```
/*
ME327 Haptics - Gold Miner Game Final Project
Author: Camille Chungyoun
*/
import processing.serial.*;

Serial myPort;        // The serial port

//*************************************************************
//****************** Initialize Variables **********************
//*************************************************************

float width = 1200;
float height = 800;
float ground_height = 100;
float gripper_x = 200;
float gripper_y = 200;
PShape gripper;  // The PShape object
float angle = radians(0.0); // angle of rope (and gripper)
float length = 0.0; // length of rope
int num_materials = 9;
int[] track_materials = new int[num_materials]; // to keep track of which obstacles have been taken. 0 means hasn't been taken, 1 means it is currently being taken, 2 means it has been taken
float[] distances = {291.22777, 490.83269, 725.0, 513.51648, 250.0, 575.0, 533.09519, 515.83269, 266.22777}; // store initial distances (in pixels) from rope origin to closest point of material
//float[] distances_to_center = {316.22777, 540.8326913, 750.0, 538.51648, 300.0, 600.0, 583.09519, 540.83269, 316.22777}; // distance from rope origin to center of material
float[][] angle_ranges = { {13.91472, 22.95518},
                           {28.40809, 38.97205},
                           {51.22095, 55.03925},
                           {65.54061, 70.85657},
                           {80.53768, 99.46232},
                           {87.61406, 92.38594},
                           {116.06267, 125.86484},
                           {143.66332, 148.95654},
                           {152.58017, 170.54993} }; // angle ranges where each material is located, in degrees
int score = 0;
int d_large = 100; // diameter of large material
int d_small = 50; // diameter of small material
float maxLength = 921.95; // max possible length of rope, in pixels
float minLengthIn = 0.0; // minimum length reeled in that can be read from the serial port
float maxLengthIn = maxLength / 1000; // maximum length reeled in that can be read from the serial port
color black = color(0);
```

```
color yellow = color(248, 252, 3);
color grey = color(150, 150, 150);
color dark_brown = color(126, 73, 3);
color orange_yellow = color(255, 200, 5);
color dark_grey = color(90, 90, 90);


//*****************************************************************
//****************** Initialize Variables *************************
//*****************************************************************

void setup () {
  // set the window size:
  size(1200, 800);

  // List all the available serial ports
  println(Serial.list());
  // Check the listed serial ports in your machine
  // and use the correct index number in Serial.list()[].

  myPort = new Serial(this, Serial.list()[1], 115200);  //make sure baud rate matches Arduino

  // A serialEvent() is generated when a newline character is received :
  myPort.bufferUntil('\n');

  background(black);      // set inital background:

}

void draw () {
  // everything happens in the serialEvent()
  background(255);
  stroke(black);     // stroke color
  strokeWeight(2);        // stroke wider

  //*****************************************************************
  //****************** Draw Objects in Scene ************************
  //*****************************************************************

  // draw line to represent ground
  line(0, ground_height, width, ground_height);

  // fill in sky gradient background
  fill(0, 200, 255);
  rect(0, 0, width, ground_height);
  color c1 = color(13, 148, 255); // dark blue
  color c2 = color(122, 189, 242); // light blue
  for (int i = 0; i ⩽ ground_height; i++) { // from https://processing.org/examples/lineargradient.html
    float inter = map(i, 0, ground_height, 0, 1);
```

```
    color c = lerpColor(c1, c2, inter);
    stroke(c);
