Individual Assignment – Introduction & Instruction

CMAA 5031 (L01), 2025 Spring

Due: 11:59 AM, 14 March 2025

Submit: Please submit a **GitHub link** including: 1) The Unity 3D Project file, and 2) a demo video of the game you made (mp4, less than 30 seconds). All files should be compressed in a **ZIP file** named by your Student ID and Name.

Aim of the task

This task aims to help students build confidence in game making and master the basic game development process.

Task Introduction

As shown in Fig 1, this project is a simple shooting game. The task consists of 8 basic tasks (80 points) and 2 advanced tasks (20 points), with a total score of 100 points.

Users control the left - and - right movement of the game character by pressing the 'A' and 'D' keys on the keyboard. The game character continuously shoots bullets forward. The "enemies" represented by small balls are continuously and randomly generated in the distance and move towards the direction of the game character. When a bullet hits an enemy, the enemy loses health points. When an enemy's health points reach 0, it disappears. Every time a player defeats an enemy, the score panel on the left - hand side of the UI interface accumulates points. For the final effect of the game, refer to the video **Demo_Video.mp4**.

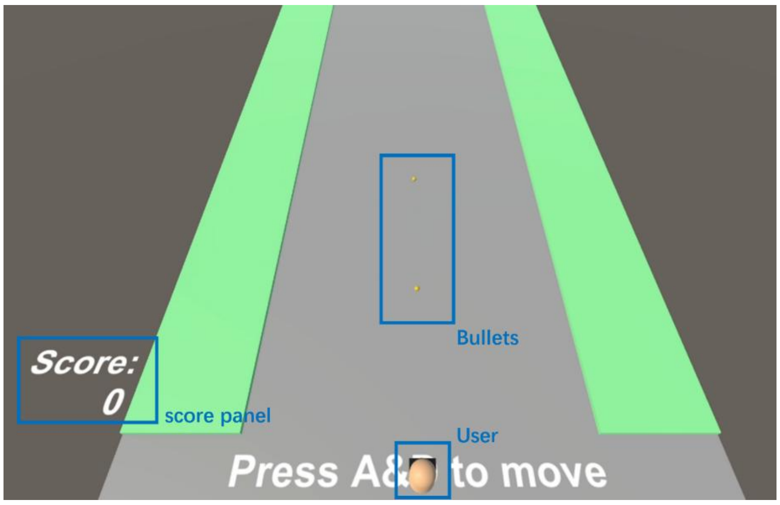


Fig 1. The game UI Demo.

As shown in Fig 2, there are a total of 8 completed scripts in the development interface (Location: Assets -> Scripts). In the 8 basic tasks, you need to connect the script codes with different game components (Fig 3) and make them run successfully. Under normal circumstances, you can complete the tasks without modifying or adding any code.

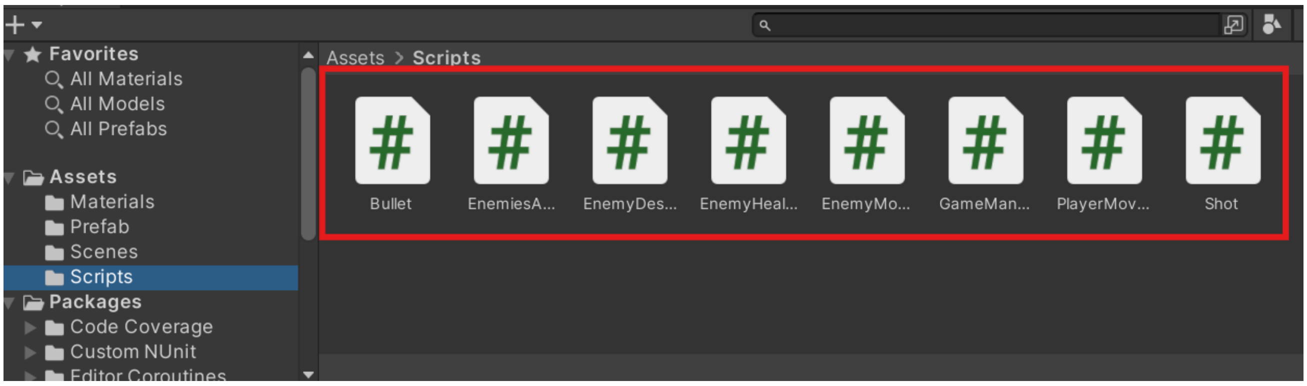


Fig 2. The 8 scripts that need to be bound.

