COMP106
Lab 10 – Collision Detection
Technical Design Document
version 5.3 – 11.10.19

1.0 Overview

The following is a **Technical Design Document** for Lab 10 that outlines:

- A description of the **behavior** of the **components**
- A **definition** of the **components**, their **interactions**, and **control structure**
- The **Work Steps** to complete **Lab 10**

Note, this is an example of what a ***Technical Design Document*** would look like; as you will be making one of these for the *Final Project*.

2.0 DESCRIPTION OF COMPONENT BEHAVIORS & INTERACTIONS

Circles

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**.

Paddle

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.

Circle & Paddle Interactions

When circles collide with the paddle the circle is removed.

3.0 DEFINITIONS OF VARs & FUNCTIONs FOR COMPONENTS

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() {

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

function moveCircles() {

    // loop 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: ... ,
        y: ...
        ...
};

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.

        //loop 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()

}
```

Setup:

```
//set the frame counter
//add the first circle
//call gameLoop() function
```

4.0 WORK STEPS

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

The starting 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 Document*** sections above.

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

(a) add the circles array

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

Add this at the top of the canvasApp() with the other component variables

We did this in the last lab.

(b) create a **genCircle()** function that will

//define variables for new random size, starting position, speed for the circle
//create a new circle object and set its values using the variables from above
//add the new circle to the circles array

We did this in the last lab.

```
function genCircle() {

        //define new random location for the circle
        - we have code for this; move the code that does the random
        generation from the variable section at the top into this function

        //create a new circle object and set its values
        - we have code for this; move the code that declares and initializes the
        circle object, var circle = {…};  , from the variable section at the top
        into this function

        //add the new circle to the circles array
        - add code that will add the circle object to the circles array
        Hint:  use the method that starts with a .p and ends with an op()

}
```

Add this function to the section of the code that has the other circle related functions.

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

- change the name to drawCircle**s**()
- remove the (**c**) parameter that is passed in

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

- inside the for loop change all the **c.**'s to **circles[i].**'s
    Example: change **c.x** to **circles[i].x**

```
function drawCircles() {
        //loop for each circle
        for ( i = 0; i < circles.length; i++ ) {

                - inside the for loop change all the c.'s to circles[i].'s
                        Example:  change c.x to circles[i].x
                do this for all the c. references
        }
}
```

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

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

- change the name to moveCircle**s**()
- remove the (**c**) parameter that is passed in

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

- inside the for loop change all the **c.**'s to **circles[i].**'s
    Example: change **c.x** to **circles[i].x**

```
function drawCircles() {
        //loop for each circle in the array
        for ( i = 0; i < circles.length; i++ ) {

                - inside the for loop change all the c.'s to circles[i].'s
                        Example:  change c.x to circles[i].x
                do this for all the c. references

        }
}
```

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

Step 2: Change the code to work for the paddle

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

Step 3: Change the **collision detection** to check for **ALL** the **circles**

- Modify the **checkCollision()** 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 all the circles array; and
        - set **Object2** to the values of **circles[i]. ;** and
        - when there is a collision, change the code to remove that circle from the **circles** array
          Hint: *.splice*

```
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].x – circles[i].size/2;
        …do this for all refences to circle.

        //if there is a collision
            //remove the circle from the array
            Hint: .splice
    }

}
```

Step 4: Include logic to **add a new circle** at a specific interval (hint: see *scratch.js*)

(a) add a parameter for when to change

```
var changeInterval = 100;
```

(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
        genCircle()
    } //if

} //newCircle()
```

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

Step 5: Add ***something more***…