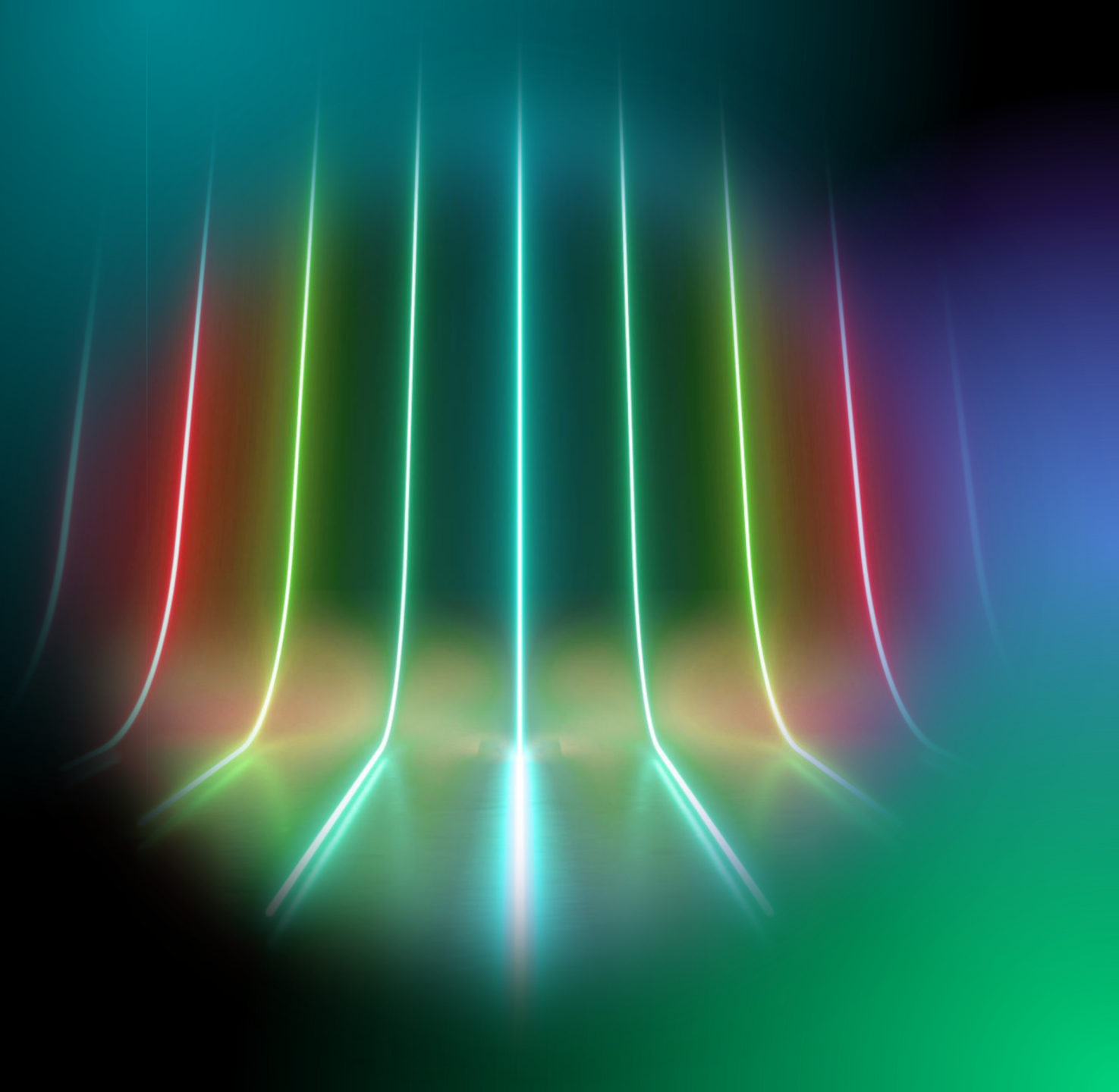
Arduino Processing - Integration Project

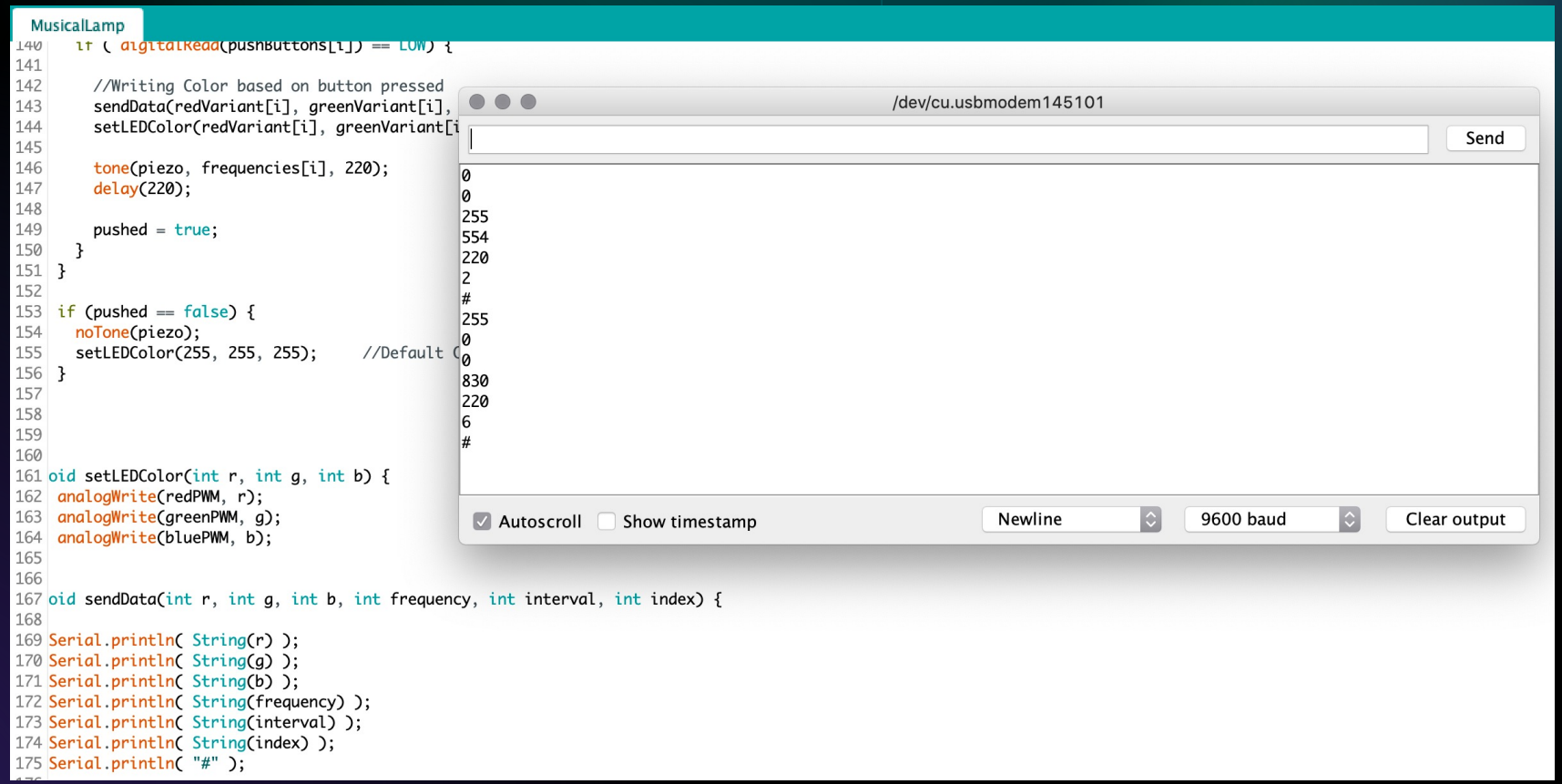
Musical Lamp