    line(0, i, width, i);
  }

  // fill in dirt gradient background
  stroke(black);
  color c3 = color(219, 168, 91); // light tan
  color c4 = color(155, 94, 8); // darker tan
  for (int i = int(ground_height); i <= height; i++) { // from https://processing.org/examples/lineargradient.html
    float inter = map(i, int(ground_height), height, 0, 1);
    color c = lerpColor(c3, c4, inter);
    stroke(c);
    line(0, i, width, i);
  }

  // draw wheel with dark brown rope
  stroke(0);
  strokeWeight(2);
  fill(150, 150, 150);
  rect(width/2 - 50, ground_height - 75, 100, 50); // draw spool
  stroke(dark_brown);
  strokeWeight(10);
  line(width/2, ground_height, width/2, ground_height - 25);
  fill(126, 73, 3);
  rect(width/2 - 25, ground_height - 75, 50, 50); // draw wrapped rope
  stroke(0);
  strokeWeight(2);
  line(width/2 - 20, ground_height - 79, width/2 - 20, ground_height - 21);
  line(width/2 - 12, ground_height - 79, width/2 - 12, ground_height - 21);
  line(width/2 - 4, ground_height - 79, width/2 - 4, ground_height - 21);
  line(width/2 + 4, ground_height - 79, width/2 + 4, ground_height - 21);
  line(width/2 + 12, ground_height - 79, width/2 + 12, ground_height - 21);
  line(width/2 + 20, ground_height - 79, width/2 + 20, ground_height - 21);

  // calculate new gripper position
  gripper_x = width/2 - length*cos(angle);
  gripper_y = ground_height + length*sin(angle);

  // check if gripper can or has captured any of the uncaptured materials, update score as necessary
  for (int i = 0; i < num_materials; i = i+1) {
    if ((length >= distances[i]) && (angle > radians(angle_ranges[i][0])) && (angle < radians(angle_ranges[i][1])) && (track_materials[i] == 0)) {
      track_materials[i] = 1; // mark the material as being currently reeled in
    }
    // if material has been fully reeled in, then update score and stop drawing the shape
    if ((track_materials[i] == 1) && (length <= 10)) {
      track_materials[i] = 2;
```

```
      if ((i == 0) || (i == 2) || (i == 5)) { // small gold
        score += 50;
      } else if ((i == 1) || (i == 6)) {      // large gold
        score += 100;
      } else if ((i == 3) || (i == 7)) {      // small stone
        score += 20;
      } else {                                // large stone
        score += 40;
      }
    }
  }

  // 1: draw small gold (top left) if it hasn't been taken yet
  drawMaterial(0, yellow, width/4, height/4, d_small, gripper_x, gripper_y, angle);

  // 2: draw large gold (left center) if it hasn't been taken yet
  drawMaterial(1, yellow, width/8, height/2, d_large, gripper_x, gripper_y, angle);

  // 3: draw small gold (bottom left) if it hasn't been taken yet
  drawMaterial(2, yellow, width/8, 7*height/8, d_small, gripper_x, gripper_y, angle);

  // 4: draw small stone (bottom left) if it hasn't been taken yet
  drawMaterial(3, grey, width/2 - 200, height/2 + 200, d_small, gripper_x, gripper_y, angle);

  // 5: draw large stone (center) if it hasn't been taken yet
  drawMaterial(4, grey, width/2, height/2, d_large, gripper_x, gripper_y, angle);

  // 6: draw small gold (bottom center) if it hasn't been taken yet
  drawMaterial(5, yellow, width/2, 7*height/8, d_small, gripper_x, gripper_y, angle);

  // 7: draw large gold (bottom right) if it hasn't been taken yet
  drawMaterial(6, yellow, 3*width/4, 3*height/4, d_large, gripper_x, gripper_y, angle);

  // 8: draw small stone (right center) if it hasn't been taken yet
  drawMaterial(7, grey, 7*width/8, height/2, d_small, gripper_x, gripper_y, angle);

  // 9: draw large stone (top right) if it hasn't been taken yet
  drawMaterial(8, grey, 3*width/4, 1*height/4, d_large, gripper_x, gripper_y, angle);

  // draw rope as dark brown line
  stroke(dark_brown);
  strokeWeight(10);        // stroke wider
```

```
  line(width/2, ground_height, gripper_x, gripper_y); // x1, y1, x2, y2

