Gabby Melamed

Paddle Game Questions

1. **Provide a written response for your video that:**
    - **identifies the programming language;**
    - **identifies the purpose of your program; and**
    - **explains what the video illustrates.**

      This code is written in p5js, a version of java script. The purpose of my program is for people to have fun playing a game. This video illustrates how the game works. The game starts on the start screen when the player can choose their game level or they can press a button to read the instructions. After the player chooses a game level, the screen will change to the actual game screen. If the player chooses the easy mode, the least number of balls will appear. Id medium mode is chosen, more ball will appear than easy mode. If hard mode is chosen, more balls will appear than medium mode. When a ball hits the top of the paddle, the player's score will increase. If the ball hits the bottom of the paddle, the player's lives will decrease by one, the balls on the screen will be cleared, and more balls will appear. If the player collects all of the balls on the screen, the screen will change to tell the player that they won the game and the player will have the option to play again or quit the game. If the player's lives get to 0, the screen will change to tell the player that they lost the game and they will also have the option to play the game again or quit the game.

2. **Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development.**

3. **Capture and paste the program code segment that implements an algorithm (marked with an oval) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve**

the intended purpose of the program. ***(Approximately 200 words)***

Update in ball.js

In the update function in the Ball class, there is a for loop and several if statements nested inside the loop. The for loop creates a starting value that is a variable, i, equal to the length of the balls array minus 1. The 2nd part of the loop, the condition, says that i has to be greater than or equal to zero. If this is true, the loop will continue. If it becomes false, the loop will end. The third part of the loop decrements i by 1 each time the loop goes through. Next, there is a nested if and an else if statement inside the for loop. The first if asks if the balls are colliding(it knows if they are or are not colliding from the isColliding function) and if the velocity is greater than 0. If both of these statements are true, one ball will be spliced(deleted) from the balls array and the score will increase by 1. If one or both of those conditions is not true the computer will go into the else if statement. This else if statement says that if the balls are colliding and the velocity is less than 0, decrement the lives by 1 and load 20 more balls.

4. **Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program.**

   Button.js class

   The Button class in the button.js file, is an abstraction I developed in this program. This Button class was very important in managing the complexity of the program because a lot of buttons appear throughout the game. The Button class makes it a lot easier to create a button each time because the code to create a button is the same for each button, except for the location and the words. By adding parameters to the buttons, it is easy to change the location and wording of each button each time I wanted to make a specific button appear. Therefore, this abstraction helped the complexity of my code because I could make as many buttons as I wanted very easily by creating a new button and then making it run.

5. **Capture and paste your entire program code in this section.**
   - **Mark with an oval the segment of program code that implements the algorithm and integrates mathematical and /or logical concepts.**
   - **Mark with a rectangle the segment of program code that represents an abstraction you developed.**
   - **Include comments or citations for program code that has been written by someone else.**

ball.js:

```
class Ball {

  //constructs values to pass into the balls
```

```
constructor(x, y, dx, dy){
  this.loc = createVector(x, y);
  this.vel = createVector(dx, dy);
  this.acc = createVector(0,0);
  this.clr = color(random(255), random(255), random(255));
 }

//runs all of the ball functions
 run(){
  this.render();
  this.checkedges();
  this.update();
 }

//makes each ball bounce when it reaches any edge of the canvas
//changes x velocity to the opposite sign when it hits the left or right
//changes y velocity to the opposite sign when it hits the top or bottom
 checkedges(){
  if(this.loc.x < 0){
   this.vel.x = -this.vel.x;
  }
  if(this.loc.x > width){
   this.vel.x = -this.vel.x;
  }
  if(this.loc.y < 0){
   this.vel.y = -this.vel.y;
  }
  if(this.loc.y > height){
   this.vel.y = -this.vel.y;
  }
 }
```

```
//deletes a ball out of the array everytime it hits the paddle coming down only
//increases the score by 1 when the ball hits the paddle
update(){

//makes the velocity have a limit so the balls don't get too fast
    this.vel.limit(7);
    this.vel.add(this.acc);
    this.loc.add(this.vel);

    //if the balls are colliding the paddle at the top, they will disappear
    //if they collide at the bottom, the balls should disappear and more balls will appear
    for(var i = balls.length - 1; i >= 0; i--) {
      if(balls[i].isColliding() && this.vel.y > 0) {
          balls.splice(i, 1);
          score = score + 1;
        }else if(balls[i].isColliding() && this.vel.y < 0) {
          loadBalls(20);
          lives = lives - 1;
        }
}
//keeps velocity the same for the easiest level
   if(gameLevel === 'easy') {
     this.vel.x = this.vel.x * 1;
     this.vel.y = this.vel.y * 1;
   }

   //makes the balls faster than easy level

   if(gameLevel === 'medium') {
     this.vel.x = this.vel.x * 1.1;
     this.vel.y = this.vel.y * 1.1;
   }
```

```
  //makes the balls faster than medium level

  if(gameLevel === 'hard') {
    this.vel.x = this.vel.x * 1.2;
    this.vel.y = this.vel.y * 1.2;
  }
 }

//makes the ball know when it hits the paddle
  isColliding() {
   if(this.loc.x + 13 > paddle.loc.x &&
       this.loc.x - 13 < paddle.loc.x + paddle.w &&
       this.loc.y + 13 > paddle.loc.y &&
       this.loc.y - 13 < paddle.loc.y + paddle.h)
       {
         return true;
       } else {
         return false;
       }
      }

//creates the ball
  render(){
    fill(this.clr);
    ellipse (this.loc.x, this.loc.y, 26, 26);
  }
}//  ++++++++++++++++++++++++++++++++++  End Ball Class
```