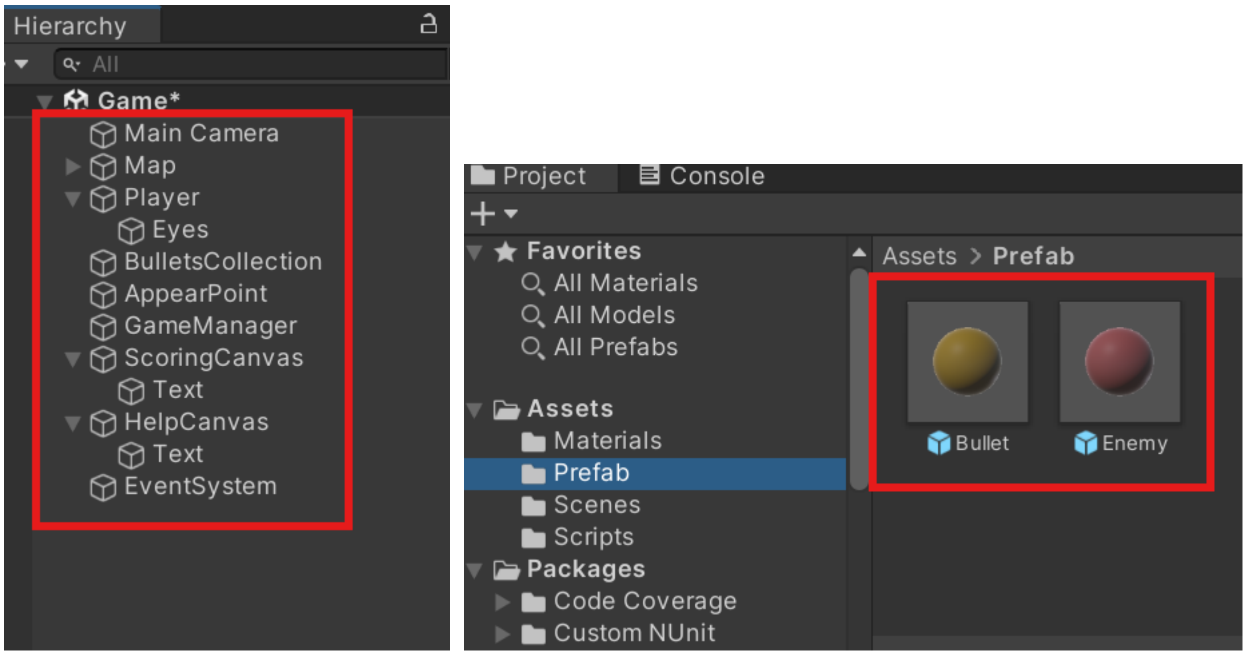


Fig 3. Objects in the game.

Basic Tasks – 80 Points

Task (1/9) 5 Points: Change UI

Objective: Add your Name to the game UI.

Explanation: Add your name to the UI of your game so that your name is visible on the screen when the game is running.

Task (2/9) 5 Points: Player

Objective: Complete the binding related to **PlayerMovement.cs**.

Script Explanation: The player's movement logic mainly involves moving the player character based on the player's input. Define a public movement speed variable `moveSpeed`. In the `Update` method, check if the player has pressed the 'A' or 'D' key. If so, move the player character in the corresponding direction while restricting the player's movement range between -15 and 15.

Task (3/9) 10 Points - Bullet 1

Objective: Complete the binding related to **Bullet.cs**.

Script Explanation: The bullet's logic includes movement, timed destruction, and collision detection. Define the bullet's damage value `damageValue`, movement speed `moveSpeed`, destruction time `destroyTime`, and a timer `timeCount`. In the `Update` method, the timer accumulates time continuously, and the bullet moves forward. When the timer exceeds the destruction time, call the `DestroySelf` method to destroy the bullet. The `DestroySelf` method uses the `Destroy` function to destroy the bullet game object. In the `OnTriggerEnter` method, when the bullet collides with another collider, check if the collider has an `EnemyHealthManagement` component. If it does, call its `GetHit` method to damage the enemy.

