KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ 1, NĂM HỌC 2024 - 2025

LÀM GAME

TRÊN PHẦN MỀM UNITY

Giáo viên hướng dẫn:

Võ Phước Hưng

Sinh viên thực hiện:

Họ tên: Nguyễn Thanh Duy

MSSV: 110122062

Lớp: DA22TTD

Trà Vinh, tháng 01 năm 2025

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

BỘ MÔN CÔNG NGHỆ THÔNG TIN

THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH

HỌC KỲ 1, NĂM HỌC 2024 - 2025

LÀM GAME

TRÊN PHẦN MỀM UNITY

Giáo viên hướng dẫn:

Võ Phước Hưng

Sinh viên thực hiện:

Họ tên: Nguyễn Thanh Duy

MSSV: 110122062

Lớp: DA22TTD

Trà Vinh, tháng 01 năm 2025

LỜI NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trước khi đi sâu vào dự án, em xin gửi lời cảm ơn chân thành đến trường Đại học Trà Vinh đã tạo điều kiện cho em thực hiện dự án này, những cá nhân đã hỗ trợ và giúp đỡ em một cách tận tình. Cũng như là sự hỗ trợ và giúp đỡ của Thầy Võ Phước Hưng, người đã đóng vai trò quan trọng trong việc phát triển và hoàn thành dự án này.

Em rất cảm kích vì sự giúp đỡ của những người quanh em, những người đã dành thời gian, công sức và kiến thức của họ để giúp đỡ em. Các ý kiến đóng góp và sự hợp tác của mọi người là nguồn động lực giúp em phát triển bản thân.

Em rất quý trọng những người đã hỗ trợ và giúp đỡ em trong suốt thời gian qua, đặt biệt là sự giúp đỡ của Thầy Võ Phước Hưng, nhờ có sự giúp đỡ của thầy mà em mới có thể thực hiện và hoàn thành dự án.

Một lần nữa, xin chân thành trường Đại học Trà Vinh và Thầy Võ Phước Hưng đã giúp đỡ và em mong rằng sẽ nhận được sự ủng hộ của mọi người trong những dự án sắp tới.

Trân trọng,

MỤC LỤC

DANH MỤC HÌNH ẢNH

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Vấn đề cần nghiên cứu

Đồ án tập trung vào việc thiết kế và phát triển một trò chơi thể loại Top-Down Shooting bằng phần mềm Unity – một công cụ phổ biến và mạnh mẽ trong ngành phát triển game. Đây là thể loại t rò chơi có góc nhìn từ trên xuống (top-down), cho phép người chơi điều khiển nhân vật di chuyển và tấn công kẻ thù trong môi trường 2D hoặc 3D.

Các hướng tiếp cận

- **Tìm hiểu và làm quen với Unity:** Khám phá các tính năng cơ bản của Unity, bao gồm giao diện, hệ thống vật lý, và cách tích hợp assets (tài nguyên game như mô hình, âm thanh, hình ảnh).
- **Xây dựng hệ thống gameplay:** Tạo ra các cơ chế cơ bản như di chuyển nhân vật, bắn đạn, và tương tác với kẻ thù.
- **Thiết kế môi trường và giao diện:** Sử dụng công cụ thiết kế tích hợp trong Unity để xây dựng bản đồ, hiệu ứng hình ảnh, và giao diện người dùng (UI).
- **Kiểm tra và tối ưu hóa:** Đảm bảo trò chơi chạy mượt mà trên các thiết bị và sửa lỗi phát sinh.

Cách giải quyết vấn đề

- **Phân tích yêu cầu:** Định nghĩa rõ các yếu tố cần thiết cho trò chơi như cốt truyện, nhân vật, kẻ thù, và các màn chơi.
- **Lập trình với C#:** Sử dụng ngôn ngữ lập trình C# trong Unity để viết mã điều khiển nhân vật, tạo các hành vi của kẻ thù, và lập trình hiệu ứng đặc biệt.
- **Sử dụng tài nguyên Unity Asset Store:** Tận dụng các tài nguyên miễn phí và trả phí để tăng tốc quá trình phát triển.
- **Quản lý dự án:** Sử dụng phương pháp Agile hoặc các công cụ như Trello để theo dõi tiến độ công việc.

