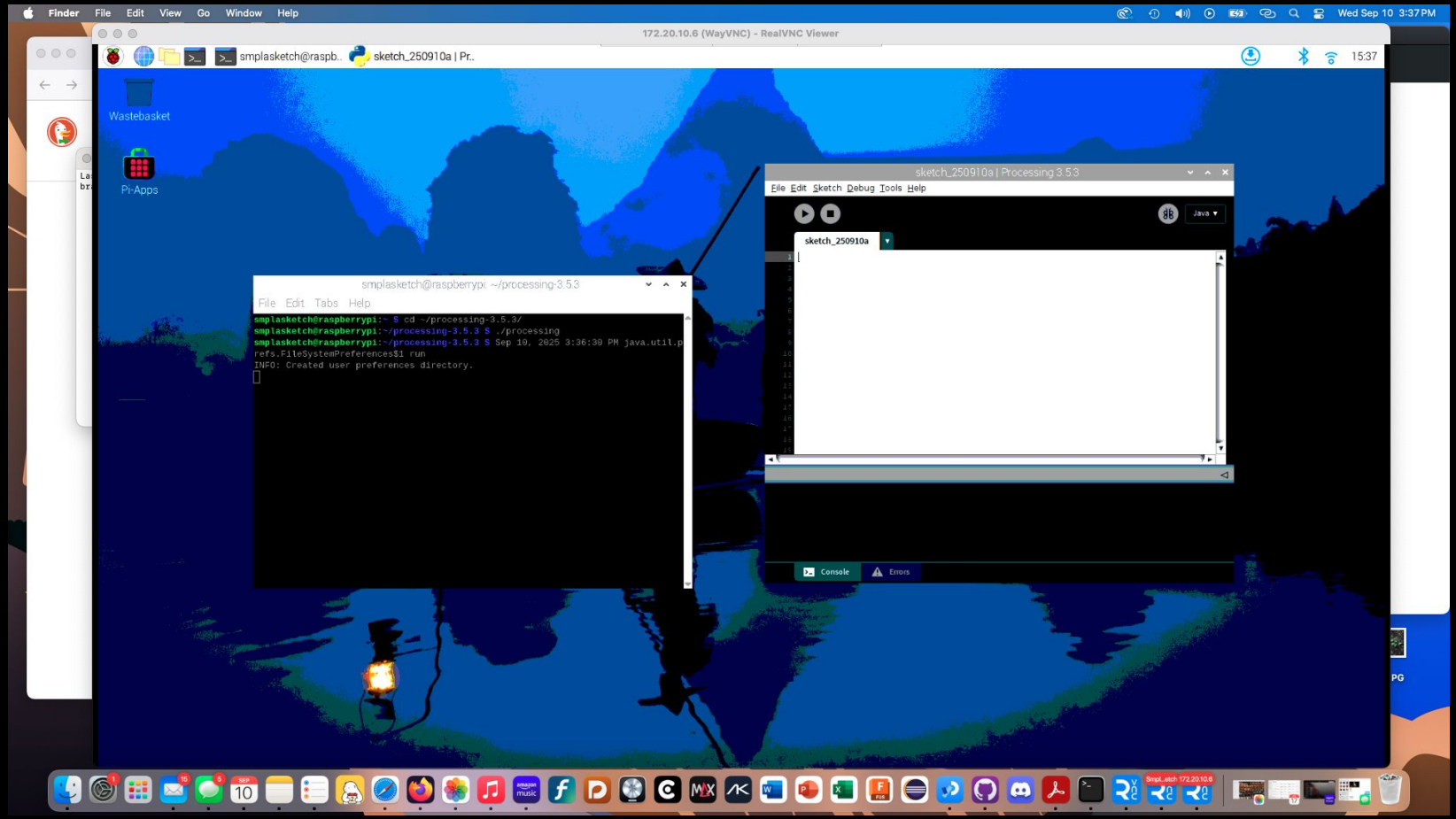
Week 2 Progress

Installed Processing On Pi - and wrote documentation on how to do it



Rewrote Image Processing Function

smplasketch | audioSampling | buttonControl | cameraCode | envGen | imageProcessing

```
PImage wavImg;
PImage envImg;
PImage modWavImg;
PImage modEnvImg;

int findAvgY(ArrayList<Integer> list) { // find average of y pixels from drawing
  int avg = 0;
  int size = list.size();
  for (int i = 0; i < size; i ++) {
    avg += list.get(i);
  }
  avg = avg/size;
  return avg;
}

float[] processImage(PImage img, PImage modImg, Boolean isWav) { //takes the image from the folder changes all red pixels to green and returns float array of drawn wave
  modImg.loadPixels(); //load display pixels
  img.loadPixels(); //load images pizels

  float[] wav = new float[width];

  for (int x = 0; x < width; x++) {
    ArrayList<Integer> readRedPix = new ArrayList<Integer>(); // create list to hold y values for a single x point
    for (int y = 0; y < height; y++) {
      int loc = x + y*width; //calculate pixel location

      // Pull out the 3 color components from pixel
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

      // Change red pixels to green to show that it was read
      if (r > 200 && b < 250 && g < 250) {
        r=0;
        b=0;
        g=255;
        readRedPix.add(y); //add the y value to the list
      }

      // Change displayed pixel to represent coordinates counted
      modImg.pixels[loc] =  color(r, g, b);
    }
    if (readRedPix.isEmpty()) {
      wav[x]=0;
    } else {
      int avg = findAvgY(readRedPix);
      float avgFl;
      if (isWav) {
        avgFl = map(avg, 0, height, 1, -1);//very confused why it's flipped but that's how the numbers work out
      } else {
        avgFl = map(avg, 0, height, 1000, 0);//very confused why it's flipped but that's how the numbers work out
      }