button.js:

```
class Button {
```

```
  //constructs values to pass into the buttons
  constructor(x, y, msg){
    this.loc = createVector(x, y);
    this.msg = msg;
    this.clr = color(random(255), random(255), random(255));
  }

  //displays the buttons on the screen
  run() {
    this.render();
  }

  //creates the button with the words in it
  render() {
    fill(this.clr);
    rect(this.loc.x, this.loc.y, 100, 100)
    fill(255, 0, 0);
    textSize(20);
    text(this.msg, this.loc.x, this.loc.y);
  }
}
```

Index.html:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Paddle Game</title>
    <script src="libraries/p5.js" type="text/javascript"></script>
    <script src="libraries/p5.dom.js" type="text/javascript"></script>
    <script src="libraries/p5.sound.js" type="text/javascript"></script>
    <script src="sketch.js" type="text/javascript"></script>
```

```
    <script src="ball.js" type="text/javascript"></script>
    <script src="paddle.js" type="text/javascript"></script>
    <script src="button.js" type="text/javascript"></script>
    <style> body {padding: 0; margin: 0;} canvas {vertical-align: top;} </style>
  </head>



  <body>
  </body>
</html>
```

paddle.js:

```
class Paddle {

  //passes in the values of the paddles
  constructor(x, y, w, h) {
    this.loc = createVector(x, y);
    this.w = w;
    this.h = h;
    //this.loc2 = createVector(w, h);
    this.clr = color(random(255), random(255), random(255));
  }

//shows the paddle on the screen and allows movement
  run() {
    this.render();
    this.update();
  }

//creates the paddle
  render() {
    fill(this.clr);
```

```
    rect(this.loc.x, this.loc.y, this.w, this.h);
  }

//allows the mouse to move the x location of the paddle
  update() {
    var paddleMouseLoc = createVector(mouseX, 700);
    this.loc = p5.Vector.lerp(this.loc, paddleMouseLoc, 0.09);
  }
}
```

sketch.js:

```
var paddle;
var balls = [];
var gameState = 1;
var score = 0;
var gameLevel;
var lives = 5;
var buttonEasy;
var buttonMedium;
var buttonHard;
var buttonPlayAgain;
var buttonEndGame;
var buttonInstructions;
var buttonBack;
var numBallsEasy = 5;
var numBallsMedium = 10;
var numBallsHard = 15;

//create canvas and background
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
```

```
  background(5, 5, 5);
  fill(200, 30, 150);
}

//The draw function is called automatically @ 30 fps
//associates a game screen with a gameState number
//calls a game screen when the gameState = the associated number
function draw() {
  background(5, 5, 5, 50);
  if (gameState === 1){
    startGame();
  } else if(gameState === 2){
    gameMode();
  } else if(gameState === 3){
    endGame();
  }else if(gameState === 4){
    instructions();
  }
}

//start screen of game
//displays buttons with different levels the player can choose from
function startGame() {
  lives = 5;
  clear();
  background(5, 5, 5);
  textSize(90);
  fill(255, 255, 255);
  text('The Paddle Game', 30, 200);

  //displays the buttons that have .run in makeButtons
  makeButtons();
```

```
  //sets game level to the level of the button the player pressed on
  //changes the gameState to gameState = 2 which calls the gameMode function
  if(mouseIsPressed) {
    if(mouseX > 120 &&
      mouseX < 220 &&
      mouseY > 600 &&
      mouseY < 700) {
        gameLevel = 'easy';
        gameState = 2;
      }else if(mouseX > 270 &&
      mouseX < 370 &&
      mouseY > 600 &&
      mouseY < 700) {
        gameLevel = 'medium';
        gameState = 2;
      }else if(mouseX > 420 &&
      mouseX < 520 &&
      mouseY > 600 &&
      mouseY < 700) {
        gameLevel = 'hard';
        gameState = 2;
      }

