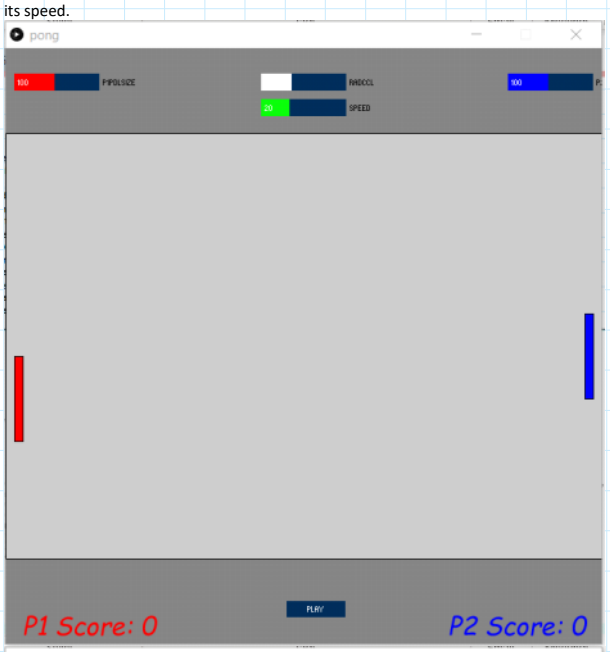
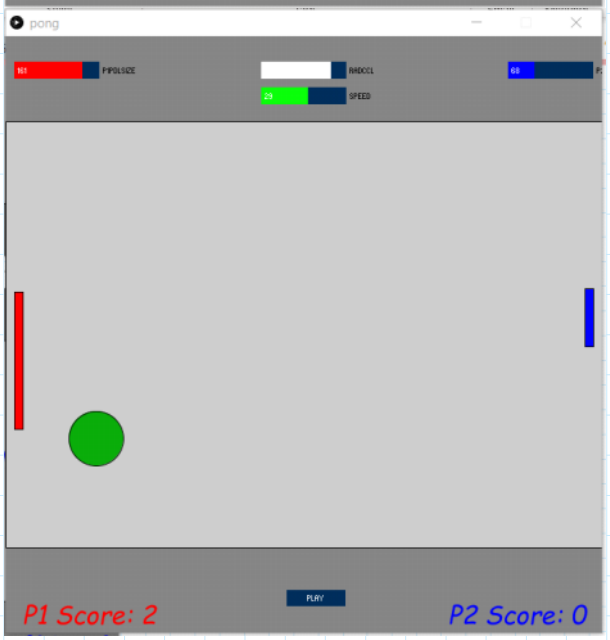
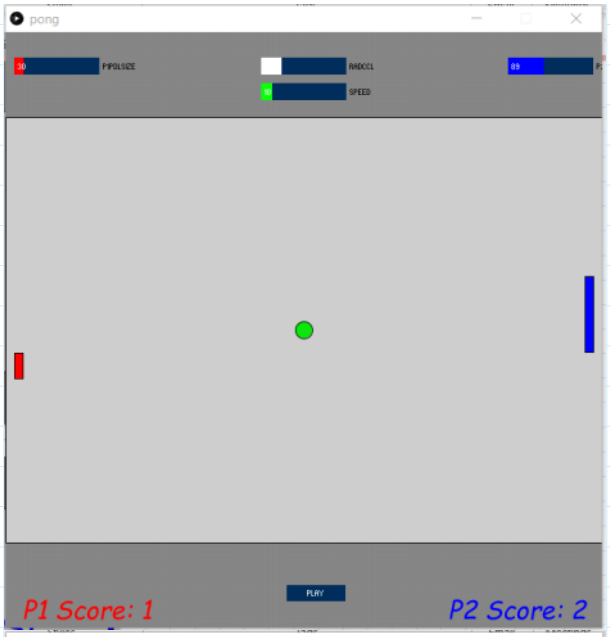
HW5 Simon Ng

Thursday, April 30, 2020 23:50

It took me a while to figure out how to have the ball bounce off objects. After considering using angles from 0 to 2pi or slopes with extensive if statements, I realized I could just change the sign of the x or y direction and this would give me the correct bouncing behavior.
