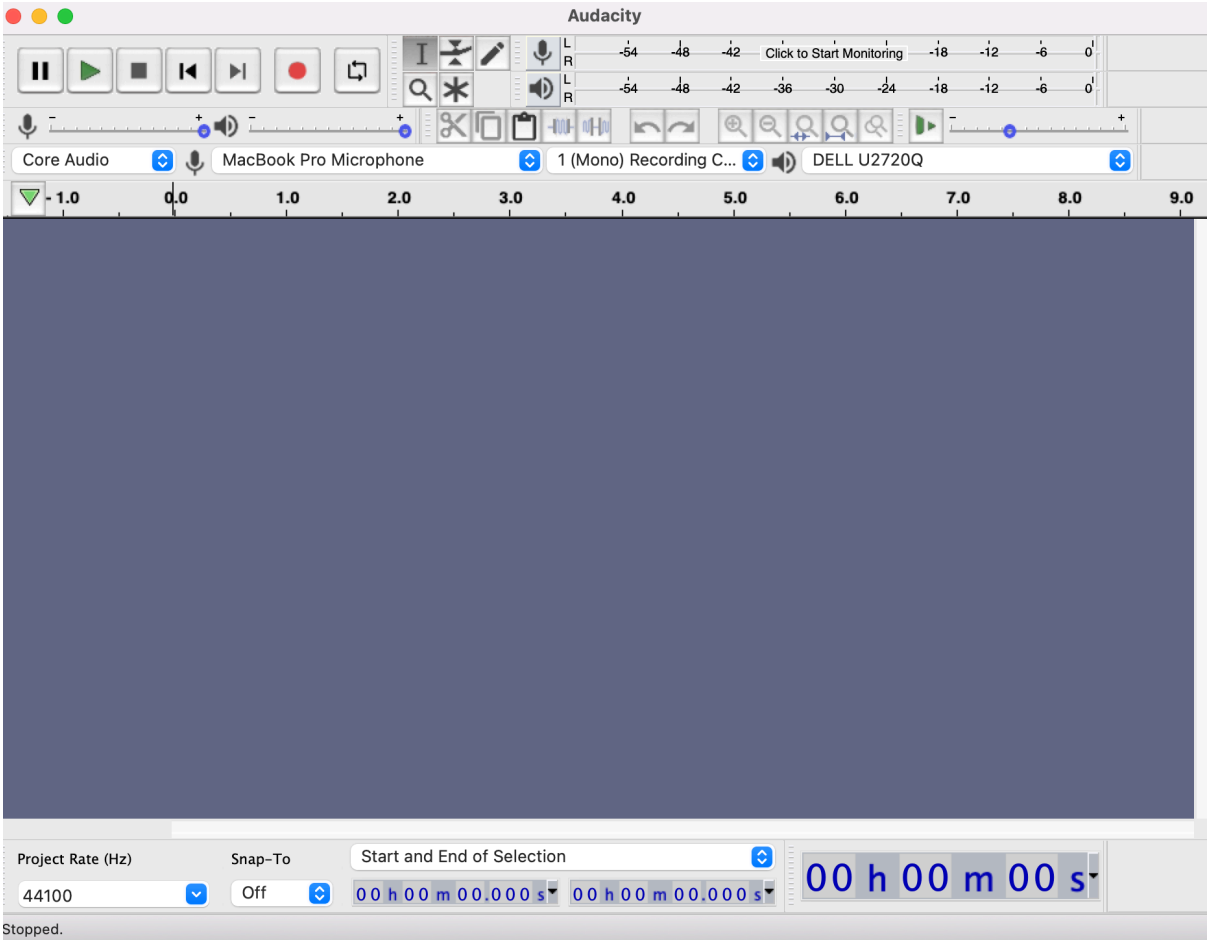
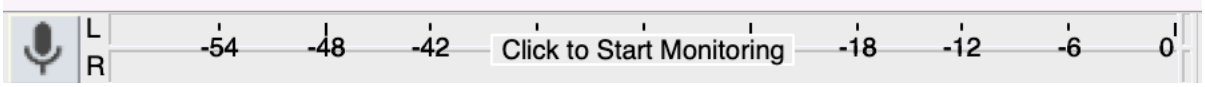
Exercise 1

This is what it looks like when you run the Audacity app. You will be greeted with a lot of buttons and functions to choose from.



