MID-TERM ASSIGNMENT REPORT

INTELLIGENT SIGNAL PROCESSING COURSEWORK 1 EXERCISE 2

Contents

Task 1:

All audio features are tested by drawing a circle using the audio features with a different scale. It is then evaluated by how much the size of the circle changes throughout the duration of the sound.

Sound1	Spectral Kurtosis	As the pitch of the sound changes throughout the duration of the sound, it will provide a good base for the visual effect. Based on circleSize = features.spectralKurtosis * 1;
	Spectral Skewness	SpectralSkewness relies on frequency and spectrum as its 2 key physical characteristics. It is seen that the values fluctuate throughout the duration of the audio. Based on circleSize = features.spectralSkewness * 10;
	Spectral Spread	As there are sounds of low and high spectral spread, it can be used to produce audio visual effects. Based on circleSize = features.spectralSpread * 10;
Sound2	Energy	As the sound goes from soft to load, audio features that measures loudness will be a good fit. Based on circleSize = features.energy * 10;
	Spectral Flatness	Determines how noisy a sound is, this can be a good fit for this audio. Based on circleSize = features.spectralFlatness * 500;
	Spectral Kurtosis	Pitch and tonality of sound changes over the duration of the sound. Based on circleSize = features.spectralKurtosis * 4;
Sound3	Spectral Kurtosis	Good as the pitch of this sound changes throughout the duration of the sound. Based on circleSize = features.spectralKurtosis * 1;
		Other audio features does not seem to fit this sound file as the loudness and brightness of the sound in this file is rather consistent and does not have any rapid or major change.

Task 2:

```
"featureExtractors": ["zcr", "spectralRolloff", "energy", "spectralCentroid", "spectralFlatness", "spectralSpread", "spectralKurtosis"],
"callback": features => {
    shape1Size = features.spectralRolloff * 0.01;
    shape2Size = features.spectralCentroid * 2;
    shape3Size = features.spectralFlatness * 1000;
    shape4Size = features.energy * 10;
    shape5Size = features.spectralSpread * 5;
    shape6Size = features.spectralKurtosis * 1;

    movement = features.zcr;
```

A total of 6 audio features are used as their values fluctuates throughout the duration of the audio, this allows shapes to have more effects visually as their sizes grow and shrink rapidly.

Spectral Rolloff is the frequency below a specified percentage of the total spectral energy which, in this case is 99%, lies. The value returned ranges between 0 and half of the sampling rate, where Meyda's default sampling rate is 44100Hz.

Spectral Centroid is an indication of how "bright" a given sound is, representing the spectral centre of gravity. If you were to take the spectrum, make a wooden block out of it and try to balance it on your finger (across the X axis), the spectral centroid would be the frequency that your finger "touches" when it successfully balances.

Spectral Flatness displays the flatness of the spectrum which determines how noisy a sound is. A pure sine wave will have a flatness that approaches 0.0, and white noise will have a flatness that approaches 1. '0.0' represents not flat while '1.0' is very flat.

Energy is an indication of how "loud" a given sound is which can be represented by an infinite integer.

Spectral Spread is an indication of how spread out the frequency content of the sound is across the spectrum like the frequency bandwidth. It can be used to differentiate between noisy (high spectral spread) and pitched sounds (low spectral spread).

Spectral Kurtosis is an indication of how pointy the spectrum is which can be viewed as the opposite of spectral flatness. This is often used to indicate "pitchiness/tonality" of a sound.

```
function draw() {
    background(backgroundColour[0],backgroundColour[1], backgroundColour[2]);
    fill(0);
    drawShape(shape, frameCount);
}
```

