“Air Hockey”

A virtual two player air hockey game

Jessica Krapfl
20250742

CS171 Computer Systems I

B.Sc.

Computer Science and Software Engineering

Department of Computer Science

Maynooth University

Maynooth, Co. Kildare

Ireland

Declaration

I hereby certify that this material, which I now submit for assessment as part of CS171 Computer Systems module, is entirely my own work and has not been taken from the work of others - save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Jessica Krapfl   Date: 12/12/2020

Acknowledgments

I would like to thank Tim Hughes for helping me test my program and also for helping me correct this project report.

Abstract

This document describes the development of a game designed to allow two people to play using the same keyboard. The game is similar to an arcade style air hockey table. Each player will use their hammer to hit a puck into the opposite goal. The hammers will be moved using keyboard input from each player.

1 Introduction

The idea for the CS171 semester one project is an air hockey game. When you first run the program a starting menu will appear that tells each player the controls they use to move their mallets. Player1 will use WASD to move and player2 will use the arrow keys. After the players have read the instructions and pressed the spacebar to continue, the next screen asks if you want to play first to 10 wins or if you want to play whoever has the most points in a time frame that the player picks. After you pick your desired way to play the game, it will start. The players will hit the puck with their hammer. The puck will bounce off of the hammer and the sides of the arena. Each player is not able to move their hammer beyond the center line that divides the screen. A player will score when they hit the puck into the opposite goal. If say player1 scored into the goal behind player2, player1 would earn a point. After player1 scores the screen will reset back to how it was at the start of the game. The game will continue until a player reaches the score of 10 or until the time runs out whichever the players chose at the start of the game. When a game finishes a screen will come up that says who won and will ask you if you want to play again. If you choose no the program will stop and if you choose yes you will be brought back to the original starting screen.

2 Making the Game

When starting this project it was easiest to start drawing out what each screen should look like next to the drawings. It was also helpful to take down notes of the coordinates of each object on the screen. Then made a numbered list of what should get done, in the order of what should be done. This was very helpful as it broke down a massive idea into steps and what each step should look like. After this was finished, it was finally time to start coding.

While making the starting screen, it was realized that if all the screens were put in the draw function the code would quickly become very jumbled and messy. It was also realized that I did not know how to go about flipping through different screens. Then I came across a processing forum [1]. This person neatly had each of their screens in their function. All that was in the draw function as if-else statements checking for an int variable called 'screen'. This helped to keep the code very organized and also easily allowed for the reuse of functions if needed.

The starting screen needed to have the directions for the user. Once the user has read the directions, the user needs to press the spacebar to change to the next screen. I also did not know how to do this which brought me to another processing forum [2]. This used 'keyPressed' to check for keys being pressed. So 'keyPressed' was used for the start screen to flip to the options screen. For the options screen, the user needed to pick either option 1: "play first to 10" or option 2: "play whoever has the most points after x minutes". For this keyPressed was used again to check if 1 or 2 was pressed. The next thing to do was to add functionality to the game screen as at this point it was only just drawn out.

The first thing for the game screen was to get the hammers to move with user input. Originally more if statements checking for if ‘keyPressed’ equaled “WASD” or any of the arrow keys. This method worked okay, it was not what I was expecting. It was really hard to move your hammer when two people were playing and pressing keys, the hammers did not move diagonally, and they did not stop when you let go of the keys. This took a very long time to figure out how to function properly. I thought this was happening because of rollover(key) [3]. I thought that my keyboard just could not handle that many inputs. Then I tried using Tim's gaming keyboard thinking that because that keyboard is designed to take in multiple keys that it would work however, it did not work with his keyboard either. After more searching came across another processing forum [4]. This person had a boolean array and a switch matching the boolean array to the key that needed to be pressed for that boolean value to be true. There was also a keyPressed() function changing the boolean value to true and a keyReleased() function to check that if it was not being pressed it would be false again. This code worked on my computer with no problem and decided to implement something similar to my project. This ended up working and I realized that keyPressed is just like any other variable and it could only equal one thing at a time.

Next was to get the puck to move, bounce off the walls and hammers and add to the score when it hits the goal. I got the puck to move and add to the scores just like the ball in lab 2. Then I needed to figure out how to get the ball to bounce off the hammers. For this, I found a few helpful sites [5][6]. This worked great until the hammer overlapped the puck, when it overlapped the puck just shook. To fix this I found this website[7]. When the hammer and the puck overlapped the normal method of bouncing off of the hammers did not work. Because each frame updates the puckDirection = -puckDirection; for every frame and because it takes more than one frame for the puck to get out from under the hammer the puckDirection = -puckDirection; is constantly being updated and so that is why it shook. Then I wanted to make the puck only move when it was hit. So I made a boolean variable called puckHit and set it equal to false. Then in the code where it checks for when a hammer hits the puck, I set puckHit to true so that the puck would move only when it was hit.

