Final Project Overview

Author: Nolan Hamel
Due: Dec. 9, 2022

The Project

For this project I chose to take audio effects and apply them to images. The reason is that I've manipulated images in this was using a free program called Audacity, but I wanted something a bit more intentional and made for images.

The Effects

For each of these effects, imagine the image as a 1D array

- y = the output image
- x = the input image

Echo

My implementation of the echo effect works by combining the current value with a value N pixels earlier.

Parameters:

- N = Delay Time, the number of pixels back to look to apply to current pixel
- gFB = Feedback gain, which controls the amount of previously affected image that is mixed into the new affected image. Increasing this leads to more echos.
- gFF = Feed forward gain, which controls the significance that the echos affects the image.

```
y[i] = gFB*y[i-N]+x[i]+(gFF-gFB)*x[i-N]
```

After processing with the above equation, the output is then normalized.

Examples:

Original image

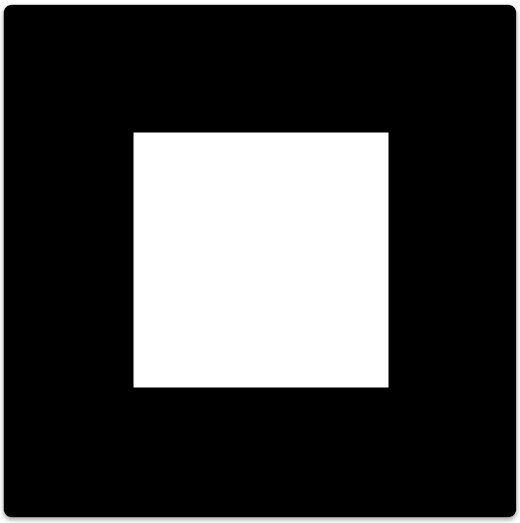
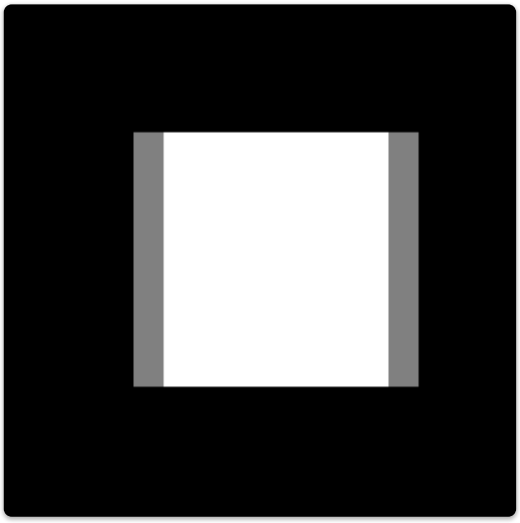
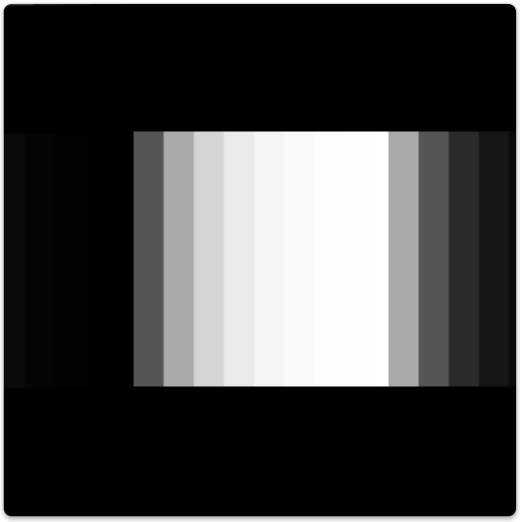


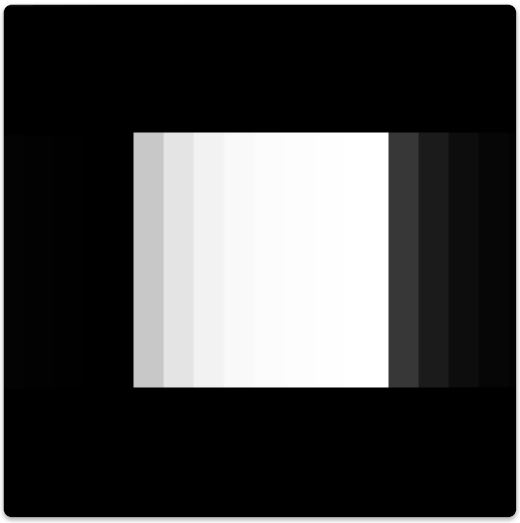
Image with feed forward gain, and no feedback gain



Increased feedback gain



Same feedback gain, decreased feed forward gain



Distortion

My implementation of the distortion effect works by clamping and amplifying the input image. It acts on each channel independently rather than on pixel values, so it can lead to some more interesting results.

Parameters:

- Clipping Level, which controls what value to clamp to. 0.4 will clamp the image to the highest 40% of possible values, so values between 255 and 153 will be clamped to 153.
- Make-up Gain, which controls how bright the affected values are. With a make-up gain of 1, the affected values will be brightened to 255.

```
const clampVal = Math.floor(255-255*clippingLevel);
if(y[i] >= clampVal) {
        outPixels[i] = clampVal;
        outPixels[i] += makeupGain*(255-outPixels[i]);
}
```

