**Assessment 3 – Matrices Application Project**

Prepared For:

Farshid Keivanian

Math102

Prepared By:

Kenbe Yeung (Student ID: A00107244)

# **Overview**

This is the remastered version of Assessment 1, part 2 now renamed as “Space Evaders”. There were several changes made during this remaster, primarily focusing on the theme and graphics. In the original version it featured a girl avoiding danger, whereas in the new version I opted for a space themed game about a spaceship dodging meteors.

To better align with the new theme, I change the graphics. The first modification was the background from a solid pink colour to a space themed backdrop. Additionally, I changed the playable character from a girl in the first version to a spaceship. Lastly, the enemy was changed into a meteor.

The game’s goal is to maneuverer the spaceship avoiding both the white borders surrounding the screen and any appearing meteors. Once, the player collides with either the border or the meteor it would result in a game over, returning the player to the title screen.

My primary aim is to fulfill the following requirements:

* Quaternions
* Collision detection
* User input response to in game/simulation objects
* Appropriate data structures will be utilised.

while ensuring the game remains engaging and interactive

# **Functionalities**

The game contains two Scenes; Scene 1 (Menu) this would be the main menu and Scene 2 (Play) this is where the game starts. Within Scene 1 (Menu) it contains two script files: MainMenu.cs and rotation.cs:

***MainMenu.cs***

This script has the critical role in managing the game’s main menu interactions. Specifically, the behaviour of the two buttons: Play and Quit. This script was taken from Assessment 1, Part1 (button) and further improved.

A computer screen shot of a black background

Description automatically generated

Whole Script of MainMenu.cs

The first method is called *“SpaceInvader”* this is connected to the Play button. Inside the *“SpaceInvader”* method contains two lines of code which will be executed when the Play button is pressed. The function first line of code inside the method will take you to the next scene (Play) and the second line of code will display the text “Starting Space Invader” in the console log.

A black screen with white text

Description automatically generated

The second method is called *“Quit”* and this is connected to the quit button. Once the button is pressed the two lines inside the method will run. Resulting the application to close and the console log will display “Quitting Application”.



***rotation.cs***

This script is for the rotation of the game object (the spaceship and meteor) in the main menu.

A screen shot of a computer program

Description automatically generated

Whole Script of rotation.cs

The first line of code is for the rotational speed of the game object the initial speed was set to 1f. The variable was written with *“SerializeField”* because this would enable adjustments for the rotation speed at any given time during the execution. The second line of code is to control the mechanism of the automatic rotation.



Inside the Update() method, are two conditional statements. The first if statement is that when the spacebar key is pressed it will call the ToggleAutomaticRotation() method. In the second if statement contains *”automaticrotation”* . If this feature is enabled the game object will continuously rotate.

A black rectangle with white text

Description automatically generated

The RotateObject() method will continuously rotate the object forward.

A computer screen with text

Description automatically generated

This method toggles the automatic rotation on/off. So, if you press the spacebar the game object will rotate but if you press it again it will stop rotating.



Within Scene 2 (Play) several scripts are crucial to the gameplay experience such as CharacterMovement.cs, EnemyMovement.cs, EnemySpawning.cs, Music.cs, and ObjectCollider.cs. However, it’s important to note that CharacterMovement.cs, EnemyMovement.cs, EnemySpawning.cs, and ObjectCollider.cs were taken from the Assesstment 1.

**CharacterMovement.cs**

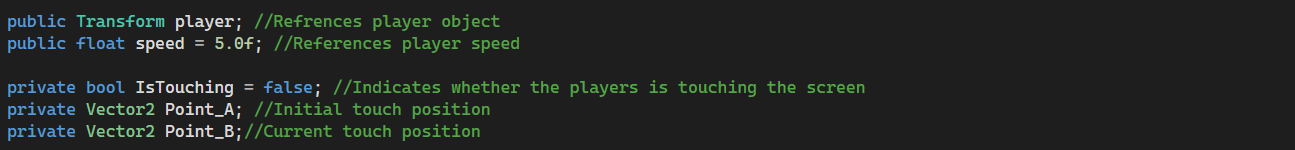
This script allows the player to move their playable character (spaceship) during the gameplay.

A screenshot of a computer program

Description automatically generated

Whole Script of CharacterMovement.cs

The first two lines would references the player object and the speed of the player. The next line will check if the player is touching the screen or is pressing the mouse button down.



Inside the Update method there are three conditional statements. The first statement is to check is the mouse button is pressed and it will record the initial touch position. The second statement will check if the mouse button is held down and will continuously update the current touch position in world coordinates. The last statement is that if the mouse button is not pressed then movement will stop.

