Paddle Game Questions

1. Provide a written response for your video that:
    - identifies the programming language;

    The programming language is p5js, which is the fifth version of JavaScript.

    - identifies the purpose of your program; and

    The purpose of my program is to create a paddle ball game in which there are methods to score, win, and lose the game. Additionally, the goal of my game is to clear all balls from the screen before the number of lives reaches zero.

    - explains what the video illustrates.

    My video begins in the start screen where the game title, instructions, and mode are displayed. I then press on easy mode and win the game by clearing all the balls from the screen. Then after pressing the "play again" button on the win screen the game returns to its start screen. The medium mode is then chosen and played until it is won. The win game screen appears. Play again is then pressed again and the game in hard mode is displayed. Losing the game is then showed because balls touch the bottom of the paddle, creating a new array of balls and causing a loss of life. When lives equals zero, the game ends and the losing screen appears. Once quit game is pressed, it goes to a screen that says, "BYE! HOPE YOU ENJOYED!:)"

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)*

```
isColliding() {
    if(this.loc.x + 20 > paddle.loc.x &&
```

```
            this.loc.x - 20 < paddle.loc.x + paddle.w &&
            this.loc.y + 20 > paddle.loc.y &&
            this.loc.y - 20 < paddle.loc.y + paddle.h) //bounce off
    paddle
            {
              return true;
              this.clr = color(random(255), random(255), random(255));
              }


             }
```

This algorithm permits the ball in my game to bounce off the paddle. The if statement in the isColliding function is a conditional statement that, if proven true, will output the action of the ball bouncing off the top of the paddle. Additionally, the algorithm provides the code that makes the ball and paddle change colors during collisions. Without the algorithm inside the function, collision would not occur and the paddleball game would therefore not work, It is as a result of the if statement that the action occurs and only at the specific time it is supposed to.

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.

```
class Paddle {
    constructor(x, y, w, h){  //factors of paddle
        this.loc = createVector(x, y);
        this.w = 300;
        this.h = 50;
        this.clr = color(random(255), random(255), random(255));
    }

    run(){
        this.render();
        this.update();
    }
    render() {
        fill(this.clr);
        rect(this.loc.x, 650, this.w, this.h);  //place paddle near
bottom of screen
    }
    update() {
        var MouseLoc = createVector(mouseX, 650);  //make paddle follow
mouse
        this.loc = p5.Vector.lerp(this.loc, MouseLoc, 0.09);
    }
```

```
}
```

This abstraction helped manage the complexity of my program by acting as a shortcut by allowing me to not have to retype the same code in multiple areas. The paddle class fits all the information concerning the paddle into one abstraction, permitting me to call upon the class whenever needed. It also organizes the program by assembling all the code regarding one aspect of the program into a single location.

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
//Olivia Cordero
//9/11/19
//PaddleBall GAME
//setup function is called upon once when program begins

class Ball {
  constructor(x, y, dx, dy){  //defining aspects of ball
   this.loc = createVector(x, y);
   this.vel = createVector(dx, dy);
   this.acc = createVector(0,0);
   this.clr = color(random(255), random(255), random(255));

  }

  run(){
    this.checkedges();
    this.update();
    this.render();
    this.isColliding();
  }

  checkedges(){
//bounce off edges
    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;
    }

    }
    update(){
         this.vel.limit(20); //how fast balls can go
         this.vel.add(this.acc);
         this.loc.add(this.vel);
        for(var i = balls.length - 1; i>= 0; i--){
          if(balls[i].isColliding() && this.vel.y > 0) {
         balls.splice(i, 1);
         score++;  //when ball touches top of paddle, ball
disappears, +1 point
       }else if(balls[i].isColliding() && this.vel.y < 0){
         loadObjects(10*2);
         lives--;  //when ball touches bottom of paddle, 20 new balls
appear, -1 life
       }
     }
   }

    render(){
        fill(this.clr);
        ellipse(this.loc.x, this.loc.y, 40, 40);  //ball appearances
       }

    isColliding() {
```

```
if(this.loc.x + 20 > paddle.loc.x &&
        this.loc.x - 20 < paddle.loc.x +
paddle.w &&
        this.loc.y + 20 > paddle.loc.y
&&
        this.loc.y - 20 < paddle.loc.y +
paddle.h) //bounce off paddle
        {
          return true;
          this.clr = color(random(255),
random(255), random(255));
         }
```

```
                }
        }
```

```
//  ++++++++++++++++++++++++++++++++++  End Ball Class

sketch.js
//  Olivia Cordero
// Sept 03
//  PaddleBall
//  The setup function function is called once when your program
begins


var paddle;
var balls = []
var score = 0;
var lives = 5;
var gameState = 1;
var button = []
var EasyButton;
var MediumButton;
var HardButton;
var PlayAgainButton;
var QuitButton;
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(5, 5, 5, 10);
  loadObjects(random(0,10)); //array of balls


}

function draw() {
background(255,255,255,10);
  //runObjects();
//  fill(0, 255, 255);
//textSize(30);
//text("Score: " + score, 10, 25);
fill(2, 2, 2); //splash screen code
```

```
if(gameState === 1){  //different screens/modes of games
  startGame();
}else if(gameState === 2){
  playGame();
}else if(gameState === 3){
  endGame();
}else if(gameState === 4){
  winGame();
}else if(gameState === 5){
  quitGame();
}
}


function loadObjects(n) { //make balls and paddle appear
  for(var i = 0; i < n; i++){
    balls[i] = new Ball(random(800),random(300), random(4, 7),
random(4, 7));
  }
    paddle = new Paddle(250, 700, 300, 50);
}

