P#9
Finger Danso

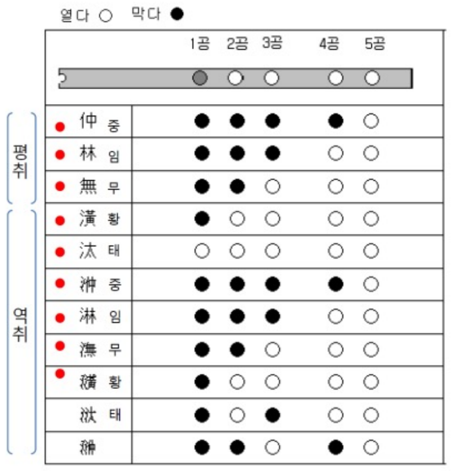
by Lesson 9: Machine Learning - Regression

20201127 이창현

Idea

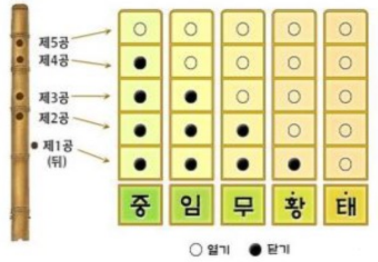
- 5 melody of Korean traditional Musical scales[중임무황태]
- Danso[단소] : korean traditional Instrument
- 5 Fingers of Hand

!! Easy to blow Danso -> Danso played with fingers !!



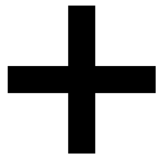
열다 ○ 막다 ●

			1공	2공	3공	4공	5공
평취	仲	중	●	●	●	●	○
	林	임	●	●	●	○	○
	無	무	●	●	○	○	○
역취	潢	황	●	○	○	○	○
	汰	태	○	○	○	○	○
	㳞	중	●	●	●	●	○
	淋	임	●	●	●	○	○
	潕	무	●	●	○	○	○
	潢	황	●	○	○	○	○
	汰	태	●	○	●	○	○
	㳞		●	●	○	●	○



	중	임	무	황	태
제5공	○	○	○	○	○
제4공	●	○	○	○	○
제3공	●	●	○	○	○
제2공	●	●	●	○	○
제1공 (뒤)	●	●	●	●	○

