Ziggy Sheynin

Mr. Ettlin

AP CSP, Period 1

6 March 2020

Practice Create Task Questions- Ecosystem

2a. Provide a written response or audio narration in your video that:

- ***Identifies the programming language***
- ***Identifies the purpose of your program and;***
- ***Explains what the video illustrates. (Must not exceed 150 words)***

My code is written in javascript, using the p5js libraries. I wrote the code in atom. The purpose of my program is to provide joy to those who wish to see art and witness the interaction of multiple objects. My video demonstrates the running of my program, demonstrating the functionality of all the buttons. The video starts with the first splash screen that shows four buttons and a title. The "Ecosystem" button has all of the objects, balls, ships, and squares. While each of the other buttons has only the main balls and the selected shape. When the screen goes to the ecosystem, there is a small grey button in the top left corner that takes the used back to the menu.

Word Count: 122

2b. 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. (Must not exceed 200 words)

I began with a program that created two main balls and other balls that were attracted or repulsed by one of the main balls based on their distance from it. I planned to add triangles and squares. I did this by adding two more classes for ships and squares, creating two abstractions enabling me to create multiple objects with fewer lines of code. Throughout this program, there were opportunities to further enhance the code. When creating the squares class, I realized that the squares and triangles should have an interaction rather than just between them and the balls. I created a main ship and only if the squares got between a short distance would they be attracted to the ship. Additionally, I decided there should be multiple options to choose from for what shapes the user would like. I had trouble getting the buttons to line up with the text in them as well as getting them to work. In order to fix these problems, I worked by myself to get the text to match up by trial and error and utilized the help of a peer to review how buttons worked to change the screen.

Word Count: 196

2c. Capture and paste a program code segment that implements an algorithm (marked with an oval in section 3 below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate 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. (Must not exceed 200 words)

```
function runEcosystem(){ //function to continue the running of
each gamestate
```

```
  if (difficulty === 'balls' || difficulty === 'ships'||
difficulty === 'squares' || difficulty === 'ecosystem'){
//checks to see which button is clicked
    if (difficulty === 'balls'){ //runs the main balls and
orbiters
      runBalls(50); //makes 50 orbiter objects
      runMainBalls();
    }else if (difficulty === 'ships'){ //runs main balls and
ships
      runShips(50);
      runMainBalls();
    }else if (difficulty === 'squares'){ //runs main balls and
squares
      runSquares(50);
      runMainBalls();
    }else if (difficulty === 'ecosystem'){ //runs all of the
possible objects
      runBalls(50);
      runShips(50);
      runSquares(50);
      runMainBalls();
    }
  }
```

```
  btnBTM.render(); //renders the grey menu button
  if (btnBTM.isClicked()=== true){ //if the menu button is
clicked, go back to the menu
   difficulty = 'startOver'; //sets a new variable for the
button
   clearEverything(); //sets the gameState back to the home
screen
 }
}//end runEcosystem
```

This function serves to establish what the program should run if a specific button is pressed. The first part of the algorithm is four if-else statements indicating which button is clicked on. The conditionals tell the computer whether the ecosystem button, ball button, ships button, or square button was pressed. If a specific button is pressed, the program always calls the function mainBalls to create two balls that have different properties according to the object that is run. For the individual shapes, if the button is pressed, the specific function that runs those objects is called. I created this algorithm on my own.

