Task 1

	Meyda audio features	Justification
Sound 1		
	Chroma	The recording plays different chords, therefore chroma can be used to detect the key or the chords that are being played, by calculating the amount of each chromatic pitch class that can be found in the signal. Thus the variation in chords can then be visualized to provide good impact.
	RMS	Since chords are being played with some echoes, RMS is a good audio feature to help visualize the difference in loudness of the signal, between the actual chord being played and its subsequent echoes.
	Loudness	Since the sound is being played on different levels of loudness, the loudness feature can be used to construct filters that corresponds better to the loudness levels that humans can perceive.
Sound 2		
	Energy	It can be used to visualize the loudness of the signal, which generally increases as the audio plays.
	Spectral Centroid	This can be used to quantify the brightness of the audio file, and will help listeners identify the various instruments being used by comparing to their general spectral centroid levels.
	Spectral Rolloff	Since the audio is being played in decreasing level of loudness, the frequency changes. This feature will then be able to get the approximate value for the signal frequency and map it visually.
Sound 3		

	Energy	It can be used to visualize the loudness of the signal, which generally decreases as the audio plays.
	Spectral Flatness	This feature can be used to measure how noisy the sound is, which will give good visualization as the flatness of the spectrum can then be measured.
	Spectral Rolloff	Since the audio is being played in decreasing level of loudness, the frequency changes. This feature will then be able to get the approximate value for the signal frequency and map it visually.

To make it similar to the sample given in the question, I have decided to create 6 different rectangles in the draw function, where each rectangle has its different visual feature be affected by the same meyda audio feature. This is done in draw().

```
50      var rectLength = rectSize*1.2;
51
52  //  rect 1
53      rect(50, 250, rectLength, rectSize);
54      fill(rectColour, rectColour, 255, rectOpacity);
55      strokeWeight(borderSize);
56      stroke(borderColour, borderColour, 255, borderOpacity);
57
58  //  rect 2
59      rect(150, 230, rectLength, rectSize*1.3);
70      fill(rectColour, 255, rectColour, rectOpacity*1.3);
71      strokeWeight(borderSize*2);
72      stroke(borderColour, borderColour, 100, borderOpacity*1.3);
73
74  //  rect 3
75      rect(290, 270, rectLength, rectSize*0.6);
76      fill(rectColour, 255, rectColour, rectOpacity*0.6);
77      strokeWeight(borderSize*1.5);
78      stroke(borderColour, 255, borderColour, borderOpacity*0.6);
79
80  //  rect 4
81      rect(360, 262, rectLength, rectSize*0.75);
82      fill(rectColour, 100, rectColour, rectOpacity*0.75);
83      strokeWeight(borderSize*1.5);
84      stroke(borderColour, 100, borderColour, borderOpacity*0.75);
85
86  //  rect 5
87      rect(430, 240, rectLength, rectSize*1.15);
88      fill(0, rectColour, rectColour, rectOpacity*1.15);
89      strokeWeight(borderSize*1.5);
90      stroke(255, borderColour, borderColour, borderOpacity*1.15);
91
92  //  rect 6
93      rect(550, 258, rectLength, rectSize*0.85);
94      fill(100, rectColour, rectColour, rectOpacity*0.85);
95      strokeWeight(borderSize*1.5);
96      stroke(100, borderColour, borderColour, borderOpacity*0.85);
```

The Meyda audio features are created in the setup function, where each feature was mapped to the corresponding visual feature and adjusted such that there is proper visual impact.

```
if (typeof Meyda ==="undefined"){
    console.log("Meyda could not be found!")
}else{
    analyzer = Meyda.createMeydaAnalyzer({
        "audioContext": getAudioContext(),
        "source": myAudio,
        "bufferSize": 512,
        "featureExtractors":[
            "rms",
            "zcr",
            "spectralCentroid",
            "spectralSlope",
            "spectralRolloff",
            "energy",
            "perceptualSharpness",
            "chroma"
        ],

        "callback": features => {
            console.log(features);
            rectSize = features.rms*1000;
            rectColour = features.zcr*10;
            borderSize = features.spectralCentroid/20;
            borderColour = features.spectralSlope*50;
            rectOpacity = features.spectralRolloff/100;
            borderOpacity = features.energy*50;
            bgColour = features.perceptualSharpness*300;
        }
    });
}
```

For the Meyda audio feature RMS, it is being called to the variable rectSize, and since RMS values are typically very small (<0.1), it is multiplied by 1000 to give it a value that is roughly below 500, such that it can be mapped to a rectangle size that is at that value and thus can be seen visually. Also RMS was chosen corresponds to the loudness of the audio, which definitely changes over time and thus can be mapped to show visual change.

For the Meyda audio feature ZCR, it is being called to the variable rectColour, and since ZCR values are typically smaller, it is multiplied by 10 to give it a value that is roughly below 255, such that it can be mapped to a colour where the individual RGB is at that value and thus can be seen visually. Also ZCR was chosen as it corresponds to the number of times the audio crosses the zero value in the buffer, which definitely changes over time and thus can be mapped to show visual change.

For the Meyda audio feature Spectral Centroid, it is being called to the variable borderSize, and since Spectral Centroid values are typically larger, it is divided by 20 to give it a value that is roughly below 5, such that it can be mapped to a rectangle size that is at that value and thus can be seen visually. Also Spectral Centroid was chosen as it is an indicator of the brightness of the audio, which definitely changes over time and thus can be mapped to show visual change.

For the Meyda audio feature spectral rolloff, it is being called to the variable rectOpacity, and since spectral rolloff values are typically very large, it is divided by 100 to give it a value that is roughly below 255, such that it can be mapped to an opacity that is at that value and thus can be seen visually. Also spectral slope was chosen as it corresponds to the measure of the inclination of the

shape of the spectrum, which definitely changes over time and thus can be mapped to show visual change.

For the Meyda audio feature energy, it is being called to the variable borderOpacity, and since energy values are typically smaller, it is multiplied by 50 to give it a value that is roughly below 255, such that it can be mapped to an opacity that is at that value and thus can be seen visually. Also spectral slope was chosen as it corresponds to the loudness of the audio, which definitely changes over time and thus can be mapped to show visual change.

For the Meyda audio feature perceptual sharpness, it is being called to the variable bgColour, and since perceptual sharpness values are typically smaller (<1), it is multiplied by 300 to give it a value that is roughly below 255, such that it can be mapped to a background colour where the individual RGB is at that value and thus can be seen visually. Also perceptual sharpness was chosen as it can detect if the audio is being sharp, which definitely changes over time and thus can be mapped to show visual change.