Ziggy Sheynin

Mr. Ettlin

AP Principles, Period 1

20 November 2019

Snake Game- Create Task

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 uses the arrow keys to move a snake around to eat food. As the snake eats food it grows in length. The player loses when the snake goes back on itself and gets tangled. My game has three different levels and three different themes; it has easy, medium, and hard and the themes are sea, forest, and garden. The purpose of the game is to entertain and give people a way to waste time.

The video illustrates a run of the program. In the video, the player plays the game, loses and switches themes. There are buttons that the person can click to go to the main menu or replay when the person loses by getting tangled. The colors change if the player clicks a different theme.

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.

I created this program collaboratively with Elena. I started with basic steps given to me by Mr. Ettlin to create three different classes, snake, sketch, and food. I started with the food class and my first goal was to get a specific number of food squares to pop up on the screen. During that process, I had to link the food class to the index and the sketch. At first, the food showed up overlapping and not in a specific grid. In order to improve the food class, I worked with Elena to make the food show up on a grid and spread out more using the random function with specific parameters. After I got the food to show up on a grid, I started to work on the snake. At first, I tried to just make the head of the snake to show up. After I succeeded, I used the p5 js reference to figure out to move the head using the arrow keys. Through some trial and error, I got the head to move when the arrow keys were pressed. At first, the snake just moved up and had a long tail. I removed the tail by redrawing the background when the arrow keys were pressed and wrote conditional if statements to make the rest of the arrow keys work. My final step in the program was creating different themes. This I did independently. I created a new splash screen with three different buttons for each of the levels. The first problem I encountered was that clicking the button for a new theme overlapped the location that chose a new level and would skip over the screen to choose a level. In order to fix this problem, I moved the theme buttons to a new location. Each button has its specific if statement in the code and depending on what theme is clicked, the food, snake, and background have a different color.

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)

```
function loadObjects(n){
  if(type === 'garden'){
  snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(227, 69, 7));
    for (var j = 0; j < n; j++){
      food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(70));
      }
    }else if (type === 'forest'){
      snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(20));
        for (var j = 0; j < n; j++){
          food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(202, 237, 0));
          }
    } else if(type === 'sea'){
      snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(162, 0, 255));
        for (var j = 0; j < n; j++){
          food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(0, 255, 229));
          }
    }
  }
```

In this algorithm, I call the Math.floor function and the Math.random function. These two functions work to choose a random integer between 26 and 800. The function loadObjects works to put all of the food and the snake on the screen. In this specific algorithm, I have conditional statements to determine the theme that was selected in order to determine which colors to show. Because the food object are in an array, they require a for loop to be printed. The number of foods is determined by the parameter when the loadObjects method is called.

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 Food{
constructor (x, y, clr){
  this.loc = createVector(x,y);
  this.clr = clr;
}//end contructor
run(){
  this.render();
  this.update();
}
render(){
    fill(this.clr);
    rect(this.loc.x, this.loc.y, 30, 30);
}
update(){
  if(snake.head.x === this.loc.x &&
   snake.head.y === this.loc.y){
  this.loc.x = Math.floor(random(0,79))*30;
  this.loc.y = Math.floor(random(0,79))*30;
}
}
}// end food class
```

This class creates the backbone for all food squares. This allowed me to write two lines of code to create every food object instead of six for every piece of food. It also allowed me to have a designated location to deal with the functions that involved the food instead of having it all in sketch.

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.

```
<!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="snake.js" type="text/javascript"></script>

    <script src="food.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>
```

```
class Button{

constructor(x, y, w, h, clr){

    this.loc = createVector(x,y);

    this.w= w;

    this.h = h;

    this.clr = clr;

  } //end constructor

render(){ // creates button

  fill(this.clr);

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

  }



isClicked(){ // to check if that button is 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

class Food{



constructor (x, y, clr){
  this.loc = createVector(x,y);

  this.clr = clr;

}


run(){

  this.render();

  this.update();

}


render(){

    fill(this.clr);

    rect(this.loc.x, this.loc.y, 30, 30);

}


update(){

  if(snake.head.x === this.loc.x &&

   snake.head.y === this.loc.y){

  this.loc.x = Math.floor(random(0,79))*30;

  this.loc.y = Math.floor(random(0,79))*30;

}

}

}// end food class
```

```
var score, header_height, snake, difficulty, type;

var gameState = 5;

var h = 10;

var food = [];

var body = [];

var btnEasy, btnMed, btnHard, btnInstructions, btnBTMI, btnBTME,
btnReplay, btnSea, btnForest, btnGarden;

function setup() {

  var cnv = createCanvas(800, 800);

  cnv.position((windowWidth-width)/2, 30);

  background(100, 200, 100);

  header_height = 800;

  score = 0;

  loadObjects(2);

  newButton();