Word Count: 103

2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a rectangle in section 3 below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (Must not exceed 200 words)

```
class Square {
  constructor(x, y, dx, dy, id){
```

```
 this.loc = createVector(x, y);
 this.vel = createVector(dx, dy);
 this.acc = createVector(0,0);
 this.clr = color(random(255), random(255), random(255));
 this.id = id;
}
run(){
  this.checkedges();
  this.update();
  this.render();
}
checkedges(){
  if(this.loc.x < 0){
    this.loc.x = width;
  }
  if(this.loc.x > width){
    this.loc.x = 0;
  }
  if(this.loc.y < 0){
    this.loc.y =height;
  }
  if(this.loc.y > height){
  this.loc.y = 0;
```

```
  }
}
update(){
  var distToMainBall;
  var distToMainShip;
  if(this.id > 2){
   distToMainBall = this.loc.dist(mainBall.loc);
   if(distToMainBall < 800){
     //add attraction
     this.acc = p5.Vector.sub(mainBall2.loc, this.loc);
     this.acc.normalize();
     this.acc.mult(0.3);
   }
   if(distToMainBall < 70){ // add repulsion
     this.acc = p5.Vector.sub(this.loc, mainBall.loc);
     this.acc.normalize();
     this.acc.mult(0.3);
   }
 }
   if(this.id > 3){
    distToMainShip = this.loc.dist(mainShip.loc);
      if(distToMainShip < 30){
        //add attraction
```

```
            this.acc = p5.Vector.sub(mainShip.loc, this.loc);
            this.acc.normalize();
            this.acc.mult(0.3);
          }
      }
      this.vel.limit(3);
      this.vel.add(this.acc);
      this.loc.add(this.vel);
    }
    render(){
      push();
      translate (this.loc.x, this.loc.y);
      //rotate (this.heading +1);
      rect(10, 10, 10, 10);
      pop();
    }
}//  ++++++++++++++++++++++++++++++++++  End ship Class
```

This code demonstrates the square class. By creating a square class, I allowed myself to write fewer lines of code in order to create more squares. If I had not abstracted the square class out, I would have had to write at least 4 lines of code for each square, and if I wanted 50 squares, I would have had to write 200 lines of code. By creating a square class I gave each square its properties and only had to create a for-loop to create any given number of squares. This abstraction utilizes the acceleration and velocity functions already in the p5js libraries. It

demonstrates logical concepts through the conditions of the squares relationship to the mainBalls.

Word Count: 118

3. Program Code Capture and paste your entire program code in this section.

➢ Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms 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 acknowledgments for program code that has been written by someone else.

INDEX:

```
//Ziggy Sheynin
//Practice Create Task
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Practice Create Task</title>
    <title>Vectors</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="ship.js" type="text/javascript"></script>
    <script src="squares.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:

```
var ships = [];//declares ships array
var balls = []; //declares array
var squares = []; //declares squares array
var mainBall, mainBall2;
var btnShips, btnSquares, btnBalls, btnAll, btnBTM;
var gameState = 1;
var difficulty;
```

```
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(5, 5, 5, 20);
   loadBalls(50);
   loadShips(50);
   loadSquares(50);
   loadMainBalls(2);
   newButton();
}



function draw() {
background(5,5,5, 20);
  if (gameState ===1){
    startEcosystem();
  }else if (gameState === 2){
    runEcosystem();
  }else if (gameState === 3){
    endSystem();
  }
} //end draw
```

```
function newButton(){ //declares location and color of all the
buttons
  btnBalls = new Button(50, 450, 220, 200, color(54, 191, 136));
  btnShips = new Button(300, 450, 220, 200, color (54, 132,
191));
  btnSquares = new Button(550, 450, 220, 200, color(173, 54,
191));
  btnAll = new Button(300, 200, 220, 200, color(191, 54, 111));
  btnBTM = new Button(10, 10, 50, 50, color(200));
}

function startEcosystem(){
  background(235, 64, 52);
  btnAll.render();
  btnShips.render();
  btnBalls.render();
  btnSquares.render();
  textSize(100); //Title  text
  fill(5);
  text("Ecosystem", 170, 150);
  textSize(45);
  fill(5);
  text ("Balls", 55, 525, 200, 200);
```

```
  text ("Squares", 560, 525, 200, 200);
  text ("Ships", 305, 530, 200, 200);
  text ("Ecosystem", 300, 250, 200, 200);
  checkButton(); // checks which difficulty is chosen
  //I got help from Elena Campell on this code
  if (difficulty === 'balls' || difficulty === 'ships'||
difficulty === 'squares' || difficulty === 'ecosystem'){
    if (difficulty === 'balls'){
      runBalls(50);
      runMainBalls();
    }else if (difficulty === 'ships'){
      runShips(50);
      runMainBalls();
    }else if (difficulty === 'squares'){
      runSquares(50);
      runMainBalls();
    }else if (difficulty === 'ecosystem'){
      runBalls(50);
      runShips(50);
      runSquares(50);
      runMainBalls();
    }
    gameState = 2; // play game
```

