Ziggy Sheynin

Mr. Ettlin

APCS-Principles, Period 1

23 September 2019

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 program was written in javascript using P5js. The index is written in html. The purpose of this program is to create a game in which the user moves their ouse in order to have balls bounce off of it. They are trying to only bounce the green balls off the paddle which lerps to the mouse. The purpose of the game is entertainment as well as for me to learn about the syntax of javascript.

The video illustrates all the functions of the program. It starts at gameState 1, which is the start screen. Then I use the easy button to change to gameState 2, but the easy version so only five balls appear on the screen, the even ones are green and the odd are red. AS long as my score stays greater than zero, I stay alive until all the green balls are gone. If I get a red ball first or if I get more red balls than green balls, I lose and the gameState changes to 4. GameState 4 is the end screen which has two buttons, restart and back to main menu. The restart button replays the level and the back to the main menu button takes you back to gameState 1, the start screen.

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)*

```
removeBall(){ //If the ball touches the top of the paddle, it is removed
if (this.vel.y > 0 ){
  for (var i = balls.length-1; i >= 0; i--){
    if (balls[i].isColliding()){
      balls.splice(i, 1);
      return true;
    }
  }
}
}
```

In this algorithm, I call the splice algorithm and the isColliding algorithm. This method removes the balls from the screen when they bounce off the top of the paddle. The isColliding algorithm allows the computer to know whether the ball hits the paddle. The splice method is a p5.js method that takes the ball out of the array. This algorithm uses two if statement to see if the ball is moving, and then also if it is colliding with the paddle. It uses a for loop to traverse the entire array of balls to see if any of them are colliding.

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 Button{
  constructor(x, y, w, h, clr){ //constructor for what every button has
    this.loc = createVector(x,y);
    this.w = w;
    this.h = h;
    this.clr = clr;
    //this.txtSize = textSize();
  }

  run(){ //calls both functions in thi class
    this.render();
    this.isClicked();
  }
  render(){ //draws the button on the screen
    fill(this.clr)
    rect(this.loc.x, this.loc.y, this.w, this.h);
    // textSize(this.txtSize);
  }

  isClicked(){ //for all buttons, what to do if they are pressed
    if(mouseIsPressed&&
    mouseX > this.loc.x&&
    mouseX < this.loc.x+this.w &&
    mouseY > this.loc.y&&
    mouseY < this.loc.y +this.h){
      return true;
    }
  }
}//End Button Class
```

This class creates the backbone for all buttons. It shows what they need in order to be buttons. This abstraction made it so that instead of having six lines of code for six different buttons, each button only has 3 lines of code.

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.

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

//Sketch
```
//  Ziggy Sheynin
//Project 916 PaddleGame
////  This is a comment
//  The setup function function is called once when your program
begins

var balls = []; //declares array
var paddle;
var difficulty;
var score =0;
var gameState = 1;
var win;
var btnEasy, btnMed, btnHard, btnInstruction, btnRestart, btnBTMI,
btnBTME;
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(5, 5, 5);
  newButton();
} //end setup

//  The draw function is called @ 30 fps
function draw() {
  background(5,5,5);
  if (gameState ===1){
    startGame(); //start screen
  }else if (gameState === 2){
    playGame(); //game screen
  }
  else if (gameState === 3){
    instructionsText(); //game over screen
    if(btnBTMI.isClicked() === true){
      gameState = 1;
    }
  }else if (gameState === 4){
    endGame(); //game over screen
  }

} //end draw

function newButton(){
  btnEasy = new Button (50, 610 ,200, 100, color(0,255,0));
  btnMed = new Button (300, 610, 200, 100, color(0,204,204));
  btnHard = new Button (550, 610, 200, 100, color(153,0,0));
  btnInstruction = new Button (550, 75, 210, 100,
color(179,179,179));
  btnBTMI = new Button (150, 450, 500, 75, color(255));
  btnReplay = new Button (50, 450, 250, 75, color (255));
  btnBTME = new Button (400, 450, 370, 75, color(255));
}

function playGame(){
  fill (255);
  textSize (40);
  text ("SCORE:" + score, 100 , 50);
  runObjects();
  if (checkRed() === true|| balls.length === 0){
    gameState= 4;
    win = 'yes';
  } else if( score < 0 ){
    gameState = 4;
    win = 'no';
  }
} //end playGame

function startGame(){//easy, medium, hard
  textSize(75);
  fill(255, 255, 240);
  textAlign(CENTER);
  text("Paddle Game", 400, 300); // title text

  btnEasy.render();
  btnMed.render();
  btnHard.render();
  btnInstruction.render();
  fill(0,0,0);
  textSize(40);
  text("Easy", 150, 675);
  text("Medium", 400, 675);
  text("Hard", 650, 675);

  fill(179,179,179); //Instructions button
  fill(0,90,0);
  textSize(40);
  text("Instructions", 655, 145);