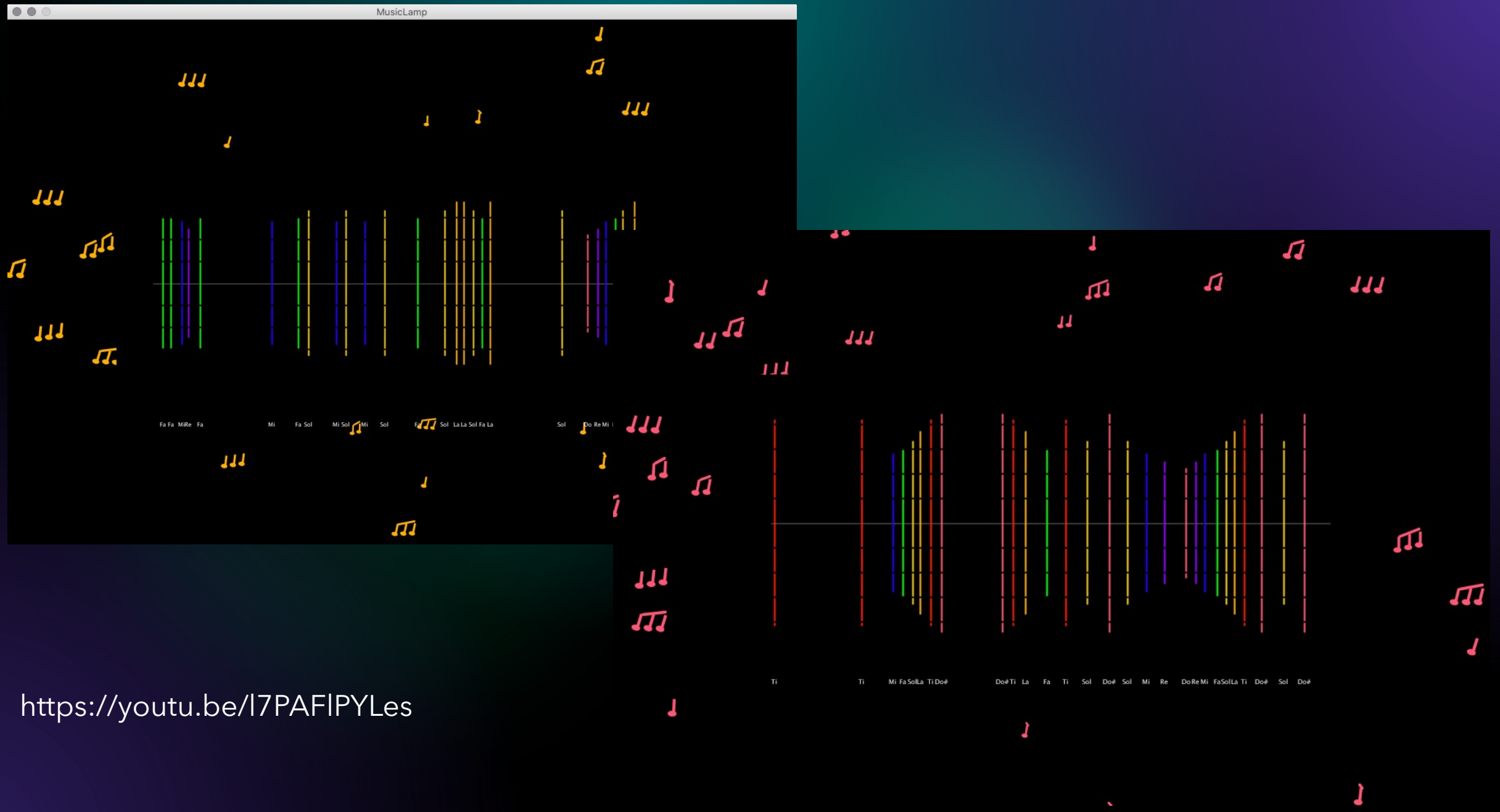
NARENDRA KONATHALA



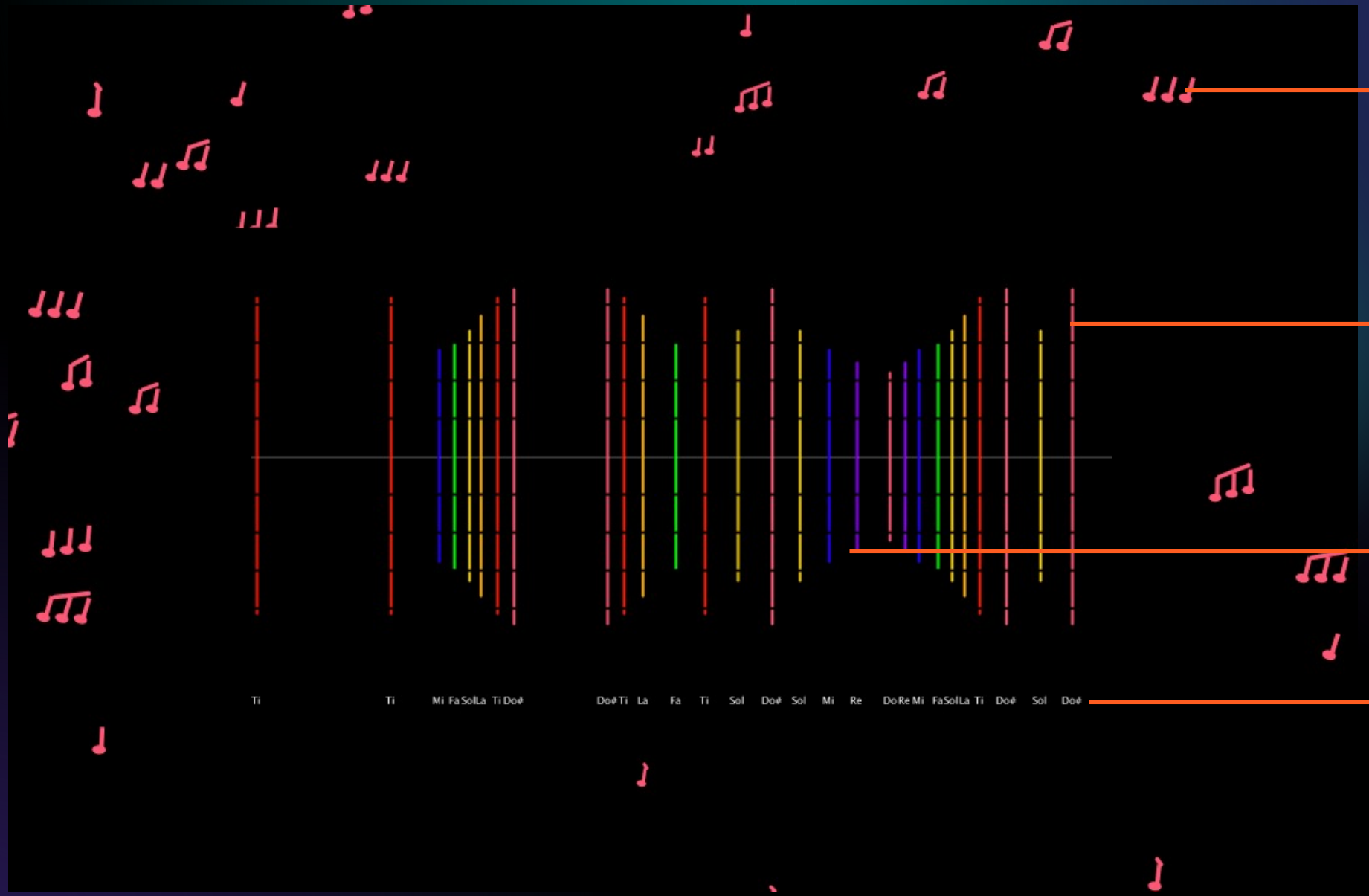
Serilog Data Sent

Every music note is sent with a set of values delimited by # as shown below.
Each record has R, G, B values, Frequency, Interval and index of musical note sent to processing Serial Event (Do, Re, Mi Fa, Sol, La, Ti, Do#).





https://youtu.be/l7PAFlPYLes



Musical Symbols created using basic shapes.

