```
//I worked independently for the entirety of this project.
//I wrote all of the code by myself. I did not borrow code from anybody else.
//The Code for "Wall Ball":

/**** START OF ALL OF THE CONSTANT VARIABLES FOR THIS PROGRAM *****/

//Constant Variables for the Game Screen:
final int PUCK_SIZE = 30;
final int BLOCK_WIDTH = 80;
final int BLOCK_HEIGHT = 20;
final int BLOCK_STARTING_POS_X = 500;
final int BLOCK_STARTING_POS_Y = 560;
final int SCORE_PLACEMENT_X = 600;
final int HIGHSCORE_PLACEMENT_X = 800;
final int SCORE_PLACEMENT_Y = 50;
final int BAR_Y = 115;
final int BAR_HEIGHT = 10;
final int NUM_DOTS = 100; //number of dots that try to block the the screen.

//Constant Variables for the Home Screen:
final int HS_BUTTON_X = 300; // for both
final int HS_BUTTON_WIDTH = 400; // "
final int HS_BUTTON_HEIGHT = 150; // "
final int HS_BUTTON_Y_TOP = 550; // for top button only
final int HS_BUTTON_Y_BOTTOM = 750; // for button button only
final int BLANK_BOX_X = 50;
final int BLANK_BOX_Y = 50;
final int BLANK_BOX_WIDTH = 900;
final int BLANK_BOX_HEIGHT = 400;
final int PADDLE_Y = 400;

//Constant Variables for Instructions
final int I_BUTTON_LEFT_X = 50;
final int I_BUTTON_RIGHT_X = 550;
final int I_BUTTON_Y = 700;
final int I_BUTTON_WIDTH = 400;
final int I_BUTTON_HEIGHT = 250;
final int I_TEXT_Y = 60;

//Constant Variables for Game Over Screen
final int FINAL_SCORE_Y = 200;
final int FINAL_HIGHSCORE_Y = 500;
/**** END OF ALL OF THE CONSTANT VARIABLES FOR THIS PROGRAM *****/




/**** START OF ALL OF THE OTHER VARIABLES FOR THIS PROGRAM ***/

int timer = 0; //used to make sure that two buttons are not clicked on in rapid succession

//Variables for the Game Screen
int blockX; //
int blockY;
float puckSpeedX;
float puckSpeedY;
float puckX;
float puckY;
int score;
```

```
int highScore;
//these arrays store information for the dots that block the screen on while playing wall ball
float[] dotsX = new float [NUM_DOTS];
float[] dotsY = new float [NUM_DOTS];
float[] dotsXSpeed = new float [NUM_DOTS];
float[] dotsYSpeed = new float [NUM_DOTS];
float[] dotsSize = new float [NUM_DOTS];

//Variables for the Home Screen
float puckXAnimation;
float puckYAnimation;
float puckXSpeedAnimation;
float puckYSpeedAnimation;
float paddleX;

String screen;
/* stores a string that determines which screen will be shown:
FOR EXAMPLE:
- "gameScreen" will load the game screen
- "homeScreen" will load the home screen
- "gameOverScreen" will load the game over screen
- "instructions" will load the instructions screen
*/
/**** END OF ALL OF THE OTHER VARIABLES FOR THIS PROGRAM *****/



void setup()
{
  size(1000, 1000);
  noStroke();
  puckSpeedX = random(-5, 5);
  puckSpeedY = random(-5, 5);
  blockX = (int)(mouseX - (0.5 * BLOCK_WIDTH));
  blockY = height - BLOCK_HEIGHT - 20;
  puckX = BLOCK_STARTING_POS_X;
  puckY = BLOCK_STARTING_POS_Y;
  score = 0;
  highScore = 0;
  screen = "homeScreen"; //will load the home screen when initially run
  puckXAnimation = ((BLANK_BOX_X + BLANK_BOX_WIDTH) / 2);
  puckYAnimation = ((BLANK_BOX_Y + BLANK_BOX_HEIGHT) / 2);
  puckXSpeedAnimation = random(-10, 10);
  puckYSpeedAnimation = random(-10, 10);
  //initializes all of the values for all of these arrays
  for (int i = 0; i < NUM_DOTS; i++)
  {
    dotsSize[i] = 2;
    dotsX[i] = random(dotsSize[i] * 0.5, width - (dotsSize[i] * 0.5));
    dotsY[i] = random(dotsSize[i] * 0.5 + (BAR_Y + BAR_HEIGHT), height - (dotsSize[i] * 0.5));
    dotsXSpeed[i] = random(-10, 10);
    dotsYSpeed[i] = random(-10, 10);
  }
}

void draw()
{
  /*
    Calling this function runs the game! Whatever string that
    "screen" gets will determine the screen that is drawn
    */
  displayScreen(screen);
}
```