Next I had issues with my ball bouncing even when it should have scored and getting "trapped" beyond boundaries. I realized the issue was the my ball would reach a point where it would always be satisfying the boundary contact case, so it would continually change direction, getting trapped. My solution was to artificially move the ball to just outside the boundary for the next iteration so it wouldn't get trapped.
I also wanted the ball to appear at the center for a second before moving towards a player's side. I had tried displaying the circle with a delay, but I realized it wasn't working because of how Processing draws at the end of the draw loop. I was able to get the ball to display for 1 second by using millis() to add a delay before the ball started moving, and during this millis() delay, I drew the ball, giving me what I wanted.
I also changed the font to Comic Sans MS italic because I really had no other choice.
My game also has sliders for the ball size and speed, and the color of the ball changes with its speed.








```
import processing.serial.*;
import cc.arduino.*;
import controlP5.*;
ControlP5 cp5;

Arduino arduino;
int ySize = 700; // pong window height
int xSize = 700; // pong window width
int xPlaySize = xSize; // if this is reduced below xSize, the xPdlPos needs to be changed
int yPlaySize = 500;
int leftBound = int(0 + 0.5*(xSize-xPlaySize)); // left play area boundary
int rightBound = int(xSize - 0.5*(xSize-xPlaySize)); // right play area boundary
int upperBound = int(0 + 0.5*(ySize-yPlaySize)); // top play area boundary
int lowerBound = int(ySize - 0.5*(ySize-yPlaySize)); // bottom play area boundary
int P1PdlSize, P2PdlSize, cclTop, cclBottom, cclLeft, cclRight, xCclPos, yCclPos, radCcl, speed; // paddle size, circle bounds, x & y coordinate of cirlce positon, circle radius, ball speed
int xPdlDim = 10; // width of paddle (doesn't change)
int pdlOffset = 10; // paddle distance from edge of play area
int yPdlDim[] = {60, 60}; // length of paddle (can change with slider)
int xPdlPos[] = {leftBound+pdlOffset, rightBound-(pdlOffset+xPdlDim)}; // sit 10 pixels away from edge (calculated for right paddle)
int yPdlPos[] = {ySize/2, ySize/2}; // player vertical position (initialized in middle)
int pdlCol[] = {color(255, 0, 0), color(0, 0, 255)}; // paddle colors
int pdlBottom[] = {0, 0}; // bottom bound of paddle (top boundary is its position)
boolean start = false; // play button
boolean run = false;
boolean initialMove;
float dxdt, dydt; // x and y ball movement
int score[] = {0, 0};
int winningScore = 4; // play to score
int winner = 0;
int t_delay; // millis storage variable to get ball to display briefly before moving

void setup() {
  size(700, 700);
  println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
```

```
  cp5 = new ControlP5(this);

  cp5.addSlider("P1PdlSize") // slider to control player 1 paddle length
    .setPosition(10, 30)
    .setSize(100, 20)
    .setRange(10, 200)
    .setValue(100)
    .setColorCaptionLabel(0)
    .setColorForeground(pdlCol[0])
    ;

  cp5.addSlider("P2PdlSize")  // slider to control player 2 paddle length
    .setPosition(xSize-100-10, 30)
    .setSize(100, 20)
    .setRange(10, 200)
    .setValue(100)
    .setColorCaptionLabel(0)
    .setColorForeground(pdlCol[1])
    ;

  cp5.addSlider("radCcl") // slider to control circle size
    .setPosition(xSize/2-50, 30)
    .setSize(100, 20)
    .setRange(1, 40)
    .setValue(15)
    .setColorCaptionLabel(255)
    .setColorForeground(color(255))
    ;

  cp5.addSlider("speed") // slider to control ball speed
    .setPosition(xSize/2-50, 60)
    .setSize(100, 20)
    .setRange(5, 50)
    .setValue(20)
    .setColorCaptionLabel(0)
    .setColorForeground(color(10, 250, 10))
    ;

  cp5.addButton("play") // start button
    .setPosition(xSize/2-20, ySize - 50)