//function loadButtons(n) {
//  button = new Button(this.loc.x, 575, 80, 80);
//}
function startGame(){ //starting splash screen
  clear();
  lives = 5;
  background(100,50,100);
  fill(46,79,200);
  textSize(30);
  text('PADDLEBALL GAME', 260, 230);
  textSize(50);
  text('HIT IT OR QUIT IT!', 176, 300);  //game title
  fill(46,79,148);
  textSize(20);
  text('Instructions: Click on one of the boxes below to choose game
mode.', 120, 350);
  text('As the difficulty level increases, so does the number of
balls.', 120, 370);
  text('Try to keep the balls from touching the bottom of the
paddle.', 120, 390);
  text('When the ball reaches the bottom, the amount of lives will
decrease by one', 120, 410);
  text('And a new array of balls will appear.', 120, 430)
```

```
  text('If the number of lives equals zero, game over!', 120, 450);
  createButtons();
  if(mouseIsPressed &&
      mouseX > 170 &&
      mouseX < 230 &&
      mouseY > 600 &&
      mouseY < 660){
        loadObjects(random(1,5));
        gameState = 2;
        console.log('easy');
}
    if(mouseIsPressed &&  //button pressed, load medium mode
        mouseX > 370 &&
        mouseX < 430 &&
        mouseY > 600 &&
        mouseY < 660){
          loadObjects(random(6,10));
          gameState = 2
          console.log('medium');
        }
      if(mouseIsPressed &&  //button pressed, load hard mode
        mouseX > 570 &&
        mouseX < 630 &&
        mouseY > 600 &&
        mouseY < 660){
          gameState = 2;
          loadObjects(random(11,15));
          console.log('hard');
        }
      }
function createButtons(){  //making buttons with different functions
  EasyButton = new Button(170, 600,'easy');  //appears in start
screen
  MediumButton = new Button(370, 600, 'medium'); //appears in start
screen
  HardButton = new Button(570, 600, 'hard'); //appears in start
screen
  PlayAgainButton = new Button(176, 450, 'new game'); //appears in
end and win game
  QuitButton = new Button(550, 450, 'quit game'); //appears in end
and win game
  EasyButton.run();  //to make buttons appear
  MediumButton.run();
  HardButton.run();
}
```

```
function playGame(){
  fill(0, 255, 255);
textSize(30);
text("Score: " + score, 10, 25); //score in top left corner
text("Lives: " + lives, 10, 60); //number of lives in top left corner
below score
runObjects();
  if(score===balls.length + score){  //win once all balls are cleared
    gameState=4;
  }
  if(lives <= 0){  //lose if lives are less than or equal to zero
    gameState = 3;  //losing screen
  }
  }

function runObjects(){
  paddle.run();
  for(var i = 0; i < balls.length; i++) balls[i].run();
}
function endGame(){ //lose screen
  background(255,50,100,10);
  textSize(50);
  text('OOPS! GAME OVER!', 160, 300);
  PlayAgainButton.run();
  QuitButton.run();
if(mouseIsPressed && //restart game
  mouseX >= 176 &&
  mouseX <= 256 &&
  mouseY >= 450 &&
  mouseY <= 530){
     paddle;  //redefining original variables
     balls = [];
     score = 0;
     lives = 5;
     gameState = 1;
    }
if(mouseIsPressed && //quit game
    mouseX >= 550 &&
    mouseX <= 630 &&
    mouseY >= 450 &&
    mouseY <= 530){
      gameState = 5; //quitting game screen

    }
```

```
  }
  function winGame(){ //win screen
    background(200,30,89,10);
    textSize(50);
    text('CONGRATULATIONS!', 160, 300);
    text('YOU WIN!', 300,350);
    PlayAgainButton.run();
    QuitButton.run();
    if(mouseIsPressed && //restart game
      mouseX >= 176 &&
      mouseX <= 256 &&
      mouseY >= 450 &&
      mouseY <= 530){
         paddle;  //redefining original variables
         balls = [];
         score = 0;
         lives = 5;
         gameState = 1;
        }
      if(mouseIsPressed && //quit game
          mouseX >= 550 &&
          mouseX <= 630 &&
          mouseY >= 450 &&
          mouseY <= 530){
            gameState = 5;
  }
}
function quitGame(){  //quit game screen
  background(200,40,98,0);
  textSize(50);
  text('BYE!', 300, 300);
  text('HOPE YOU ENJOYED!:)', 150, 360);
}
```

Paddle.js

```
// Olivia Cordero
//9/11 PaddleBall
// CollisionDetection
// The setup function is called once when your program begins
```

```
class Paddle {
  constructor(x, y, w, h){  //factors of
paddle
   this.loc = createVector(x, y);
   this.w = 300;
   this.h = 50;
   this.clr = color(random(255), random(255),
random(255));
  }

  run(){
    this.render();
    this.update();
  }
  render() {
    fill(this.clr);
    rect(this.loc.x, 650, this.w, this.h);
//place paddle near bottom of screen
  }
  update() {
    var MouseLoc = createVector(mouseX, 650);
//make paddle follow mouse
    this.loc = p5.Vector.lerp(this.loc,
MouseLoc, 0.09);
}

  }
```

```
//Olivia Cordero
//9.11.19
//PaddleBallGame
//setup function is called once when program begins

class Button {
  constructor(x, y, msg) {
    this.loc = createVector(x, y);
    this.msg = msg;  //text determining function of button
    this.clr = color(20, 100, 130);
  }//determining parts of buttons


run(){
  this.render();
}
```

```
      render() {  //create button with all its info
        fill(this.clr);
        rect(this.loc.x, this.loc.y, 80, 80); // square shaped button
        fill(20, 100, 130);
        textSize(25);
        text(this.msg, this.loc.x, this.loc.y - 20);
      }
      }
```

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>CollisionDetection</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>
```