In the draw function, we will call the drawShape function to draw the visual effect shapes according to the current shape input.

```
function drawShape(shape, frameCount) {
    if(shape=="square"){
        spectralRolloffShape("square");
        perceptualCentroidShape("square");

        fill(shape3Colour, 30, 255);
        square(350, 100, shape3Size);

        fill(shape4Colour, 30, 255);
        square(350, 250, shape4Size);

        spectralSpreadShape("square");
        spectralKurtosisShape("square");
    }
    if(shape=="circle"){
        spectralRolloffShape("circle");
        perceptualCentroidShape("circle");

        // spectralFlatness
        fill(shape3Colour, 30, 255);
        circle(350, 100, shape3Size);

        // energy
        fill(shape4Colour, 30, 255);
        circle(350, 250, shape4Size);

        spectralSpreadShape("circle");
        spectralKurtosisShape("circle");
    }
    if(shape=="triangle"){
        spectralRolloffShape("triangle");
        perceptualCentroidShape("triangle");

        fill(shape3Colour, 30, 255);
        triangle(350, 100, 350 + shape3Size/2, 100-shape3Size, 350+shape3Size, 100);

        fill(shape4Colour, 30, 255);
        triangle(350, 250, 350 + shape4Size/2, 250-shape4Size, 350+shape4Size, 250);

        spectralSpreadShape("triangle");
        spectralKurtosisShape("triangle");
    }
    if(shape=="pentagon"){
        polygon(350,100,100,5);
        spectralRolloffShape("pentagon");
        perceptualCentroidShape("pentagon");
        fill(shape3Colour, 30, 255);
        polygon(350, 100, 100, 5);
        polygon(0, 0, 200, 7);

        fill(shape3Colour, 30, 255);
        polygon(350, 100, shape3Size * 10, 5);

        fill(shape4Colour, 30, 255);
        polygon(350, 250, shape4Size * 10, 5);

        spectralSpreadShape("pentagon");
        spectralKurtosisShape("pentagon");
    }
}
```

drawShape function will handle the shape inputs and call all other respective shape functions to build the shape. Only shape 1 (spectral rolloff), shape 2 (perceptual centroid), shape 5 (spectral spread) and shape 6 (spectral kurtosis) has their own function as they are the only shapes that I will be implementing the rotation and movement effects while the 2 effects on the centre will stay in their position and only change their shape sizes.

```
function spectralRolloffShape(shape) {
    push();
    translate(100, 100);
    rotate(frameCount);
    translate(movement, 0);
    fill(shape1Colour, 30, 255);
    if(shape == "square") square(0, 0, shape1Size);
    if(shape == "circle") circle(0, 0, shape1Size);
    if(shape == "triangle") triangle(0, 0, shape1Size/2, -shape1Size, shape1Size, 0);
    if(shape == "pentagon") polygon(0, 0, shape1Size, 5);
    pop();
}

function perceptualCentroidShape(shape) {
    push();
    translate(100, 300);
    rotate(-frameCount);
    translate(movement, 0);
    fill(shape1Colour, 30, 255);
    if(shape == "square") square(0, 0, shape2Size);
    if(shape == "circle") circle(0, 0, shape2Size);
    if(shape == "triangle") triangle(0, 0, shape2Size/2, -shape2Size, shape2Size, 0);
    if(shape == "pentagon") polygon(0, 0, shape2Size, 5);
    pop();
}

function spectralSpreadShape(shape) {
    push();
    translate(600, 100);
    rotate(-frameCount-180);
    translate(movement, 0);
    fill(shape1Colour, 30, 255);
    if(shape == "square") square(0, 0, shape5Size);
    if(shape == "circle") circle(0, 0, shape5Size);
    if(shape == "triangle") triangle(0, 0, shape5Size/2, -shape5Size, shape5Size, 0);
    if(shape == "pentagon") polygon(0, 0, shape5Size, 5);
    pop();
}

function spectralKurtosisShape(shape) {
    push();
    translate(600, 300);
    rotate(frameCount-180);
    translate(movement, 0);
    fill(shape1Colour, 30, 255);
    if(shape == "square") square(0, 0, shape6Size);
    if(shape == "circle") circle(0, 0, shape6Size);
    if(shape == "triangle") triangle(0, 0, shape6Size/2, -shape6Size, shape6Size, 0);
    if(shape == "pentagon") polygon(0, 0, shape6Size, 5);
    pop();
}
```

For each of those shapes, we will wrap all commands in push pop() to specify that all translate and rotate functions apply to the respective shapes only.

