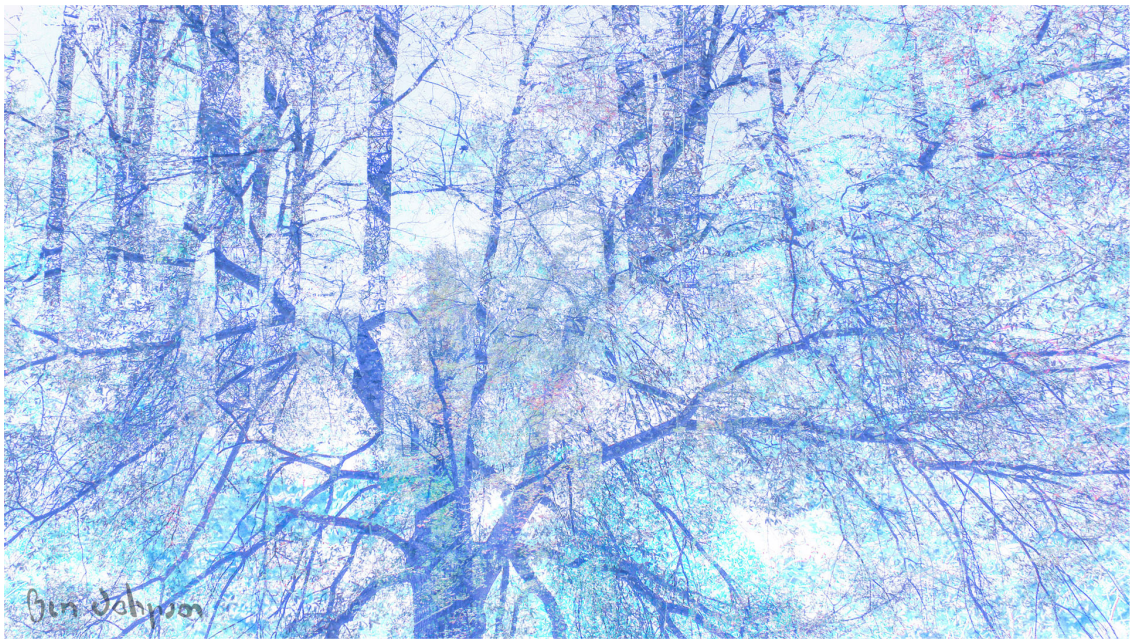
2 SEASONS



BEN JOHNSON

Table of Contents:

I. Artist Statement

Claude Monet & Antonio Vivaldi; I have always considered these two artists to be on the same conceptual plane. For this piece, I decided to combine the two for a multimedia work primarily inspired by Vivaldi's *Four Seasons*. Using Processing, I set out to create an interactive installation similar to the work of Golan Levin.

My first conceptual draft for *2Seasons* involved using a Microsoft Kinect sensor to gain audio and video input from the viewer to change an image on a projector screen. Inspired in part by Levin's *Messa di Voce*, I used a code library called "Open Kinect for Processing" by Processing co-founder Daniel Shiffman to create a demo program. This program used the space at a certain depth point in the Kinect's range to draw a color across the screen.

In my use of the Kinect sensor, I re-discovered *Fantasia: Music Evolved*, a game I used to play with my cousins when the Xbox Kinect was still popular as a game controller. Developed by Harmonix (creator of *Guitar Hero* and the *Rock Band* series), the game focuses on using hand movements to perform and re-mix songs from a variety of genres.

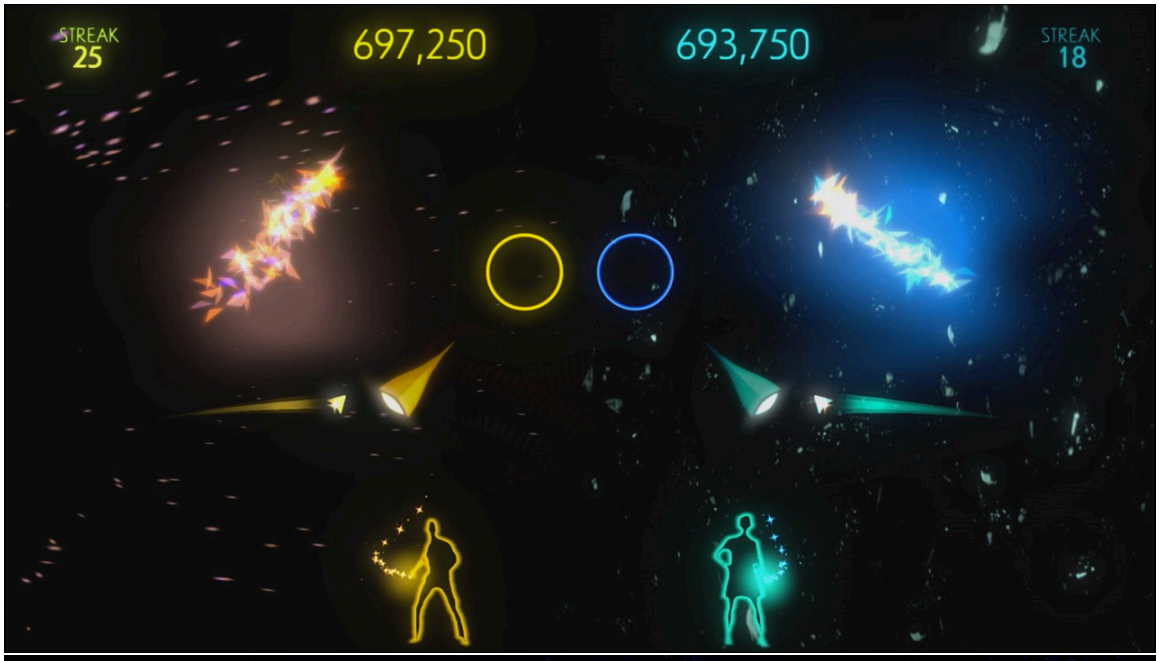
Re-experiencing this game is what inspired me to add an audio element to the piece. Using "Sound," another Processing library by Shiffman and the Processing Foundation, I created a program that changes the color of a photo on the screen and the mix of an audio track depending on the position of the mouse. In this initial build, I used the original multi-track recordings for David Bowie's song "Moonage Daydream," allowing the user to hear the full song if the cursor is on the right side of the screen and Bowie's isolated vocal track if the cursor is on the left side of the screen.