The last thing was to figure out the timer for game option 2. The first part of this was getting user input. Using what we did in lab 5 part 3 where the program asked for a float from the user. So I added the JOptionPane library to help with user input. I thought it would be helpful to have a processing library that handled the timer and found on the processing website a library called countdownTimer. From the examples given I was able to figure out how to use this to create a timer at the top of the screen that ends the game once it is done counting down.

3 Evaluation

Then finally it was time to test the game. One of the first things I noticed was if I pressed option 1 and played the game it would not go to the end screen it would go right back to the start screen. I figured out this was because option 1 on the end screen was to play again and the key function in processing remembered that and automatically started the game again even though the user may or may not have wanted that. To fix this I had the game options screen use 1 and 2 and have the end game screen use 3 and 4.

4 Conclusion

To conclude I am very happy with how the game turned out overall. There are a few things that I would change. One of these things being that sometimes the puck gets stuck outside the bounds of the arena. This is because somehow the hammer hitting the puck precedes the arena wall. Other things that could improve is the puck physics in general. Sometimes the way it bounces off the hammer does not make much sense and the puck never just goes in one direction the x and y values are always changing together.

References

[1] Website - https://discourse.processing.org/t/how-to-make-a-start-game-screen/23776

[2] Website - https://discourse.processing.org/t/problem-with-keypressed-and-space/7468

[3] Website - https://en.wikipedia.org/wiki/Rollover_(key)

[4] Website -
https://discourse.processing.org/t/keypressed-for-multiple-keys-pressed-at-the-same-time/18892

[5] Website - https://happycoding.io/tutorials/processing/collision-detection

[6] Website - https://processing.org/reference/dist_.html

[7] Website - https://forum.processing.org/two/discussion/24966/bouncing-balls-collision