Kết quả đạt được

Tạo ra một trò chơi *Top-Down Shooting* hoàn chỉnh với gameplay hấp dẫn, đồ họa sắc nét, và hiệu ứng âm thanh chân thực.

Thể hiện được khả năng sử dụng Unity trong việc xây dựng game từ khâu ý tưởng đến sản phẩm cuối cùng.

Phát triển kỹ năng lập trình, thiết kế game, và quản lý dự án.

CHƯƠNG 1. TỔNG QUAN

1.1. Nghiên cứu chính

Trong đồ án này là xây dựng một trò chơi bắn súng *Top-Down Shooting* sử dụng Unity, nghiên cứu các kỹ thuật thiết kế gameplay, lập trình AI (trí tuệ nhân tạo) cho kẻ thù, tối ưu hóa hiệu suất, và tạo ra môi trường chơi tương tác và hấp dẫn. Unity là công cụ chính được sử dụng, với những tính năng vượt trội trong việc phát triển game 2D và 3D, bao gồm hệ thống vật lý mạnh mẽ, khả năng tối ưu hóa cho nhiều nền tảng và giao diện người dùng (UI) trực quan.

1.2. Mục tiêu

Nghiên cứu và phát triển một game *Top-Down Shooting* cơ bản với các tính năng như di chuyển nhân vật, bắn đạn, tạo AI cho kẻ thù, thiết kế môi trường, và tối ưu hóa hiệu suất. Qua đó, đồ án sẽ không chỉ giúp làm quen với các kỹ thuật lập trình game mà còn cung cấp cái nhìn tổng quan về quy trình phát triển game từ ý tưởng đến sản phẩm hoàn chỉnh.

1.3. Các vấn đề cụ thể mà đồ án này tập trung giải quyết bao gồm:

- Xây dựng cơ chế di chuyển nhân vật và bắn đạn trong môi trường game.
- Thiết kế AI cho kẻ thù và lập trình hành vi của chúng sao cho hợp lý và thách thức người chơi.
- Tạo môi trường chơi.
- Tối ưu hóa hiệu suất để game có thể chạy mượt mà.
- Thiết kế giao diện người dùng (UI) dễ sử dụng và thân thiện.
- Nhờ vào việc sử dụng Unity và các công cụ mạnh mẽ đi kèm, đồ án sẽ giúp hiểu rõ hơn về quá trình phát triển game.

1.4. Tổng quan về Visual Studio

1.4.1. Khái niệm về Visual Studio

Visual Studio là một môi trường phát triển tích hợp (IDE) mạnh mẽ do Microsoft phát triển, được thiết kế để hỗ trợ lập trình viên trong việc xây dựng các ứng dụng cho nhiều nền tảng khác nhau như Windows, macOS, web, di động, và đám mây. Visual Studio hỗ trợ một loạt các ngôn ngữ lập trình, bao gồm **C#**, **C++**, **Python**, **JavaScript**, **TypeScript**, và nhiều ngôn ngữ khác.

IDE này cung cấp một loạt các công cụ tích hợp để viết mã, kiểm thử, gỡ lỗi, và triển khai ứng dụng, giúp tối ưu hóa hiệu quả làm việc của lập trình viên.

Hình 1.1 Visual Studio

1.5. Tổng quan về C#

1.5.1. Khái niệm về C#

C# (C Sharp) là một ngôn ngữ lập trình hướng đối tượng (OOP) được phát triển bởi **Microsoft** vào năm 2000 như một phần của nền tảng .NET Framework. C# được thiết kế để trở thành một ngôn ngữ đơn giản, hiện đại, và an toàn, kế thừa các ý tưởng từ các ngôn ngữ như **C++, Java**, nhưng cải tiến để tăng hiệu suất và giảm lỗi lập trình.