    ;
  textFont(createFont("Comic Sans MS Italic", 30));
  //String[] fontList = PFont.list();
  //printArray(fontList);
}

void draw() {
  yPdlDim[0] = P1PdlSize; //update paddle 1 size
  yPdlDim[1] = P2PdlSize; //update paddle 2 size
  background(130);


  drawPlayArea();
  drawPaddle();
  if (winner != 0) {
    textSize(80);
    fill(pdlCol[winner-1]);
    text("P" + (winner) + " Wins!", xSize/2-130, ySize/2);
  }
  initializeBall();

  if (millis() < t_delay+1000) {
    fill(10, 255-speed*3, 10);
    circle(xSize/2, ySize/2, radCcl*2); // update circle position
  } else if (run) {
    //Draw Circle
    fill(10, 255-speed*3, 10);
    circle(xCclPos, yCclPos, radCcl*2); // update circle position
    cclLeft = xCclPos-radCcl;
    cclRight = xCclPos+radCcl;
    cclTop = yCclPos-radCcl;
    cclBottom = yCclPos+radCcl;

    if (initialMove) {
      xCclPos += dxdt*10; // slower initial speed
      yCclPos += dydt*10; // slower initial speed
    } else {
      xCclPos += dxdt*speed; // update x position at slider speed
      yCclPos += dydt*speed; // update y position at slider speed
    }
    collision();
  }
}



public void play() {
  start = true;
}

public void drawPlayArea() {
  fill(200);
  rect(leftBound, upperBound, xPlaySize, yPlaySize);
  textSize(30);
  fill(pdlCol[0]);
  text("P1 Score: " + score[0], 20, ySize-10);
  fill(pdlCol[1]);
  text("P2 Score: " + score[1], xSize-180, ySize-10);
}

public void drawPaddle() {
  // READ AND DRAW PADDLE POSITION
  for (int i = 0; i < 2; i++) {
    yPdlPos[i] = int(map(float(arduino.analogRead(i)), 0, 1023, lowerBound-yPdlDim[i], upperBound)); // read paddle location from potentiometers
```

```
    fill(pdlCol[i]);
    rect(xPdlPos[i], yPdlPos[i], xPdlDim, yPdlDim[i]); // update rectangle position
    pdlBottom[i] = int(yPdlPos[i] + yPdlDim[i]);
  }
}

public void initializeBall() {
  if (start) { // initialize movement
    xCclPos = xSize/2; // x coordinate of cirlce position
    yCclPos = ySize/2;

    dxdt = random(-1, 1); // random
    dydt = random(-1, 1); // random
    while ((dxdt < 0.1 && dxdt > -0.1) || dydt == 0) { // bad starting angle
      dxdt = random(-2, 2); // random
      dydt = random(-1, 1); // random
    }
    start = false;
    run = true;
    initialMove = true;
    t_delay = millis();
    winner = 0;
  }
}

public void collision() { // determine if the ball has hit something
  if (cclTop < upperBound) { // ball hit top boundary
    dydt = -dydt;
    yCclPos = upperBound+radCcl;// reset ball to boundary
  } else if (cclBottom > lowerBound) { // ball hit bottom boundary
    dydt = -dydt;
    yCclPos = lowerBound-radCcl;// reset ball to boundary
  }

  if (cclLeft < leftBound+(pdlOffset+xPdlDim)) { // ball has reached left side
    if ((yCclPos > yPdlPos[0] && yCclPos < pdlBottom[0])) { // ball center within paddle space
      dxdt = -dxdt;
      xCclPos = leftBound+(pdlOffset+xPdlDim)+radCcl; // reset ball to boundary
      initialMove = false;
      if (cclTop == upperBound || cclBottom == lowerBound) {
        dydt-=0.5;
      }
    } else if (cclRight < leftBound) { // P2 scores
      goal(1);
    }
  } else if (cclRight > rightBound-(pdlOffset+xPdlDim)) {// ball has reached right side
    if ((yCclPos > yPdlPos[1] && yCclPos < pdlBottom[1])) { // ball center within paddle space
      dxdt = -dxdt;
      xCclPos = rightBound-(pdlOffset+xPdlDim)-radCcl; // reset ball to boundary
      initialMove = false;
      if (cclTop == upperBound || cclBottom == lowerBound) {
        dydt-=0.5;
      }
    } else if (cclLeft > rightBound) { // P1 scores
      goal(0);
    }
  }
}

public void goal(int pointTo) {
  score[pointTo]++;
  start = true;
  run = false;
  if (score[pointTo] == winningScore) {
    start = false;
    score[0] = 0;
    score[1] = 0;
    winner = pointTo+1;
  }
}
```