Maya Kelly Finlay - S2024180

BSc (Hons) Computer Games (Design) – P01628

*I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award.*



Table of Contents

[1. Project Description - Brick Breaker 2](#_Toc61210290)

[2. Code Description 2](#_Toc61210291)

[2.1 Manager Script 2](#_Toc61210292)

[2.2 Ball Script 3](#_Toc61210293)

[2.3 Bat Script 3](#_Toc61210294)

[3. Challenging Code 4](#_Toc61210295)

# Project Description - Brick Breaker

For our Introduction to Programming Coursework, we were tasked to recreate a game like “Brickout”. The game consists of 4 walls of bricks on the top of the screen, a bouncing ball, and a horizontally movable bat on the bottom of the screen. The objective of the game is to break all the blocks by hitting them with the ball by bouncing it with the bat. Each brick destroyed awards points. If the ball falls off screen the player loses a live and causes a game over if all lives are lost.

# Code Description

## Manager Script

The Manager script handles methods such as the creation and deletion of the ball by instantiating a clone of a premade ball object then activating/deactivating these clones as required. The script also runs miscellaneous game elements such as lives, score, and game over conditions. The game can be completed two ways; by winning through destroying all the bricks or losing by dropping the ball enough to lose all 3 lives. I created an array to hold the bricks and had the program check when there were no more bricks left. Similarly, the program checks for when the player’s lives have hit zero and calls the GameOver function.

## Ball Script

This script covers how the ball moves within the game space, how it handles collisions with the surrounding objects. It uses an OnCollisionEnter function with if and switch statements to check for the walls, the bricks, the bat, and the area off screen to trigger a life lost.The script also contains statements so that when the ball is respawned it is set to the bat’s position and the spacebar can be pressed to send the ball upwards.

## Bat Script

The bat script also handles movement and collisions (particularly with the two two side walls.) To detect movement inputs, I used public keycode variables to detect left and right inputs and Boolean inputs to check whether the bat could move in the desired direction. The program then checks for collisions with each wall using a switch case for both OnCollisionEnter and OnCollisionExit to disallow and allow movement, respectively. I then use an If statement to check for each button press and set the direction to move to each input if the bat can move in that direction.

# Challenging Code

The element of the code I struggled with implementing the most was the collision with the bricks. As there were numerous bricks to be destroyed, writing separate code to check for each brick’s name and preform the same functions upon collision would have been inefficient. Hence, I tagged each brick with its own “Bricks” tag to collate them into one searchable group and attempted to add this tag as a case into the switch statement I had previously developed. Unfortunately, I found that the switch case could not retrieve the tag and would allow the ball to pass through the bricks.

I had thought the issue was the program not looking for a tag but an object name so I then tried to use transform.CompareTag within the switch case to make sure it was searching for tags, but instead found this code was also incompatible with the switch statement. As the switch statement was more efficient use of code for the other collisions than an if statement, I did not want to swap it completely. This meant I had to move the brick collision check into its own if statement and place the switch statement within the else check.