Lab 11 – Collision Detection

version 5.0 – 4/16/18

1.0 Overview

The following is a **Technical Design Document** that outlines:

- A description of the **behavior** of the **components**
- A **definition** of the **components**, their **interactions**, and **control structure**

This is intended to be an example of what a **Technical Design Document** would look like for the *Final Project*.

This also then contains suggested **Work Steps** to complete **Lab 11**.

2.0 COMPONENT DESCRIPTIONS

<u>The Circles</u>

The circles start above the top of the canvas and move down in a straight line until they pass through the bottom of the canvas.

When a circle moves beyond the bottom the canvas the circle restarts at a new random location above the canvas and continues to fall.

The app starts with a single circle.

A new circle is added at a **specified interval**.

<u>The Paddle</u>

The paddle is a rectangle that moves laterally across the bottom of the canvas by mouse movement.

The paddle is restricted to not go past the left and right boundaries of the canvas.

<u>The Interactions</u>

When circles collide with the paddle the circle is removed.

3.0 COMPONENT DEFINITIONS

The Circles

```
var circles = [];

function genCircle() {

    //define new random location for the circle

    //create a new circle object and set its values

    //add the new circle to the circles array

}

function drawCircles() {

    // for each circle in the circles array
        //draw the circles[i]
}

function moveCircles() {

    // for each circle in the circles array

        //move the circles[i] to the next location

        //if the circles[i] has passed the bottom of the canvas

            //reset the circles[i] to new random starting location at the
            top of the canvas

}

function addCircle() {

    //at a specific interval
        // call the function genCircle() to add a new random circle

    Hint:  See scratch.js

}
```

The Paddle

```
var paddle = {
        x: ... ,
        ...
};

function drawPaddle() {

        //draw the paddle

}

function movePaddle( mouseX, mouseY ) {

        //move the paddle left or right

        //if the paddle is at the left boundary
                //stay at the left boundary

        //if the paddle is at the right boundary
                //stay at the right boundary

}
```

Add an **event listener** on the window for when the mouse moves

Add an **event handler** called by the listener that then calls **movePaddle()** with the mouse's X and Y location

The Interactions (collision)

```
function checkCollision() {

        //setup Object1 to the values of the paddle.

        //for each circle on the circles array

                //set Object 2 to the value of the circles[i].

                //if there is a collision

                        //remove the circles[i].from the array
                        Hint:  use .slice()
```

The Canvas

```
function clearCanvas() {

    //write a background color over the entire canvas

}

function drawCanvas() {

    //clear the canvas

    //move all the objects
    - move all the circles

    //draw the objects
    - draw all the circles
    - draw the paddle (box)

    //check for collisions

}
```

The Game Loop

```
function gameLoop() {

    //get a new animation frame [this replaces setInterval]

    //increment the frame counter

    //add a new circle based on an interval

    //call drawCanvas()

}

//set the frame counter
//call gameLoop() function
```

4.0 LAB 11 – WORK STEPS

Here are some suggested **Work Steps** to guide you through Lab 11.

The provided code works for 1 circle. We need to make this work for an **array of circles** based on the **definitions** of the **behaviors** in the ***Technical Design Documen*t** sections above.

Step 1: Change the code to work for **multiple circles**

(a) add the circles array

```
var circles = [];
```

(b) create the **genCircles()** function

- create the function

    - move the random generation into the function

    - move the circle object creation into the function

    - add the circle to the circles array

```
function genCircle() {

    //define new random location for the circle

    //create a new circle object and set its values

    //add the new circle to the circles array

}
```

(c) modify the **drawCircle()** function to work for **ALL** the circles

- change the name to drawCircle**s**()

- add a **for** loop to draw each circle in the circles array

```
for ( i = 0; i < circles.length; i++ ) {

    - change all the circle.'s To circles[i].'s

}
```

- change the function name in **drawCanvas()**

(d) modify the **moveCircle()** function to work for **ALL** the **circles**

- change the name to moveCircle**s**()

- add a for loop to draw each **circles[i]** in the **circles** array

```
for ( i = 0; i < circles.length; i++ ) {

    //move the circles[i].'s

    //check the boundaries for circles[i].'s

}
```

- change the function name in drawCanvas()

<u>Step 2:</u> Change the code to work for the paddle

The paddle does not change; so nothing needs to change here in the code

<u>Step 3:</u> Change the **collision detection** to check **for ALL the circles**

- Modify the **checkCollisions()** function.

- **Object1** is still the paddle; so no changes to that.

- **Object2** is now each circle on the circles array; so add a for loop to step through the circles array and set Object2 to the values of **circles[i]**

```
function checkCollision() {

    //setup Object 1 to the values of the paddle (box)

    //for each circles[i] on the circles array
    for ( i = 0; i < circles.length; i++ ) {

        //set Object 2 to the value of the circle
        Object2X = circles[i].

        //if there is a collision
            //remove the circle from the array
```

```
                  Hint: .slice(i,1)
      }
```

Step 4: Include logic to **add a new circle** at a specific interval

(a) add a parameter for when to change

```
var changeInterval = 1000;
```

(b) add a function that will add a circle when the interval is hit

```
function newCircle() {

   //add new circle based on a change interval
   if ( (frameCounter % changeInterval) == 0 ){

      //add a new circle
      addCircle();

   } //if

} //newCircle()
```

(c) call **newCircle()** in the **gameLoop()**

Step 5: Add ***something more***…