Further Development Ideas:

```
var speechRec = new p5.SpeechRec('en-US', parseResult);
speechRec.continuous = true;
speechRec.interimResults = true;

function preload() {
    soundFormats('mp3', 'wav');
    mySound = loadSound('/sounds/Kalte_Ohren_(_Remix_).mp3')
}

function setup() {
    createCanvas(700, 500);
    background(180);
    backgroundColour = [180,180,180];
    speechRec.start();
```

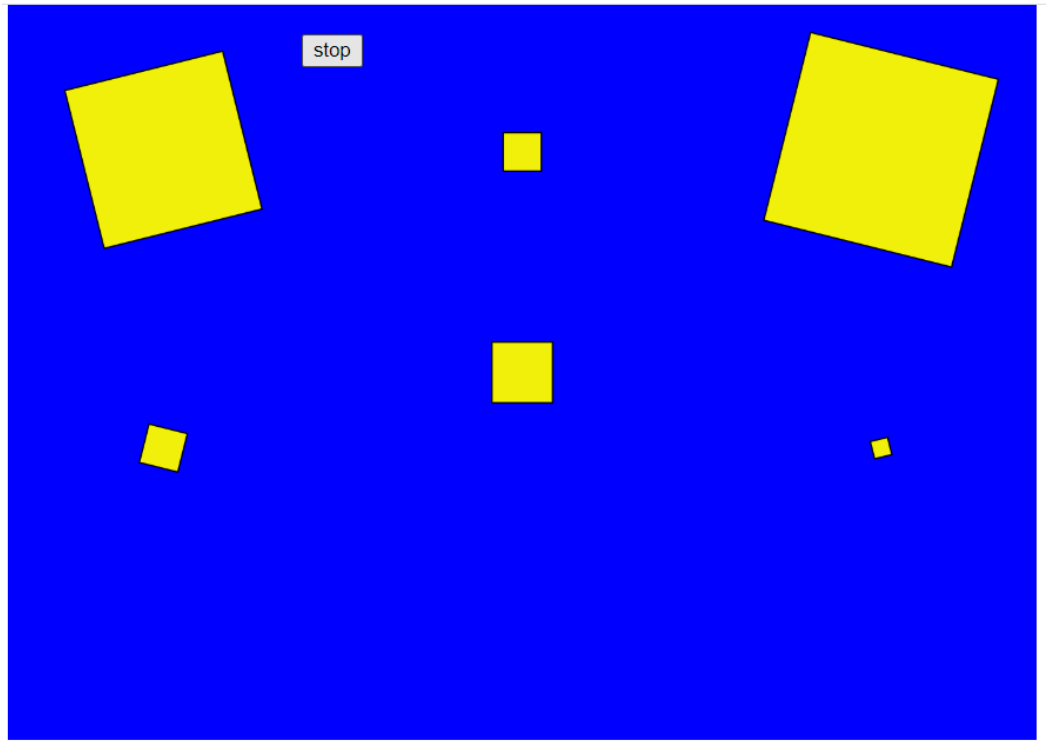
We will set up US English speech recognition, set continuous and interim results to true so that the speech engine will continuously give results and will give faster partial results instead of waiting for

the speaker to pause. Once the setup is done, we will start the speech recognition engine with speechRec.start().

```
102  function parseResult()
103  {
104      var mostRecentWord = speechRec.resultString.spli
105      if(mostRecentWord.indexOf("black")!==-1) {
106          backgroundColour=[0,0,0];
107          shape1Colour = [240, 240, 240];
108          shape2Colour = [240, 240, 240];
109          shape3Colour = [240, 240, 240];
110          shape4Colour = [240, 240, 240];
111          shape5Colour = [240, 240, 240];
112          shape6Colour = [240, 240, 240];
113      }
114      if(mostRecentWord.indexOf("white")!==-1) {
115          backgroundColour=[255,255,255];
116          shape1Colour = [10, 10, 10];
117          shape2Colour = [10, 10, 10];
118          shape3Colour = [10, 10, 10];
119          shape4Colour = [10, 10, 10];
120          shape5Colour = [10, 10, 10];
121          shape6Colour = [10, 10, 10];
122      }
123      if(mostRecentWord.indexOf("red")!==-1) {
124          backgroundColour=[255,0,0];
125          shape1Colour = [10, 240, 240];
126          shape2Colour = [10, 240, 240];
127          shape3Colour = [10, 240, 240];
128          shape4Colour = [10, 240, 240];
129          shape5Colour = [10, 240, 240];
130          shape6Colour = [10, 240, 240];
131      }
132      if(mostRecentWord.indexOf("green")!==-1) {
133          backgroundColour=[0,255,0];
134          shape1Colour = [240, 10, 240];
135          shape2Colour = [240, 10, 240];
136          shape3Colour = [240, 10, 240];
137          shape4Colour = [240, 10, 240];
138          shape5Colour = [240, 10, 240];
139          shape6Colour = [240, 10, 240];
140      }
141      if(mostRecentWord.indexOf("blue")!==-1) {
142          backgroundColour=[0,0,255];
143          shape1Colour = [240, 240, 10];
144          shape2Colour = [240, 240, 10];
145          shape3Colour = [240, 240, 10];
146          shape4Colour = [240, 240, 10];
147          shape5Colour = [240, 240, 10];
148          shape6Colour = [240, 240, 10];
149      }
150      if(mostRecentWord.indexOf("square")!==-1) {
151          shape = "square"
152      }
153      if(mostRecentWord.indexOf("triangle")!==-1) {
154          shape = "triangle"
155      }
156      if(mostRecentWord.indexOf("circle")!==-1) {
157          shape = "circle"
158      }
159      if(mostRecentWord.indexOf("pentagon")!==-1) {
160          shape = "pentagon"
161      }
```