  //   //checks if user presses easy, medium or hard button
  checkDifficulty();
  //moves to next splash screen

if(difficulty==='easy'||difficulty==='medium'||difficulty==='hard'||d
ifficulty === 'instructions'|| difficulty=== 'startOver'){
    if(difficulty==='easy'){
      loadObjects(5);
        gameState=2;
    }
    if(difficulty==='medium'){
      loadObjects(10);
        gameState=2;
    }
    if(difficulty==='hard'){
      loadObjects(25);
        gameState=2;
    }

    if(difficulty==='instructions'){
      instructionsText();
      gameState=3;
    }
  }
}//end startGame

  function loadObjects(x){
    paddle = new Paddle (400, 500, 150, 40);
    for(var i = 0; i < x; i++){
      balls[i]=new Ball(random(width), 0 , 4,4, i);
    }
  } //end loadObjects

  function checkDifficulty(){
    //if mouse touches easy
    if(mouseIsPressed&&
        mouseX>50&&
        mouseX<250&&
        mouseY>600&&
        mouseY<700){
          difficulty='easy'
        }
        //if mouse touches medium
    if(mouseIsPressed&&
        mouseX>300&&
        mouseX<500&&
        mouseY>600&&
        mouseY<700){
          difficulty='medium'
        }
        //if mouse touches hard
    if(mouseIsPressed&&
        mouseX>550&&
        mouseX<750&&
        mouseY>600&&
        mouseY<700){
          difficulty='hard'
        }
        if(mouseIsPressed&&
          mouseX>550&&
          mouseX<700&&
          mouseY>50&&
          mouseY<210){
            difficulty='instructions'
          }
}// end checkDifficulty

function endGame(){ //function for game over
if (win === 'yes'){ //if the score is greater than zero, you win
    textSize(80);
    fill (10, 255, 50);
    text ("YOU WIN", 350, 200);
    text ("SCORE:" + score, 350, 325);
  }else if (win === 'no'){ //if you get a red ball first or your
score goes below zero, you lose
    textSize(80);
    fill (255, 10, 10);
    text ("YOU LOSE", 350, 250);
  }
  fill(50, 100, 150) //Restart
  rect(50, 450, 250, 75);
  fill(40, 200, 100);
  textSize(30);
  text("Restart", 175, 500)

  if(mouseIsPressed&&
    mouseX>50&&
    mouseX<300&&
    mouseY>450&&
    mouseY<525){
      gameState = 1;
      clearEverything();
    }
    fill(50, 100, 150) //Back to Main Menu
    btnBTME.render();
    fill(40, 200, 100);
    textSize(20);
    text("Back to Main Menu", 600, 500)
    if (btnBTME.isClicked()){
      clear();
      gameState =1;
      difficulty = 'startOver'
      clearEverything();
}
} //end of endgame

function instructionsText(){

  textSize(20);
  fill(255);
  text("Move the mouse around the screen to move the paddle.", 400,
100);
  text("Try to get only the green balls, if you touch a red ball your
score goes down", 400, 200);
  text("Once you have removed all the green balls, the game is over",
400, 300);
  text("Good Luck!", 400, 400);

  fill(50, 100, 150) //back to main menu button
//  rect(150, 450, 500, 75);
btnBTMI.render();
  fill(40, 200, 100);
  textSize(20);
  text("Back to Main Menu", 400, 500)

  if(btnBTMI.isClicked()=== true){
    gameState = 1;
    clearEverything();
    difficulty = 'clearEverything';
  }
// if(gameState === 3){
//   if(mouseIsPressed&&
//     mouseX>150&&
//     mouseX<650&&
//     mouseY>450&&
//     mouseY<525){
//       clearEverything();
//      }
  //}
} //end of instructionsText

function runObjects(){
paddle.run();
for(var i = 0; i < balls.length; i++){
    balls[i].run();
} //end RunObjects
}

function checkRed(){ //checks to see if there are only red balls left
  var numRed = 0;
  for (var i = 0 ; i < balls.length; i++){
   if (balls[i].getID()% 2===0){
numRed++;     }
  }
  if (balls.length === numRed){
    return true;
  }
}//end checkRed

function clearEverything(){ //returns to origianl state
  gameState =1;
  score = 0;
}
```

//Ball

