Index:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Create Task 2020</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>
```

Sketch:

```
var score, header_height, snake, difficulty, type, choice;
var gameState = 1;
var h = 10;
var food = [];
var body = [];
var btnEasy, btnMed, btnHard, btnInstructions, btnBTME, btnBTMI,
btnSea, btnDesert, btnGarden;

function setup() {
  var cnv = createCanvas(600, 600);
  cnv.position((windowWidth-width)/2, 30);
  background(163, 163, 194);
  header_height = 600;
  score = 0; //iniital score
  newButton();
}

function draw(){ //decides what screen to go to
```

```
    if (gameState ===1){
      pickSnakeType(); //start screen
    }else if (gameState === 2){
      instructionsText(); //game screen
    }else if (gameState === 3){ //game over screen
      startGame();
    } else if(gameState === 4){ //screen to pick theme
      playGame();
    }else if(gameState === 5){ //insturctuctions screen
      endGame();
    }
} //end draw

function newButton(){ //declares location and color of all the
buttons
  btnEasy = new Button(25, 350, 150, 100, color(78, 219, 18));
  btnMed = new Button(225, 350, 150, 100, color (250,250,7));
  btnHard = new Button(420, 350, 150, 100, color(250, 0, 0));
  btnSea = new Button(25, 250, 150, 100, color(0, 0, 255));
  btnDesert = new Button(225, 250, 150, 100, color (153, 102,
51));
  btnGarden = new Button(420, 250, 150, 100, color(15, 71, 38));
  btnInstructions = new Button (110, 440, 400, 100, color(5));
  btnBTMI = new Button(110, 540, 400, 100, color(255, 179,
179));
  btnBTME = new Button(250, 50, 150, 100, color(100));
}

function pickSnakeType(){ //allows you to pick the theme
  background(128, 128, 255); //background for theme choosing
screen
  btnSea.render(); //draws buttons
  btnDesert.render();
  btnGarden.render();
  btnInstructions.render();

  textFont('Georgia')
  textSize(80); //Snake game text
  fill(191, 64, 128);
  text("SNAKE GAME", 40, 150);
```

```
  textSize (40); //text for buttons
  fill(255);
  text ("Sea", 65, 278, 200, 200);
  text ("Desert", 240, 278, 200, 200);
  text ("Garden", 430, 278, 200, 200);

  text ("Instructions", 200, 500);

   pickSnake(); // checks which difficulty is chosen
    if (type === 'sea' || type === 'desert'|| type ===
'garden'|| type === 'instructions'){
      if (type === 'sea'){
        startGame();
        gameState = 3;
      }else if (type === 'desert'){
        startGame();
        gameState = 3;
      }else if (type === 'garden'){
        startGame();
        gameState = 3;
      }else if (type === 'instructions'){
        instructionsText();
        gameState = 1;
    }

}// end pickSnake


function instructionsText(){//function for the instructions
  background(0, 163, 204)
  textSize(20);
  fill(5);
  text("Objective: Make the snake as long as possible", 100,
100);
  text("How: Use the arrow keys to move the snake around to eat
the food", 10, 150);
  text("The game ends if the snake hits the edges of the
screen", 50, 200);
  text("or if the snake hits itself", 200, 250);
  text("Your score increases by one for each piece of food you
eat", 50, 300);
```

```
  text("Good Luck!", 250, 350);

  fill(50, 100, 150) //back to main menu button
  btnBTMI.render();
  fill(5);
  textSize(40);
  text("Back to Main Menu", 140, 600)

  if(btnBTMI.isClicked()=== true){
    pickSnakeType();
  }
}
}//end function instructionsText

function startGame(){
  textSize(80);
  background(204, 153, 255);
  fill(96, 0, 128);
  text ("Snake Game", 100, 150); //title

  btnEasy.render(); //draws buttons
  btnMed.render();
  btnHard.render();

  textSize (35); //text for buttons
  fill(5);
  text ("Easy", 65, 378, 200, 200);
  text ("Medium", 235, 378, 200, 200);
  text ("Hard", 450, 378, 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 = 4; // play game
```

```
    }
}//end startGame

function playGame(){ //function to play the game
  frameRate(10); //makes snake go at normal speed

  if(type === 'garden'){ //makes background specific to theme
    background(15, 71, 38);

  } else if (type === 'desert'){
    background(153, 102, 51);