Appendix

```
/*
Jessica Krapfl
CS171
Project
Air hockey
last edit: 15/12/20
*/
//******things for the timer********//
import javax.swing.JOptionPane;//to get user input for the time
//learned how to do the JOptionPane this from Lab 5 part b where the user was routinely
//asked to enter a float
import com.dhchoi.CountdownTimer;
import com.dhchoi.CountdownTimerService;
//came across this library when going through processings site to find a library that
//would help me with time keeping i learned how to make a timer from the examples given

final long SECOND_IN_MILLIS = 1000;
final long HOUR_IN_MILLIS = 36000000;

CountdownTimer timer;
int timeToElapse;
String timeText = "";
int timeTextSeconds = 0, timeTextMinutes = 0; // the seconds and minutes to be displayed

//*********boolean array for key detection*********//
boolean[] moveHammer = new boolean[8];//the origional way i was moving the hammers only
//allowed one button to be pressed. this was very hard to figure out how i could allow
//more keys to be pressed and i came across another processing forum that was discussing
//key jamming and the idea of using a boolean array instead of what i was origionally using

//*****other gobal variables*******//
int screen = 0;
//if screen =0: start screen/
```

```
//if screen =1: options screen/
//if screen =2: game screen(first to 10)/
//if screen =3: game screen(winner after x min)
//if screen =4: end screen

boolean puckHit = false;

char space = ' ';
char option1 = '1';
char option2 = '2';
char option3 = '3';
char option4 = '4';

int p1_x = 312, p1_y = 375,p1DirectionX = 5, p1DirectionY = 5;//
int p2_x = 903,p2_y = 375, p2DirectionX = 5, p2DirectionY = 5;
int puckX = 625, puckY = 375, puckDirectionX = 5, puckDirectionY = 5;//starting pos of the puck
and the direction it will be going
int p1Score = 0;
int p2Score = 0;

int radiusOfPuck = 37;
int radiusOfHam = 75;

void setup()
{
  size(1250,750);// create window

                                                                  timer                    =
CountdownTimerService.getNewCountdownTimer(this).configure(SECOND_IN_MILLIS,
HOUR_IN_MILLIS);
    updateTimeText();
}

//************DRAW*********//
void draw()
{
```

```
//i got the idea for doing the screens like this from a processing forum and i really wanted to
implement
  //this idea into my own code because it keeps the draw function very tidey and if something is
wrong with
  //a certian screen i dont have to go through the whole draw function to figure out where the
error is

  if(screen ==0)
  {
    startScreen();
    //reset scores
    p1Score =0;
    p2Score = 0;
    if(key == space)//once space is pressed screen is 1
    {
      screen = 1;
    }
  }
  else if(screen ==1)//option screen
  {
    optionScreen();
    if(key == option1)
    {
      screen = 2;
    }
    else if(key == option2)
    {
            String s = JOptionPane.showInputDialog("Enter a time (300 is 5 min):",
JOptionPane.QUESTION_MESSAGE);
      try
      {
        timeToElapse = parseInt(s);
      }
      catch(NumberFormatException e)
      {
        println("you did not enter a number!");
}
        screen = 3;
      }
  }
  else if(screen ==2)//first to 10 game
  {
    gameScreen();
    puck();
    p1Hammer();
    p2Hammer();
    fill(0,0,0);
    //text(mouseY,595,25);
  }
  else if(screen ==3)//most points in x minutes
  {
    gameScreen();
    puck();
    p1Hammer();
    p2Hammer();
    fill(0,0,0);
    text(timeText, 595, 25);
    timer.start();
  }
  else if (screen == 4)//end game screen
  {
    endGame();
    if(key == option3)
    {
      screen = 0;
    }
    else if(key == option4)
    {
      exit();
    }
  }
```

```
 }

//***************SCREENS**************//
//*****screen 0******//
void startScreen()
{
  background(255,255,255);//make the background white

    textSize(64);//title text
    fill(233,87,72);
    text("Air Hockey",468,247);

    textSize(30);//p1 instructions
    fill(233,87,72);
    text("P1 use WASD keys",156,562);

    textSize(30);//P2 instructions
    fill(64,145,245);
    text("P2 use IJKL keys",781,562);

    textSize(30);
    fill(0,0,0);
    text("Press space to continue",464, 700);
}

//*******screen 1******//
void optionScreen()
{
    background(255,255,255);

    textSize(64);//title how do you want to play
    fill(233,87,72);
    text("How do you want to play?",242,187);

    textSize(30);//first to 10
 fill(0,0,0);
   text("Press   1   to play first to 10",418,297);

   textSize(30);//or
   fill(0,0,0);
   text("or",625,425);

   textSize(30);//most points in x amount of time
   fill(0,0,0);
   text("Press   2   to play most points in x minutes",313,597);
}

//*****screen 2*******//
void gameScreen()
{
  background(255,255,255);

  //************set up the arena**************//
  strokeWeight(5);//arena bounds
  fill(255,255,255);
  rect(25,30,1200,700);

  strokeWeight(5);//dividing line
  line(625,30,625,727);

  //******************P1 Stuff***********//
  strokeWeight(1);//p1 goal box
  fill(233,87,72);
  rect(1220,213,1250,305);

  textSize(24);//P1 score text
  fill(233,87,72);
  text("P1 SCORE: "+p1Score,25,25);

  //*******************p2 stuff**************//
```

```
strokeWeight(1);//p2 goal box
 fill(64,145,245);
 rect(0,213,30,305);

 textSize(24);//p2 score text
 fill(64,145,245);
 text("P2 SCORE: "+p2Score,903,25);

 //***********Score***********//