As seen from line 102 to 161, it will recognise voice commands such as black, white, red, green and blue which will change the variable “backgroundColour” and all shape colours. Once these values get updated, the draw function will draw the background and shapes with those specific RGB colours.

Other voice commands that can be recognised includes square, triangle, circle and pentagon which changes the shape variable, which will determine what shapes to be drawn for the effects representing the audio features.

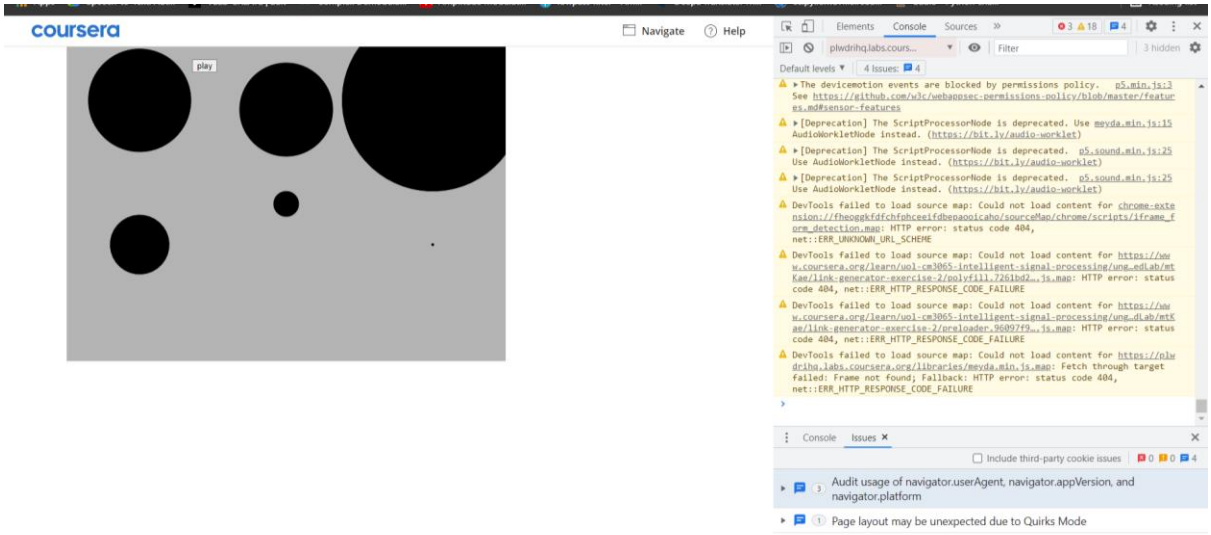


Every feature works as it should except for the pentagon shape as seen in the screenshot below.

Shareable Lab Link:

https://hub.labs.coursera.org:443/connect/sharedzvwnaguf?forceRefresh=false&path=%2FvMBxTlyvUcM5DXb37Upsye6EhaJPW4z7oTLHiHdIrBJcLsAIkbpg8LFmKA6qdMB0%2F

After uploading the zip files, the application works but the speech recognition feature does not seem to launch, there are no prompts for microphone usage despite that feature working on the same code while running on my local server.



I have tried opening this link but received this result:

To avoid an issue of inability to access my submission files, I have created a google drive shareable link with the same files (“Ex2_task2_code.zip”) included in it.

Google drive link:
https://drive.google.com/file/d/1DscYeQ4tg2K0hiceq305gs9vsrxxJn6H/view?usp=sharing