.js` file, these 3 blocks displayed only one sketch in iframe' link, the last one. You can find the `<iframe />` tag rendered in class `Element` and we import `element` as a component in `Index.js` by these code: `import Element from "./element.js";` in the begin of `Index.js`, and `<Elemnet />` is inside render, in the logic place where we want to see it in design [24]. Apparently that didn't suit the aim of this project, so I came up two ideas as solution to try to include `<iframe />` tag in my React project.

The first one is separate the sketch source links inside of one `<iframe />` into 3 independent `iframe`, keeping them in the `element` compontent.The structure and where iframe be called are similar as in sandboxfile except use a `<div>` block to store the 3 `<iframe />` tags, and on each`<iframe />` tag there only one `src` link.

```
const link = ["https://editor.p5js.org/HXMou/embed/5LOoD5I4a",
"https://editor.p5js.org/HXMou/embed/cMNgGu8hY",
"https://editor.p5js.org/HXMou/embed/xjSOBGUeH",
"https://editor.p5js.org/HXMou/embed/y0ruVGAPx"];
```

Thought about the second idea was because in Vis.js in my React project, it renders the 3 blocks by gaining id from a array, so the id of array maybe will also work for iframe if the source links stored in an array too. For doing this, first need replace iframe from imported Element component to Vis.js file and generate a array parameter for all the links needed. The relative code is as follows:

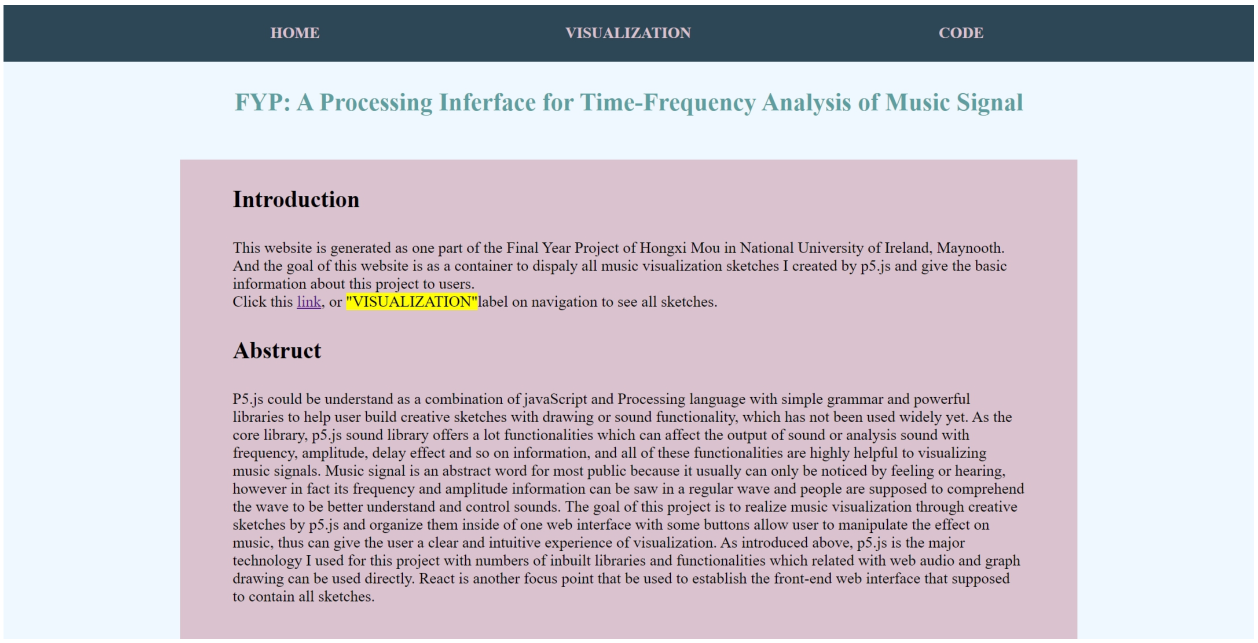
Second, direct includes `<iframe />` tag in render of the `Vis` class in the logic place we want and set the value of `src` property of `<iframe />` tag as every element of the array link by Javascript code as follow:

```
src={link[item.id]}
```

4.5 Verification

4.5.1 Architecture

In the final vision of goal website, the user interface be made up of 3 pages, Home Visualization and Code, that can switch to each other through the navigation bar. On Home page (figure 15), it includes an introduction and abstract information about the whole website; Visualization is where shows all sketches and Code has the source code of each sketch in Visualization.



FYP: A Processing Inferface for Time-Frequency Analysis of Music Signal

Introduction

This website is generated as one part of the Final Year Project of Hongxi Mou in National University of Ireland, Maynooth. And the goal of this website is as a container to dispaly all music visualization sketches I created by p5.js and give the basic information about this project to users.
Click this link, or "VISUALIZATION"label on navigation to see all sketches.

Abstruct

P5.js could be understand as a combination of javaScript and Processing language with simple grammar and powerful libraries to help user build creative sketches with drawing or sound functionality, which has not been used widely yet. As the core library, p5.js sound library offers a lot functionalities which can affect the output of sound or analysis sound with frequency, amplitude, delay effect and so on information, and all of these functionalities are highly helpful to visualizing music signals. Music signal is an abstract word for most public because it usually can only be noticed by feeling or hearing, however in fact its frequency and amplitude information can be saw in a regular wave and people are supposed to comprehend the wave to be better understand and control sounds. The goal of this project is to realize music visualization through creative sketches by p5.js and organize them inside of one web interface with some buttons allow user to manipulate the effect on music, thus can give the user a clear and intuitive experience of visualization. As introduced above, p5.js is the major technology I used for this project with numbers of inbuilt libraries and functionalities which related with web audio and graph drawing can be used directly. React is another focus point that be used to establish the front-end web interface that supposed to contain all sketches.

Figure 15. Home page of the React web project

4.5.2 Low-Level Design

As the reason of why I designed the functionalities and logical relationship between parameters can be verified through the in-build examples about amplitude, delay and how to add key broad into sketches in p5.js website. [2] [18] [19]

It turned out the first solution for iframe problem doesn’t work ---- they load these 3 sketches in the same block and at the same position. But thankful the second solution solved this problem to make each sketch showed in their own block individually.