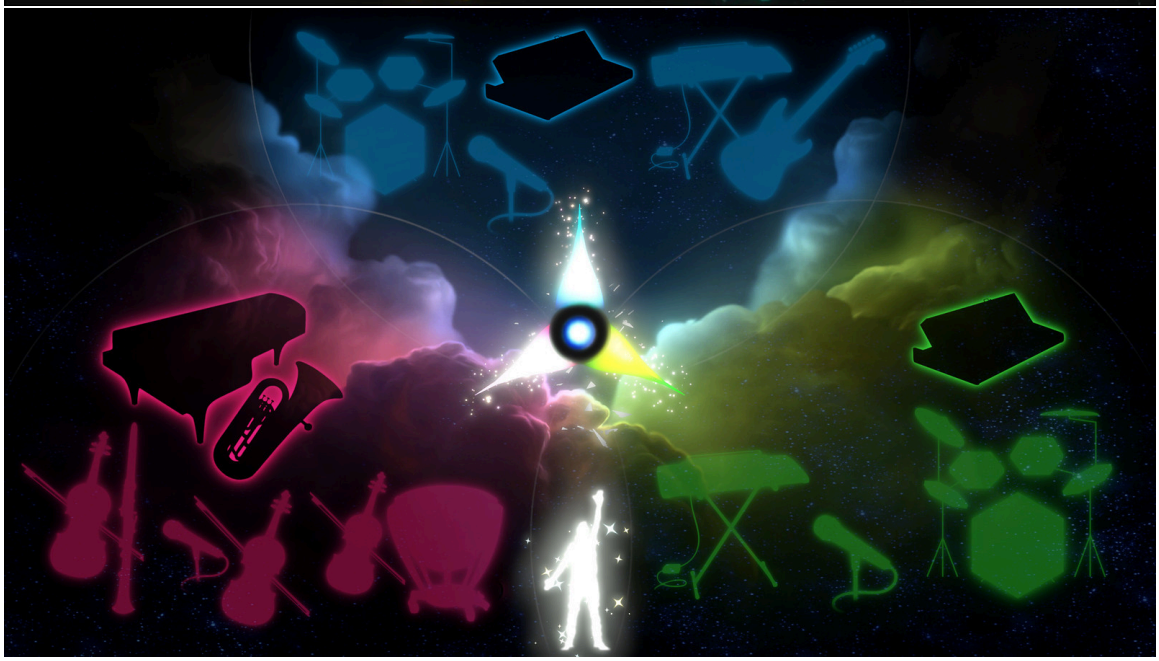
Conceptually, with this piece I wanted to explore the change of nature through seasons. This lead to my use of Vivaldi's *Four Seasons* as an element to manipulate and re-contextualize the work. When looking for a way to visually represent this change, I was drawn to Monet's impressionist pieces and his process of painting different versions of the same scene. The final version of this piece uses a touch screen for input, as the Kinect sensor proved too error-prone for long-term use.

Using overlaying techniques in Photoshop, I imitated Monet's signature technique of layering colors over a canvas by combining digital photography. Two separate pieces can be seen in this display and touching each side of the screen will reveal either winter or autumn. Corresponding to this visual change, you will hear a different selection from *Four Seasons*--I've selected the first movement of *Winter* and the second movement of *Autumn* for this display, as they are the most sonically similar of the twelve movements. Both pieces have been transposed to D minor and abstracted to reflect the visual pieces they accompany.

II. Research Portfolio

Screenshots and gameplay from *Fantasia: Music Evolved*, taken from Gaming Trend's 2014 Review (http://gamingtrend.com/reviews/disneys-fantasia-music-evolved-imagination-has-no-age/).





Images of Golan Levin's *Messa di Voce* and *Rectified Flowers* taken from Levin's website Flong.com

Images of Claude Monet's *San Giorgio Maggiore at Dusk* and *Water Lilies and the Japanese Bridge* taken from Wikipedia Commons under Public Domain (https://en.wikipedia.org/wiki/Claude_Monet#/media/File:Water-Lilies-and-Japanese-Bridge-(1897-1899)-Monet.jpg and https://en.wikipedia.org/wiki/San_Giorgio_Maggiore_at_Dusk#/media/File:Claude_Monet,_Saint-Georges_majeur_au_cr%C3%A9puscule.jpg)

Source example code for Daniel Shiffman and The Processing Foundation's *Sound* and *Open Kinect* libraries taken from Shiffman's GitHub page (https://github.com/shiffman/)