      wav[x] = avgFl;
    }
  }
```

Wrote Envelope Generation Code

smplasketch | audioSampling | buttonControl | cameraCode | envGen | imageProcessing

```
Env env = new Env(this);

float sustainLevel = 0.3; //keep the same

float envMult = 1;

float[] getTimeVals() { //translate envImage processing code into values needed to play the envelope back
  float[] timeVals = new float[3];
  float[] splits = processEnvImage();
  timeVals[0] = (splits[0]/100) * 0.45;
  timeVals[1] = ((splits[1]-splits[0])/100) * 0.45;
  timeVals[2] = ((100-splits[1])/100) * 0.45;
  return timeVals;
}

void playEnv(AudioSample sample) { //takes in a sample object and using the variables plays an envelope
  float[] time = getTimeVals();
  float attackTime = time[0]; //0.001;
  float sustainTime = time[1]; //0.004;
  float releaseTime = time[2]; //0.4;
  env.play(sample, attackTime * envMult, sustainTime * envMult, sustainLevel * envMult, releaseTime);
}
```

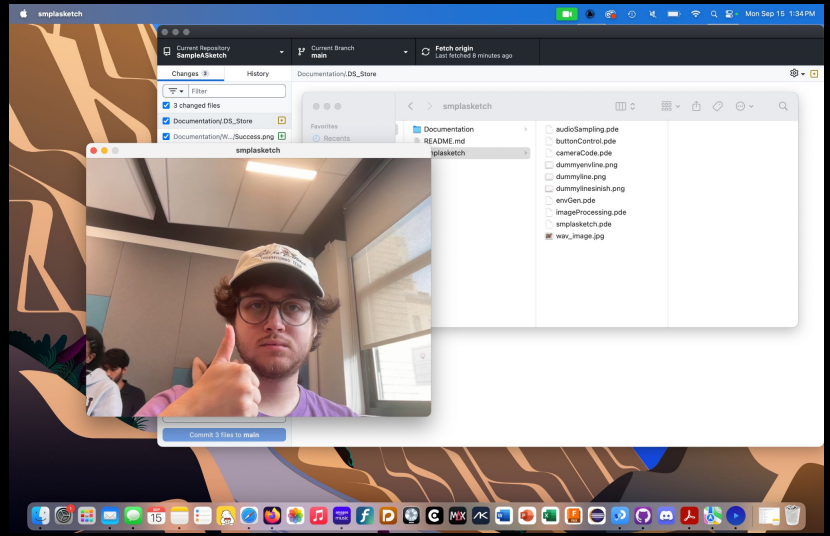
smplasketch | audioSampling | buttonControl | cameraCode | envGen | imageProcessing

```
boolean isOnStraight(float f1, float f2, float f3) { //returns a value is surrounded by two like values
  int i1 = Math.round(f1);
  int i2 = Math.round(f2);
  int i3 = Math.round(f3);
  if ((i1 == i2) && (i2 == i3)) {
    return true;
  } else {
    return false;
  }
}

float[] processEnvImage() { //feeds in env imgs to processing function and returns splits
  float[] splits = new float[2];
  float[] scan = processImage(envImg, modEnvImg, false);
  int counter = 1;
  while (!isOnStraight(scan[counter-1], scan[counter], scan[counter+1])) {
    counter++;
  }
  splits[0] = map(counter, 0, width, 0, 100);
  while (isOnStraight(scan[counter-1], scan[counter], scan[counter+1])) {
    counter++;
    if (counter == (scan.length - 1)) {
      break;
    }
  }
  splits[1] =  map(counter, 0, width, 100, 0);
  return splits;
}
```