```
123
124 /*
125 This function determines which functions to call based on what the parameter currentScreen gets.
126 The string variable "screen" gets passed into this parameter when this function is called
127 This function also serves to make sure that only one screen is being displayed at a time
128 */
129 void displayScreen(String currentScreen)
130 {
131   background(255);
132   if (currentScreen == "gameScreen")
133   {
134     noCursor();
135     updateGameScreen();
136     renderGameScreen();
137   }
138   else if (currentScreen == "homeScreen")
139   {
140     cursor();
141     updateHomeScreenAnimation();
142     drawHomeScreenAnimation();
143     homeScreenButtons();
144   }
145   else if (currentScreen == "gameOverScreen")
146   {
147     cursor();
148     gameOverButtons();
149     displayFinalScore();
150   }
151   else if (currentScreen == "instructions")
152   {
153     cursor();
154     instructionButtons();
155     displayInstructionsText();
156   }
157 }
158
159 /*This abstraction allows me to efficiently create new
160 buttons and choose what screen they take you to when they are clicked.*/
161 void button(int x, int y, int bWidth, int bHeight, String pickScreen)
162 {
163   if (mouseX > x && mouseX < x + bWidth && mouseY > y && mouseY < y + bHeight)
164   {
165     fill(125);
166     if (mousePressed && timer == 0)
167     {
168     screen = pickScreen;
169     if(pickScreen == "gameScreen")
170     {
171         reset();
172     }
173     timer = 30;
174     }
175   }
176   else
177   {
178     if (timer > 0)
179     {
180     timer--;
181     }
182     fill(0);
183   }
184   rect(x, y, bWidth, bHeight);
185 }
```

```
186
187 //buttons that will be drawn on the home screen
188 void homeScreenButtons()
189 {
190   //Play Game Button
191   button(HS_BUTTON_X, HS_BUTTON_Y_TOP, HS_BUTTON_WIDTH, HS_BUTTON_HEIGHT, "gameScreen");
192
193   //Go to Instructins Button
194   button(HS_BUTTON_X, HS_BUTTON_Y_BOTTOM, HS_BUTTON_WIDTH, HS_BUTTON_HEIGHT, "instructions");
195   fill(255);
196   textAlign(CENTER, CENTER);
197   textFont(createFont("impact", 50));
198   text("Start Game", HS_BUTTON_X + (0.5 * HS_BUTTON_WIDTH), HS_BUTTON_Y_TOP + (0.5 * HS_BUTTON_HEIGHT));
199   text("How to Play", HS_BUTTON_X + (0.5 * HS_BUTTON_WIDTH), HS_BUTTON_Y_BOTTOM + (0.5 * HS_BUTTON_HEIGHT));
200 }
201
202 //buttons that will be drawn on the instruction screen
203 void instructionButtons()
204 {
205   //back button
206   button(I_BUTTON_LEFT_X, I_BUTTON_Y, I_BUTTON_WIDTH, I_BUTTON_HEIGHT, "homeScreen");
207
208   //start playing button
209   button(I_BUTTON_RIGHT_X, I_BUTTON_Y, I_BUTTON_WIDTH, I_BUTTON_HEIGHT, "gameScreen");
210   fill(255);
211   textAlign(CENTER, CENTER);
212   textFont(createFont("impact", 50));
213   text("Back", I_BUTTON_LEFT_X + (0.5 * I_BUTTON_WIDTH), I_BUTTON_Y + (0.5 * I_BUTTON_HEIGHT));
214   text("Start Game", I_BUTTON_RIGHT_X + (0.5 * I_BUTTON_WIDTH), I_BUTTON_Y + (0.5 * I_BUTTON_HEIGHT));
215 }
216
217 //buttons that will be drawn on the game over screen
218 void gameOverButtons()
219 {
220   //back button
221   button(I_BUTTON_LEFT_X, I_BUTTON_Y, I_BUTTON_WIDTH, I_BUTTON_HEIGHT, "homeScreen");
222
223   //start playing button
224   button(I_BUTTON_RIGHT_X, I_BUTTON_Y, I_BUTTON_WIDTH, I_BUTTON_HEIGHT, "gameScreen");
225   fill(255);
226   textAlign(CENTER, CENTER);
227   textFont(createFont("impact", 50));
228   text("Home Screen", I_BUTTON_LEFT_X + (0.5 * I_BUTTON_WIDTH), I_BUTTON_Y + (0.5 * I_BUTTON_HEIGHT));
229   text("Start Game", I_BUTTON_RIGHT_X + (0.5 * I_BUTTON_WIDTH), I_BUTTON_Y + (0.5 * I_BUTTON_HEIGHT));
230 }
231
232
233