Chapter 5

Evaluation

5.1 Software Verification

I've never test React project before so this time I started to do fundamental test of it by following the instruction on Test page of Create React App [25].

Because I built this React project through `npm` comment Create React App, I don't need to do any thing to install test tool Jest. On the terminal window, enter into the path of my project and then run `npm test` comment, React will automatically create App.test.js setupTests.js file. The detailed code of test can be found in Appendix D.

5.2 System Validation

After finish the programming of this project, the first direct test I did was go straightly to my roommates. I showed them all my sketches without any verbal explanation and asked them to use it, so they were the user who only know all the information from the interface. The feedback gave by them could help me realize what is missed in my project. At the first time, they told me they didn't find out those sketches are used to music visualization. They noticed the audio signal changes by their control however as they didn't know the aim of music visualization before, thinking about an animation sketch is to visualize a signal is impossible. So after that feedback, I added a short description in the top of each sketch.

Chapter 6

Conclusions

This paper discussed the process of generate audio signal visualization by p5.js step-by-step. From the basics of analyzing sound signals to the coding of a visualization program with p5.js, this paper covers it all. The aim of music visualization is not to allow the music listener to analyze the music accurately, but as an aid to help the listener to appreciate the music more fully. Thus, the design of a website with an embedded interactive music visualization interface is intended to achieve this aim and to promote visualization as a way of understanding and appreciating music. The analysis and design process given in the thesis is, in my opinion, universally applicable to all programming based on Processing and p5.js, because firstly, the temporal and frequency analysis of the music signal covered in this paper is limited to the basic mathematics principles, and the design steps are summarized on the basis of the built-in examples in the editors of these two languages. So for those who also use the p5.js language for music visualization, could find the sample URLs given in the reference and combine them with the design steps in this article for their further implementation of visualization. As the sound analysis functions used in this project are called directly from existing libraries, it is not visible how the music signal is calculated and extracted internally, which cause the limitation of the accuracy of the audio signal information transforming to visualization in this project.

Referring to exist sketches in Openprocessing website contribute a lot in design, and it is also the major approach used to start each sketch of this project. Not everyone is very talented and creative, so learning and building on what others have done is a way of learning and research that cannot be denied. I have to admit that all visualization sketches created in this project are low-level animated and designed, and only can control the size and position of shapes by sound elements now. In the further work, there should be work on the color information of shapes, also thinking about use input audio source like microphone.

References

[1] P5.js organization led by Moira Turner. Reference | p5.Delay [online]. Available from: https://p5js.org/zh-Hans/reference/#/p5.Delay

[2] P5.js organization led by Moira Turner. Reference | p5.Amplitude [online]. Available from: https://p5js.org/zh-Hans/reference/#/p5.Amplitude

[3] Britannica.com, The Editors of Encyclopaedia Britannia. Timber [online]. "Timbre is determined by an instrument's shape (e.g., the conical or cylindrical pipe of a wind instrument), by the frequency range within which the instrument can produce overtones, and by the envelope of the instrument's sound."

[4] Robyn Taylor, Pierre Boulanger and Daniel Torres. (2006). Real-time Music Visualization Using Responsive Imagery [online]. Available from: https://dkit.ie.libguides.com/harvard/citing-referencing

[5] Jimmy Azar, Hassan Abou Saleh, and Mohamad Adnan Al-Alaoui. (2007). Sound Visualization for the Hearing Impaired. International Journal of Emerging Technologies in Learing (iJET) [online]. 2(1). Available from: https://www.researchgate.net/publication/26537086_Sound_Visualization_for_the_Hearing_Impaired

[6] Jimmy Azar, Hassan Abou Saleh, and Mohamad Adnan Al-Alaoui. (2007). Sound Visualization for the Hearing Impaired. International Journal of Emerging Technologies in Learing (iJET) [online]. "In this paper, we investigate several means of visualizing both ambient and speech sounds and present a fusion of different visualization displays into one program package that would help provide the hearing impaired with a means to an enhanced awareness of their surroundings."

[7] Jimmy Azar, Hassan Abou Saleh, and Mohamad Adnan Al-Alaoui. (2007). Sound Visualization for the Hearing Impaired. International Journal of Emerging Technologies in Learing (iJET) [online]. "The overall usefulness of the entire program prototype was given an average of 88% grade by the participants."

[8] Jimmy Azar, Hassan Abou Saleh, and Mohamad Adnan Al-Alaoui. (2007). Sound Visualization for the Hearing Impaired. International Journal of Emerging Technologies in Learing (iJET) [online]. "Center Clipping the speech signal by considering only the peaks that overpass 68% of the minimum or maximum values of the signal eliminates these undesirable peaks and hence allows for a more accurate detection of pitch."

[9] Wu, H. H. and J. Bello. (2010). Audio-Based Music Visualization for Music Structure Analysis [online]. "The proposed approach can be subdivided into three main stages: first we extract low-level, harmonic features from the audio signal; second, we project the feature sequence into phase space and compute a recurrence plot from this data; and finally, we generate an arc diagram characterizing the repetitions in the music data stream." Available from: https://www.semanticscholar.org/paper/Audio-Based-Music-Visualization-For-Music-Structure-Wu-Bello/5b3aed100b884a8466929a915712608624bdfd4b

[10] Wu, H. H. and J. Bello. (2010). Audio-Based Music Visualization for Music Structure Analysis [online]. "In our analysis we use chroma features to represent har-monic content in the music signal."

[11] David W. Fourney and Deborah I. Fels. (2009). Creating Access to Music through Visualization [online]. Science and Technology for Humanity (TIC-STH 2009), IEEE Toronto International Conference, September 26-27, Toronto, ON, Canada. "Using five different displays created from three different visualization tools (Music Animation Machine, iTunes, and Motion Pixels of Music), 18 prototype visualizations of different popular music tracks were presented to D/HH participants in a focus group setting." Available from:

https://www.researchgate.net/publication/236660379_Creating_access_to_music_through_visualization

[12] P5.js organization led by Moira Turner. *Reference | p5.SoundFile* [online]. Available from: https://p5js.org/reference/#/p5.SoundFile

[13] P5.js organization led by Moira Turner. *P5.sound library* [online]. Available from: https://p5js.org/zh-Hans/reference/#/libraries/p5.sound

[14] Tutorials on Happy Coding. *Array Functions, p5.js Tutorials* [online]. "This code uses an array of p5.Vector instances to show a trail that follows the mouse." Availble from: https://happycoding.io/tutorials/p5js/array-functions

[15] Wikipedia. *STFT* [online]. Available from: https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[16] Wikipedia. *Wigner Distribute Function* [online]. Available from: https://en.wikipedia.org/wiki/Wigner_distribution_function#Time-frequency_analysis_example

[17] Christopher Pramerdorfer. *An Introduction to Processing and Music Visualization.* Vienna University of Technology.