To start recording an audio, the first thing we need to check is the recording device, located at the top left-hand corner. I have selected MacBook Pro Microphone as it is the only source of recording device I have. Secondly, we need to check the number of recording channels which is located just beside it. I only have single recording channel, 1 (Mono), and hence I selected that being the only option.



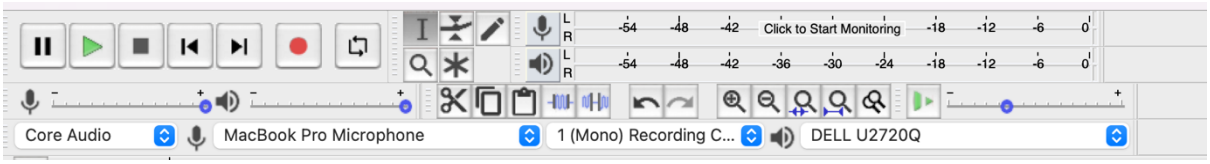
We can activate the Start Monitoring option by clicking this button. This option allows us to adjust the recording input volume. This is crucial; we must ensure that the level does not fall below zero dB; otherwise, the recording will be distorted. The ideal level range is typically minus 12 to minus six decibels. This gives you the best signal-to-noise ratio while also ensuring that the source signal has enough room to avoid clipping.

We can now select our project sample rate (Hz), located on the bottom left-hand corner. The sample rate is the number of samples per second required to convert a continuous

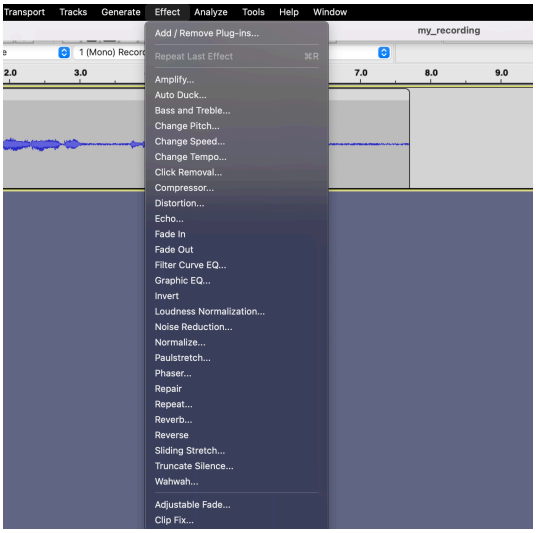
signal to a digital signal. Although 44.1KHz is a standard sampling frequency in digital audio, 48KHz is the most practical sample rate for most applications.
To start recording, we can press the Record button, which is the red circular button, on the top hand of the page.

As we picked mono recording, the audio track being recorded will only have one channel. We can now listen to the recording by pressing the green coloured Play button.