C# là một ngôn ngữ **đa năng**, phù hợp cho nhiều loại ứng dụng: từ phần mềm desktop, ứng dụng web, game (với Unity), đến các dịch vụ đám mây

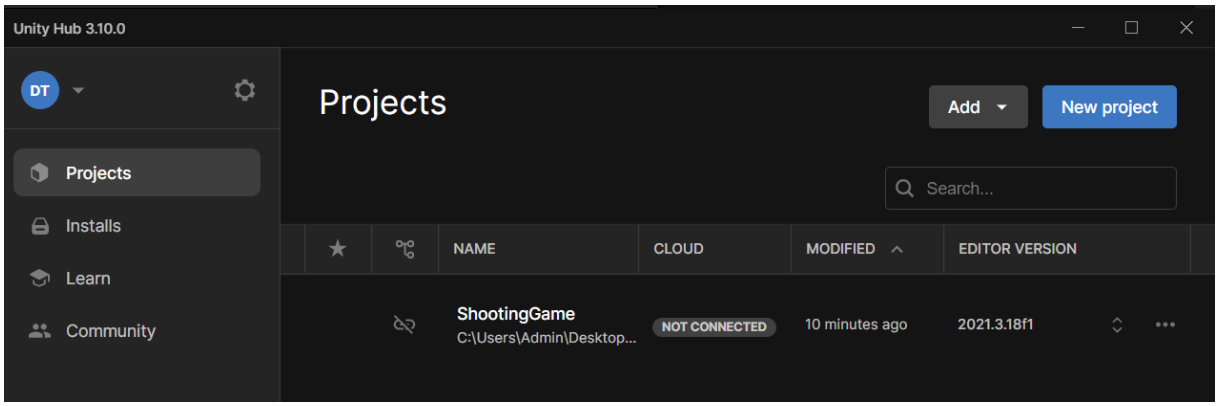
CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT

2.1. Các khái niệm trong Unity

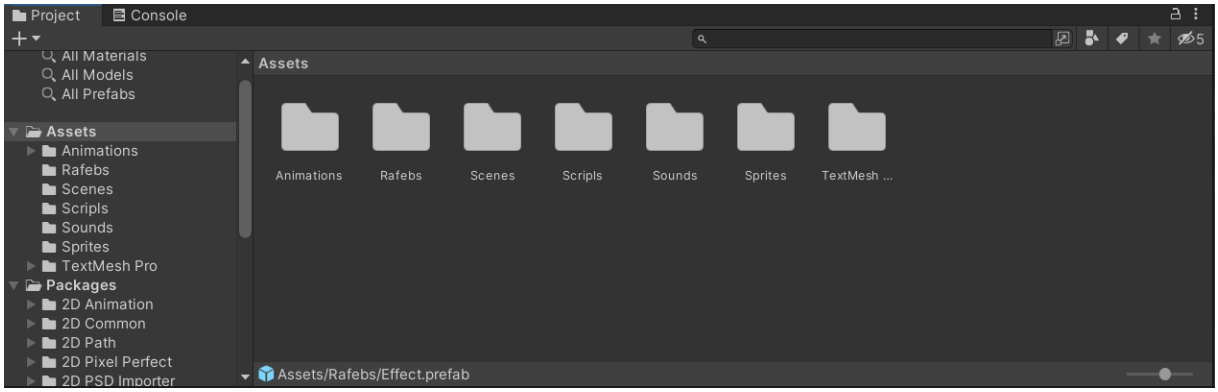
2.1.1. Project (Dự án)

Định nghĩa:

Trong Unity, **Project** là toàn bộ cấu trúc dự án mà bạn tạo ra khi bắt đầu phát triển một trò chơi hoặc ứng dụng. Dự án Unity bao gồm tất cả các tệp và tài nguyên như **Assets**, **Scenes**, **Scripts**, **Settings**, và các dữ liệu liên quan khác. Project chính là nơi bạn làm việc và nơi Unity lưu trữ tất cả thông tin cần thiết để xây dựng sản phẩm.



Hình 2.1 Project hiển thị tron unity Hub.



Hình 2.2 Project hiển thị tron unity Editor.

