**8-2 Assignment Change Documentation**

In this assignment, I have made changes to the number of bricks, how bricks change on ball collision, how Circle objects “reflect” off the window/brick edges, and how Circle objects are generated into the scene.

I increased the total number of bricks to 10 and arranged them into interlocking pentagrams. The Brick objects with an initial color of red are REFLECTIVE type Bricks while those with an initial color of green are DESTRUCTABLE type Bricks and will be deleted once collided with by a ball. When a REFLECTIVE Brick is collided with by a ball, the Brick cycles through red, green, and blue colors in that order. The color cycling is handled by a custom enum “BRICKCOLOR” defined at the top of the file. This enum is used in the CheckCollision method of the Circle class – specifically only for REFLECTIVE type Bricks. The Circle “ball” reflects off the surface of the Brick by adjusting the direction of travel to a supplementary angle to the angle of incidence – the angle at which the ball hits the surface – to give a natural effect of bouncing off a flat surface.

To keep track of the Brick objects, I added a global vector to store them. I utilized this vector to check for ball collision and for drawing the Circle objects in every frame. Additionally, this vector is used to check if a ball would be spawned inside of a Brick and move the ball if true. Left click can be used anywhere inside the window to spawn a ball at the click location. If the ball would overlap with the window edge or a Brick object when spawned, it will be moved outside the Brick / inside the window edge. When the Circle object (ball) is being created, it is assigned a random color, and, if the new ball would cause there to be a total of more than 5 Circle objects in the scene, the oldest ball will be deleted. The Circle object is created using the mouse\_button\_callback function which is set as the mouse callback function in the main function. From the Circle class, I moved the GetRandomDirection() method to the global scope as I needed to use it to set a random direction on every new ball that is spawned.

In the rendering loop, I changed the Circle movement for loop to contain an inner for loop to check for collision on each ball against every Brick that is ON. I also added a for loop to draw every Brick that is ON as opposed to having one line to draw each Brick. Lastly, in the rendering loop, I also made the scene objects maintain their aspect ratio in the Viewport by changing the Viewport size and location if the width of the GL window is greater than the height of the GL window. This enables the Viewport to be a square, even when the GL window is not.

After making my changes, I noticed a strange phenomenon that even though my Circle objects are assigned a random direction on spawn, and even when the objects are spawned in different click locations, they tend to fall into sync – each ball following the same bouncing path as all others. I am unsure of exactly why this happens.