Its color changes based on the lamp light

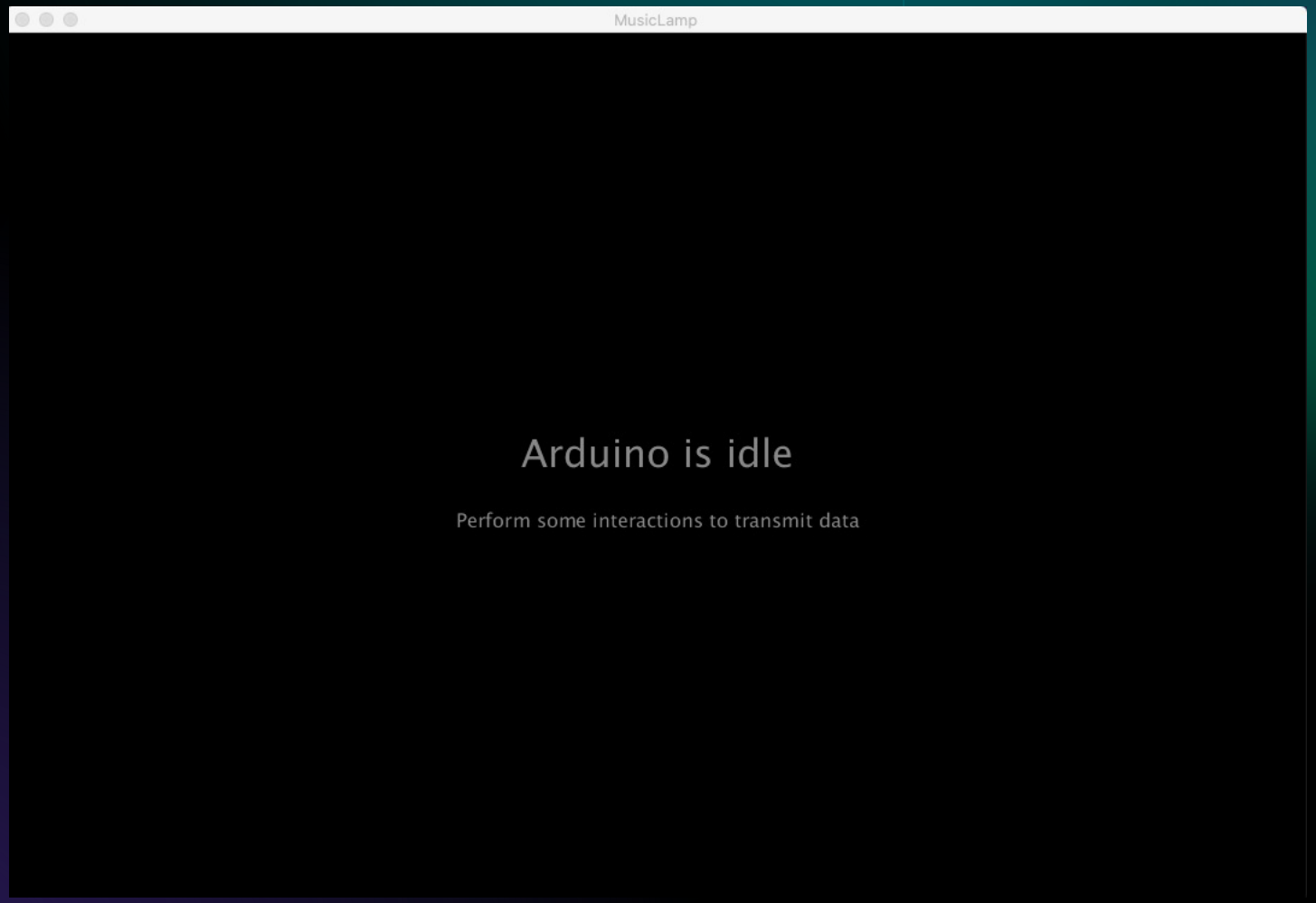
Magnitude of each line represent the Frequency (Hz) of sound

Its color changes based on the lamp light

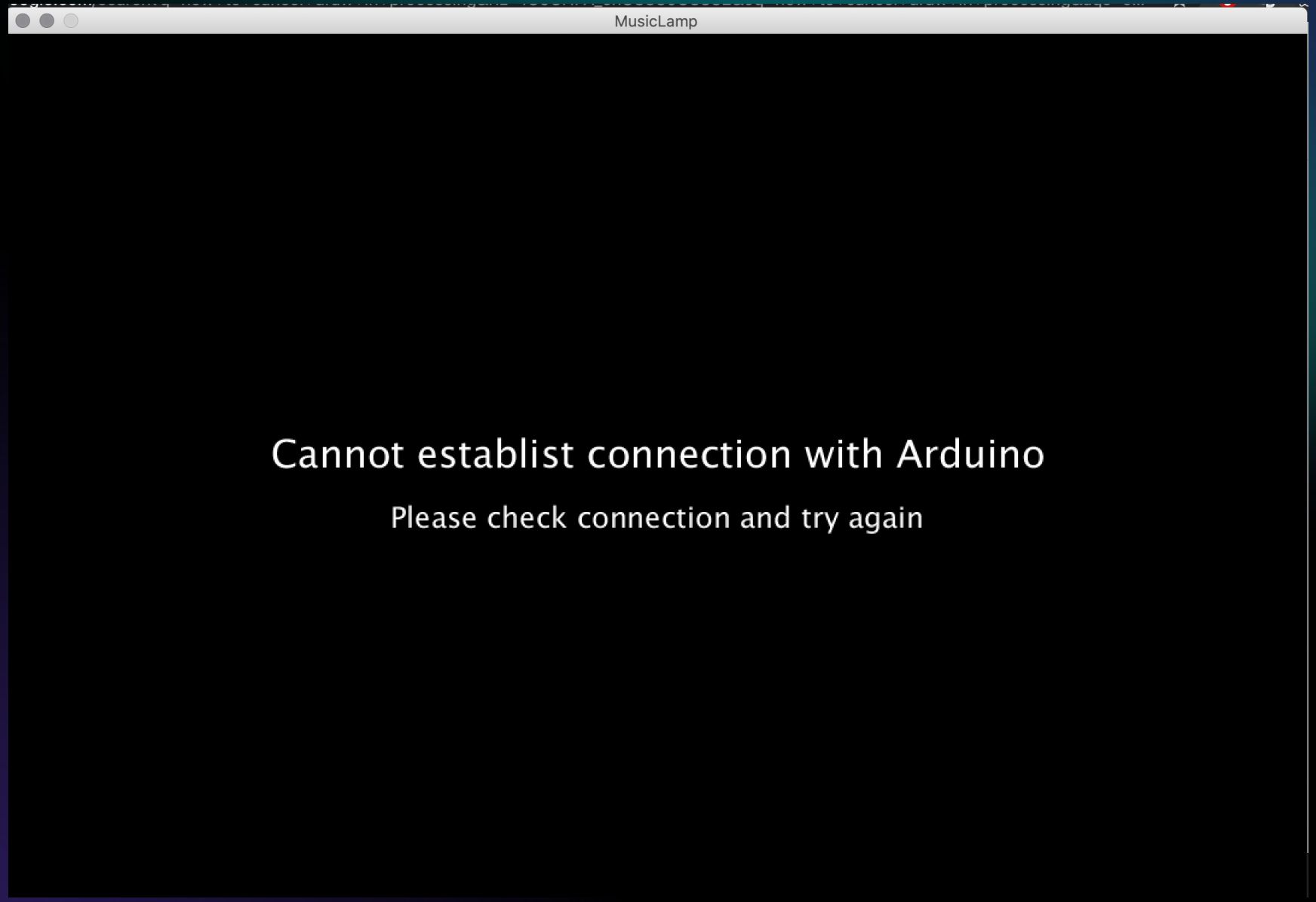
Time based graph , kind of sound wave. Gaps in between shows pause in music.

Music note value is written in the bottom for correlation.

Empty State is displayed when no data is transmitted for idle time of 10 secs.



Error Handling, when no port is available or Arduino not responding.



