Ian McKay

MART 191

12/11/16

Pong in P5: Build Process

For my final project for Creative Coding 1 I chose to recreate the original video game, *Pong*, within P5. It's something I had thought about doing before I had been assigned the project, and I'm thankful I got to work on it. In my initial final project proposal I had planned on including sound effects, a functioning scoreboard, and a computer combatant. The scoreboard and sound effects are there, but I failed to implement a computer combatant, so unfortunately my *Pong* recreation can only be played in two player mode. Nonetheless, I'm proud at what I managed to accomplish and glad that my game is functioning.

The initial work schedule I had planned was as followed: By the end of the coming weekend (11/20) I plan to have an outline of the classes and variables I'll need. By the end of the following week, I want to have a working version minus the computer combatant. The following 2 weeks after that will be dedicated to programming the computer combatant and working out any bugs. Obviously, I didn't follow this schedule faithfully; I underestimated how long creating a working version of pong (minus a computer combatant) would take me. Planning my variables out was initially helpful, but because I didn't know what I would end up needing, so my initial variable list doubled by the end of the project.

```
var ballX=20;
    ballY=20;
    ballR=3;
    dX=2;
    dY=2;
    paddleY=10;
    paddleX=385;
    paddleH=30;
    paddleW=10;

// ball
var ballX=20;
    ballY=20;
    ballR=3;
// ballspeed
    dX=4;
    dY=4;
    paddleY=10;
    paddleX=10;
    paddleH=120;
    paddleW=10;
    paddleY2=10;
    paddleW2=10;
    paddleH2=120;
    paddleX2=40;
    paddle5peed=10;
    mapwidth=800;
    mapheight=500;
    player1score=1;
    player2score=1;
```

Ian McKay

MART 191

12/11/16

I started my project with the creation of my *Pong* ball. I wanted it to being moving about my canvas, bound by the walls of the window, before I made my paddles. I successfully implemented movement by using two independent speed variables to modify the x and y coordinates of an ellipse.

```
ballX=ballX+dX;
ballY=ballY+dY;
```

I then bound it within the walls of my canvas by utilizing if statements. I initially included additional if statements that prevented the ellipse from leaving either side of the canvas, which I would later remove once I implemented paddles and scoring.

```
if (ballY < 0){
  dY = -dY;
}
if (ballY > height){
  dY = -dY;
}
```

I had done projects with similar bouncing ball mechanics previously in the semester, so this part went swimmingly; filling me with a false sense of confidence.

I moved on to creating my paddles. My plan was to have one player control their paddle with the up and down arrow keys and the other player using w and s. After creating a appropriately sized rectangle for my first paddle I started playing around with it's movement. At first, I utilized the keyPressed event function in conjuncture with keyCode to achieve my movement. Figuring this aspect out took way longer than it ever should have, mainly because I didn't understand how keyCode worked. A huge breakthrough which helped me hurdle this and move on was discovering the keycode website at keycode.info. The website tells you the

Ian McKay

MART 191

12/11/16

keyCode values for every key and it allowed me to implement movement onto my second paddle.

```
function keyPressed(){
  if (keyCode===DOWN_ARROW){
    paddleY=paddleY+paddleH;
  }else if(keyCode===UP_ARROW){
    paddleY=paddleY-paddleH;
  }

}
```

Being able to move my paddle lead to additional roadblocks. The movement of my paddle was choppy and users (me) had to press the movement keys repeatedly to move up or down. Additionally, keyPressed was not working in a two paddle system, it couldn't handle simultaneous inputs from both paddles. My next goals were to create smoother paddle movement and figure out how to have both paddles be controllable simultaneously. Thankfully my professor pointed me towards the keyIsDown function, which proved to solve both problems. My paddle movement was now smooth and automatic, and both paddles could now simultaneously move.

```
  if (keyIsDown(DOWN_ARROW)){
    paddleY=paddleY+paddleSpeed;
  }
  if(keyIsDown(UP_ARROW)){
    paddleY=paddleY-paddleSpeed;
  }
// Paddle up and down movement tied to w and s keys for player 1.
  if(keyIsDown(87)){
    paddleY2=paddleY2-paddleSpeed;
  }
  if(keyIsDown(83)){
    paddleY2=paddleY2+paddleSpeed;
  }
```

Ball to paddle detection was my next problem to solve. I knew I would just be utilizing if statements similarly to how I used them to prevent my ball from escaping my canvas. I understood that the x and y coordinates of my rectangle represented the top left corner of the