Camera Connection/Capture Code

smplasketch | audioSampling | buttonControl | **cameraCode** | envGen | imageProcessing

```
import processing.video.*;

Capture cam;


void searchForCamera() {
  String[] cameras = Capture.list();

  if (cameras.length == 0) {
    println("There are no cameras available for capture.");
    exit();
  } else {
    println("Available cameras:");
    for (int i = 0; i < cameras.length; i++) {
      println(cameras[i]);
    }

    cam = new Capture(this, cameras[0]);//Needs to be changed for webcam/testing
    cam.start();
  }
}

void takePicture(String imageName) {
  delay(1000);
  if (cam.available() == true) {
    cam.read();
  }
  cam.save(imageName+".jpg");
  image(cam, 0, 0);
}
```



Controller Code

smplasketch | audioSampling | buttonControl | cameraCode | envGen | imageProcessing

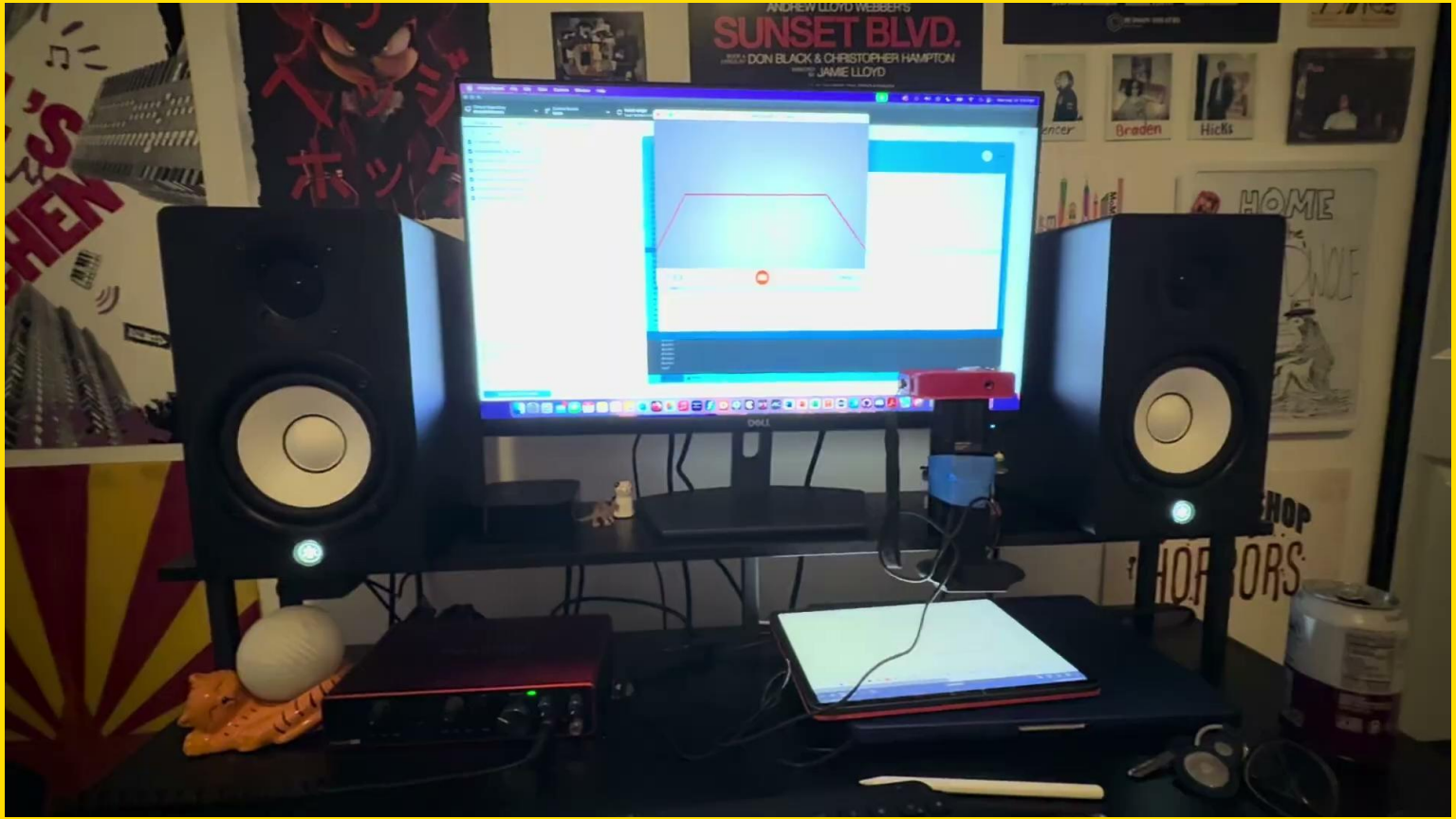
```
void wavSnap() {
  //light on
  //delay
  takePicture("wav_image");
  //light off
  wavImg = loadImage("wav_image.jpg");
}

void envSnap() {
  //light on
  //delay
  takePicture("env_image");
  //light off
  envImg = loadImage("env_image.jpg");
}

void incWavMult() {
  if (wavMult < 2.0) {
    wavMult += 0.1;
  }
}

void decWavMult() {
  if (wavMult > 0.5) {
    wavMult -= 0.1;
  }
}

void incEnvMult() {
  if (envMult < 2.0) {
    envMult += 0.1;
  }
}

void decEnvMult() {
  if (envMult > 0.5) {
    envMult -= 0.1;
  }
}

void reset() {
  wavImg = loadImage("dummylinesinish.png");
  envImg = loadImage("dummyenvline.png");
}

void smpl() {
  float[] wav = processWavImage();
  AudioSample smpl = createSample(wav);