[18] P5.js organization led by Moira Turner. *Example of delay* [online]. Available from: https://p5js.org/zh-Hans/examples/sound-delay.html

[19] P5.js organization led by Moira Turner. *Example of snake game* [online]. Available from: https://p5js.org/examples/interaction-snake-game.html

[20] Jason Labbe. (2017). *Fire brush* [online]. Available from: https://openprocessing.org/sketch/415191

[21] T318069. (2021). *T318069_応用課題* [online]. Available from: https://openprocessing.org/sketch/1059397

[22] Ant Design. *Components Overview | components* [online]. Available from: https://ant.design/components/overview/

[23] Ant Design. *Use in create-react-app | Doc* [online]. Available from: https://ant.design/docs/react/use-with-create-react-app?theme=dark

[24] Tom Lysaght. (2021). *sandboxfiles* [online]. Available from: https://codesandbox.io/s/sandboxfiles-13cj1

[25] React, Facebook Open Source. (2021). *Testing Recipes* [online]. Available from: https://reactjs.org/docs/testing-recipes.html

[26] Wikipedia. *Music Visualization* [online]. Available from: https://en.wikipedia.org/wiki/Music_visualization

Appendix A

Overview of Appendices

In this project, two visualization sketches generated on existing examples on Openprocessing community. Here I will put the source code and screenshot when run these examples in Appendix B Design Documents.
On Appendix C source code includes 4 p5.js source codes and 1 source code for website in React.

Appendix B Design Documents

1. Source code of Fire brush.

```
/*
Fire brush
Spawns particles that try to mimic fire and smoke.
Controls:
- Drag the mouse.
Author:
  Jason Labbe
Site:
  jasonlabbe3d.com
*/

var smokeParticles = [];
var fireParticles = [];
var maxSmokeLife = 30;
var maxFireLife = 50;


// Floats up while curling, then dissapears.
function Smoke(x, y, offset, parentAngle, pointSize) {

  this.pos = new p5.Vector(x, y);
  this.offset = offset;
  this.push = random(-100, -10);

  this.parentAngle = parentAngle;
  this.angle = random(-45, 45);
  this.angleRate = random(-2, 2);

  this.pointSize = pointSize
  this.alpha = random(10, 150);

  this.life = maxSmokeLife;
  this.lifeRate = random(0.4, 1.25);

  this.move = function() {
    this.life -= this.lifeRate;
    this.angle += this.angleRate;
  }

  this.display = function() {
    push();

    // This would be much easier if I had its parent's world position!
    // Instead just mimic its parent at the time of its spawn.
    translate(this.pos.x, this.pos.y);
    rotate(radians(this.parentAngle));
    translate(this.offset, 0);
    rotate(radians(-this.parentAngle));

    rotate(radians(this.angle));
    translate(0, map(this.life, maxSmokeLife, 0, 0, this.push));
    scale(map(this.life, maxSmokeLife, 0, 1, 0));

    stroke(0, 0, 0, this.alpha);
    strokeWeight(this.pointSize);
    point(0, 0);

    pop();
  }
}


// Slightly follows mouse direction, falls, then dissapears.
function Fire(x, y) {

  this.pos = new p5.Vector(x, y);

  this.push = 100;

  this.vel = new p5.Vector(mouseX, mouseY);
  this.vel.sub(new p5.Vector(pmouseX, pmouseY));
  this.vel.mult(0.1);

  this.angle = random(0, 360);

  this.life = maxFireLife;
  this.lifeRate = random(0.35, 1);

  this.particles = []; // [[p5.Vector, hue], ..]
  for (var i = 0; i < 3; i++) {
    this.particles.push([new p5.Vector(random(-10, 10), random(-10, 10)),
random(0, 50)]);
  }

  this.move = function() {
    this.life -= this.lifeRate;

    // Add gravity.
    this.vel.y += 0.05;

    this.pos.add(this.vel);

    // Has a chance to spawn smoke particles.
    if (int(random(5)) == 0) {
      var spawnCount = int(random(3))+1;

      for (var i = 0; i < spawnCount; i++) {
        this.spawn();
      }
    }
  }
  this.spawn = function() {
    var offset = map(this.life, maxFireLife, 0, 0, this.push);

    var pointSize = random(25, 50)*map(this.life, maxFireLife, 0, 1, 0);

    smokeParticles.push(new Smoke(this.pos.x, this.pos.y, offset, this.angle,
pointSize));
  }

  this.display = function() {
    push();

    translate(this.pos.x, this.pos.y);
    rotate(radians(this.angle));
    translate(map(this.life, maxFireLife, 0, 0, this.push), 0);
    scale(map(this.life, maxFireLife, 0, 1, 0));

    // Instead of displaying a single point, it'll display several.
    for (var i = 0; i < this.particles.length; i++) {
      var particlePos = this.particles[i][0];
      var particleHue = this.particles[i][1];

      stroke(particleHue, 255, 255, 20);
      strokeWeight(80);
      point(particlePos.x, particlePos.y);

      stroke(this.particles[i][1], 255, 255, 100);
      strokeWeight(30);
      point(particlePos.x, particlePos.y);
    }

    pop();
  }
}


function setup() {
  createCanvas(windowWidth, windowHeight);

  colorMode(HSB, 255);

  background(255);
}


function draw() {
  // Increases red hue the more there is fire on-screen.
  background(255, fireParticles.length, 255);

  // Spawn fire particles.
  if (mouseIsPressed && mouseButton == LEFT) {
      fireParticles.push(new Fire(mouseX, mouseY));
  }
  // Move and show smoke particles.
  for (var i = smokeParticles.length-1; i > -1; i--) {
    smokeParticles[i].move();
    smokeParticles[i].display();

    if (smokeParticles[i].life < 0) {
      smokeParticles.splice(i, 1);
    }
  }

  // Move and show fire particles.
  // Always draw in front of smoke.
  for (var i = fireParticles.length-1; i > -1; i--) {
    fireParticles[i].move();
    fireParticles[i].display();

    if (fireParticles[i].life < 0) {
      fireParticles.splice(i, 1);
    }
  }
}
```

