**Spike:** 22

**Title:** Collisions

**Author:** Ben Holmes, 103024841

**Goals / deliverables:**

1. Implement at least box-based (axis-aligned rectangles) intersection testing, and circle-circle intersection testing:
   1. Display two or more boxes, one of which is a fixed position box and the other(s) moving. Detect, using axis-aligned rectangle testing, one box overlaps another (collision detected) and visually display the “collided” status to the user
   2. Display two or more circles where one or more of the circles are moving. Detect, using circle-circle collision testing, when one circle overlaps another, and change the visual appearance to represent that a collision has occurred.
2. Incorporate extensibility in your collision data structure, so that you would be able to swap out collision detection method for another

**Technologies, Tools, and Resources used:**

* Visual Studio
* Word
* GitHub
* <https://lazyfoo.net/tutorials/SDL/index.php>

**Tasks undertaken:**

* Create file structure
* Create moving circle rendering
* Create frame and tick capping
* Create moving square
* Create square collision detection
* Create circle collision detection

**What we found out:**

First was the creation of the file structure, which has a class file for each shape, a function file for each shapes world and a file for the collision functions (and a timer file).

Then I began to make the circle rendering, this was done first as the tutorial on motion on LazyFoo did a circle and I thought it would be harder to do the circle second.

Here is the class:

A screenshot of a computer program

Description automatically generated

Circle needed to use a texture in the rendering as a result of not having an easy rendering function unlike squares who have SDL\_RenderFillRect().

Functionality:

A screen shot of a computer code

Description automatically generated

The set position values are so that the circle (and squares later on) will overlap after the moving one loops for ease of testing.

A screenshot of a computer program

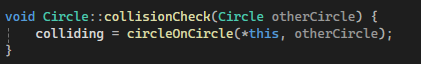
Description automatically generated

This move function bounces the circle off the wall when it gets to close and flips the velocity (as the circle position is the top left corner of the render rather than the centre of the circle, mPosX < 0 is all that’s needed for the left and top sides of the screen)

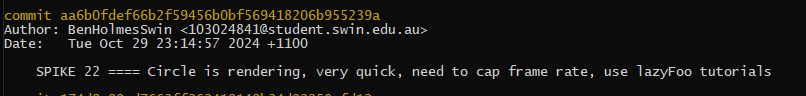
A screen shot of a computer program

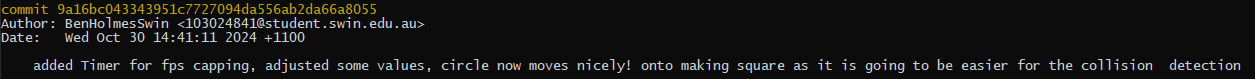
Description automatically generated

This is the circle render function, it needs all those extra values to use renderCopyEx and the if statement is the attempt at the color changing which did not work the way I hoped. It flips from red to black rather than red to green but still flips correctly.



The collision check which will be explained after implementation of squares.





I also had to add a timer class in order to be able to cap the frame rate to prevent the shapes from moving incredibly fast

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A computer screen with text

Description automatically generated

On to the square

A screenshot of a computer program

Description automatically generated

Uses a very similar structure to the circle, didn’t use inheritance as I did not want to have to worry about it this time around with all the differences.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

The renderer was far simpler for the square.

There weren’t too many changes from previous SDL window setup:

A computer screen with many colorful text

Description automatically generatedthis is in SDLcircles LoadMedia, this loads the texture for the moving circle, with the stationary circle being near identical below this one. This function and the rendering of the circle textures were the main reason for the splitting of the files rather than switching the shapes in a single file.

A screen shot of a computer program

Description automatically generated

The assignment of the capTimer and what was changed about the loop, the moving circle is rendered after the stationary circle so that it is render above it.

Square:

A screen shot of a computer program

Description automatically generated

No media needed loading for the square.

A screen shot of a computer program

Description automatically generated

And a nearly identical while loop.

The collision functions:

A computer screen shot of a code

Description automatically generated

This is the box on box, it compares each edge to see if there is overlap, and if they individually don’t, returns false, other wise returns true

Square running:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

This is deliverable 1 part a.

A black screen with numbers and letters

Description automatically generated

Circle:

A computer screen with text on it

Description automatically generated

This uses the squared version of the circle-circle comparison in order to avoid the sqrt. I was using ^ for squaring which does not work, did not realise, feel like an absolute dumbass for not checking that until after I was done. There were so many issues cause by that.

Circle running:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

This is deliverable 1 part b.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

For deliverable 2, when I created the collision methods, I specifically made it so they could be switched out, this is why the collision functions are not directly attached to the objects, but are passed to.