Sample.pde

```
/*This is a sound file player. */

import processing.sound.*;

SoundFile soundfile;

void setup() { size(640,360); background(255);

//Load a soundfile

soundfile = new SoundFile(this, "vibraphon.aiff");

// These methods return useful infos about the file

println("SFSampleRate= " + soundfile.sampleRate() + " Hz"); println("SFSamples= " +
soundfile.frames() + " samples"); println("SFDuration= " + soundfile.duration() + " seconds");

// Play the file in a loop

soundfile.loop(); }

void draw() {// Map mouseX from 0.25 to 4.0 for playback rate. 1 equals original playback //
speed 2 is an octave up 0.5 is an octave down.soundfile.rate(map(mouseX, 0, width, 0.25, 4.0));

// Map mouseY from 0.2 to 1.0 for amplitude

soundfile.amp(map(mouseY, 0, width, 0.2, 1.0));

// Map mouseY from -1.0 to 1.0 for left to right

soundfile.pan(map(mouseY, 0, width, -1.0, 1.0)); }
```

AveragePointTracking2.pde

```
// Daniel Shiffman and Thomas Sanchez Lengeling// Tracking the average location beyond a
given depth threshold // Thanks to Dan O&amp;apos;Sullivan

// https://github.com/shiffman/OpenKinect-for-Processing // http://shiffman.net/p5/kinect/

import org.openkinect.processing.*;// The kinect stuff is happening in another class

KinectTracker tracker;

void setup() { size(640, 520);
```

```
tracker = new KinectTracker(this); }

void draw() { background(255);

// Run the tracking analysis

tracker.track();

// Show the image

tracker.display();

// Let&apos;s draw the raw location

PVector v1 = tracker.getPos(); fill(50, 100, 250, 200); noStroke();ellipse(v1.x, v1.y, 20, 20);

// Let&apos;s draw the "lerped" location

PVector v2 = tracker.getLerpedPos(); fill(100, 250, 50, 200);noStroke();ellipse(v2.x, v2.y, 20,
20);

// Display some info

int t = tracker.getThreshold();fill(0);text("threshold: " + t + " " + "framerate: " + int(frameRate) +
" " +

"UP increase threshold, DOWN decrease threshold", 10, 500); }

// Adjust the threshold with key presses

// Adjust the threshold with key presses

void keyPressed() {int t = tracker.getThreshold(); if (key == CODED) {

if (keyCode == UP) {t +=5; tracker.setThreshold(t);

} else if (keyCode == DOWN) { t -=5;

tracker.setThreshold(t); }

} }
```

KinectTracker

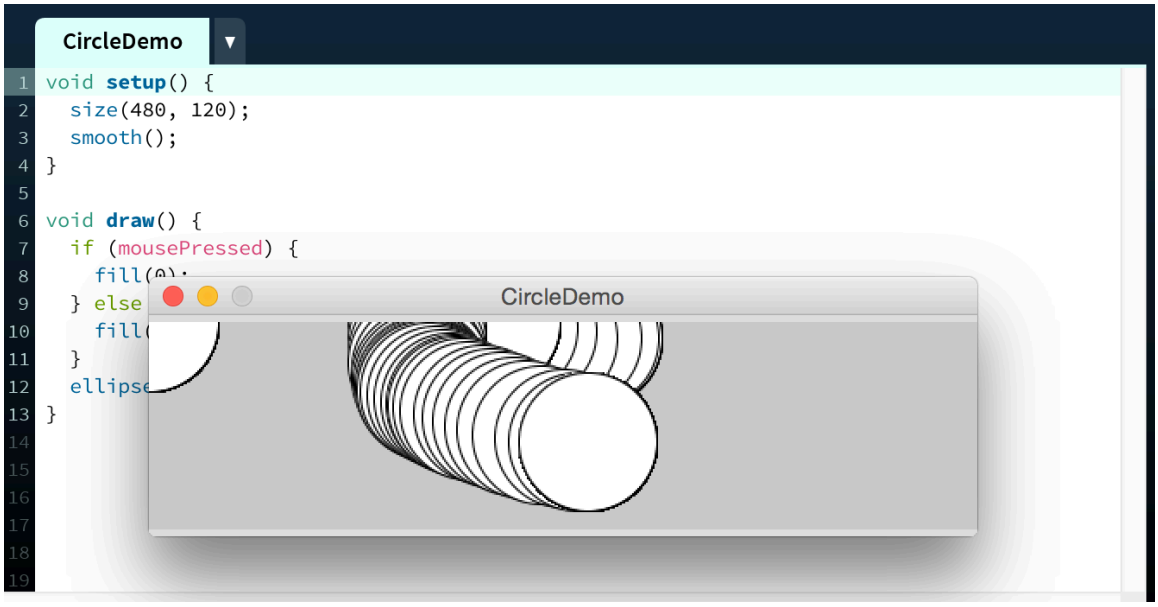
```
// Daniel Shiffman// Tracking the average location beyond a given depth threshold // Thanks to
Dan O&apos;Sullivan

// https://github.com/shiffman/OpenKinect-for-Processing // http://shiffman.net/p5/kinect/

class KinectTracker { // Depth threshold

int threshold = 745; // Raw location
```

```
PVector loc;// Interpolated location

PVector lerpedLoc; // Depth data

int[] depth;// What we&apos;ll show the user

PImage display; //Kinect2 class

Kinect2 kinect2; KinectTracker(PApplet pa) {

//enable Kinect2

kinect2 = new Kinect2(pa); kinect2.initDepth(); kinect2.initDevice();

kinect2.initDevice();

// Make a blank image

display = createImage(kinect2.depthWidth, kinect2.depthHeight, RGB);

// Set up the vectors

loc = new PVector(0, 0);

lerpedLoc = new PVector(0, 0); }

void track() {// Get the raw depth as array of integers depth = kinect2.getRawDepth();

// Being overly cautious here

if (depth == null) return;

float sumX = 0; float sumY = 0; float count = 0;

for (int x = 0; x < kinect2.depthWidth; x++) { for (int y = 0; y < kinect2.depthHeight; y++) {

// Mirroring the image

int offset = kinect2.depthWidth - x - 1 + y * kinect2.depthWidth; // Grabbing the raw depthint
rawDepth = depth[offset];

// Testing against threshold

if (rawDepth > 0 && rawDepth < threshold) { sumX += x;sumY += y;count++;

} }

}

// As long as we found something

if (count != 0) {loc = new PVector(sumX/count, sumY/count);
```

