Cooper Rapp

CS321 Music Programming 1

23 December 2021

Project Description Document

The interface design of my application is split into two distinct sections. In the left section of the application is the mini game I programmed. It is based on the mobile application called magic piano. The idea is that you hit the corresponding keys when the colored block is in the right area to get points. If you start to miss some blocks, then the music will slow down but will regain its normal speed upon getting more points. The functionality of the slowing down of the song is down with the TickRate and an integer. I use the integer to keep track of the number of blocks the user misses (i.e blocks they don't get any points from) and slow down the song by a certain amount depending on their number of missed blocks. If the user, however, does get points from a block then the integer that tracks the missed blocks will be subtracted by one, putting the song back or closer to its normal speed. The blocks coming down the screen was done by using rect() and constantly changing the y coordinate of the rect() to move it down the screen. I created variables for each lane that keeps track of where the block is in the lane, so we use that variable to give the user X amount of points depending on if/when they tap the lane key or if the block goes off the screen then we can just move it back to the top. The song playing is done through the FilePlayer object where we use the selectInput() function to allow the user to pick the file from their personal song library. I get the name of the file and pass it into the FilePlayer object.

```
// LANE ONE
if(timer.time() >= timer1) {
  fill(247, 108, 27); // orange
  rect(0, moving1, defaultWidth, defaultHeight);
  moving1 += 3; // speed of block
  laneOne = 575 - (moving1 + (defaultWidth/2));
  if(laneOne < -100) {
    moving1 = 0;
    missedTiles++;
  }

}
```

```
if(key == '1') {
  col1 = color(255, 255, 255, 150);

  if(laneOne >= -30 && laneOne <= 30) {
    score += 50;
    moving1 = 0;
    if(missedTiles > 0) missedTiles--;
  }
  else if(laneOne >= -55 && laneOne <= 55) {
    score += 25;
    moving1 = 0;
    if(missedTiles > 0) missedTiles--;
  }
  else if(laneOne >= -75 && laneOne <= 75) {
    score += 10;
    moving1 = 0;
    if(missedTiles > 0) missedTiles--;
  }
```

In the right section of the application consists of the UI elements to control the song and the Oscillator. The UI buttons were done by checking if the mouse position is inside of the specific rect() when the user clicks the mouse. The CP5 elements were done by creating specific functions to control the values set by them. Those value can then be used to control the gain and pan, for example, by passing the variables through the AudioOutput object (out.setPan(value);).

```
void mousePressed() {
  // check if clicking play button
  if(mouseX <= 630 && mouseX >= 570 && mouseY <= 150 && mouseY >= 110) {
    colPlay = color(126, 140, 194);
    play = true;
    if(songPicked) {
      player.loop();
      timer = new ControlTimer();
    }

  }
  // check if clicking the stop/pause button
  else if(mouseX <= 710 && mouseX >= 650 && mouseY <= 150 && mouseY >= 110) {
    colStop = color(126, 140, 194);

    moving1 = moving2 = moving3 = 0;

    if(songPicked && play) {
      player.rewind();
      player.pause();
    }
    play = false;

  }
  // check if clicking the restart button
  else if(mouseX <= 790 && mouseX >= 730 && mouseY <= 150 && mouseY >= 110) {
    colRestart = color(126, 140, 194);
    player.rewind();
    score = 0;
    moving1 = moving2 = moving3 = 0;

  }
  // check if clicking the pick-a-song button
  else if(mouseX <= 780 && mouseX >= 580 && mouseY <= 225 && mouseY >= 165 && !play) {
    selectInput("Select a file to process:", "fileSelected");

  }

}
```