○ 열기 ● 닫기

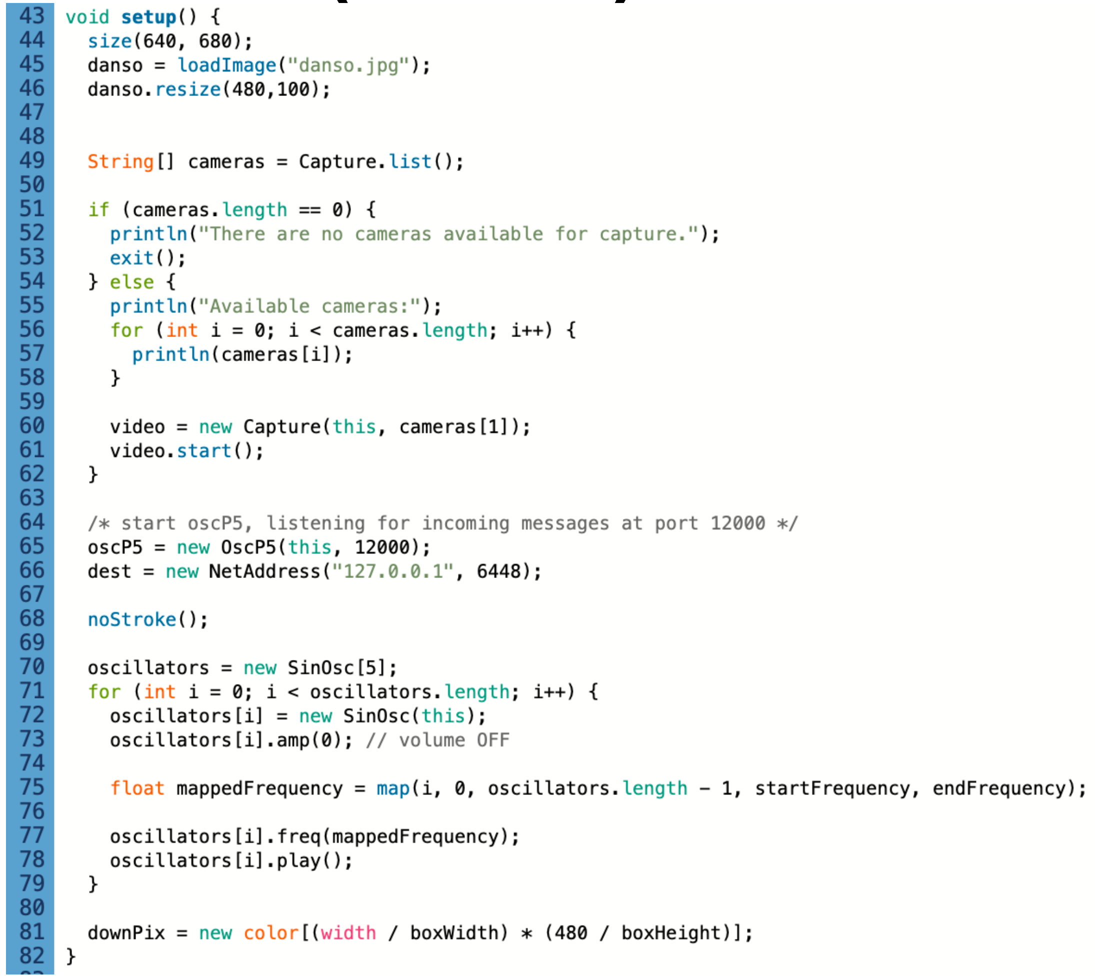


source : https://blog.naver.com/lesson-amy/220774284172

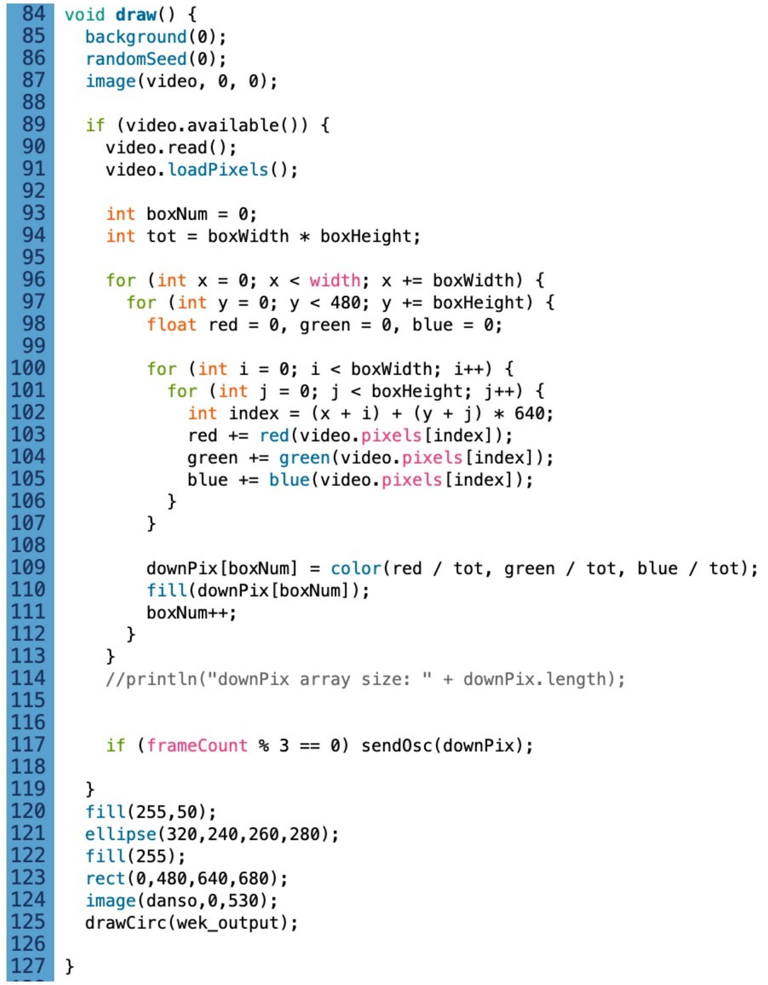
Process

- Define [SinOscillator] class for managing sound oscillations.
- Set up an array of SinOscillators with diverse frequencies and volume controls.
- Extract pixel data from camera video for Wekinator input.
- Dynamically modify sound frequencies based on Wekinator output.
- Visualize video, display a transparent guide circle at the user's hand position, and show Wekinator-based text.

Code(main)

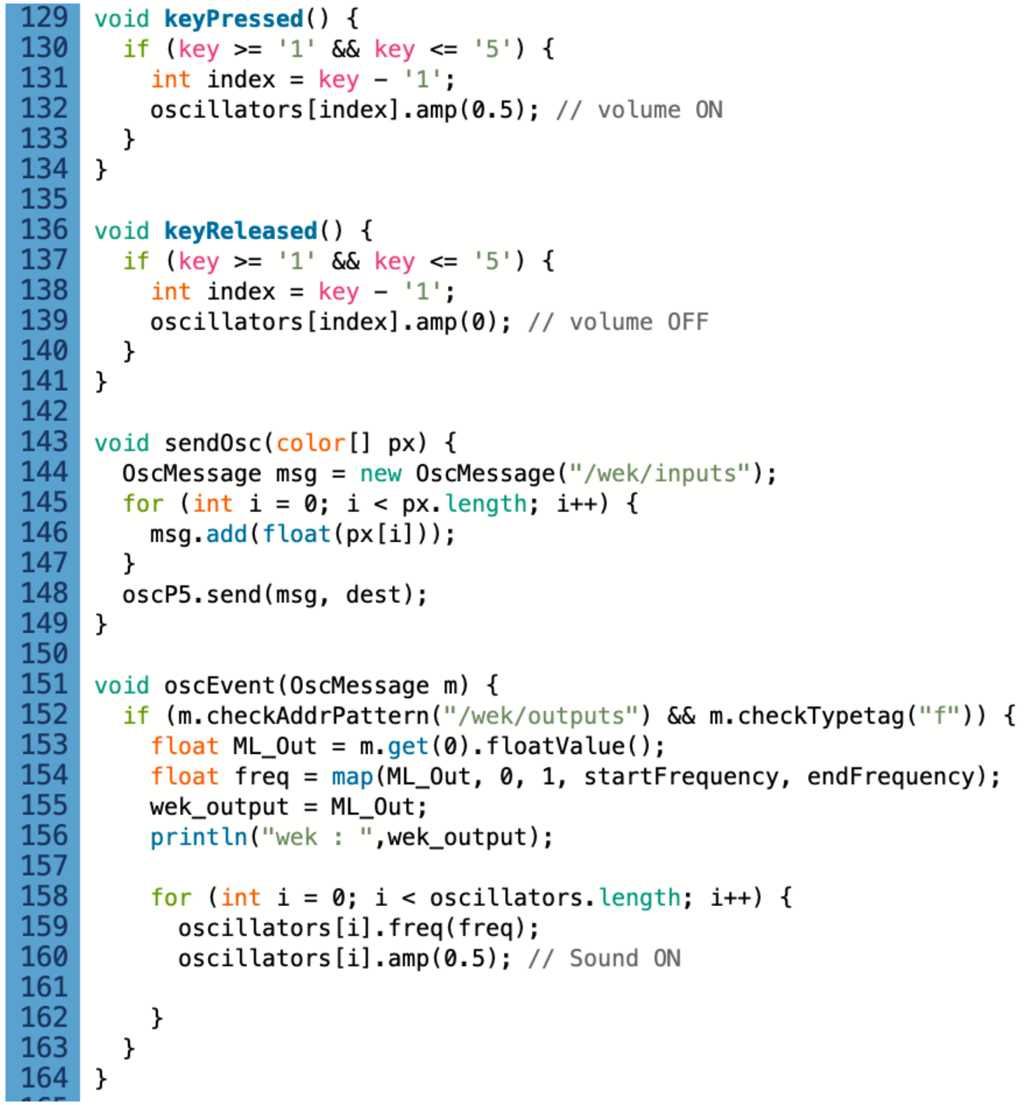


```
void setup() {
  size(640, 680);
  danso = loadImage("danso.jpg");
  danso.resize(480,100);


  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]);
    }

    video = new Capture(this, cameras[1]);
    video.start();
  }

  /* start oscP5, listening for incoming messages at port 12000 */
  oscP5 = new OscP5(this, 12000);
  dest = new NetAddress("127.0.0.1", 6448);

  noStroke();

  oscillators = new SinOsc[5];
  for (int i = 0; i < oscillators.length; i++) {
    oscillators[i] = new SinOsc(this);
    oscillators[i].amp(0); // volume OFF

    float mappedFrequency = map(i, 0, oscillators.length - 1, startFrequency, endFrequency);

    oscillators[i].freq(mappedFrequency);
    oscillators[i].play();
  }

  downPix = new color[(width / boxWidth) * (480 / boxHeight)];
}
```