Ian McKay

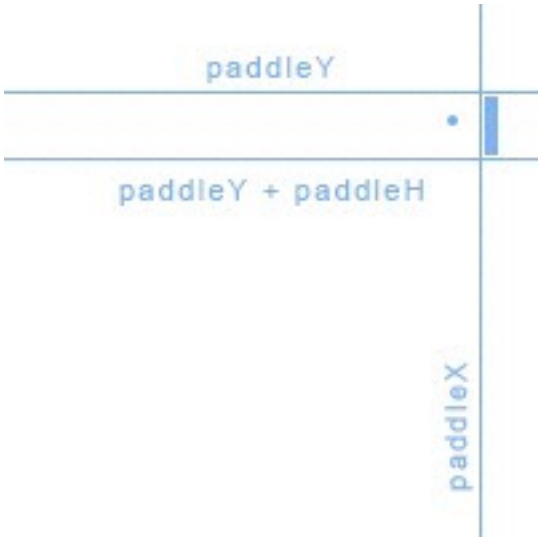
MART 191

12/11/16

rectangle, so I wrote my if booleans with that in mind. I added the paddle height variable to my paddle y coordinate in order to create a length which represented the face of the paddle. And by combining that with a length represented my paddle's x coordinate I was able to create a boundary representing the face of my paddle which I would use within an if statement to bounce my ball off of my paddle. I was overwhelmed with relief when my ball bounced off my paddle for the first time. I tripled checked the accuracy of my paddle boundaries and decided to move on.

```
function collision(){
  return (ballX >= paddleX) && (ballX <= paddleX + paddleW) &&  (ballY >= paddleY) && (ballY <= paddleY+paddleH);

}
```



Sound effects were easy to implement. I went to youtube and ripped mp3 files from videos featuring the sound effects I wanted, I put said mp3's into an audio editing program and made shorter files featuring just the sound effects that I would end up using. I put both mp3 files in my project folder and used the preload function to load them into my sketch. I then called on my sounds whenever my collision function would trigger or when the ball would leave the

Ian McKay

MART 191

12/11/16

window. I made some stupid errors having to do with file names which temporarily halted my progress, but I got sound implemented relatively quickly once that was cleared up.

```
function preload() {
  sound = loadSound('pong.mp3');
  sound2= loadSound('ballout.mp3');
}
```

Up until this point, whenever the ball would escape I simply would reload my sketch to play again. This wasn't very efficient for testing and I knew eventually I would need some sort of way to have the game automatically restart once someone scored. To create a reset function I simply named a function reset and re-listed all of my variables underneath it, so that if it gets called my variables would return to default and the game would restart.

```
function reset(){
  ballX=windowHeight/2;
  ballY=windowWidth/2;
  ballR=3;
  dX=4;
  dY=4;
  paddleY=windowHeight/2;
  paddleX=windowWidth-80;
  paddleH=windowHeight/7;
  paddleW=10;
  paddleY2=windowHeight/2;
  paddleW2=10;
  paddleH2=windowHeight/7;
  paddleX2=80;
  paddleSpeed=10;
}
```

A scoreboard was the next item on my agenda. I had been putting off working on the scoreboard because I thought it was going to be fairly difficult to figure out, but in the end it didn't take very long at all. I started by creating variables for both players scores.

```
player1score=1;
player2score=1;
```

I then created a scoreboard function which utilized the text function to draw the scores for both players in the top middle of my canvas.

```
function scoreboard(){
  textSize(30);
  textAlign(LEFT);
  text(player1score, windowWidth/2-50,30);
  textAlign(RIGHT);
  text(player2score, windowWidth/2+50,30);
}
```

Ian McKay

MART 191

12/11/16

I added player1score++ and player2score++ to my if statements that triggered whenever a ball would escape the canvas. And it was simple as that, I initially had player1score corresponding to the wrong side of the canvas, but I fixed that easily and moved on.

```
// If the ball escapes from the right, player 1 scores and the game is reset. And a sound effect.
if (ballX > width)  {
  sound2.play();
  reset();
  player1score++;
// If the ball escapes from the left, player 2 scores and the game is reset. And a sound effect.
}else if(ballX < 0){
  sound2.play();
  reset();
  player2score++;
}
```

My last priority was having the game be able to be played in any window size. Up until this point, my game was always played on a 400 by 400 canvas. I changed my canvas size to the variables mapwidth and mapheight which I passed the P5 variables windowHeight and windowWidth into.

```
mapwidth=windowWidth;
mapheight=windowHeight;
createCanvas(mapwidth,mapheight);
```

My canvas was now reliant on the browser windows size, which in turn really fucked up my paddles. I needed to change my variables so that, they too, were reliant upon window size. I utilized a new variable list within the setup function in order to do just that.

```
ballX=windowWidth/2;
ballY=windowHeight/2;
paddleY=windowHeight/2;
paddleY2=windowHeight/2;
paddleH=windowHeight/7;
paddleH2=windowHeight/7;
paddleX=windowWidth-80;
paddleX2=80
```

My paddle length was now consistently a seventh the size of however big the window height was. Paddle and ball starting positions were also changed so that they are always centered at

the start of each round. I also changed my scoreboard so that the scores were always 50 pixels off center in either direction.

And that was it. There was a lot more I wanted to do, but I was out of time. If I continued to work on this project I would've immediately worked to make my paddles have the ability to "spike" the ball in a whichever direction. My game lacks a lot of user implemented competitiveness because I wasn't able to do that; which sucks, but I'm still pleased with my final product nonetheless. I'm not sure if I'll ever have to take another coding class ever again, but I enjoyed this one enough to where I'd certainly want to.