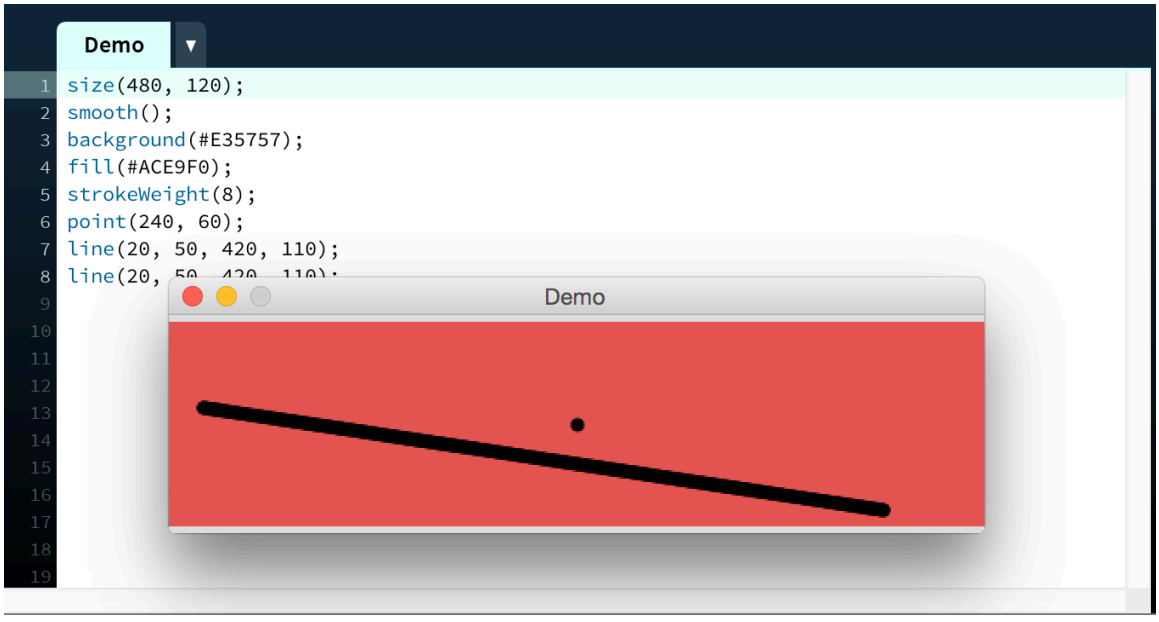
```
}

// Interpolating the location, doing it arbitrarily for now

lerpedLoc.x = PApplet.lerp(lerpedLoc.x, loc.x, 0.3f);

lerpedLoc.y = PApplet.lerp(lerpedLoc.y, loc.y, 0.3f); }

PVector getLerpedPos() { return lerpedLoc;

}

}

PVector getPos() { return loc;

}void display() {

PImage img = kinect2.getDepthImage(); // Being overly cautious here

if (depth == null || img == null) return;

// Going to rewrite the depth image to show which pixels are in threshold // A lot of this is
redundant, but this is just for demonstration purposes display.loadPixels();for (int x = 0; x <
kinect2.depthWidth; x++) {

for (int y = 0; y < kinect2.depthHeight; y++) {// mirroring imageint offset = (kinect2.depthWidth
- x - 1) + y * kinect2.depthWidth; // Raw depthint rawDepth = depth[offset];int pix = x +
y*display.width;if (rawDepth > 0 && rawDepth < threshold) {

// A red color instead

display.pixels[pix] = color(150, 50, 50); } else {

display.pixels[pix] = img.pixels[offset]; }

} }

display.updatePixels();

// Draw the image

image(display, 0, 0); }

int getThreshold() { return threshold;

}

void setThreshold(int t) { threshold = t;

} }
```

III. Artist Portfolio

CircleDemo: the first program I built, simply draws a circle where the cursor is.