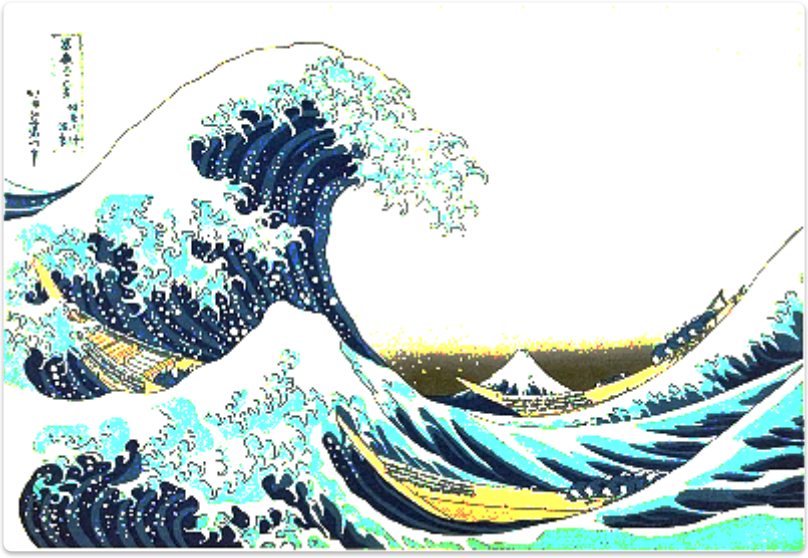
Examples:

Original Image

Image with Clip Level of 0.5

Original Image

Image with Clip Level of 0.5, full make-up gain



Without makeup gain

Chorus

My implementation of the chorus effect is like an echo with a Low Frequency Oscillator (LFO) affecting the delay time.

Parameters:

- Chorus Gain, controls how much the affected image is mixed with the original image
- Delay Time, controls the number of pixels back to look to apply to current pixel
- Sweep Width, controls the amplitude of the LFO
- Frequency, controls how many times the LFO repeats

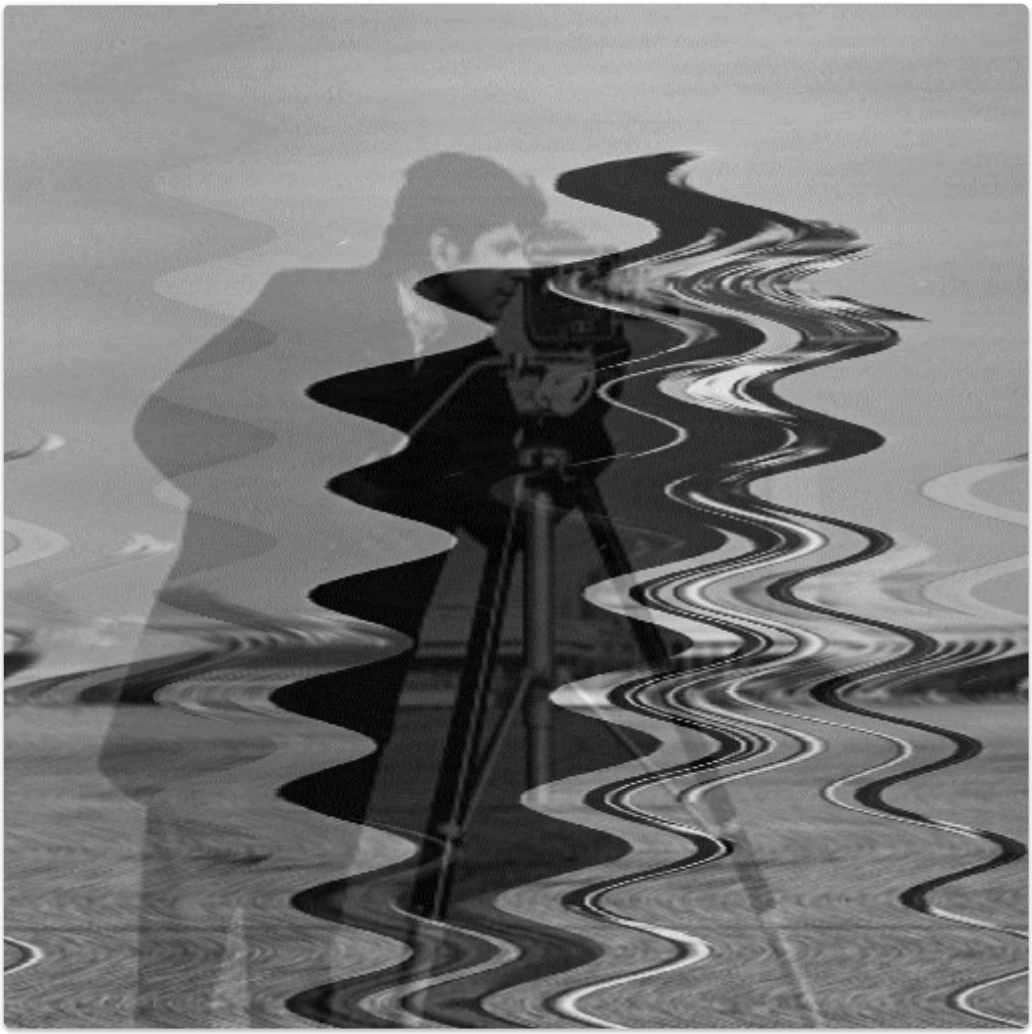
```
// calcualte lfo value
let lfo = (0.5 + 0.5 * Math.sin(2.0 * Math.PI * phase))

// calculate total Delay amount
let totalDelay = Math.round(chorusDelaySamples + ((sweepWidth/2) * (1 +
lfo)))

// make sure the delay applies across same channels
totalDelay = totalDelay - totalDelay%4

// apply the delay to pixel, n is current index
for(var i = n; i <= n+3; i++) {
        y[i] = (1-chorusGain)*x[i] + y*(x[i-totalDelay]);
}

// change the phase of lfo
phase += frequency /(imageSize/4);
if(phase >= 1.0) phase -= 1.0;
```

Examples:

Original Image

Default chorus effect

Increased Sweep Width



Increased frequency, increased chorus gain