      //when the player presses on the instructions button, the game screen changes to the
instructions screen
      else if(mouseX > 570 &&
      mouseX < 670 &&
      mouseY > 600 &&
      mouseY < 700) {
        gameState = 4;
      }
```

```
    }

    //loads a different amount of balls depending on the game level that is selected
    if(gameLevel === 'easy'){
      loadBalls(numBallsEasy);
    }
    if(gameLevel==='medium'){
      loadBalls(numBallsMedium);
    }
    if(gameLevel==='hard'){
      loadBalls(numBallsHard);
    }
}

//instruction screen
//displays the game instructions

function instructions() {
  clear();
  background(5, 5, 5);
  fill(255,192,203);
  textSize(30);
  text('INSTRUCTIONS', 290, 50);
  textSize(18);
  text('Welcome to The Paddle Game! Your score and lives are displayed in the top left corner.',
10, 100);
  text('The goal of this game is to collect all of the balls on the screen.', 10, 130);
  text('To collect the balls, move the paddle so that that ball(s) land on the paddle.', 10, 160);
  text('When the ball(s) hit the top of the paddle, they will disappear.', 10, 190);
  text('But, do not let the balls hit the bottom of the paddle.', 10, 220);
  text('Randomely, if they do, the remaining balls on the screen will disappear and more balls
will appear.', 10, 250);
```

```
  text('You will win the game when you have collected all of the balls on the screen', 10, 280);
  text('Everytime new balls appear, you will lose a life.', 10, 310);
  text('You will start with 5 lives and once your lives = 0, you lose the game.', 10, 340);
  text('After you have won or lost the game, you will have the option to quit or play again.', 10,
370);
  text('Good luck!', 50, 400);

  //displays the back button
  //when the back button is pressed, the gameState = 1 again the screen changes back to the
start screen
  buttonBack.run();
  if(mouseIsPressed &&
    mouseX > 570 &&
    mouseX < 670 &&
    mouseY > 450 &&
    mouseY < 550) {
      gameState = 1;
    }
}

//creates all of the buttons
//only runs the buttons that appear in startGame
function makeButtons() {
  buttonEasy = new Button(120, 600, 'EASY');
  buttonMedium = new Button(270, 600, 'MEDIUM');
  buttonHard = new Button(420, 600, 'HARD');
  buttonPlayAgain = new Button(500, 110, 'PLAY AGAIN?');
  buttonEndGame = new Button(200, 110, 'QUIT?');
  buttonInstructions = new Button(570, 600, 'INSTRUCTIONS');
  buttonBack = new Button(570, 450, 'BACK');
  buttonEasy.run();
  buttonMedium.run();
```

```
  buttonHard.run();
  buttonInstructions.run();

}

function gameMode(){

    //displays a score in the game screen
    fill(255, 0, 0);
    textSize(35);
    text('score:' + score, 30, 30);
    text('lives:' + lives, 30, 70);

    //drops the balls down from the top and displays the paddle
    runBalls();

    //when all of the lives have been used, screen switches to gameState = 3

    if(lives === 0) {
      gameState = 3;
    }

    //when all of the balls have been collected for a level, screen switches to gameState = 3
    if (score === balls.length + score) {
      gameState = 3;
    }
  }

//the end screen of the game
function endGame() {
    clear();
    background(5, 5, 5);
```

```
  //displays buttonPlayAgain and buttonEndGame
  buttonPlayAgain.run();
  buttonEndGame.run();

  //if player collected all of the balls, 'YOU WIN' is displayed on the screen
  if(score === balls.length + score)  {
    textSize(100);
    fill(255, 10, 10);
    text('YOU WIN!!!', 130, 500);
  }

  //if player runs out of lives, 'YOU LOST' is displayed on the screen
  if(lives === 0) {
    fill(255, 0, 0);
    textSize(100);
    text('YOU LOST!', 120, 400);
  }

    //if player presses buttonPlayAgain, gameState is reset to one and the start screen will
appear
    if(mouseIsPressed &&
      mouseX > 500 &&
      mouseX < 600 &&
      mouseY > 110 &&
      mouseY < 210) {
      score = 0;
      gameState = 1;
      balls = [ ];
      gameLevel = '';
      }
```

```
      //if player presses buttonEndGame, everything on the screen is erased and the screen
turns white
     if(mouseIsPressed &&
       mouseX > 200 &&
       mouseX < 300 &&
       mouseY > 110 &&
       mouseY < 210) {
         remove();
       }
}

//creates the paddle and the array of balls
function loadBalls(x) {
  paddle = new Paddle(250, 700, 300, 25);
  for(var i = 0; i < x; i++){
    balls[i] = new Ball(random(0, 800), random(0, 100), random(1, 5), random(1, 5));
  }
}

//displays the paddle and the array of balls
function runBalls(x) {
  paddle.run();
  for(var i = 0; i < balls.length; i++){
    balls[i].run();
  }
}
```