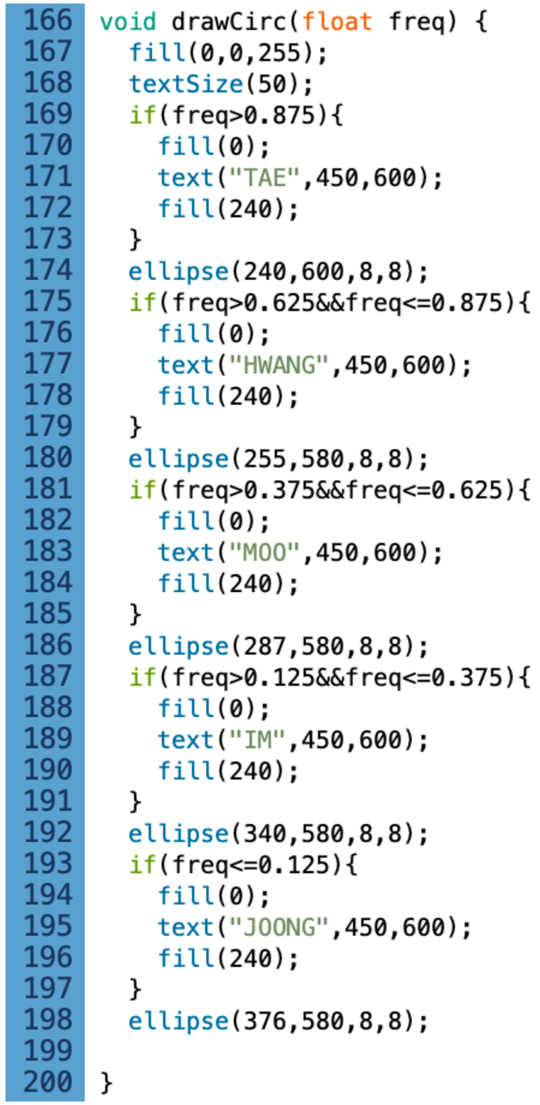


```
void draw() {
  background(0);
  randomSeed(0);
  image(video, 0, 0);

  if (video.available()) {
    video.read();
    video.loadPixels();

    int boxNum = 0;
    int tot = boxWidth * boxHeight;

    for (int x = 0; x < width; x += boxWidth) {
      for (int y = 0; y < 480; y += boxHeight) {
        float red = 0, green = 0, blue = 0;

        for (int i = 0; i < boxWidth; i++) {
          for (int j = 0; j < boxHeight; j++) {
            int index = (x + i) + (y + j) * 640;
            red += red(video.pixels[index]);
            green += green(video.pixels[index]);
            blue += blue(video.pixels[index]);
          }
        }

        downPix[boxNum] = color(red / tot, green / tot, blue / tot);
        fill(downPix[boxNum]);
        boxNum++;
      }
    }
    //println("downPix array size: " + downPix.length);


    if (frameCount % 3 == 0) sendOsc(downPix);

  }
  fill(255,50);
  ellipse(320,240,260,280);
  fill(255);
  rect(0,480,640,680);
  image(danso,0,530);
  drawCirc(wek_output);

}
```