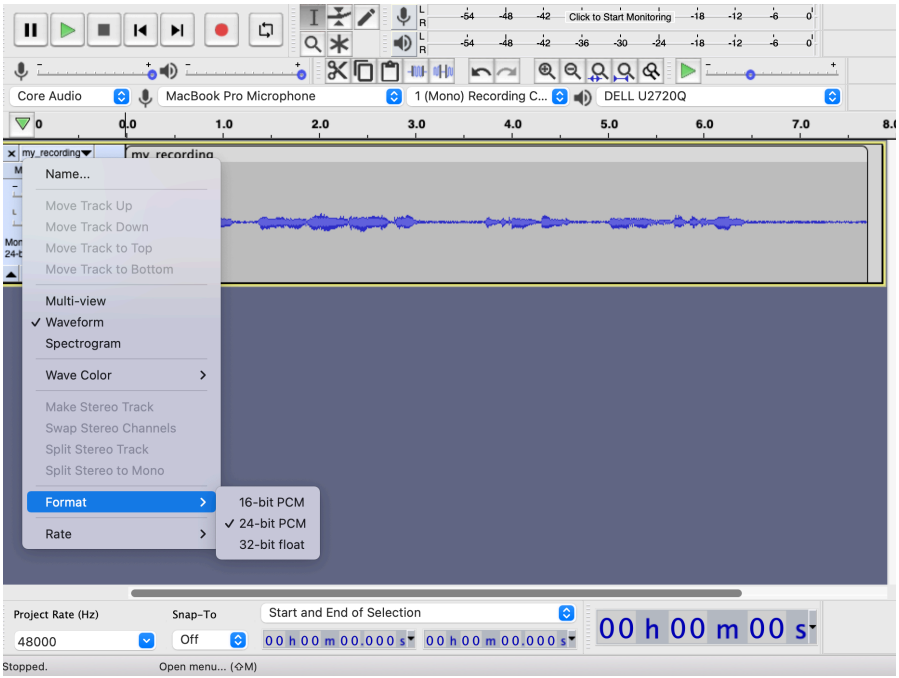
There are several tools which allow us to edit our file.
We can click the 'I' button to select and highlight a part of the track we would like to edit. There are zoom functionalities as well. Functionalities such as Cut, Copying and Pasting are also available.



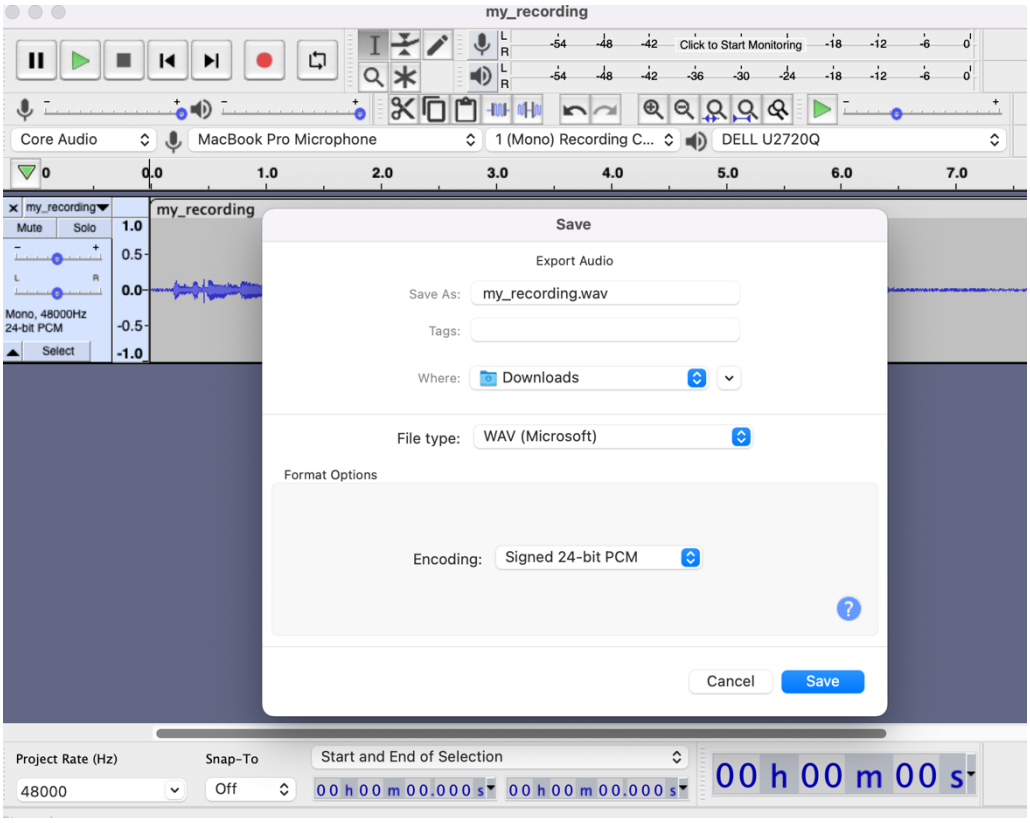
Besides that, we can also add effects to our track by clicking on the effects tab on the top. From there, we can select the effect which we would like to apply on our file.