```
import processing.serial.*;
import java.util.LinkedList;
import java.util.Queue;
import java.time.*;

Serial arduinoPort;
final String portStart = "/dev/cu.usbmodem";

Queue<NoteData> queue = new LinkedList();   // Used to store the note frequncies using 1st in 1st out method


final int numOfNotes = 50;
Note[] notes = new Note[numOfNotes];
NoteData activeNode = null;                 //Used to change the background color
int offset = 200;                             //Used to centered the frequencies of notes
ZonedDateTime pastEventTime;                //To calculate idle time of Arduino
String keyNote[] = {"Do", "Re", "Mi", "Fa", "Sol", "La", "Ti", "Do#"};
boolean badPort = false;

void setup() {
  //fullScreen();
  size(1080, 720);
  background(0);

  arduinoPort = getPort();

  if (arduinoPort == null) {                //Error handling - No Port found
      translate(width/2, height/2);
      textAlign(CENTER);
      textSize(32);
      text("Cannot establist connection with Arduino", 0, 0);
      textSize(24);
      text("Please check connection and try again", 0, 50);
      noLoop();
      badPort = true;
      return;
  }

  for (int i = offset; i < width-offset; i++) {
    queue.add(new NoteData());              //Initiate an empty queue for the length of the graph
  }

  for (int i = 0; i < numOfNotes; i++) {    //Notes - Reused for performance
    Note note = new Note();
    notes[i] = note;
  }

}


void draw() {
  if (badPort == true) { return ;}
  background(0);
```

```
54
55   if (pastEventTime == null || isArduinoIdle(pastEventTime)) {
56     pushMatrix();
57     fill(125);
58     translate(width/2, height/2);
59     textAlign(CENTER);
60     textSize(32);
61     text("Arduino is idle", 0, 0);        //Notify user the idle case - when
   no data is transmitted
62     textSize(16);
63     text("Perform some interactions to transmit data", 0, 50);
64     popMatrix();
65     return;
66   }
67
68   readSerialData(arduinoPort);             //Reads data to a queue
69
70   drawBackground();
71   drawFrequencyGraph();
72   drawAxis();
73 }
74
75
76
77 void drawFrequencyGraph() {               //Draw frequencies with color codes
78   int i = offset;                         //starts from offset to the length
   of the queue
79
80   for (NoteData element : queue) {
81     pushMatrix();
82     translate(0, height/2);
83
84     if (element.frequency == 0) {
85       strokeWeight(1);
86       stroke(element.noteColor, 50);
87       line(i, 0, i, 1);                   //Flat Line when no sound is made
88     } else {
89       float val = map(element.frequency, 0, 1200, 0, height / 4);  //Map Hz
   to Height
90       strokeWeight(2);
91       stroke(element.noteColor);
92       line(i, val, i, -1 * val);
93
94       fill(255);                          //In addition to color code display
   the note on
95       textAlign(CENTER);                  //the bottom for better data
   representation
96       textSize(8);
97       text( keyNote[element.index] , i, height / 4 + 16);
98     }
99     popMatrix();
100    i++;
101  }
102 }
103
104 void drawBackground() {
105   for (int j = 0; j < numOfNotes; j++) {
106     if (activeNode == null) {
107       notes[j].draw(-1, -1);
```

```
    }else {                             //Highlight the note which is active in the background
      notes[j].draw(activeNode.index, activeNode.noteColor);
    }
  }

  pushMatrix();                         //Block middle area for Graph
  translate(0, height/2);
  noStroke();
  fill(0);
  rect(offset - 50, -1 * height / 4, width - 2 * (offset - 50), height / 2);
  popMatrix();
}

void drawAxis() {                       //Frequencies - Y Axis
    for (int i = -1200 ; i <= 1200; i = i + 200) {
      if (i == 0) continue;
      pushMatrix();
      textAlign(CENTER);
      textSize(8);
      translate(0 ,height/2);
      stroke(0);
      strokeWeight(2);
      int y = (int) map(i, -1200, 1200, -1 * height / 4, height / 4);
      line(offset, y, width - offset, y);
      popMatrix();
    }
}

void serialEvent(Serial p) {            //Used for calculating idle time of Arduino
  pastEventTime = ZonedDateTime.now();
}

boolean isArduinoIdle(ZonedDateTime oldTime) {
  Duration duration = Duration.between(oldTime, ZonedDateTime.now());
  if (duration.getSeconds() > 10) {    //Idle timeout set to 10 sec...