Code(main)

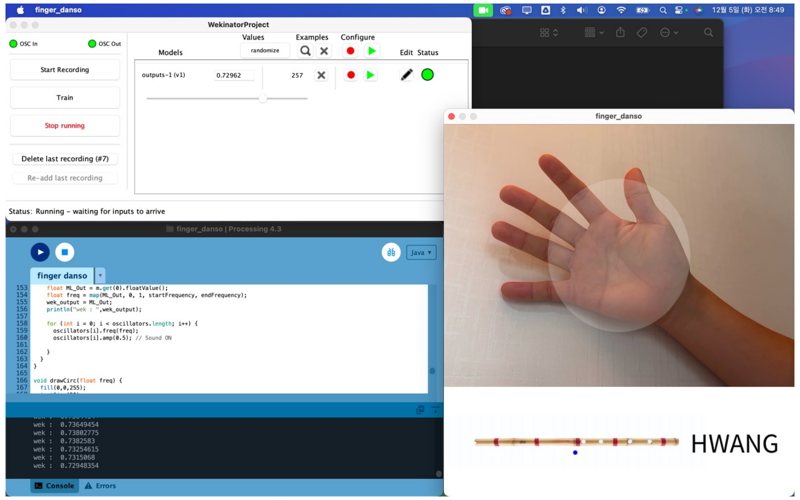


```
void keyPressed() {
  if (key >= '1' && key <= '5') {
    int index = key - '1';
    oscillators[index].amp(0.5); // volume ON
  }
}

void keyReleased() {
  if (key >= '1' && key <= '5') {
    int index = key - '1';
    oscillators[index].amp(0); // volume OFF
  }
}

void sendOsc(color[] px) {
  OscMessage msg = new OscMessage("/wek/inputs");
  for (int i = 0; i < px.length; i++) {
    msg.add(float(px[i]));
  }
  oscP5.send(msg, dest);
}

void oscEvent(OscMessage m) {
  if (m.checkAddrPattern("/wek/outputs") && m.checkTypetag("f")) {
    float ML_Out = m.get(0).floatValue();
    float freq = map(ML_Out, 0, 1, startFrequency, endFrequency);
    wek_output = ML_Out;
    println("wek : ",wek_output);

    for (int i = 0; i < oscillators.length; i++) {
      oscillators[i].freq(freq);
      oscillators[i].amp(0.5); // Sound ON

    }
  }
}
```



```
void drawCirc(float freq) {
  fill(0,0,255);
  textSize(50);
  if(freq>0.875){
    fill(0);
    text("TAE",450,600);
    fill(240);
  }
  ellipse(240,600,8,8);
  if(freq>0.625&&freq<=0.875){
    fill(0);
    text("HWANG",450,600);
    fill(240);
  }
  ellipse(255,580,8,8);
  if(freq>0.375&&freq<=0.625){
    fill(0);
    text("MOO",450,600);
    fill(240);
  }
  ellipse(287,580,8,8);
  if(freq>0.125&&freq<=0.375){
    fill(0);
    text("IM",450,600);
    fill(240);
  }
  ellipse(340,580,8,8);
  if(freq<=0.125){
    fill(0);
    text("JOONG",450,600);
    fill(240);
  }
  ellipse(376,580,8,8);

}
```