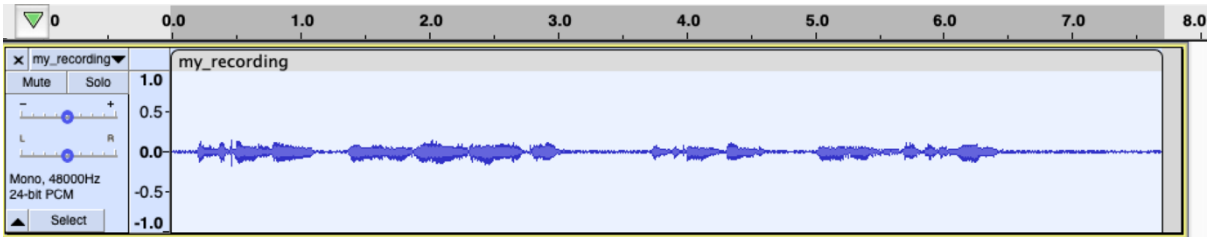
We can also select the format of our audio file by following below.



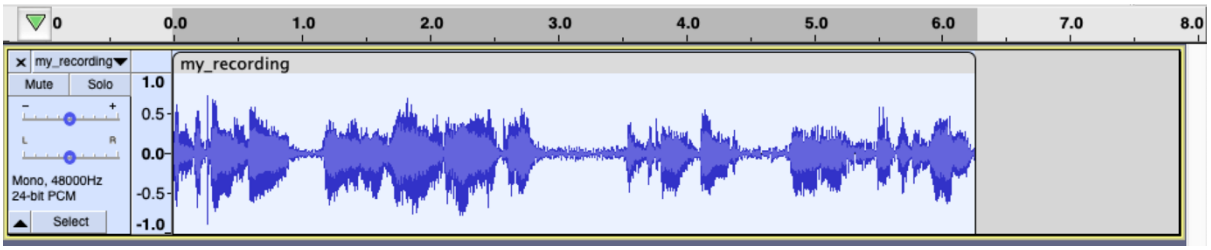
To save our recorded audio, click Export, then Export the file to your desired sound format. Afterwards, you can rename the file and then choose the encoding option which should be the same format as the one you have chosen previously.



This is the original recorded voice, before removing silences at the start and end of the file and without normalising it. I have recorded my voice at an optimal level without clipping.



This is the edited voice. I have removed silences at the start and at the end of the file. I have also normalised it.



I have set these parameter for my recording above:
Sample Rate: 48000 Hz
Format was 24bits
Silence at beginning and end of the file has been removed.
File normalized to -1dB
File saved as .wav

Low-pass filter

A low-pass filter (LPF) is an audio signal processor that filters out undesired frequencies above a preset cutoff frequency. It gradually filters out (attenuates) the high-end over its cutoff frequency while letting the low-end to pass through with few modifications, ideally.

Below shows how I have programmed it:

Under the setup() function

```
filterLowPass = new p5.LowPass();
```

Under the draw() function

```
//low pass filter
//frequency (10Hz) to the highest (22050Hz) that humans can hear
filterFreq = map(lp_cutOffSlider.value(), 0, 100, 10, 22050);

//Map to resonance (volume boost) at the cutoff frequency
filterRes = map(lp_resonanceSlider.value(), 0, 100, 15, 5);

//set filter parameters
filterLowPass.set(filterFreq, filterRes);

filterLowPass.drywet(lp_dryWetSlider.value());
filterLowPass.amp(lp_outputSlider.value());
```

Under the gui_configuration() function

```
// low-pass filter
textSize(14);
text('low-pass filter', 10,80);
textSize(10);
lp_cutOffSlider = createSlider(0, 1, 0.5, 0.01);
lp_cutOffSlider.position(10,110);
text('cutoff frequency', 10,105);
lp_resonanceSlider = createSlider(0, 1, 0.5, 0.01);
lp_resonanceSlider.position(10,155);
text('resonance', 10,150);
lp_dryWetSlider = createSlider(0, 1, 0.5, 0.01);
lp_dryWetSlider.position(10,200);
text('dry/wet', 10,195);
lp_outputSlider = createSlider(0, 1, 0.5, 0.01);
lp_outputSlider.position(10,245);
text('output level', 10,240);
```

Dynamic Compressor

The dynamic range of a sound is reduced when it is compressed. It reduces the volume of the loudest parts of the sound while increasing the volume of the quietest sections. The loudness is more constant as a result, but the dynamic range is reduced.
In music, compression is used to minimise the dynamic range of signals that contain both loud and quiet portions so that both may be heard clearly.

