

* Name of student: Nikolaos Antoniadis
* Student Number: S1829313
* Programme of study: Computer gaming design

Introduction to Games Programming

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.

Nikolaos Antoniadis

Table of Contents

[Introduction 2](#_Toc61213627)

[Main Body 2](#_Toc61213628)

[Ball movement 2](#_Toc61213629)

[Bat movement 3](#_Toc61213630)

[The Most Challenging 3](#_Toc61213631)

# Introduction

During my course throughout the year, we have been tasked to make a developed 2D game like a brick breaker. The game should be hand-coded, and we can get extra information from the course labs. The game must include movement, collision, textures, and any extra material if we can put in. This coursework is to test our ability and knowledge of coding and help us improve our way of thinking and skills. In this documentation I will illustrate the struggles I faced and the codes. The deadline for the game is until 11/01/2021 and we must use unity to create the game with C#.

# Main Body

# Ball movement

This script has everything that ball be able to do and collide with. It has an” if statement “for the velocity that helps the ball not to remain on the same place if it goes up and down. It also contains an “if statement “about the ball when the game starts. The ball will not move until the space bar is pressed the ball and then it will lunch. The game will start in a random position on top of the paddle.

{

    if (Input.GetButtonDown("Jump") && !inMovement) {

      GetComponent<Rigidbody2D>().velocity = Random.insideUnitCircle.normalized \* speed;

      inMovement = true;

    }

Vector2 NormalizeVector(Vector2 v) {

    if (v.x < 1f && v.x > -1f) {

      v.x = 1f;

    }

    if (v.y < 1f && v.y > -1f) {

      v.y = 1f;

    }

    return v;

  }

# 

# Bat movement

The paddle movement and the wall collision it was used in this script. I took the distance before the paddle hits the wall from both sides and used an “if statement” so when the paddle reaches the distance it will not be able to go further.

 if (transform.position.x < boundXLeft)

        {

            transform.position = new Vector3(boundXLeft, -4.14f, 0);

        }

        if (transform.position.x > boundXRight)

        {

            transform.position = new Vector3(boundXRight, -4.14f, 0);

        }

        input = Input.GetAxisRaw("Horizontal");

# The Most Challenging

The most challenging code that I made many mistakes and I had to try many times was the respawn. I wanted to make the paddle and the ball to respawn on the same position so when the player loses, then he continues the game from the start position. I could not put the ball in the right position when the player loses, and the ball after the restart went down with higher speed and instantly loses the game. After many tries, I made it and I created two scripts the respawn and the bounce. The paddle to be respawned I used a heritage method. I applied two scripts so when the ball gets out of bounce the first script (bound), then it will call the second script (respawn) using an “if statement” to respawn everything on the starting position to continue again the game.

 void OnTriggerEnter2D(Collider2D other) {

        Ball ball = other.GetComponent<Ball>();

        bool isBall = ball != null;

        if (isBall) {

            respawnPoint.RespawnEverything();

        }

    }