}



function draw(){

    if (gameState ===1){

      startGame2(); //start screen

    }else if (gameState === 2){

      playGame(); //game screen
    }else if (gameState === 3){
      instructionsText();
    }else if (gameState === 4){ //game over screen
      endGame();
  } else if(gameState === 5){
    pickSnakeType();
  }
}


function newButton(){
  btnEasy = new Button(50, 450, 200, 200, color(78, 219, 18));
  btnMed = new Button(300, 450, 200, 200, color (250,250,7));
  btnHard = new Button(550, 450, 200, 200, color(250, 0, 0));
  btnBTME = new Button(550, 450, 200, 200, color(200));
  btnReplay = new Button(50, 450, 200, 200, color(100));
  btnSea = new Button(50, 200, 200, 200, color(0, 0, 255));
  btnForest = new Button(300, 200, 200, 200, color (23, 200, 100));
  btnGarden = new Button(550, 200, 200, 200, color(15, 71, 38));
}


function pickSnakeType(){
```

```
background(73, 50, 173);

btnSea.render(); //draws buttons

btnForest.render();

btnGarden.render();



textSize(100);

fill(5);

text("Snake Game", 50, 150);



textSize (45); //text for buttons

fill(255);

text ("Sea", 70, 250, 200, 200);

text ("Forest", 575, 250, 200, 200);

text ("Garden", 320, 250, 200, 200);



 pickSnake(); // checks which difficulty is chosen

if (type === 'sea' || type === 'forest'|| type === 'garden'){

  if (type === 'sea'){

    // background(40, 150, 200);

    startGame1();

    gameState = 1;

  }else if (type === 'forest'){
      startGame1();

      gameState = 1;

    }else if (type === 'garden'){

      startGame1();

      gameState = 1;

    }

     // play game

  }

}// end Pick Snake



function startGame1(){

//change look of this

  textSize(80);


  background(100, 200, 100);

  fill(121, 76, 222);

  textAlign(RIGHT);

  textFont('Times New Roman')

  text ("Snake Game", 600, 200); //title

  textAlign(CENTER);
  btnEasy.render(); //draws buttons

  btnMed.render();

  btnHard.render();



  textSize (45); //text for buttons

  fill(255);

  text ("EASY", 55, 525, 200, 200);

  text ("HARD", 560, 525, 200, 200);

  text ("MEDIUM", 305, 530, 200, 200);



    checkDifficulty(); // checks which difficulty is chosen

    if (difficulty === 'easy' || difficulty === 'medium'|| difficulty
=== 'hard'){

      if (difficulty === 'easy'){

        loadObjects(7);

      }else if (difficulty === 'medium'){

        loadObjects (5);

      }else if (difficulty === 'hard'){

        loadObjects (2);

      }

      gameState = 2; // play game

    }

}
```

```
function startGame2(){

//change look of this

  textSize(80);



  background(100, 200, 100);

  fill(121, 76, 222);

  textAlign(RIGHT);

  textFont('Times New Roman')

  text ("Snake Game", 600, 200); //title

  textAlign(CENTER);



  btnEasy.render(); //draws buttons

  btnMed.render();

  btnHard.render();



  textSize (45); //text for buttons

  fill(255);

  text ("EASY", 55, 525, 200, 200);

  text ("HARD", 560, 525, 200, 200);

  text ("MEDIUM", 305, 530, 200, 200);
```

```
    checkDifficulty(); // checks which difficulty is chosen

    if (difficulty === 'easy' || difficulty === 'medium'|| difficulty
=== 'hard'){

      if (difficulty === 'easy'){

        loadObjects(7);

      }else if (difficulty === 'medium'){

        loadObjects (5);

      }else if (difficulty === 'hard'){

        loadObjects (2);

      }

      gameState = 2; // play game

    }

}



function playGame(){

  frameRate(10);

  if(type === 'garden'){

    background(100, 200, 100);

  } else if (type === 'forest'){

    background(19, 97, 50);

  }else if(type === 'sea'){

    background(11, 75, 179);

  }
```

```
   runObjects();

   text ("Score: " + score, 100, 50); //score

   checkTangled();

 }


function endGame(){

background(255,21,21);

 fill(5);

 textSize(100);

 text("GAME OVER!", 400, 300);

 textSize(45);


 btnBTME.render();

 btnReplay.render();

 fill(20)

 text("Menu",560, 525, 200, 200);

 text("Replay", 55, 525, 200, 200);

 if (btnBTME.isClicked()){ // go back to main menu

 //  gameState = 1;

   difficulty = 'startOver';


   clearEverything();
 }

 if (btnReplay.isClicked()=== true){ // replay level

   clearEverything();

   }

 }




function loadObjects(n){

  if(type === 'garden'){

  snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(227, 69, 7));

    for (var j = 0; j < n; j++){

      food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(70));

      }

    }else if (type === 'forest'){

      snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(20));

        for (var j = 0; j < n; j++){

          food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(202, 237, 0));