Below shows how I have programmed it:

Under the setup() function

```
filterDynComp = new p5.Compressor();
```

Under the draw() function

```
//dynamic compressor
filterDynComp.attack(dc_attackSlider.value());
filterDynComp.knee(dc_kneeSlider.value());
filterDynComp.release(dc_releaseSlider.value());
filterDynComp.ratio(dc_ratioSlider.value());
filterDynComp.threshold(dc_thresholdSlider.value());
filterDynComp.drywet(dc_dryWetSlider.value());
filterDynComp.amp(dc_outputSlider.value());
```

Under the gui_configuration() function

```
// dynamic compressor
textSize(14);
text('dynamic compressor', 210,80);
textSize(10);
dc_attackSlider = createSlider(0, 1, 0.003, 0.01);
dc_attackSlider.position(210,110);
text('attack', 210,105);
dc_kneeSlider = createSlider(0, 40, 30, 0.01);
dc_kneeSlider.position(210,155);
text('knee', 210,150);
dc_releaseSlider = createSlider(0, 1, 0.25, 0.01);
dc_releaseSlider.position(210,200);
text('release', 210,195);
dc_ratioSlider = createSlider(1, 20, 12, 0.01);
dc_ratioSlider.position(210,245);
text('ratio', 210,240);
dc_thresholdSlider = createSlider(-100, 0, -24, 0.01)
dc_thresholdSlider.position(360,110);
text('threshold', 360,105);
dc_dryWetSlider = createSlider(0, 1, 0.5, 0.01);
dc_dryWetSlider.position(360,155);
text('dry/wet', 360,150);
dc_outputSlider = createSlider(0, 1, 0.5, 0.01);
dc_outputSlider.position(360,200);
text('output level', 360,195);
```

Reverb

The persistence of sound after it has been generated is known as reverb. When a sound or signal is reflected off a surface, it causes many reflections to accumulate. The sound and reflections are subsequently absorbed by the surfaces of the items surrounding it, which causes them to decay. The original sound source can be turned off, but the reflections will continue to amplify (volume) until they reach zero. The way a sound resides in space is defined by reverb. We are constantly exposed to the impacts of reverb on the sounds we hear in our daily lives.

Below shows how I have programmed it:

Under the setup() function

```
reverb = new p5.Reverb();
```

Under the gui_configuration() function

```
// reverb
textSize(14);
text('reverb', 10,305);
textSize(10);
rv_durationSlider = createSlider(0, 10, 0, 0.01);
rv_durationSlider.position(10,335);
text('duration', 10,330);
rv_decaySlider = createSlider(0, 100, 2, 1);
rv_decaySlider.position(10,380);
text('decay', 10,375);
rv_dryWetSlider = createSlider(0, 1, 0, 0.01);
rv_dryWetSlider.position(10,425);
text('dry/wet', 10,420);
rv_outputSlider = createSlider(0, 1, 0, 0.01);
rv_outputSlider.position(10,470);
text('output level', 10,465);
rv_reverseButton = createButton('reverb reverse');
rv_reverseButton.position(10, 510);
  rv_reverseButton.mousePressed(reverbReverse);
```

I created a reverbReverse() function

```
function reverbReverse() {
    if(state == 0) {
        reverb.set(rv_durationSlider.value(), rv_decaySlider.value(), true);
        state++;
    }
    else if (state == 1) {
        reverb.set(rv_durationSlider.value(), rv_decaySlider.value(), false);
    }
}
```

Under the draw() function

```
//reverb

reverb.drywet(rv_dryWetSlider.value());
reverb.amp(rv_outputSlider.value());
```

Waveshaper Distortion

Waveshaping is a sort of distortion synthesis used in electronic music to create complex spectra from basic tones by changing the waveform shape.