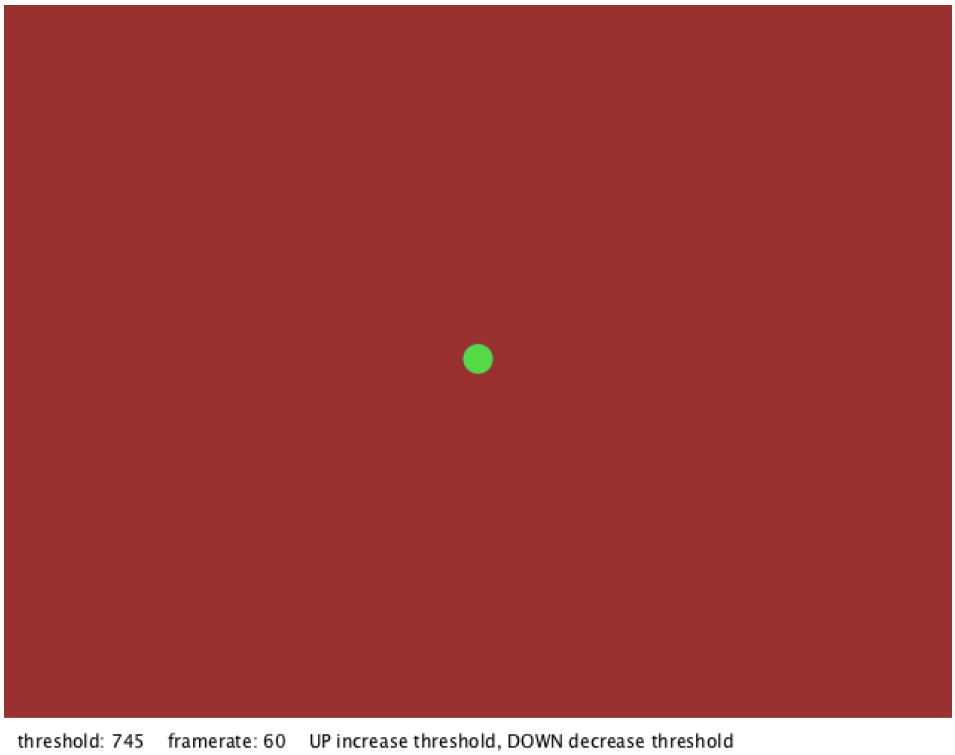


Demo: the second program I built, draws a dot where the cursor is and with a line below it. This is where I began experimenting with colors.



Drawing02: draws with a unique brush where the cursor is, later implemented in one of the first Kinect demos I made (not pictured)

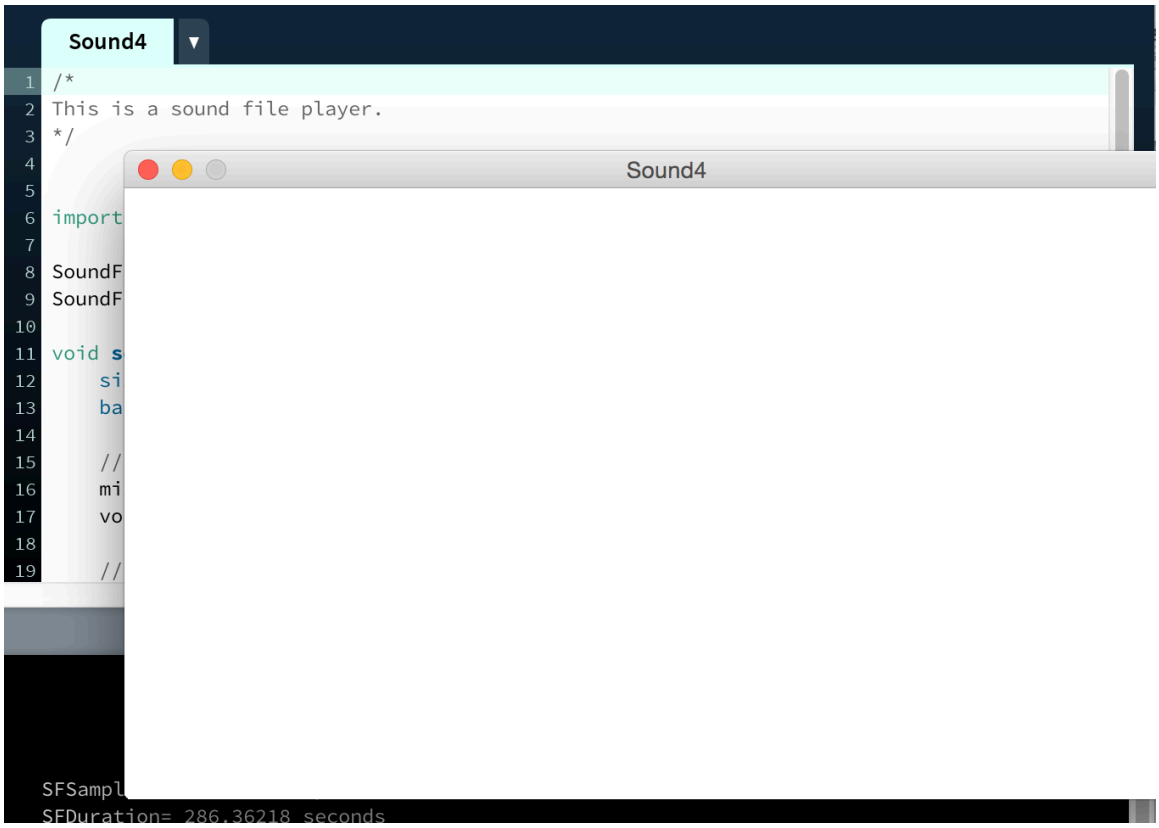
KinectTest: modified version of AveragePointTracking code to show more info and test different colors.



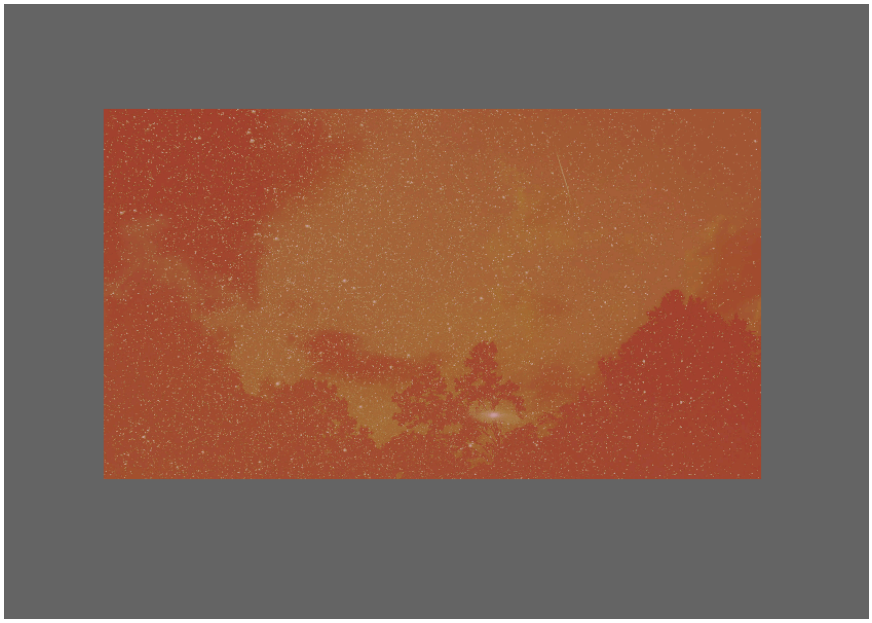
Hue2: used to experiment with assigning colors to an array and changing an image based on user input

Fall: uses color array to draw with a gradient. This is my first work with the concept of changing seasons.