  // draw gripper
  gripper = createShape();
  gripper.beginShape();
  gripper.noFill();
  gripper.strokeWeight(6);
  gripper.vertex(gripper_x - 25*sin(angle) - 25*cos(angle), gripper_y - 25*cos(angle) + 25*sin(angle)) ;
  gripper.vertex(gripper_x - 25*sin(angle), gripper_y - 25*cos(angle));
  gripper.vertex(gripper_x + 25*sin(angle), gripper_y + 25*cos(angle));
  gripper.vertex(gripper_x + 25*sin(angle) - 25*cos(angle), gripper_y + 25*cos(angle) + 25*sin(angle)) ;
  gripper.endShape();
  gripper.setStroke(color(255, 0, 0)); // red gripper
  shape(gripper);

  // display the current score
  stroke(0);
  strokeWeight(4);
  fill(255, 255, 255);
  rect(30, 20, 235, 60, 28);
  textSize(50);
  fill(255, 0, 0);
  text("Score: " + score, 40, 65);

  //// test changing length
  //if (length > 0) {
  //  length = length - 5;
  //}

  //// test changing angle
  //if ((angle >= 0) & (angle < radians(180))) {
  //  angle = angle + radians(1);
  //}

  //<>//////////////////////////////////////////////////////////////////
  //<>//////////////////// Draw Objects in Scene ////////////////////////
  //<>//////////////////////////////////////////////////////////////////
}


// draw a material based on its current location, size, color, and if it is being reeled in
void drawMaterial(int idx, color material_color, float x_loc, float y_loc, int diameter, float gripper_x, float gripper_y, float angle) {
  color new_color = material_color;
  float x_pos; // depends on if the material is moving or not
  float y_pos; // depends on if the material is moving or not

  if ((track_materials[idx] == 0) || (track_materials[idx] == 1)) {
```

```
    if (material_color == yellow) {
      new_color = orange_yellow;
    } else {
      new_color = dark_grey;
    }

    if (track_materials[idx] == 0) { // material is stationary
      x_pos = x_loc;
      y_pos = y_loc;
    } else { // material is currently being reeled in
      x_pos = gripper_x - (diameter/2)*cos(angle);
      y_pos = gripper_y + (diameter/2)*sin(angle);
    }

    // draw gradient of colors
    for (int i = 0; i <= diameter/2; i++) {
      float inter = map(i, 0, diameter/2, 0, 1);
      color c = lerpColor(material_color, new_color, inter);
      stroke(c);
      noFill();
      ellipse(x_pos, y_pos, i*2, i*2);
    }
  }

}

void serialEvent (Serial myPort) {
  //////////////////////////////////////////////////////////////
  //** Read in Handle and Mass Positions from Serial Port (START) **
  //////////////////////////////////////////////////////////////

  // (1) read the input string. HINT: use myPort.readStringUntil() with the appropriate argument
  String data = myPort.readStringUntil('\n');

  if (data == null) { // (2) if the input is null, don't do anything else
    return;
  } else {
    data = data.trim(); // (3) else, trim and convert string to a number
    String[] list = split(data, ' ');
    float button_val = float(list[0]); // indicates whether or not button has been pressed
    float new_val = float(list[1]); // either angle or length that gripper has been reeled in

    if (!Float.isNaN(button_val) && !Float.isNaN(new_val)) {
      if (button_val == 0) { // button has NOT been pressed
          angle = new_val; // second value is an angle (radians)
      } else if (button_val == 1) { // button has been pressed
          length = map(new_val, minLengthIn, maxLengthIn, 0, maxLength); // second value is the length that the gripper has been reeled in
```

```
273         } else {
274             println("Error: button_val must be 0 or 1");
275         }
276       }
277     }
278 
279     //*****************************************************************
280     //**** Read in Handle and Mass Positions from Serial Port (END) ***
281     //*****************************************************************
282 }
```