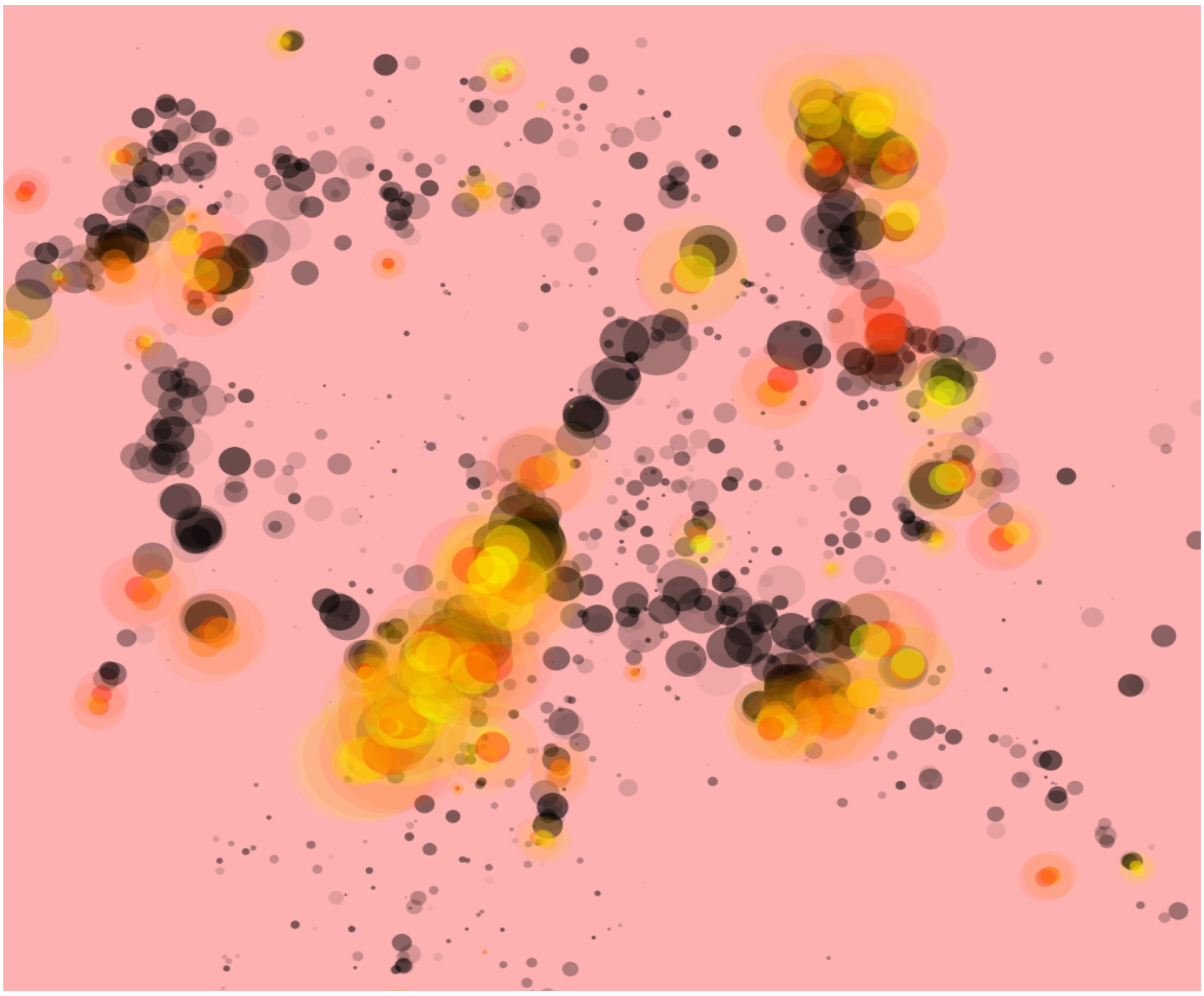


Figure 1. Fire brush screenshot

2. Source code of T318069_応用課題

```
float time = 0;
int runs = 0;
int fps = 60;
int figureRadius = 70;

color leftColor = #337C64;
color rightColor = #03FFAD;

void setup() {
  size(640, 640);
  rectMode(CENTER);
  frameRate(fps);
  noStroke();

  runs++;
}

void draw() {
  time += 1f/fps;
  float p = time;

  if (time >= 1) {
    time -= 1;
    runs++;
  }

  boolean even = runs%2==0;

  translate(width/2, height/2);

  float x = (even?1:-1) * width/3*sin(PI*p);

  fill(leftColor);
  rect(x-width/2, 0, width, height*2);
  fill(rightColor);
  rect(x+width/2, 0, width, height*2);

  if (x < 0) {
    fill(leftColor);
    drawFigure(p);

    translate(x, 0);
    drawWave(p, 1);
    drawDrops(p, 1);
  }

  if (x > 0) {
    fill(rightColor);
    drawFigure(p);

    translate(x, 0);
    drawWave(p, -1);
    drawDrops(p, -1);
  }
}

void drawFigure(float p) {
  float t = 2*(p<=0.5?p:1-p);
  float radius = figureRadius*0.5*(1+t);
  ellipse(0, 0, radius, radius);
}

void drawWave(float p, int dir) {
  beginShape();
  for (float a = -PI/2; a < 3*PI/2; a += radians(2)) {
    float wx = dir*figureRadius*0.5*(sin((p>0.5?0.5:p)*2*PI)*(sin(a)+1))-dir;
    float wy = map(a, 0, PI, -figureRadius, figureRadius);
    vertex(wx, wy);
  }
  endShape();
}

void drawDrops(float p, int dir) {
  for (int i = 0; i < 10; i++) {
    float a = map(i, 0, 10, -PI/2, PI/2) + (dir==-1?PI:0);
    float r = lerp(0, 5*figureRadius, p);
    float k = 2*(1-p);

    pushMatrix();
    translate(r*cos(a), r*sin(a));
    rotate(a-PI/2);
    for (int j = 1; j < 6; j++) {
      ellipse(0,j*2*k,j*k,j*k);
    }
    popMatrix();
  }
}
```