          }
    } else if(type === 'sea'){

      snake = new Snake
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,30,
color(162, 0, 255));

        for (var j = 0; j < n; j++){

          food[j] = new Food
(Math.floor(Math.random()*26)*30,Math.floor(Math.random()*26)*30,
color(0, 255, 229));

          }

    }

  }



function runObjects(){

    snake.run();



  for(var i = 0; i< food.length; i++){

    food[i].run();

  }

}



function checkTangled(){

  if(snake.tangled() === true){

    gameState  = 4;

  }
```

```
}



function checkDifficulty(){ //check which difficulty button is
isClicked

  if (btnEasy.isClicked()=== true){

     difficulty = 'easy';

   }

 if (btnMed.isClicked()===true){

    difficulty = 'medium';

  } if (btnHard.isClicked()=== true){

    difficulty = 'hard';

  }

}

function pickSnake(){ //check which difficulty button is isClicked

  if (btnSea.isClicked()=== true){

     type = 'sea';

   }

 if (btnForest.isClicked()===true){

    type = 'forest';

  } if (btnGarden.isClicked()=== true){

    type = 'garden';

  }

}
```

```
function clearEverything() { //clear gamestate and score for
restarting level

  gameState = 1;

  // startGame();

  score = 0 ;

  timerValue = 10;

  food = [];

}

class Snake{

  constructor(x, y, w, c){

    this.head =createVector(x,y);

    this.vel = createVector(0,0);

    this.w = 30;

    this.clr = c;

    this.body = [];



  }// end constructor



  run(){

    this.update();

    this.render();

  } //end run
  update(){

    this.keyPressed();

   //  this.checkEdges();

     for(var i = 0; i< food.length; i++){

     if(this.head.x === food[i].loc.x &&

        this.head.y === food[i].loc.y){

          this.loadSegment();

          if (type === 'garden'){

          food.push(new Food
(Math.floor(Math.random()*25)*30,Math.floor(Math.random()*25)*30,
color(70)));

          score++;

        }else if (type === 'forest'){

          food.push(new Food
(Math.floor(Math.random()*25)*30,Math.floor(Math.random()*25)*30,
color(202, 237, 0)));

          score++;

        }else if(type === 'sea'){

          food.push(new Food
(Math.floor(Math.random()*25)*30,Math.floor(Math.random()*25)*30,
color(0, 255, 229)));

          score++;

        }

      }

    }
```

```
     // update the body

     for (i = this.body.length-1; i>=0; i--){

     if (i >= 1){

       this.body[i].x = this.body[i-1].x;

       this.body[i].y = this.body[i-1].y;

    }  if (i === 0){

       this.body[i].x = this.head.x;

       this.body[i].y = this.head.y;

     }

  }

    // update the head

  //this.head.add(this.vel);


}//end update


  render(){

 // render head

    fill(this.clr);

    rect(this.head.x, this.head.y, this.w, this.w);

 // render the body

    for(var i = 0; i < this.body.length; i++){

      rect(this.body[i].x, this.body[i].y, 30, 30);
    }

  }



   loadSegment(){

     this.body.push(createVector(this.head.x, this.head.y));

   }



   tangled(){

     //for loop checking each segment in the segment array

   for(var i = 1; i < this.body.length; i++){

     //if stament checking if the headations are equal to each other

     if(this.head.x == this.body[i].x && this.head.y ==
 this.body[i].y){

       return true;

     }

   }

   }



    keyPressed(){

      this.head.add(this.vel);

     if(keyCode === UP_ARROW){

         this.vel.x = 0;

         this.vel.y = -30;
     }

     if(keyCode === DOWN_ARROW){

       this.vel.x = 0;

       this.vel.y = 30;


   //    this.head.y = this.head.y + this.w;

       }

     if(keyCode === LEFT_ARROW){

       this.vel.y = 0;

       this.vel.x = -30;

   //  this.head.x = this.head.x - this.w;

       }

     if(keyCode === RIGHT_ARROW){

       this.vel.y = 0;

       this.vel.x = 30

   //  this.head.x = this.head.x + this.w;


       }

   }//end keyPressed


   checkEdges(){ //keep snake inside screen //doesnt work

     // if(this.head.x< 0) {this.vel. x = -this.vel.x;}
```

```
    // if (this.head.x> width) {this.vel.x = -this.vel.x;}

    // if (this.head.y < 0) {this.vel.y = - this.vel.y;}

    // if(this.head.y> height) {this.vel.y = -this.vel.y;}

    //   // if(this.head.x< 0){

      //   this.head.x + this.w

      // if (this.head.x> width) this.head.x + this.w;

      // if (this.head.y < 0) this.head.y + this.w;

      // if(this.head.y> height) this.head.y - this.w;

  }

} //++++++++++++++++ End Snake
```