    return true;                        //Returns idle if no data is transmitted for 10Secs
  }
  return false;
}

void readSerialData(Serial port){      //Function to read each set of Serial data trasmitted from Arduino

  String inBuffer = port.readStringUntil('#');
  NoteData data = null;
    if (inBuffer != null) {
      data = new NoteData(inBuffer.trim());
      activeNode = data;
    } else {
      data = new NoteData();          //No sound
    }
    queue.poll();                       //Removes the 1st element from queue
    queue.add(data);                    //Add new data to the end.
}


```

```
Serial getPort() {                    //Gets the Arduino port and creates a listener
  String[] portList = Serial.list();
  for (int i = 0; i < portList.length; i++) {
    if (portList[i].startsWith(portStart)) {
      return new Serial(this, portList[i], 9600);
    }
  }
  return null;
}

class NoteData {                      //Data structure to hold each note data recieved from Arduino
  public color noteColor;
  public int frequency;
  public int interval;
  public int index;

  public NoteData() {                //Empty Note -  default constructor
    noteColor = color(255);          //When no music or data trasmitted
  }

  public NoteData(String rawText) { //override constructor
    try {
      String arr[] = rawText.split("\\n");
      this.noteColor = color(Integer.parseInt(arr[0].trim()), Integer.parseInt(arr[1].trim()), Integer.parseInt(arr[2].trim()));
      this.frequency = Integer.parseInt(arr[3].trim());
      this.interval = Integer.parseInt(arr[4].trim());
      this.index = Integer.parseInt(arr[5].trim());
    } catch (Exception e) {
      println(e);
    }
  }
}

public class Note {                   //Data to represent background
    public float x;
    public float y;
    int size = 0;
    float distance;
    float angle = 0;
    final float boundRadius = width /2;
    int rindex = (int) (Math.random() * 8);
    float inclination = (float) Math.random() * TWO_PI / 20;
    float scale = 0.1;

    int type = (int) (Math.random() * 2 + 1);
    int type1 = (int) (Math.random() * 3 + 1);

    public Note() {                   //Random point on the canvas
        angle = (float) Math.random() * TWO_PI;
        distance = (float) Math.sqrt(Math.random()) * boundRadius + 200;
        setX(); setY();
    }

    private void setX() {
        x =  (float) (distance * cos( (float) angle));
    }
```

```
219
220     private void setY() {
221         y = (float) (distance * sin( (float) angle));
222     }
223
224     private void move() {            //For 3D feel
225         distance = distance + 2;
226         size++;
227         if (distance > boundRadius + 200) {
228           distance = (float) Math.sqrt(Math.random()) * boundRadius+ 200;
229           scale = 0.1;
230         }
231         size = (int) ((distance) * 0.04);
232         scale = distance / 1000;
233
234         if ( size < 4 ) { size = 4 ;};
235         setX(); setY();
236     }
237
238     void draw(int index, color noteColor) {
239         move();                                          //Move the object
towards the circumference
240         pushMatrix();                                    //This function
draws random notes.
241         fill(noteColor);
242         translate(width/2, height/2);
243         scale(scale);
244         rotate(inclination);
245         noStroke();
246
247         for (int i = 0; i < type1; i++) {
248           rect(x + 20 * i, y - i * 5, 5, 25);               //# string
249           ellipse(x + 20 * i, y - i * 5 + 25,  15, 10);
250
251           if (type > 1 && i == type1 -1) {
252             strokeWeight(5);
253             stroke(noteColor);
254             line(x  + 5, y, x + 20 * (i), y  - 5 * (i +1));  //With Bar
255           }
256         }
257         popMatrix();
258     }
259
260
261
262 }
```