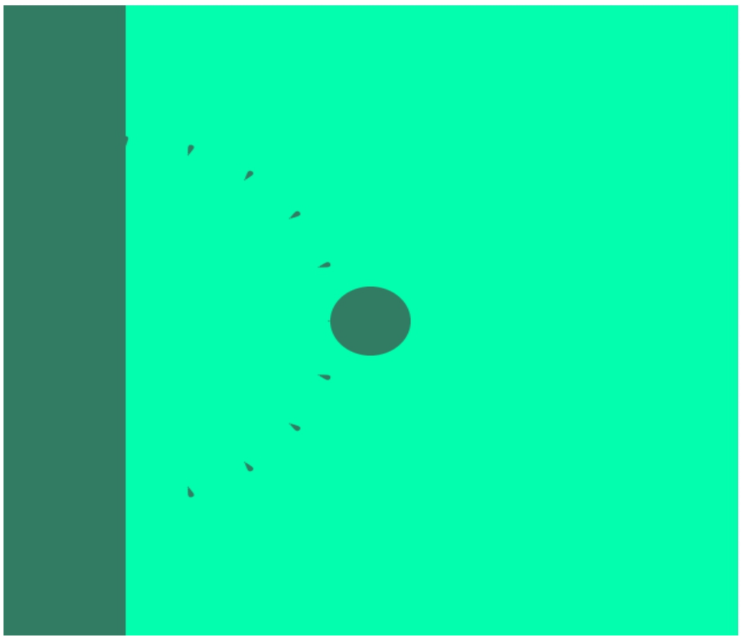


Figure 2. Display of *T318069_応用課題*

Appendix C Source Code

1. Mouse trace

```
var smokeParticles = [];
var fireParticles = [];
var maxSmokeLife = 30;
var maxFireLife = 50;
var soundFile;

// Floats up while curling, then dissapears.
function Smoke(x, y, offset, parentAngle, pointSize) {

  this.pos = new p5.Vector(x, y);
  this.offset = offset;
  this.push = random(-100, -10);

  this.parentAngle = parentAngle;
  this.angle = random(-45, 45);
  this.angleRate = random(-2, 2);

  this.pointSize = pointSize
  this.alpha = random(10, 150);

  this.life = maxSmokeLife;
  this.lifeRate = random(0.4, 1.25);

  this.move = function() {
    this.life -= this.lifeRate;
    this.angle += this.angleRate;
  }

  this.display = function() {
    push();

    translate(this.pos.x, this.pos.y);
    rotate(radians(this.parentAngle));
    translate(this.offset, 0);
    rotate(radians(-this.parentAngle));

    rotate(radians(this.angle));
    translate(0, map(this.life, maxSmokeLife, 0, 0, this.push));
    scale(map(this.life, maxSmokeLife, 0, 1, 0));

    stroke(100, 200, 200, this.alpha);
    strokeWeight(this.pointSize);
    point(0, 0);

    pop();
  }
```

```
}

function Fire(x, y) {
  this.pos = new p5.Vector(x, y);
  this.push = 100;

  this.vel = new p5.Vector(mouseX, mouseY);
  this.vel.sub(new p5.Vector(pmouseX, pmouseY));
  this.vel.mult(0.1);

  this.angle = random(0, 360);

  this.life = maxFireLife;
  this.lifeRate = random(0.35, 1);

  this.particles = [];
  for (var i = 0; i < 3; i++) {
    this.particles.push([new p5.Vector(random(-10, 10), random(-10, 10)),
random(110, 160)]);
  }

  this.move = function() {
    this.life -= this.lifeRate;

    // Add gravity.
    this.vel.y += 0.05;

    this.pos.add(this.vel);

    // Has a chance to spawn smoke particles.
    if (int(random(5)) == 0) {
      var spawnCount = int(random(3))+1;

      for (var i = 0; i < spawnCount; i++) {
        this.spawn();
      }
    }
  }

  this.spawn = function() {
    var offset = map(this.life, maxFireLife, 0, 0, this.push);

    var pointSize = random(25, 50)*map(this.life, maxFireLife, 0, 1, 0);

    smokeParticles.push(new Smoke(this.pos.x, this.pos.y, offset, this.angle,
pointSize));
  }

  this.display = function() {
    push();

    translate(this.pos.x, this.pos.y);
    rotate(radians(this.angle));
    translate(map(this.life, maxFireLife, 0, 0, this.push), 0);
    scale(map(this.life, maxFireLife, 0, 1, 0));
    for (var i = 0; i < this.particles.length; i++) {
      var particlePos = this.particles[i][0];
      var particleHue = this.particles[i][1];

      stroke(particleHue, 255, 255, 20);
      strokeWeight(80);
      point(particlePos.x, particlePos.y);

      stroke(this.particles[i][1], 255, 255, 100);
      strokeWeight(30);
      point(particlePos.x, particlePos.y);
    }

    pop();
  }
}

function preload(){
  soundFormats('mp3', 'wav');
  soundFile = loadSound('assets/mixedBeat.wav');
}

function setup() {
  let cnv = createCanvas(windowWidth, windowHeight);
  cnv.mousePressed(pressed);

  colorMode(HSB, 255);

  background(255);

  filter = new p5.BandPass();
}


function draw() {
  // Increases red hue the more there is fire on-screen.
  background(170, fireParticles.length, 255);

  let volRate = map(mouseX, 0, width, -0.5, 0.5);
  soundFile.setVolume(0.5 + volRate);

  let playbackRate = map(mouseY, 0.1, height, 2, 0);
  playbackRate = constrain(playbackRate, 0.1, 5);
  soundFile.rate(playbackRate);

  // Spawn fire particles.
  if (mouseIsPressed && mouseButton == LEFT) {
      fireParticles.push(new Fire(mouseX, mouseY));
  }

  for (var i = smokeParticles.length-1; i > -1; i--) {
    smokeParticles[i].move();
    smokeParticles[i].display();
    if (smokeParticles[i].life < 0) {
      smokeParticles.splice(i, 1);
    }
  }

  for (var i = fireParticles.length-1; i > -1; i--) {
    fireParticles[i].move();
    fireParticles[i].display();

    if (fireParticles[i].life < 0) {
      fireParticles.splice(i, 1);
    }
  }
}

function pressed(){
  soundFile.loop();
}

function mouseReleased(){
  soundFile.pause();
}
```