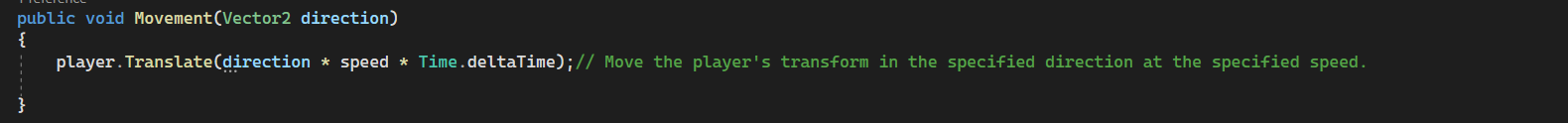
A black background with text

Description automatically generated

In the Fixed update method, the movement is calculated as the player moves. It normalizes the direction of the player and ensures that the player moves to the direction intended.

A black screen with green text

Description automatically generatedLastly, this is to make sure the player’s transform is moved in the specified direction and speed.



**EnemyMovement.cs**

This controls the speed and movement of the meteors.

A screenshot of a computer program

Description automatically generated

Whole Script of EnemyMovement.cs

The first line of code is the speed of the game object (meteor). Subsequently, the second line of the code references to the Rigidbody2D this is very helpful when you want the objects to collide.



Inside the Start method, the first line of code gets the reference to the Rigidbody component attached to the Game object and the second line computes the speed of the game object.

A black background with green text

Description automatically generated

**EnemySpawning.cs**

**A screenshot of a computer program

Description automatically generated**

Whole Script of EnemySpawning.cs

**ObjectCollider.cs**

This script is attached to the playable character (spaceship), enemies (meteors), and the borders surrounding the screen. This is if the player crashes onto them it would result in a game over. There is also a sound effect added in.

**A computer screen shot of a program code

Description automatically generated**

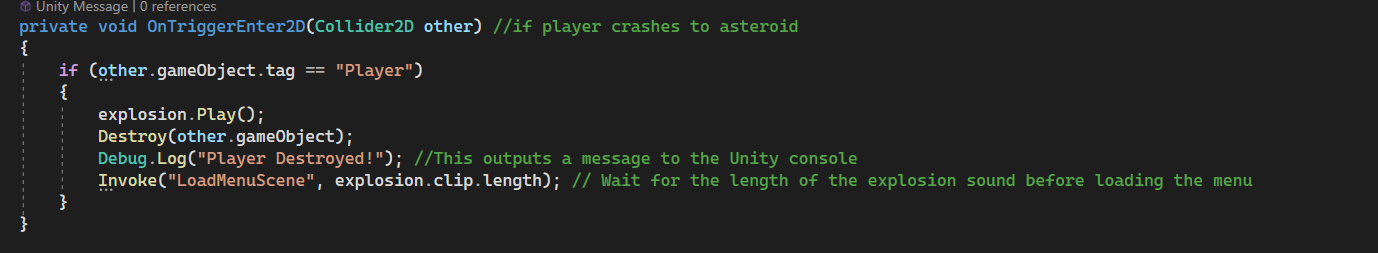
Whole Script of ObjectCollider.cs

The first line of code is to reference the sound effect explosion. Inside the start method, this would get the audio source attached to the game object.

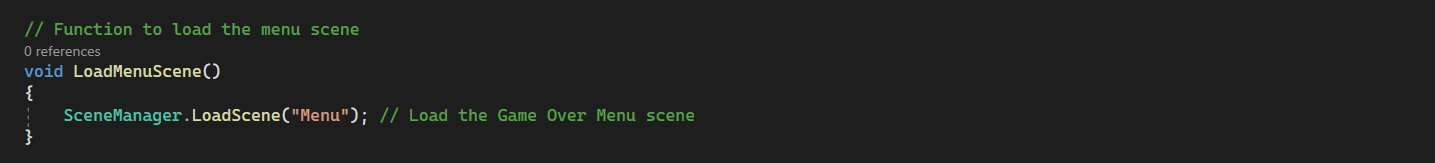
**A black rectangle with white text

Description automatically generated**

Inside this game method, that if the player collides with either the border or meteor the sound effect will play and the console log will display “Player Destroyed!”. Additionally, there will be a slight pause before it returns the player to the menu screen.

****

Lastly, this function will take you back to the menu screen.



**Music.cs**

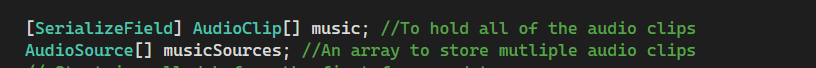
I wanted that every time the game is played a random music will be played in the background.

**A computer screen shot of a program code

Description automatically generated**

Whole Script of Music.cs

The two lines of code are an array. This is so that it would be able to hold multiple audio clips that would be played during the game. The first line has a *“SerializedField”* this is so that you can change, add, or deduct music from the array in any given point.



In the PlayAtRandom method, it would iterate through each audioSource and choose a random clip. Also, will check if an AudioSource is not playing.

A screen shot of a computer

Description automatically generated

Inside the Update method, it would check if the musicSources array is not null and if an AudioSource is available it would play a random audio clip.

A black screen with white text

Description automatically generated

# Demonstration

Having poured countless hours of effort and dedication into the development of Space Evaders. I am thrilled to unveil the final out put of the game. To showcase the game I have provided the demonstration of the game via the link provided below:

<https://youtu.be/52Q6USwp5Sc>