234 /********************Start of Home Screen Functions*****************/
235 //draws the title
236 void drawTitle()
237 {
238   fill(100);
239   textAlign(CENTER, CENTER);
240   textFont(createFont("GodOfWar", 150));
241   text("Wall\nBall", width/2, (BLANK_BOX_Y + BLANK_BOX_HEIGHT) / 2);
242 }
243
244 //A function to update all the variables needed to run the animation
245 void updateHomeScreenAnimation()
246 {
247   //moves the puck
248   puckXAnimation += puckXSpeedAnimation;
```

```
  puckYAnimation += puckYSpeedAnimation;
  //makes the puck bounce off walls and paddle
  if (puckXAnimation + PUCK_SIZE > BLANK_BOX_X + BLANK_BOX_WIDTH || puckXAnimation < BLANK_BOX_X)
  {
    puckXSpeedAnimation *= -1;
  }
  if (puckYAnimation < BLANK_BOX_Y || puckYAnimation + PUCK_SIZE > PADDLE_Y)
  {
    puckYSpeedAnimation *= -1;
  }
  //keeps the paddle from going outside the box
  paddleX = (puckXAnimation - (0.5 * BLOCK_WIDTH)) + (0.5 * PUCK_SIZE);
  if (paddleX < BLANK_BOX_X)
  {
    paddleX = BLANK_BOX_X;
  } else if (paddleX + BLOCK_WIDTH > BLANK_BOX_X + BLANK_BOX_WIDTH)
  {
    paddleX = BLANK_BOX_X + BLANK_BOX_WIDTH - BLOCK_WIDTH;
  }
}

//uses the data created from updateHomeScreenAnimation() to draw the animation
void drawHomeScreenAnimation()
{
  //draws the empty box
  noFill();
  stroke(0);
  strokeWeight(10);
  rect(BLANK_BOX_X, BLANK_BOX_Y, BLANK_BOX_WIDTH, BLANK_BOX_HEIGHT);

  //draw puck
  noStroke();
  fill(0);
  rect(puckXAnimation, puckYAnimation, PUCK_SIZE, PUCK_SIZE);

  //draws paddle
  fill(0);
  rect(paddleX, PADDLE_Y, BLOCK_WIDTH, BLOCK_HEIGHT);
  drawTitle();
}
/******************End of the Home Screen Functions*************/




/***********Start of Instruction Screen Functions******************/
//writes out the instructions on the instruction screen
void displayInstructionsText()
{
  fill(0);
  textAlign(CENTER, CENTER);
  textFont(createFont("serif", 50));
  text("How to Play Wall Ball:", width/2, I_TEXT_Y);
  text("1: Use the mouse to control the paddle\nat the bottom of the screen.", width/2, I_TEXT_Y + 100);
  text("2. Use the paddle to hit the puck away from you.", width/2, I_TEXT_Y + 220);
  text("3. As the game goes on, your vision\nwill get start to get fuzzier!", width/2, I_TEXT_Y + 340);
  text("4. You get 1 point every time you hit the puck.", width/2, I_TEXT_Y + 460);
  text("Try to beat your highscore!", width/2, I_TEXT_Y + 560);

}
/************End of Instruction Screen Functions**************/
```

```
312
313
314 /******************Start of the game over screen functions*************/
315 //writes out the score and highscore on the game over screen
316 void displayFinalScore()
317 {
318   fill(0);
319   textAlign(CENTER, CENTER);
320   textFont(createFont("impact", 100));
321   text("Final Score:\n" + score, width/2, FINAL_SCORE_Y);
322   text("High Score:\n" + highScore, width/2, FINAL_HIGHSCORE_Y);
323 }
324 /******************End of the game over screen functions*************/
325
326
327
328
329 /******************Start of the game screen functions*************/
330 /*
331 This function updates every
332 variable needed on the game screen
333 by calling many functions
334 */
335 void updateGameScreen()
336 {
337   movePlayer();
338   bounceDots();
339   updatePuckSpeed();
340   movePuck();
341   bouncePuckOffPaddle();
342   updateHighScore();
343   endGame();
344   updateDots();
345 }
346
347 //the function that will draw everything on the game screen
348 void renderGameScreen()
349 {
350   drawPuck();
351   drawDots();
352   drawPlayer();
353   displayScore();
354   drawBar();
355 }
356
357 /*
358 moves the dots based on the values of dotsXSpeed[] and dotsYSpeed[].