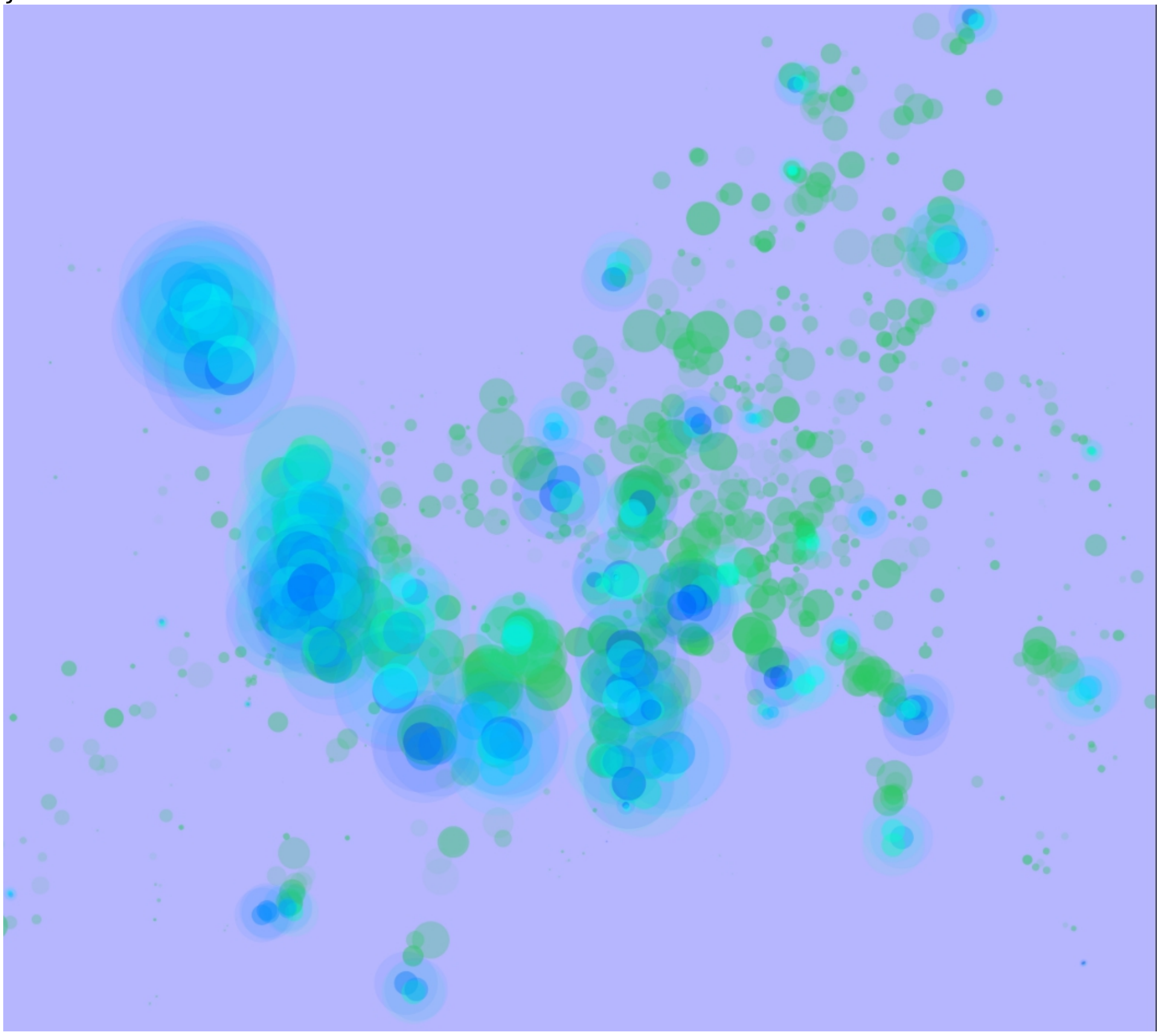


Figure 3. Display of Mouse trace.

2. Jumping ball

```
var time = 0;
var runs = 0;
var fps = 60;
var figureRadius = 70;

var leftColor, rightColor;

var osc, freq;

function setup() {
  let cnv = createCanvas(1000, 600);
  leftColor = color(127,255,212);
  rightColor = color(0,191,255);

  rectMode(CENTER);
  frameRate(fps);
  noStroke();

  runs++;

  osc = new p5.Oscillator('sine');
  osc.start();
}

function draw() {
  time += 1/fps;
  let p = time;

  if (time >= 1) {
    time -= 1;
    runs++;
  }

  let even = runs%2==0;

  translate(width/2, height/2);

  let x = (even?1:-1) * width/3*sin(PI*p);

  fill(leftColor);
  rect(x-width/2, 0, width, height*2);
  fill(rightColor);
  rect(x+width/2, 0, width, height*2);

  if (x < 0) {
    fill(leftColor);
    drawFigure(p);

    translate(x, 0);
    drawWave(p, 1);
    drawDrops(p, 1);
  }
  if (x > 0) {
    fill(rightColor);
    drawFigure(p);

    translate(x, 0);
    drawWave(p, -1);
    drawDrops(p, -1);
  }

  freq = constrain(map(x, 0, width, 100, 500), 100, 500);
  osc.freq(freq, 0.1);
}

function drawFigure(p) {
  let t = 2*(p<=0.5?p:1-p);
  let radius = figureRadius*0.5*(1+t);
  ellipse(0, 0, radius, radius);
}

function drawWave(p, dir) {
  beginShape();
  for (let a = -PI/2; a < 3*PI/2; a += radians(2)) {
    let wx = dir*figureRadius*0.5*(sin((p>0.5?0.5:p)*2*PI)*(sin(a)+1))-dir;
    let wy = map(a, 0, PI, -figureRadius, figureRadius);
    vertex(wx, wy);
  }
  endShape();
}

function drawDrops(p, dir) {
  for (var i = 0; i < 10; i++) {
    let a = map(i, 0, 10, -PI/2, PI/2) + (dir==-1?PI:0);
    let r = lerp(0, 5*figureRadius, p);
    let k = 2*(1-p);

    push();
    translate(r*cos(a), r*sin(a));
    rotate(a-PI/2);
    for (var j = 1; j < 6; j++) {
      ellipse(0,j*2*k,j*k,j*k);
    }
    pop();

  }
}
```