Sound4: modified version of the Sample example of the Sound library, changes the volume level of two stereo files instead of the stereo effect of one mono file

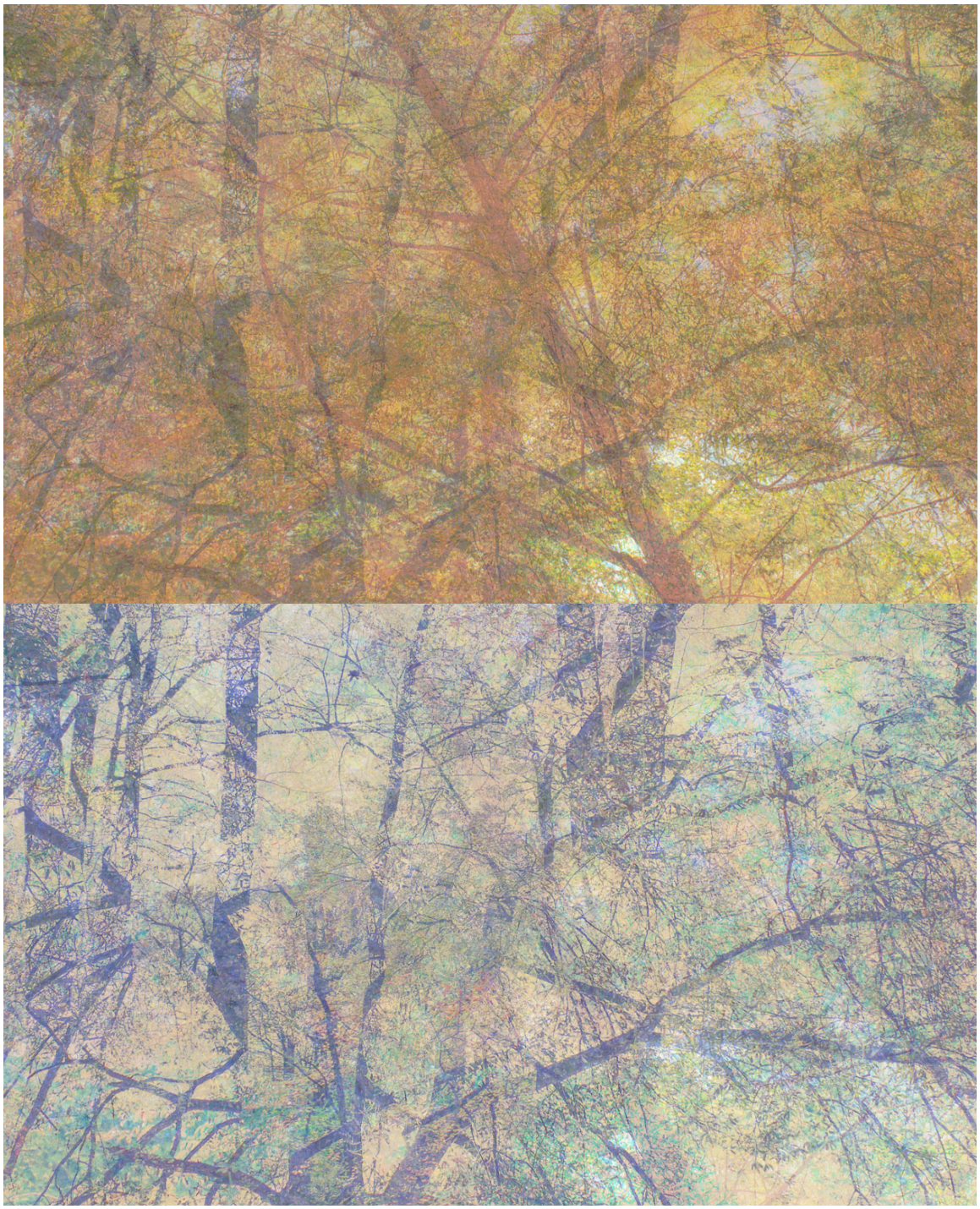


MoonageDaydream: Sound4 implemented with a David Bowie song and a changing image. (See artist statement for more info)