 if (puckX >(width-65) && 248 < puckY && puckY < 480)//p1 scores
  {
   p1Score =p1Score +1;
   puckHit = false;//reset puck in the field

   //reset where the hammers are
   p1_x = 312;
   p1_y = 375;

   p2_x = 903;
   p2_y = 375;

    if(p1Score == 10 && screen ==2)//when playing first to 10 once 10 is reached go to the end
screen
   {
     screen = 4;
   }
  }
 if(puckX < 55 && 248 < puckY && puckY < 480)//p2 scores
  {
   p2Score = p2Score + 1;
   puckHit = false;//reset puck in the field

   //reset where the hammers are
   p1_x = 312;
   p1_y = 375;
p2_x = 903;
     p2_y = 375;

      if(p2Score == 10 && screen == 2)//when playing first to 10 once 10 is reached go to the end
screen
     {
       screen = 4;
     }
   }
}

//********Screen 4*********//
void endGame()
{
  background(255,255,255);
  //***winner text***
  if(p1Score > p2Score)//if p1 wwins
  {
    textSize(64);//title text
    fill(233,87,72);
    text("P1 Wins!",468,247);
  }
  else if(p2Score > p1Score)//if p2 wins
  {
    textSize(64);
    fill(64,145,245);
    text("P2 Wins!",500,247);
  }

  textSize(30);
  fill(0,0,0);
  text("Press 3 to play again",480,400);

  textSize(30);
 fill(0,0,0);
```

```
  text("or",625,500);

  textSize(30);
  fill(0,0,0);
  text("Press 4 to end",525,600);
}

//*************FUNCTIONS***************/
//***********puck*********//
void puck()
{
  strokeWeight(1);//puck
  fill(0,0,0);
  ellipse(puckX,puckY,75,75);
  //ellipse(mouseX,mouseY,75,75);

  //***********Moving the puck*********//
  //boundries
  if(puckHit == true)
  {
    puckX = puckX + puckDirectionX;
    if(puckX<55)  puckDirectionX = -puckDirectionX;//reverse if hit boundry
    if(puckX>(width-65)) puckDirectionX = -puckDirectionX;//reverse if hit boundry

    puckY = puckY +puckDirectionY;
    if(puckY<60)  puckDirectionY = -puckDirectionY;//reverse if hit boundry
    if(puckY>(height -65))  puckDirectionY = -puckDirectionY;//reverse if hit boundry
  }
  else if(puckHit == false)
  {
    puckX = 625;
    puckY = 375;
  }
  //*******bounceing off hammers****//
  //bounce off p1 hammer
  if(dist(puckX,puckY,p1_x,p1_y) <(radiusOfPuck + radiusOfHam))
  {
    puckHit = true;
    if(puckX < p1_x)
    {
      puckDirectionX = -abs(puckDirectionX);
    }
    if(puckX > p1_x)
    {
      puckDirectionX = abs(puckDirectionX);
    }
    if(puckY < p1_y)
    {
      puckDirectionY = -abs(puckDirectionY);
    }
    if(puckY < p1_y)
    {
      puckDirectionY = abs(puckDirectionY);
    }
  }
  //bounce off p2 hammer
  if(dist(puckX,puckY,p2_x,p2_y) <(radiusOfPuck + radiusOfHam))
  {
    puckHit = true;
    if(puckX < p2_x)
    {
      puckDirectionX = -abs(puckDirectionX);
    }
    if(puckX > p2_x)
    {
      puckDirectionX = abs(puckDirectionX);
    }
    if(puckY < p2_y)
{
      puckDirectionY = -abs(puckDirectionY);
    }
    if(puckY < p2_y)
    {
      puckDirectionY = abs(puckDirectionY);
    }
  }
}
//********p1 hammer*******//
void p1Hammer()
{
  strokeWeight(1);//p1 hammer
  fill(233,87,72);
  ellipse(p1_x,p1_y,150,150);

  //****************Moving p1 hammer**************//
   //*****direct direction******//
    if(moveHammer[0] == true && p1_y > 105)
    {
      p1_y= p1_y - p1DirectionY;
    }
    if(moveHammer[1] == true && p1_x > 100)
    {
      p1_x = p1_x - p1DirectionX;
    }
    if(moveHammer[2] == true && p1_y < 650)
    {
      p1_y = p1_y +p1DirectionY;
    }
    if(moveHammer[3] == true && p1_x < 550)
    {
      p1_x = p1_x + p1DirectionX;
    }
    /*the original way this was done:
    if(keyPressed == 'w' && p1_y > 105)
    {
      p1_y= p1_y - p1DirectionY;
    }
    if(keyPressed == 'a' && p1_x > 100)
    {
      p1_x = p1_x - p1DirectionX;
    }
    if(keyPressed == 's' && p1_y < 650)
    {
      p1_y = p1_y +p1DirectionY;
    }
    if(keyPressed == 'd' && p1_x < 550)
    {
      p1_x = p1_x + p1DirectionX;
    }
    */
}
//*********p2 Hammer*********//
void p2Hammer()
{
  strokeWeight(1);//p2 hammer
  fill(64,145,245);
  ellipse(p2_x,p2_y,150,150);

 //**************Moving p2 Hammer***********//

    if(moveHammer[4] == true && p2_y>105)
    {
      p2_y= p2_y - p2DirectionY;
    }
    if(moveHammer[5] == true && p2_x > 700)
    {
      p2_x = p2_x - p2DirectionX;
 }
    if(moveHammer[6] == true && p2_y <650)
    {
```

```
    p2_y = p2_y +p2DirectionY;
  }
  if(moveHammer[7] == true && p2_x < 1150)
  {
    p2_x = p2_x + p2DirectionX;
  }
}

//**************Timer functions**********//
void updateTimeText() {
 timeTextSeconds = timeToElapse % 60;
 timeTextMinutes = timeToElapse / 60;
 timeText = nf(timeTextMinutes, 2) + ':' + nf(timeTextSeconds, 2);
}

void onTickEvent(CountdownTimer t, long timeLeftUntilFinish)
{
 --timeToElapse;
 updateTimeText();

 if(timeToElapse ==0)
 {
  screen = 4;
 }
}

//***********bool array swtich function to set the correct keys*********//

void setMovement(int k, boolean b)
{
 switch (k)
 {
  case 'w': moveHammer[0] = b;break;
  case 'a': moveHammer[1] = b;break;
  case 's': moveHammer[2] = b;break;
```

```
    case 'd': moveHammer[3] = b;break;
    case 'i': moveHammer[4] = b;break;
    case 'j': moveHammer[5] = b;break;
    case 'k': moveHammer[6] = b;break;
    case 'l': moveHammer[7] = b;break;
  }
}
//******if key pressed key = true when key released = false
void keyPressed() {
  setMovement(key, true);
}

void keyReleased() {
  setMovement(key, false);
}
```