Below shows how I have programmed it:

Under the setup() function

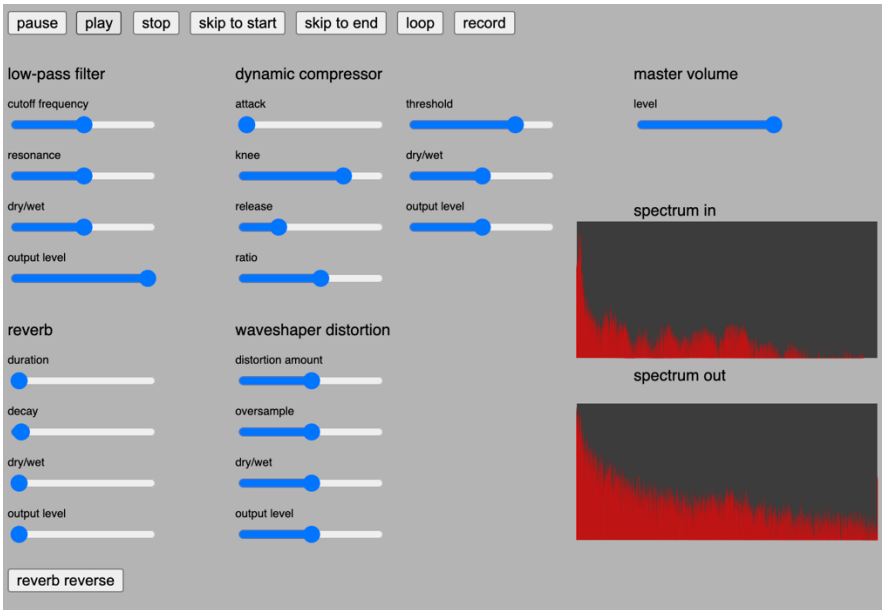
```
filterWaveShaperDistortion = new p5.Distortion();
```

Under the draw() function

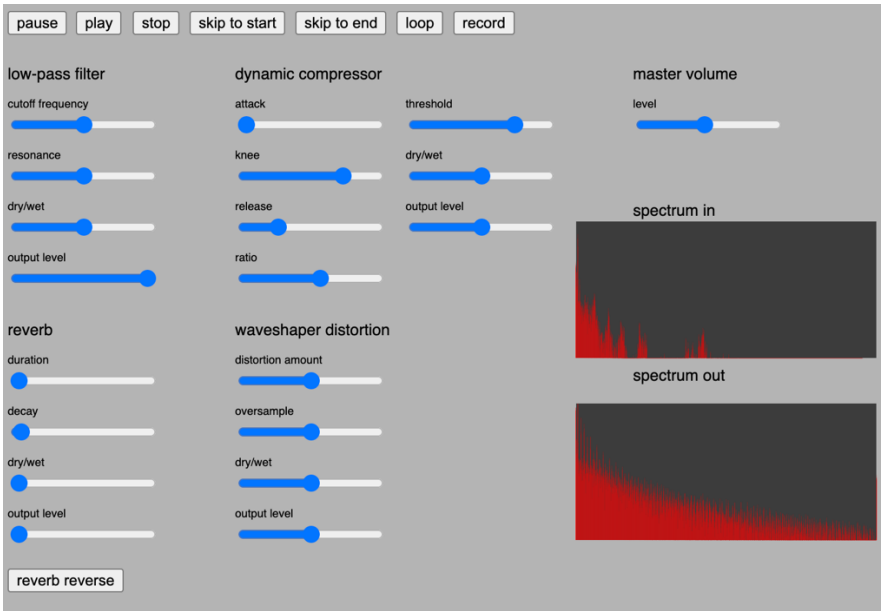
```
//waveshaper distortion
if(wd_oversampleSlider.value()==0){
    filterWaveShaperDistortion.set(wd_amountSlider.value(), + "none");
}
else {
    filterWaveShaperDistortion.set(wd_amountSlider.value(), str(wd_oversampleSlider.value()) + "x");
}

filterWaveShaperDistortion.drywet(wd_dryWetSlider.value());
filterWaveShaperDistortion.amp(wd_outputSlider.value());
```