2.1.2. Scene (Cảnh)

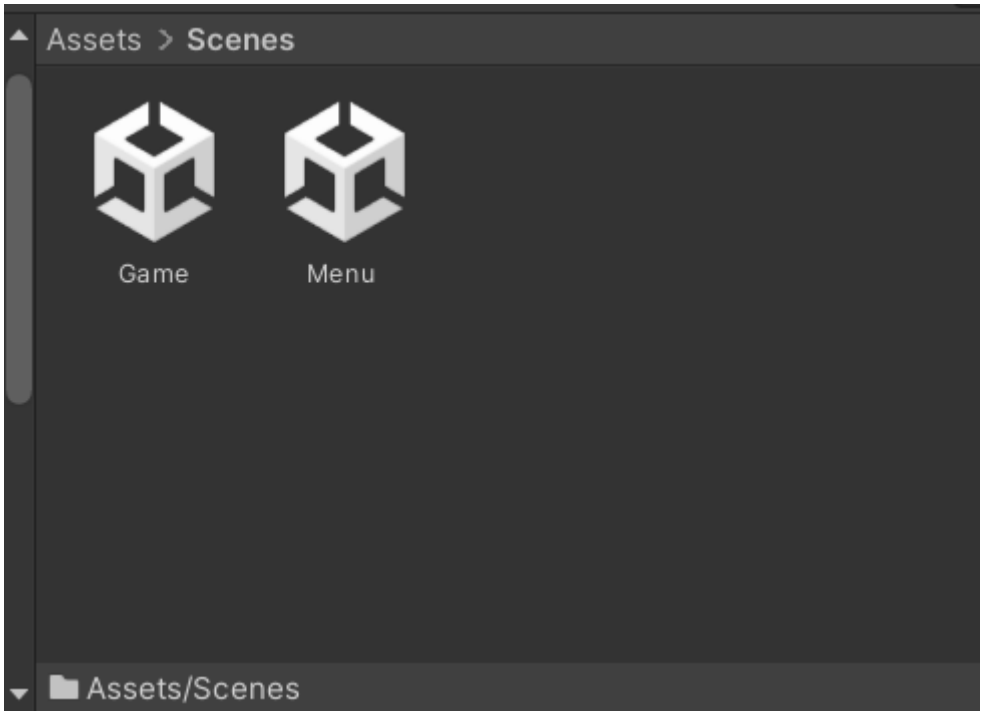
- **Định nghĩa:**

Scene là không gian làm việc trong Unity, nơi bạn đặt các đối tượng (GameObjects) để tạo ra trò chơi hoặc ứng dụng.

- **Chức năng:**

Scene có thể được coi như một cấp độ (level) hoặc một màn chơi trong game, hoặc có thể là giao diện chính của ứng dụng.

- Tính năng:
    - Có thể chứa đối tượng 2D hoặc 3D.
    - Có thể sử dụng nhiều Scene trong một dự án và chuyển đổi qua lại trong runtime.

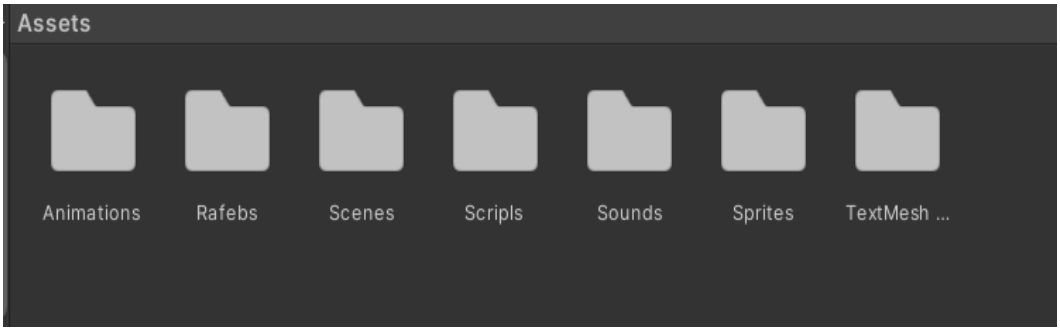


Hình 2.3 Các Scenes trong Assets

2.1.3. Assets (Tài nguyên)

 Định nghĩa:

Assets là tất cả các tài nguyên được sử dụng trong dự án Unity, bao gồm hình ảnh, âm thanh, mô hình 3D, tập lệnh, video, hiệu ứng hạt, và nhiều loại tài nguyên khác. Các Assets này là thành phần cơ bản để xây dựng trò chơi hoặc ứng dụng.