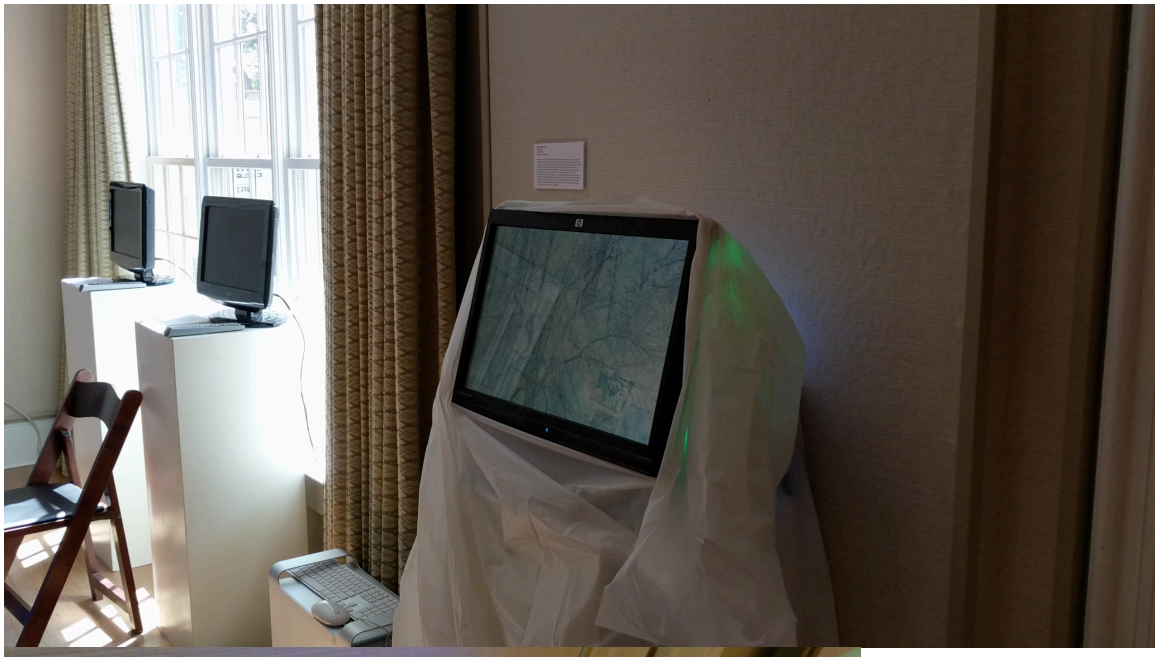


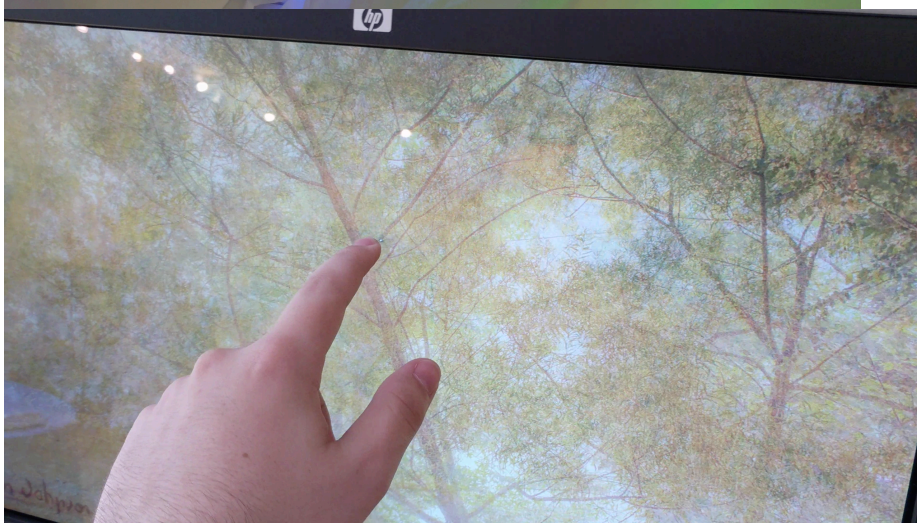
IV. Final Work

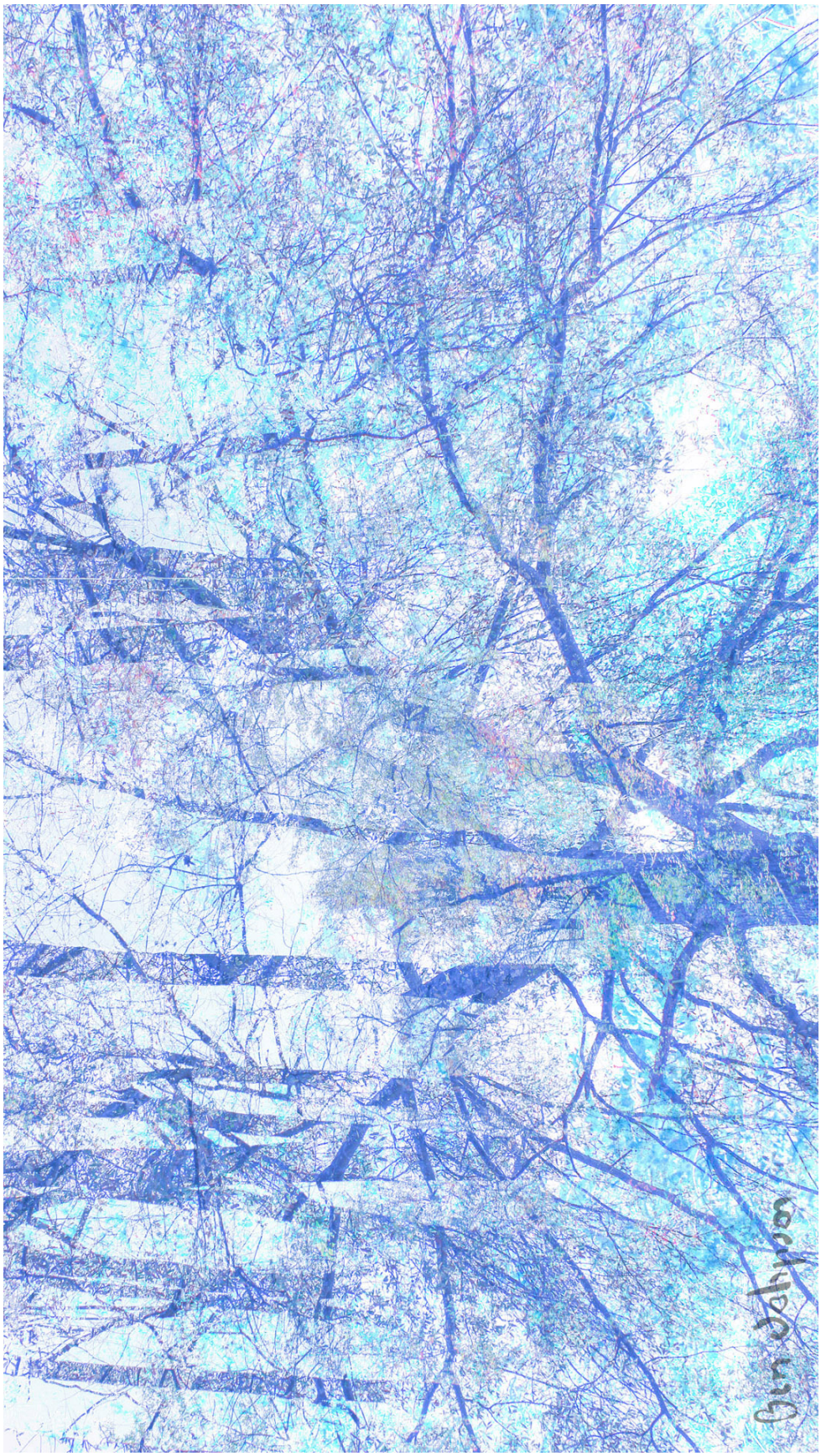
In action:



In gallery:







Source code:
TwoSeasonsTouchFinal.pde

```
/**Copyright 2017, Ben Johnson. Created using examples from Daniel Shiffman, Thomas Sanchez and The Processing Foundation **/

import processing.sound.*;import org.openkinect.processing.*;

PImage autumnImg; PImage winterImg;

SoundFile leftFile; SoundFile rightFile;

void setup() {size(1920,1080);//fullScreen(P2D, SPAN); background(0);autumnImg = loadImage("Autumn.jpg"); winterImg = loadImage("Winter.jpg");

//Load a soundfile

leftFile = new SoundFile(this, "left_mono.wav"); rightFile = new SoundFile(this, "right_mono.wav");

// These methods return useful infos about the file //println("SFSampleRate= " + soundfile.sampleRate() + " Hz"); //println("SFSamples= " + mixfile.frames() + " samples"); //println("SFDuration= " + mixfile.duration() + " seconds");

// Play the file in a loop

leftFile.loop(); rightFile.loop();

}

void draw() {// Map mouseX from 0.25 to 4.0 for playback rate. 1 equals original playback // speed 2 is an octave up 0.5 is an octave down. //soundfile.rate(map(mouseX, 0, width, 0.25, 4.0));

//if (millis() % 1800000 == 0) {// every 30 minutes, restart; 600000

//leftFile.stop();

// every 30 minutes, restart; 600000

//leftFile.stop(); //rightFile.stop(); //setup(); //draw();

//}

// Map mouseY from 0.2 to 1.0 for amplitude

leftFile.amp(map(mouseX, 0, height, 0.0, 1.0)); leftFile.pan(-1.0);rightFile.amp(map(mouseX, 0, height, 1.0, 0.0)); rightFile.pan(1.0);

image(autumnImg, 0, 0, width, height); tint(255, map(mouseX, 0, height, 255, 1)); image(winterImg, 0, 0, width, height);
```

```
// Map mouseY from -1.0 to 1.0 for left to right

//soundfile.pan(map(mouseY, 0, width, -1.0, 1.0));

}

void keyReleased(){if (key==&amp;apos; &amp;apos;) {

leftFile.stop(); rightFile.stop(); setup();

} }
```

IV. Annotated Bibliography

Harmonix. (2014). *Fantasia: Music Evolved* [Xbox 360]. Glendale, CA: Disney Interactive.

Fantasia: Music Evolved is what initially inspired me to use the Kinect in a musical way. While most of the songs on this game are contemporary, what caught my interest are the levels that use classical pieces (such as Tchaikovsky's *Nutcracker Suite*).

Levin, Golan. *Messa Di Voce*. 2003. Interactive video installation.

Levin, Golan. *Rectified Flowers*. 2010. Interactive video installation.

Researching Levin's work is where I found inspiration to use the Kinect as a tool for an interactive installation. In *Messa Di Voce*, the viewer can interact with the piece through sound. With *Rectified Flowers*, a still piece is changed with the interaction of a viewer.

Monet, Claude. *Water Lilies and the Japanese bridge*. 1899. Oil on canvas.

Monet, Claude. *San Giorgio Maggiore at Dusk*. 1908. Oil on canvas.

I was inspired by Monet's impressionist pieces for the visual element of my piece. I have always been interested in Monet's technique of overlaying colors and wanted to replicate that through digital photography. Using Photoshop, I overlaid eight landscape photographs in two

compositions (“Winter” and “Autumn.”) I also replicated Monet’s iconic signature with my own name in the bottom left corner of each piece.

Shiffman, Daniel, and Thomas Sanchez. *Open Kinect for Processing*. Code library. *Processing.org*. Vers. 1.0. The Processing Foundation. Web.

Shiffman, Daniel. *Sound Library for Processing*. Code library. *Processing.org*. Vers. 1.3.2. The Processing Foundation. Web.

I used Processing to produce my final program. The final code structure primarily implements these two libraries and parts of my source code are directly inspired by a few examples presented in each of these libraries.

Vivaldi, Antonio. *Four Seasons*. George Pollen, 2009. MIDI.

Vivaldi’s Four Seasons is the main piece of art that drove my project. Using a MIDI file of Vivaldi’s original composition that was transposed digitally in 2009 by George Pollen, I transcribed my own versions of two of the pieces in D minor and digitally stretched and added reverb to the two resulting files.