Under the gui_configuration() function

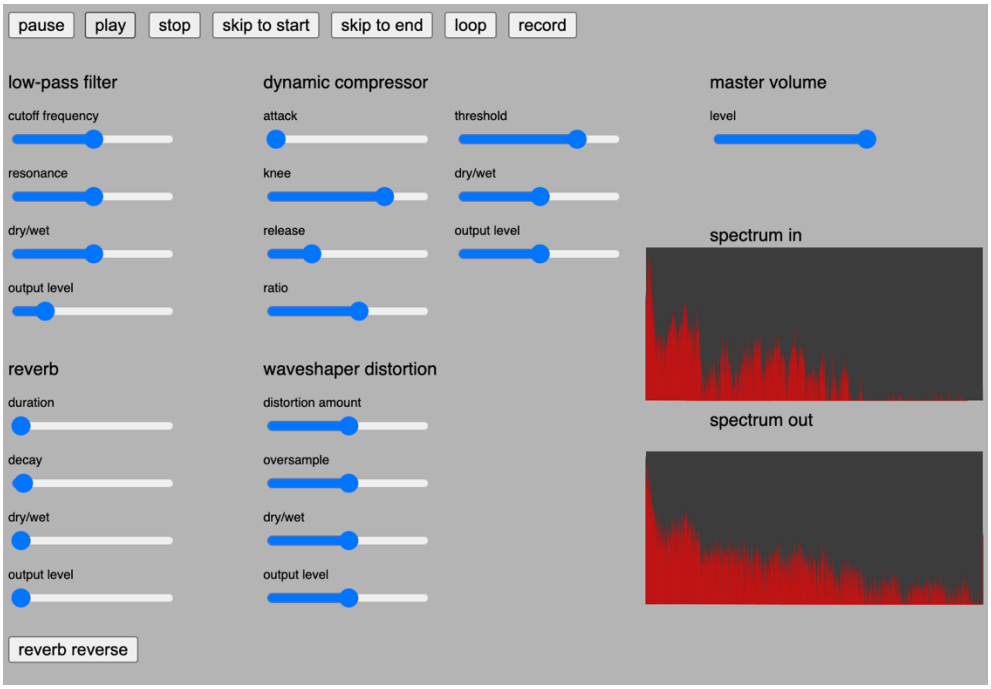
```
// waveshaper distortion
textSize(14);
text('waveshaper distortion', 210,305);
textSize(10);
wd_amountSlider = createSlider(0, 1, 0.5, 0.01);
wd_amountSlider.position(210,335);
text('distortion amount', 210,330);
wd_oversampleSlider = createSlider(0, 1, 0.5, 0.01);
wd_oversampleSlider.position(210,380);
text('oversample', 210,375);
wd_dryWetSlider = createSlider(0, 1, 0.5, 0.01);
wd_dryWetSlider.position(210,425);
text('dry/wet', 210,420);
wd_outputSlider = createSlider(0, 1, 0.5, 0.01);
wd_outputSlider.position(210,470);
text('output level', 210,465);
```



As you can see from the picture above, at maximum master volume and maximum low-pass filter, we can see that the graph tends to be higher.



At medium master volume, and maximum low-pass filter, we can see that the graph is more evenly distributed/spread out.



At maximum master volume, and with a low value of low-pass filter, we can see that the graph is somewhat relatively the same as the original audio output.

In conclusion, the master volume affects the spectrum of the sound. When there is an increase in volume, the spectrum will increase both horizontally and vertically.

Lab Link:
Generated shareable link file:
https://hub.labs.coursera.org:443/connect/sharedshekwhxv?forceRefresh=false&path=%2FvMBxTlyvUcM5DXb37Upsye6EhaJPW4z7oTLHiHdIrBJcLsAIkbpg8LFmKA6qdMB0%2F

However, the generated link above does not work. Need to trim the URL, remove everything after the word false.

This is the working link:
https://hub.labs.coursera.org:443/connect/sharedshekwhxv?forceRefresh=false