Screenshot

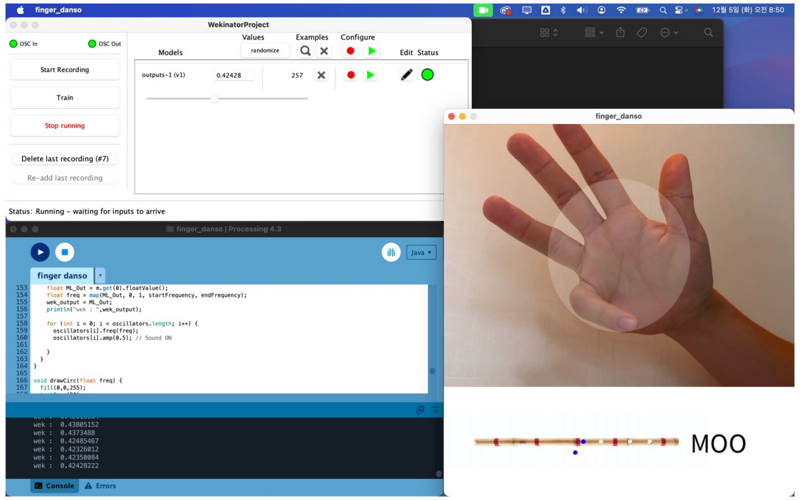


outputs-1 (v1) 0.72962 257

Status: Running - waiting for inputs to arrive

```
    float ML_Out = m.get(0).floatValue();
    float freq = map(ML_Out, 0, 1, startFrequency, endFrequency);
    wek_output = ML_Out;
    println("wek : ",wek_output);

    for (int i = 0; i < oscillators.length; i++) {
      oscillators[i].freq(freq);
      oscillators[i].amp(0.5); // Sound ON

    }
  }
}

void drawCirc(float freq) {
  fill(0,0,255);
```

```
wek :  0.73649454
wek :  0.73802775
wek :  0.7382583
wek :  0.73254615
wek :  0.7315068
wek :  0.72948354
```

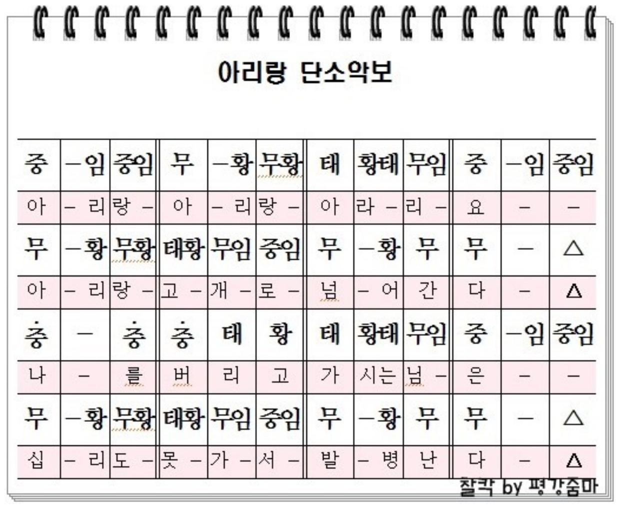


outputs-1 (v1) 0.42428 257

Status: Running - waiting for inputs to arrive

```
    float ML_Out = m.get(0).floatValue();
    float freq = map(ML_Out, 0, 1, startFrequency, endFrequency);
    wek_output = ML_Out;
    println("wek : ",wek_output);

    for (int i = 0; i < oscillators.length; i++) {
      oscillators[i].freq(freq);
      oscillators[i].amp(0.5); // Sound ON

    }
  }
}

void drawCirc(float freq) {
  fill(0,0,255);
```

```
wek :  0.43805152
wek :  0.4373488
wek :  0.42485467
wek :  0.42326012
wek :  0.42350084
wek :  0.42428222
```



아리랑 단소악보

중	–임	중임	무	–황	무황	태	황태	무임	중	–임	중임
아	– 리	랑 –	아	– 리	랑 –	아	라 –	리 –	요	–	–
무	–황	무황	태황	무임	중임	무	–황	무	무	–	△
아	– 리	랑 –	고 –	개 –	로 –	넘	– 어	간	다	–	△
충	–	충	충	태	황	태	황태	무임	중	–임	중임
나	–	를	버	리	고	가	시는	님 –	은	–	–
무	–황	무황	태황	무임	중임	무	–황	무	무	–	△
십	– 리	도 –	못 –	가 –	서 –	발	– 병	난	다	–	△

찰칵 by 평강줌마

Let's play [Arirang] using Finger Danso