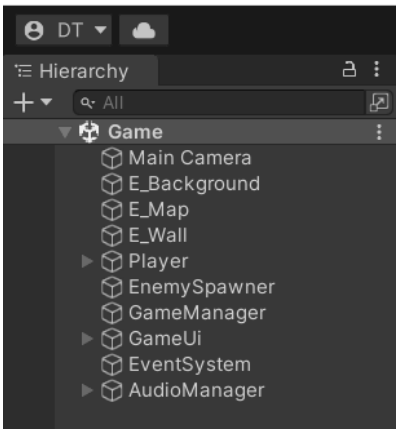
Hình 2.4 Assets

2.1.4. GameObject (Đối tượng trò chơi)

GameObject là thành phần cơ bản nhất trong Unity, đại diện cho mọi đối tượng xuất hiện trong Scene.

Một GameObject có thể là bất kỳ thứ gì, từ nhân vật, camera, ánh sáng, đến các vật thể vô tri như cây cối hoặc đồ vật..

Mỗi GameObject có thể chứa nhiều **Component** để định nghĩa hành vi hoặc thuộc tính của nó.



Hình 2.5 GameObject

2.1.5. Component (Thành phần)

Component là thành phần gắn vào GameObject để mở rộng tính năng hoặc định nghĩa hành vi cho nó

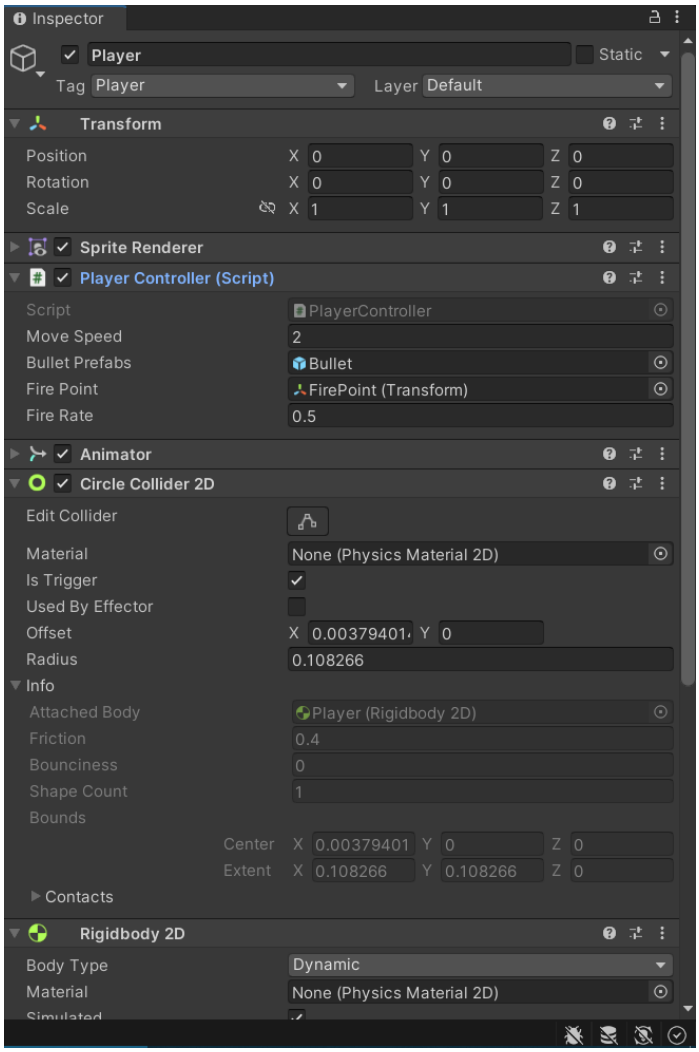
- **Các loại Component phổ biến**:

Transform: Xác định vị trí, xoay và tỷ lệ của GameObject trong Scene.

Rigidbody: Thêm tính chất vật lý như trọng lực và va chạm.