```
//  Ziggy Sheynin
//Project 916 PaddleGame
////  This is a comment
//  The setup function function is called once when your program
begins

class Ball{

  constructor(x, y, dx, dy, id){//constructor for ball objects
    this.loc = createVector(x, y);
    this.vel = createVector (dx, dy);
    this.acc = createVector (0, .7);
    this.id = id;
  }

run(){
  this.checkEdges();
  this.update();
  this.render();
  this.removeBall();
  this.score();
}

checkEdges(){ //checks the edges of the screen so the ball can do
something when it arrives there
  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(){//makes the ball bounce
this.vel.add(this.acc);
  this.loc.add(this.vel);
}

render(){//makes the balls show up on the screen
if (this.id%2 === 0){ //makes half balls red
    fill (250, 0, 0);
  }else if (this.id%2 === 1){ //half the balls green
    fill (0, 250, 0);
  }
  ellipse(this.loc.x, this.loc.y, 30, 30);
}

isColliding(){ //Is the ball touching the paddle
  if (this.loc.x> paddle.loc.x &&
    this.loc.x < paddle.loc.x +paddle.w &&
  this.loc.y > paddle.loc.y && this.loc.y <
     paddle.loc.y +paddle.h && this.vel.y >0){
    return true;
  } else{
    return false;
  }
```

```
}

removeBall(){ //If the ball touches the top of the paddle, it is
removed
if (this.vel.y > 0 ){
  for (var i = balls.length-1; i >= 0; i--){
    if (balls[i].isColliding()){
      balls.splice(i, 1);
      return true;
    }
  }
}
}

score(){//adds points if green ball is removed and takes them away if
red ball is removed
  if (this.isColliding()===true && this.id %2 === 1){
    score ++;
  }else if (this.isColliding() === true && this.id %2 ===0){
    score--;
  }
}

getID(){
  return this.id;
}
}//  +++++++++++++++++++++++++++++++++++  End Ball Class

//Paddle
//  Ziggy Sheynin
//Project 916 PaddleGame
////  This is a comment
//  The setup function function is called once when your program
begins

class Paddle { //constructor to create paddle
  constructor(x, y, w, h){
    this.loc = createVector(x, y);
    this.w = w;
    this.h= h;
    this.clr = color(random(255), random(255), random(255));
  }

  run(){
    this.update();
    this.render();
}
update(){ //paddle follows mouse
  var mouseLoc = createVector (mouseX, 600);
  this.loc = p5.Vector.lerp(this.loc, mouseLoc, .09); //lerp function
allows paddle to follow mouse
}

render(){ //makes paddle show up on screen
  fill(this.clr);
  rect(this.loc.x, this.loc.y, this.w, this.h);
}
} //*** end class
```

```
//Button
//  Ziggy Sheynin
//Project 916 PaddleGame
////  This is a comment
//  The setup function function is called once when your program
begins

class Button{
  constructor(x, y, w, h, clr){ //constructor for what every button
has
    this.loc = createVector(x,y);
    this.w = w;
    this.h = h;
    this.clr = clr;
    //this.txtSize = textSize();
  }

  run(){ //calls both functions in thi class
    this.render();
    this.isClicked();
  }
  render(){ //draws the button on the screen
    fill(this.clr)
    rect(this.loc.x, this.loc.y, this.w, this.h);
    // textSize(this.txtSize);
  }

  isClicked(){ //for all buttons, what to do if they are pressed
    if(mouseIsPressed&&
    mouseX > this.loc.x&&
    mouseX < this.loc.x+this.w &&
    mouseY > this.loc.y&&
    mouseY < this.loc.y +this.h){
      return true;
    }
  }
}//End Button Class
```