Task (4/9) 10 Points - Bullet 2

Objective: Complete the binding related to **Shot.cs**.

Script Explanation: The shooting script is responsible for generating bullets at regular intervals. Define the bullet's prefab `bulletPrefab`, bullet collection `bulletsCollection`, shooting time interval `shotTimeInterval`, and a timer `timeCount`. In the `Update` method, when the timer exceeds the shooting time interval, use the `Instantiate` method to generate a bullet under the bullet collection and set the bullet's initial position.

Task (5/9) 10 Points – Enemy Movement

Objective: Complete the binding related to **EnemyMovement.cs**.

Script Explanation: Define a public movement speed variable `moveSpeed`. In the `Update` method, use the `Translate` method to make the enemy move backward.

Task (6/9) 10 Points – Enemy Destruction

Objective: Complete the binding related to **EnemyDestory.cs**.

Script Explanation: The enemy's destruction logic mainly involves determining whether the enemy has exceeded a certain position. If it has, destroy it. In the `Update` method, check if the enemy's z - coordinate is less than -15. If so, call the `DestroySelf` method to destroy the enemy. The `DestroySelf` method uses the `Destroy` function to destroy the game object.

Task (7/9) 10 Points – Enemy Health

Objective: Complete the binding related to **EnemyHealthManagement.cs**.

Script Explanation: The enemy health management script needs to handle the enemy's health points and the logic when it is hit. Define a public total health variable `totalHealth` and a score value `value`. In the `Update` method, check if the health points are less than 0. If so, call the `EnemyDestory.DestroySelf` method to destroy the enemy and call the `GameManager.UpdateScore` method to update the score. At the same time, add a public method `GetHit` to handle the logic of reducing the enemy's health points when it is hit.

Task (8/9) 10 Points – Enemy Spawn

Objective: Complete the binding related to **EnemiesAppearManagement.cs**.

Script Explanation: The enemy spawn management script is responsible for generating enemies at regular intervals. Define the enemy's prefab `enemyPrefab`, spawn time interval `appearTimeInterval`, spawn point `appearPoint`, and a timer `timeCount`. In the `Update` method, when the timer exceeds the spawn time interval, use the `Instantiate` method to generate an enemy at the spawn point and randomly adjust the enemy's position.

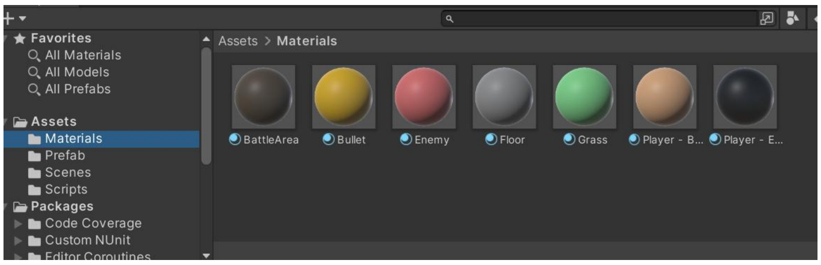
Task (9/9) 10 Points – Score Panel

Objective: Complete the binding related to **GameManager.cs**.

Script Explanation: This script is responsible for managing the overall state of the game, such as score display. First, define a static score variable `score`. In the `Start` method, initialize the UI text display. In the `Update` method, update the score display in real - time. Finally, add a static method `UpdateScore` to update the score. This ensures that this method can be easily called in other scripts to update the score.

Advanced task 1 – 10 Points

In this section, you need to **replace the materials** of components. This can include color replacements, material substitutions, etc., to give the visible objects in your game a different appearance. You can search for and download materials from the internet on your own.



Advanced task 2 – 10 Points

In this additional part, you need to **replace the models** of different components by yourself. For example, the game character controlled by the player, bullets, enemies, the game map, etc. Try your best to make the overall appearance of your game more exquisite.