**Brickout**

**Introduction to Games Programming**

Finn Atkinson · S1915107 · Computer Games (Design)

*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.*

*-Finn Atkinson*

**Table of Contents**

1 [Introduction](https://github.com/atkinsonfinn01/breakout#Introduction) – Page 3

2 Description of Code – Page 3

* + Ball
  + Bat

3 Issues – Page 4

**1 Introduction**

We have been tasked with creating the game "Brickout". I have used Unity to create the project and Visual Studio using C#.

**2 Description of Code**

**Ball**

To make the ball move I gave it physics using the unity engine and used a C# script to give it an initial momentum which would propel it towards the bat. The bounce physics material was used to maintain this momentum.

void GoBall()

{

float rand = Random.Range(0, 2);

if (rand < 1)

{

rb2d.AddForce(new Vector2(30, -15));

}

else

{

rb2d.AddForce(new Vector2(-30, -15));

}

}

**Bat**

To make the Bat move, I set the Bat to having the "Kinematic" physics option so it would only be moved by the instructions presented to the Bat in the code. The script will detect the Left Arrow or Right Arrow key inputs and make the Bat move to the left or right respectively when it receives these inputs. If no input is detected, the Bat will remain/return to stationary

bool isLeftPressed = Input.GetKey(moveLeftKey);

bool isRightPressed = Input.GetKey(moveRightKey);

if (isLeftPressed && canMoveLeft)

{

direction = -1.0f;

}

else if (isRightPressed && canMoveRight)

{

direction = 1.0f;

}

{

direction = 0.0f;

}

**3 Issues**

My main issue was when I tried to make the ball move. Initially, I tried to make the ball move entirely through the code but struggled to figure this out and as a result changed my balls movement to be mainly physics based through the unity engine. I plan on coming back to this project and learning different ways to make the ball move to aid me in future projects. Previously, I tried this code

else if (Input.GetButtonDown("Jump") && transform.position.x < 0 && !moving)

{

moving = true;

rb.AddForce(Vector2.up \* speed);

rb.AddForce(Vector2.left \* (speed / 2));

}

else if (Input.GetButtonDown("Jump") && transform.position.x == 0 && !moving)

{

moving = true;

rb.AddForce(Vector2.up \* speed);

}

This approach did not work for me, I didn't know where I went wrong so I tried the approach I used above. This leaned more heavily on the unity physics engine than the script which I found easier to work with. The script was used to exert a force to create momentum in one of two directions towards the Bat, and maintained this momentum using the Bounciness feature of the physics material 2d. Initially I set this value to 2, but found that increasing the Bounciness value above 1 would cause it to gain momentum exponentially and fly off the screen. To allow the ball to fly off the bat at an angle, I changed my collider from being a rectangle to being a polygon, so I could create a raised triangle on the top of the bat which disallows the ball to get stuck in a loop going up and down.