```
  }
}

function checkButton(){ //check which button is isClicked
  //I got help from Elena Campell on this code
  if (btnBalls.isClicked()=== true){
     difficulty = 'balls';
   }
 if (btnShips.isClicked()===true){
    difficulty = 'ships';
  }
   if (btnSquares.isClicked()=== true){
    difficulty = 'squares';
  }
 if (btnAll.isClicked()=== true){
  difficulty = 'ecosystem';
}
if (btnBTM.isClicked()=== true){
 difficulty = 'startOver';
}
}// end checkDifficulty
```

```
function runEcosystem(){ //function to continue the running o
each gamestate
  //I got help from Elena Campell on this code
  if (difficulty === 'balls' || difficulty === 'ships'||
difficulty === 'squares' || difficulty === 'ecosystem'){
//checks to see which button is clicked
    if (difficulty === 'balls'){ //runs the main balls and
orbiters
      runBalls(50); //makes 50 orbiter objects
      runMainBalls();
    }else if (difficulty === 'ships'){ //runs main balls and
ships
      runShips(50);
      runMainBalls();
    }else if (difficulty === 'squares'){ //runs main balls an
sq ares
      nSquares(50);
      runMai lls();
    }else if (difficulty      stem'){ //runs all of the
possible objects
      runBalls(50);
      runShips(50);
      runSquares(50);
```

```
      runMainBalls();
    }
  }
  btnBTM.render(); //renders the grey menu button
  if (btnBTM.isClicked()=== true){ //if the menu button is
clicked, gor back to the menu
   difficulty = 'startOver'; //sets a new variable for the
button
   clearEverything(); //sets the gameState back to the hom
creen
 }
}//end f aEcosystem


function loadMainBalls(){
  mainBall = new Ball(random(width/2), random(height/2), random
(-.4,.4), random(-.4,.4), -1);
  mainBall2 = new Ball(random(width/2), random(height/2), random
(-.4,.4), random(-.4,.4), 0);
}


function loadBalls(x){
  for(var i = 0; i < x; i++){
```

```
    balls[i]=new Ball(random(width), random(height), random
(-1,1), random(-1,1), i+2);
  }
}//end loadBalls

function runMainBalls(){
  mainBall.run();
  mainBall2.run();
}
function runBalls(){
  for(var i = 0; i < balls.length; i++){
    balls[i].run();
  }
}//end runBalls

function loadShips(y){
   mainShip = new Ship(random(width/2), random(height/2),
random(-0.4,.4), random(-.4,.4), -5)
  for(var i = 0; i < y; i++){
    if(y % 2 == 0){
    ships[i]=new Ship(random(width), random(height), random
(-1,1), random(-1,1), i+2);
  } else {
```

```
    ships[i]=new Ship(random(width), random(height), random
(-0.5,0.5), random(-0.5,0.5), i+4);
  }
  }
}//end loadShips

function runShips(){
  for (var i = 0; i < ships.length; i++){
    ships[i].run();
  }
}//end runShips

function loadSquares(n){
  for(var i = 0; i < n; i++){
    squares[i]=new Square(random(width), random(height), random
(-1,1), random(-1,1), i+6);
  }
}//end function loadSquares

function runSquares(){
  for(var i = 0; i < squares.length; i++){
    squares[i].run();
  }
```

```
}//end function runSquares

function clearEverything() { //clear gamestate and score for

restarting level

  gameState = 1;

}//end sketch
```

Ball:

```
class Ball {

  constructor(x, y, dx, dy, id){

   this.loc = createVector(x, y);

   this.vel = createVector(dx, dy);

   this.acc = createVector(0,0);

   this.clr = color(random(255), random(255), random(255));

   this.id = id;

  }

  run(){

    this.checkedges();

    this.update();

    this.render();

  }

  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;
    this.loc.y = height -2;
  }
}
update(){
  var distToMainBall;
  if(this.id > 0){
   distToMainBall = this.loc.dist(mainBall.loc);
   if(distToMainBall < 250){
     //add attraction
     this.acc = p5.Vector.sub(mainBall.loc, this.loc);
     this.acc.normalize();
     this.acc.mult(0.1);
   }
   if(distToMainBall < 150){//
     this.acc = p5.Vector.sub(this.loc, mainBall.loc);
```