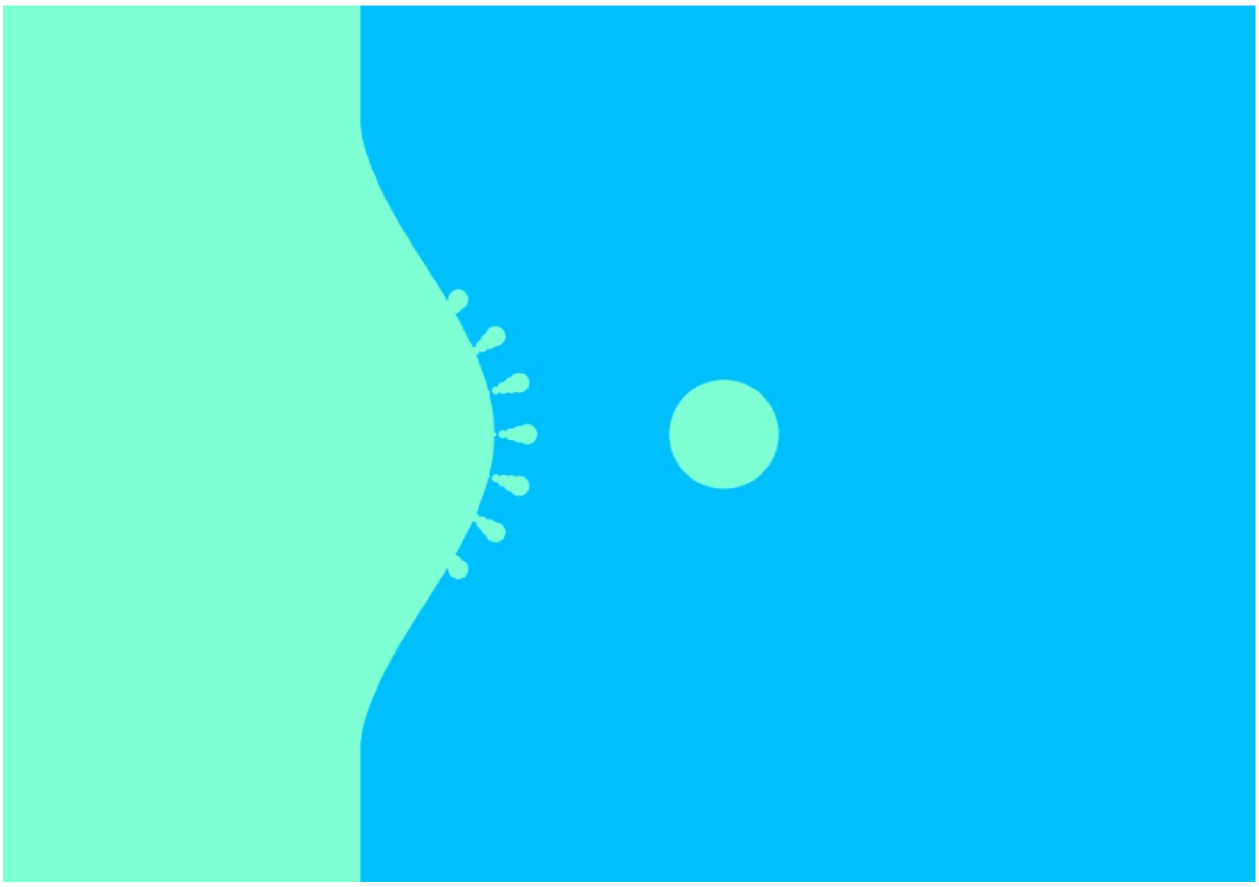


Figure 4. A screenshot when the line and ball is changing.

3. Delay Circle

```
let soundFile, delay, analyzer;
let cX, cY;
let sel;
let file1, file2, file3, file4,sound;

function preload(){
  soundFormats('ogg', 'wav');

  file1 = loadSound('assets/drum1.wav');
  file2 = loadSound('assets/mixedBeat.wav');
  file3 = loadSound('assets/latin-guitar.wav');
  file4 = loadSound('assets/koto-loop.wav');

  sound = file1;
}

function setup() {

  let cnv = createCanvas(windowWidth, windowHeight);

  delay = new p5.Delay();
  analyzer = new p5.Amplitude();

  delay.process(sound, 0.12, 0.7, 2300);
  delay.setType('pingPong');
  textAlign(CENTER);
  sel = createSelect();
  sel.position(10,10);
  sel.option('drum');
  sel.option('mixedBeat');
  sel.option('latin-guitar');
  sel.option('koto-loop');
  sel.changed(mySelectEvent);

  cnv.mousePressed(start);
  frameRate(10);
}

function draw(){
  background(255);

  let level = analyzer.getLevel();
  let levelHeight = map(level, 0, 0.1, 2, 40);

  let theColor = color(random(255),random(255),random(255));
  noStroke();
  fill(theColor);
  circle(mouseX, cY, levelHeight);

  let filterFreq = map(mouseX, 0, width, 60, 15000);
  filterFreq = constrain(filterFreq, 60, 15000);
  let filterRes = map(mouseY, 0, height, 3, 0.01);
  filterRes = constrain(filterRes, 0.01, 3);
  delay.filter(filterFreq, filterRes);
}

function mySelectEvent(){
  let item = sel.value();
  background(220);

  if(sound.isPlaying()){
    sound.pause();
  }

  switch (item){
    case 'drum':
      sound = file1;
      break;

    case 'mixedBeat':
      sound = file2;
      break;
    case 'latin-guitar':
      sound = file3;
      break;
    case 'koto-loop':
      sound = file4;
      break;
  }
  sound.playMode('restart');
  sound.loop();

  delay.process(sound, 0.12, 0.7, 2300);
}

function keyPressed(){
  if(keyCode === UP_ARROW){
    cY = cY - 10;
  }else if(keyCode === DOWN_ARROW) {
    cY = cY + 10;
  }
    let delTime = map(cY, 0, height, 0.01, 0.5);
    delTime = constrain(delTime, 0.01, 0.5);
    delay.delayTime(delTime);
}

function start() {
  sound.loop();
  cY = mouseY;
}

function mouseReleased() {
  sound.stop();
}
```