  playWav(smpl);
  playEnv(smpl);
}
```

Reorganized Code/Added Keybind Simulation

smplasketch | audioSampling | buttonControl | cameraCode | envGen | imageProcessing

```
import java.util.Map;
import processing.sound.*;

void settings() { // using variables for size must be done in settings()
  int[] sizeArr = {640, 480}; //fix width to match pi cam(mac cam is 640 480)
  int width = sizeArr[0];
  int height = sizeArr[1];
  size(width, height);
}

void setup() {
  modWavImg = createImage(width, height, RGB); // Create image to write other data to
  modEnvImg = createImage(width, height, RGB); // Create env image to write other data to
  wavImg = loadImage("dummylinesinish.png"); // Load janky image from folder "dummyline.png" | Load sinishimage from folder "dummylinesinish.png"
  envImg = loadImage("dummyenvline.png");
  searchForCamera();
  //noLoop();//turns off infinite loop of draw()
}

void draw() {
//keypressed code will go here for buttons
}

void keyPressed() {
  if (key == 'w') {
    wavSnap();
    println("wavSnap");
  } else if (key == 'e') {
    envSnap();
    println("envSnap");
  } else if (key == '[') {
    decEnvMult();
    println("decEnv");
  } else if (key == ']') {
    incEnvMult();
    println("incEnv");
  } else if (key == '-') {
    decWavMult();
    println("decWav");
  } else if (key == '+') {
    incWavMult();
    println("incWav");
  } else if (key == 's') {
    smpl();
    println("smpl");
  } else if (key == 'r') {
    reset();
    println("reset");
  }
}
```

Demo


SUNSET BLVD.
DON BLACK & CHRISTOPHER HAMPTON
JAMIE LLOYD
Braden
Hicks
HOME