```
      this.acc.normalize();
      this.acc.mult(0.5);
    }
   }
   var distToMainBall2;
   if(this.id > 0){
    distToMainBall2 = this.loc.dist(mainBall2.loc);
    if(distToMainBall2 < 250){
      //add attraction
      this.acc = p5.Vector.sub(mainBall2.loc, this.loc);
      this.acc.normalize();
      this.acc.mult(0.1);
    }
    if(distToMainBall2 < 150){//
      this.acc = p5.Vector.sub(this.loc, mainBall2.loc);
      this.acc.normalize();
      this.acc.mult(0.5);
    }
   }
   this.vel.limit(5);
    this.vel.add(this.acc);
 this.loc.add(this.vel);
}
```

```
  render(){
    fill(this.clr);
    if (this.id == -1 || this.id == 0){
      ellipse (this.loc.x, this.loc.y, 40, 40);
    }else {
    ellipse(this.loc.x, this.loc.y, 15, 15);
  }
  }
}//  ++++++++++++++++++++++++++++++++++  End Ball Class
```

Ships:

```
class Ship {
  constructor(x, y, dx, dy, id){
   this.loc = createVector(x, y);
   this.vel = createVector(dx, dy);
   this.acc = createVector(0,0);
   this.angle = 0;
   this.clr = color(random(255), random(255), random(255));
   this.id = id;
  }
  run(){
    this.checkedges();
    this.update();
    this.render();
```

```
}
checkedges(){
  if(this.loc.x < 0){
    this.loc.x = width;
  }
  if(this.loc.x > width){
    this.loc.x = 0;
  }
  if(this.loc.y < 0){
    this.loc.y =height;
  }
  if(this.loc.y > height){
  this.loc.y = 0;
  }
}
update(){
  var distToMainBall;
  if(this.id > 2){
   distToMainBall = this.loc.dist(mainBall2.loc);
   if(distToMainBall < 800){
     //add attraction
     this.acc = p5.Vector.sub(mainBall.loc, this.loc);
     this.acc.normalize();
```

```
        this.acc.mult(0.3);
      }
      if(distToMainBall < 80){ // add repulsion
        this.acc = p5.Vector.sub(this.loc, mainBall.loc);
        this.acc.normalize();
        this.acc.mult(0.3);
      }
    }
    this.vel.limit(3);
    this.vel.add(this.acc);
    this.loc.add(this.vel);
  }
  render(){
    this.heading = this.vel.heading();
    fill(this.clr);
    this.angle = this.angle +1;
    push();
    translate (this.loc.x, this.loc.y);
    rotate (this.heading +1);
    triangle (-5, 8, 5,8,0, -8);
    pop();
  }
}//  ++++++++++++++++++++++++++++++++++  End ship Class
```

Squares:

```
class Square {
  constructor(x, y, dx, dy, id){
   this.loc = createVector(x, y);
   this.vel = createVector(dx, dy);
   this.acc = createVector(0,0);
   this.clr = color(random(255), random(255), random(255));
   this.id = id;
  }
  run(){
    this.checkedges();
    this.update();
    this.render();
  }
  checkedges(){
    if(this.loc.x < 0){
      this.loc.x = width;
    }
    if(this.loc.x > width){
      this.loc.x = 0;
    }
    if(this.loc.y < 0){
      this.loc.y =height;
```

```
  }
  if(this.loc.y > height){
  this.loc.y = 0;
  }
}
update(){
  var distToMainBall;
  var distToMainShip;
  if(this.id > 2){
   distToMainBall = this.loc.dist(mainBall.loc);
   if(distToMainBall < 800){
     //add attraction
     this.acc = p5.Vector.sub(mainBall2.loc, this.loc);
     this.acc.normalize();
     this.acc.mult(0.3);
   }
   if(distToMainBall < 70){ // add repulsion
     this.acc = p5.Vector.sub(this.loc, mainBall.loc);
     this.acc.normalize();
     this.acc.mult(0.3);
   }
 }
   if(this.id > 3){
```

```
      distToMainShip = this.loc.dist(mainShip.loc);
        if(distToMainShip < 30){
          //add attraction
          this.acc = p5.Vector.sub(mainShip.loc, this.loc);
          this.acc.normalize();
          this.acc.mult(0.3);
        }
    }
    this.vel.limit(3);
    this.vel.add(this.acc);
    this.loc.add(this.vel);
  }
  render(){
    push();
    translate (this.loc.x, this.loc.y);
    //rotate (this.heading +1);
    rect(10, 10, 10, 10);
    pop();
  }
}//  ++++++++++++++++++++++++++++++++++  End Square Class
```