359 Each dot has a corresponding x and y speed in these arrays
360 */
361 void updateDots()
362 {
363   for (int i = 0; i < NUM_DOTS; i++)
364   {
365     dotsX[i] += dotsXSpeed[i];
366     dotsY[i] += dotsYSpeed[i];
367   }
368 }
369
370 //makes the dots bounce off of the edges of the screen
371 void bounceDots()
372 {
373   for (int i = 0; i < NUM_DOTS; i++)
374   {
```

```
    if (dotsX[i] - dotsSize[i] < 0 || dotsX[i] + dotsSize[i] > width)
    {
    dotsXSpeed[i] *= -1;
    }
    if (dotsY[i] - dotsSize[i] < BAR_Y + BAR_HEIGHT || dotsY[i] + dotsSize[i] > height)
    {
    dotsYSpeed[i] *= -1;
    }
  }
}

//displays the dots onto the screen
void drawDots()
{
  for (int i = 0; i < NUM_DOTS; i++)
  {
    fill(30);
    ellipse(dotsX[i], dotsY[i], dotsSize[i], dotsSize[i]);
  }
}

//moves the paddle based on mouseX and keeps the the paddle on the screen
void movePlayer()
{
  blockX = (int)(mouseX - (0.5 * BLOCK_WIDTH));
  if (blockX < 0)
  {
    blockX = 0;
  } else if (blockX > width - BLOCK_WIDTH)
  {
    blockX = width - BLOCK_WIDTH;
  }
}

//displays the paddle
void drawPlayer()
{
  fill(0);
  rect(blockX, blockY, BLOCK_WIDTH, BLOCK_HEIGHT);
}

/*
resets the game by resetting a lot of differnent variables
to their default values. Is only called when a button
that takes a player to the game screen is clicked
*/
void reset()
{
  puckY = BLOCK_STARTING_POS_Y;
  puckX = BLOCK_STARTING_POS_X;
  puckSpeedX = random(-5, 5);
  puckSpeedY = random(-5, 5);
  score = 0;
  for (int i = 0; i < NUM_DOTS; i++)
  {
    dotsSize[i] = 2;
  }

}

//if the puck hits the bottom of the screen
void endGame()
{
```

```
  if (puckY + PUCK_SIZE > height)
  {
    screen = "gameOverScreen";
  }
}



//changes the speed of the puck when it hits a wall
void updatePuckSpeed()
{
  if (puckX > width - PUCK_SIZE || puckX < 0)
  {
    puckSpeedX *= -1;
  }
  if (puckY < BAR_Y + BAR_HEIGHT)
  {
    puckSpeedY *= -1;
  }
}

//makes the puck bounce off of the paddle
void bouncePuckOffPaddle()
{
  if (puckY+PUCK_SIZE>blockY&&puckY<blockY+BLOCK_HEIGHT&&puckX+PUCK_SIZE>blockX&&puckX<blockX+BLOCK_WIDTH)
  {
    puckSpeedY *= -1;
    puckSpeedY--;
    score++;
    for (int i = 0; i < NUM_DOTS; i++)
    {
    dotsSize[i]++;
    }
  }
}

//moves the puck based upon the values of puckSpeedX and puckSpeedY
void movePuck()
{
  puckX += puckSpeedX;
  puckY += puckSpeedY;
}




//if the score exceeds the highscore then change highscore
void updateHighScore()
{
  if (score > highScore)
  {
    highScore = score;
  }
}

//writes the score to the top right hand corner of the screen
void displayScore()
{
  fill(30);
  textAlign(LEFT);
  textFont(createFont("impact", 30));
```

```
  text("Score\n" + score, SCORE_PLACEMENT_X, SCORE_PLACEMENT_Y);
  text("Highscore\n" + highScore, HIGHSCORE_PLACEMENT_X, SCORE_PLACEMENT_Y);
}

//draws a bar that separates the top part of the screen from the bottom part of the screen
void drawBar()
{
  fill(0);
  rect(0, BAR_Y, width, BAR_HEIGHT);
}

//draws the puck using the variables puckX, puckY, which are definied in movePuck()
void drawPuck()
{
  fill(0);
  rect(puckX, puckY, PUCK_SIZE, PUCK_SIZE);
}
/***************End of the Game Screen Functions*****************/


```