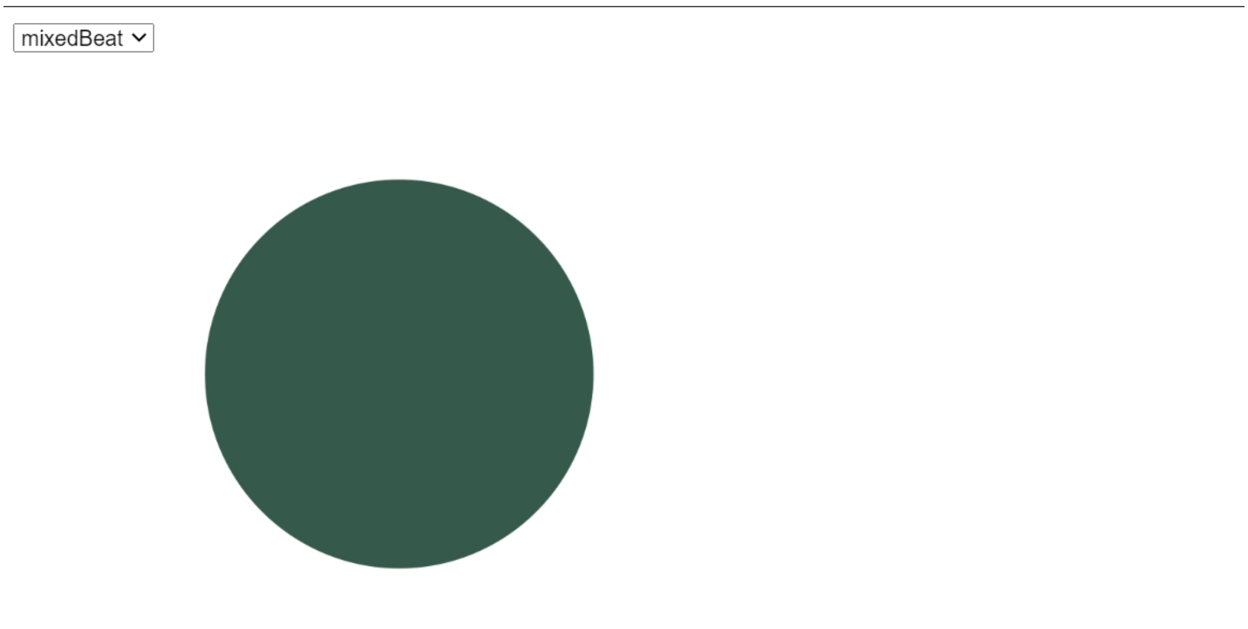


Figure 5. A screenshot of Delay Circle sketch

4. Frequency wave

```
let modulator; // this oscillator will modulate the frequency of the carrier

let sound;
let analyzer;

// min/max ranges for modulator
let modMaxFreq = 112;
let modMinFreq = 0;
let modMaxDepth = 150;
let modMinDepth = -150;

function preload(){
  soundFormats('ogg', 'wav');
  sound = loadSound('assets/mixedBeat.wav');
}

function setup() {
  let cnv = createCanvas(800, 400);
  noFill();

  analyzer = new p5.FFT();
  sound.amp(0.2);
  cnv.mouseClicked(startPlay);
  frameRate(30);
}

function draw() {
  background(30);

  // analyze the waveform
  let waveform = analyzer.waveform();

  // draw the shape of the waveform
  stroke(255);
  strokeWeight(10);
  beginShape();
  for (let i = 0; i < waveform.length; i++) {
    let x = map(i, 0, waveform.length, 100, 700);
    let y = map(waveform[i], -1, 1, -height / 2, height / 2);
    vertex(x, y + height / 2);
  }
  endShape();

  line(0, height/2+1, 100, height/2+1);
  line(700, height/2+1, width, height/2+1);
}

function startPlay(){
  if(sound.isPlaying()){
    sound.pause();
  }else{
    sound.loop();
  }
}
```

Figure 6. Screenshot of Frequency wave

Appendix D Test Details

App.test.js:

```
import { render, screen } from '@testing-library/react';
import React from 'react';
import { shallow } from 'enzyme';
import Home from './components/Home';
import Vis from './components/Vis';

test('renders learn react link', () => {
  render(<Home />);
  const linkElement = screen.getByText(/learn react/i);
  expect(linkElement).toBeInTheDocument();
});

it('renders welcome message', () => {
  const wrapper = shallow(<Vis />);
  const welcome = <h2>Welcome to React</h2>;
  // expect(wrapper.contains(welcome)).toBe(true);
  expect(wrapper.contains(welcome)).toEqual(true);
});
```

Result:

```
 FAIL  src/__tests__/Home.test.js
  ● Test suite failed to run

    ENOENT: no such file or directory, open 'D:\final-project\src\__tests__\Home.test.js'

      at runTestInternal (node_modules/jest-runner/build/runTest.js:202:27)

 FAIL  src/App.test.js
  ● Test suite failed to run

    Target container is not a DOM element.

      93 | }
      94 |
    > 95 | render(<Vis />, document.getElementById("root"));

      93 | }
      94 |
    > 95 | render(<Vis />, document.getElementById("root"));
      96 |
      97 | export default Vis;
      at render (node_modules/react-dom/cjs/react-dom.development.js:26091:13)
      at Object.<anonymous> (src/components/Vis.js:95:1)
      at Object.<anonymous> (src/App.test.js:5:1)

  Test Suites: 2 failed, 2 total
  Tests:       0 total
  Snapshots:   0 total
  Time:        6.408 s
  Ran all test suites related to changed files.
```