  }else if(type === 'sea'){
    background(0, 0, 255);
  }
  runObjects(); //calls runOjects function
  text ("Score: " + score, 100, 50); //score
  //checkTangled(); //if tangled, game Over
} //end playGame

function endGame(){ //created end screen
  background(128, 0, 0); //red background
  fill(5);
  textSize(80);
  text("GAME OVER!", 50, 300); //game over text

  btnBTME.render(); //puts buttons on screen
  fill(20);
  textSize(30);
  text("Menu", 290, 80, 200, 200);

  pickSnake();
  if (type === 'menu1'){
    if (type === 'menu1'){
      clearEverything();
    }
  }
 } //end function endGame

function loadObjects(n){ //function to declare snake and food
objects
```

```
  if(type === 'garden'){ //checks to see what type is chosen,
then prints colors specific to that theme
  snake = new Snake
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
,30, color(227, 69, 7));
    for (var j = 0; j < n; j++){
      food[j] = new Food
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
, color(70));
      }
    }else if (type === 'desert'){
      snake = new Snake
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
,30, color(20));
        for (var j = 0; j < n; j++){
          food[j] = new Food
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
, color(202, 237, 0));
          }
    } else if(type === 'sea'){
      snake = new Snake
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
,30, color(162, 0, 255));
        for (var j = 0; j < n; j++){
          food[j] = new Food
(Math.floor(Math.random()*16)*30,Math.floor(Math.random()*16)*30
, color(0, 255, 229));
          }
    }
  }// end loadObjects

function runObjects(){
    snake.run();
    for(var i = 0; i< food.length; i++){//renders food objects
based on length of array
      food[i].run();
    }
  } //end runOjects

function checkTangled(){ //checks to see if snake is tangled
  if(snake.tangled() === true){
```

```
    gameState = 5; //game over
  }
} //end checkTangled

function checkDifficulty(){ //check which difficulty button is
isClicked
  if(btnEasy.isClicked()=== true){
     difficulty = 'easy';
   } else if (btnMed.isClicked()===true){
    difficulty = 'medium';
  } else if (btnHard.isClicked()=== true){
    difficulty = 'hard';
  }

}// end checkDifficulty

function pickSnake(){ //check which difficulty button is
isClicked
  if (btnSea.isClicked()=== true){
     type = 'sea';
   }else if (btnDesert.isClicked()===true){
    type = 'desert';
  }else if (btnGarden.isClicked()=== true){
    type = 'garden';
  }else if (btnInstructions.isClicked()=== true){
    type = 'instructions';
  }else if (btnBTMI.isClicked() === true){
    type = 'menu';
  }else if (btnBTME.isClicked() === true){
    type = 'menu1';
  }
} //end pickSnake


function clearEverything() { //clear gamestate and score for
restarting level
  gameState = 1;
  score = 0;
  food = [];

} //end clearEverything
```

Snake Class:

```
class Snake{ //class snake
  constructor(x, y, w, c){ //constructor for snkae objects
    this.head =createVector(x,y);
    this.vel = createVector(0,0);
    this.w = 30;
    this.clr = c;
    this.body = [];

  }// end constructor

  run(){ //class all methods in this class
    this.update();
    this.render();
  } //end run

  update(){ //updates location of snake
    this.keyPressed(); //goes to where key is pressed
    this.checkEdges();
     for(var i = 0; i< food.length; i++){ //traverses whole food
array
       if(this.head.x === food[i].loc.x &&
          this.head.y === food[i].loc.y){ //if head is on food
          this.loadSegment();
          if (type === 'garden'){ //makes new food objects
appear for the given theme
          food.push(new Food
(Math.floor(Math.random()*25)*30,Math.floor(Math.random()*25)*30
, color(70)));
          score++; //if the snake hits a food object, the score
increases
        }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;
     }
  }
}//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);
   }
 }//end render

  loadSegment(){
    this.body.push(createVector(this.head.x, this.head.y));
  } //end loadSegment

  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;
    }
  }
}//end tangled
```

```
keyPressed(){ //function to move snake with arrow keys
    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;
      }
      if(keyCode === LEFT_ARROW){
        this.vel.y = 0;
        this.vel.x = -30;
      }
      if(keyCode === RIGHT_ARROW){
        this.vel.y = 0;
        this.vel.x = 30
      }
  }//end keyPressed

  checkEdges(){ //keep snake inside screen, if not, game over
    if(this.head.x< 0 || this.head.x > 600|| this.head.y < 0 ||
this.head.y> 600){
        gameState = 5;
    }
  }//end checkEdges
} //+++++++++++++++++ End Snake
```

Food Class:

```
class Food{ //food class

constructor (x, y, clr){ //constructor that decides property of
food objects
  this.loc = createVector(x,y);
  this.clr = clr;
}

run(){ //calls all methods in this class
  this.render();
  this.update();
```

```
}

render(){ //makes button appear on screen once declared in
sketch
    fill(this.clr);
    rect(this.loc.x, this.loc.y, 30, 30);
}

update(){ //remakes food when it is eaten by snake
    if(snake.head.x === this.loc.x &&
      snake.head.y === this.loc.y){ //if statement to see if
snake head eats food
      this.loc.x = Math.floor(random(0,79))*30;
      this.loc.y = Math.floor(random(0,79))*30;
    }
  }
}// end food class
```

Button Class:

```
class Button{ //button class
constructor(x, y, w, h , clr){ //constructor- decides properties
of buttons
    this.loc = createVector(x,y);
    this.w= w;
    this.h = h;
    this.clr = clr;
  }

render(){ // creates button on screen when declared in sketch
  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
```