Collider: Xác định vùng va chạm cho đối tượng.

Script: Thành phần do lập trình viên tạo ra để định nghĩa hành vi tùy chỉnh cho GameObject.



Hình 2.6 Component

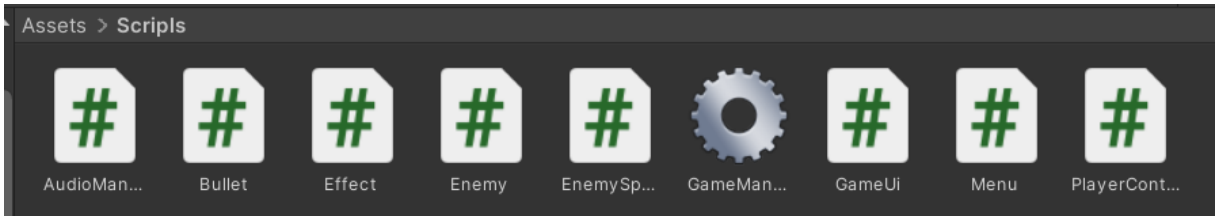
2.1.6. Script (Mã nguồn)

Script là một đoạn mã lập trình được viết bằng **C#**, được sử dụng để định nghĩa hành vi và logic cho GameObject.

Chức năng:

Tạo các tương tác như di chuyển, điều khiển nhân vật, hoặc xử lý các sự kiện trong game.

Kết hợp với các Component khác để điều chỉnh hành vi của GameObject.



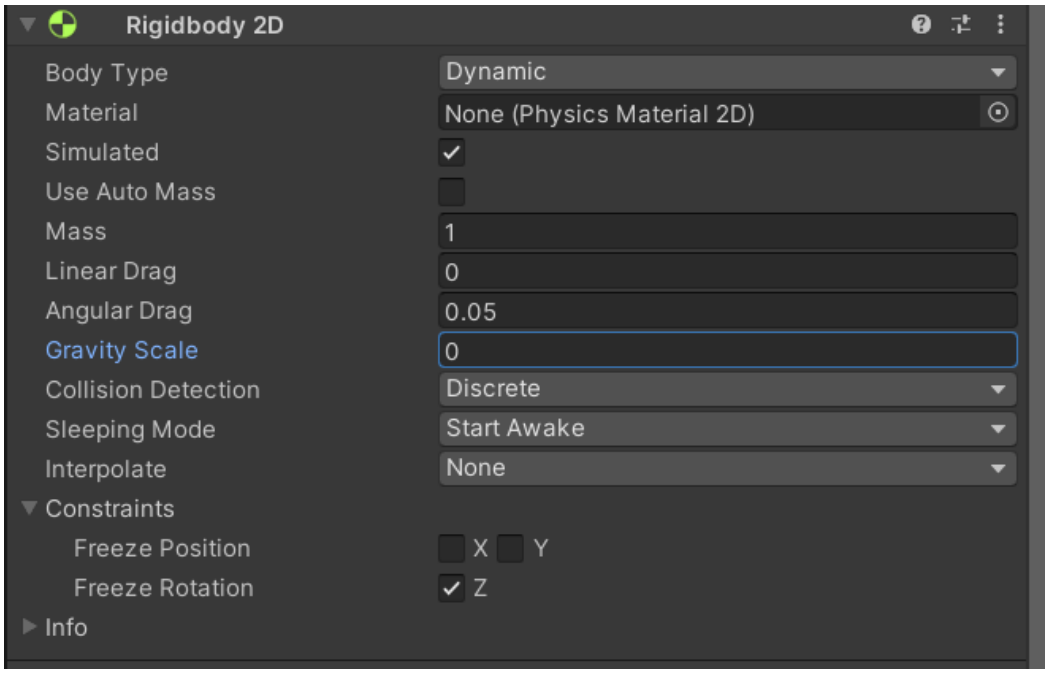
Hình 2.7 Script

2.1.7. Physics (Vật lý)

Unity tích hợp sẵn các engine vật lý như **PhysX** để mô phỏng trọng lực, va chạm, và chuyển động thực tế.

Các thành phần vật lý phổ biến:

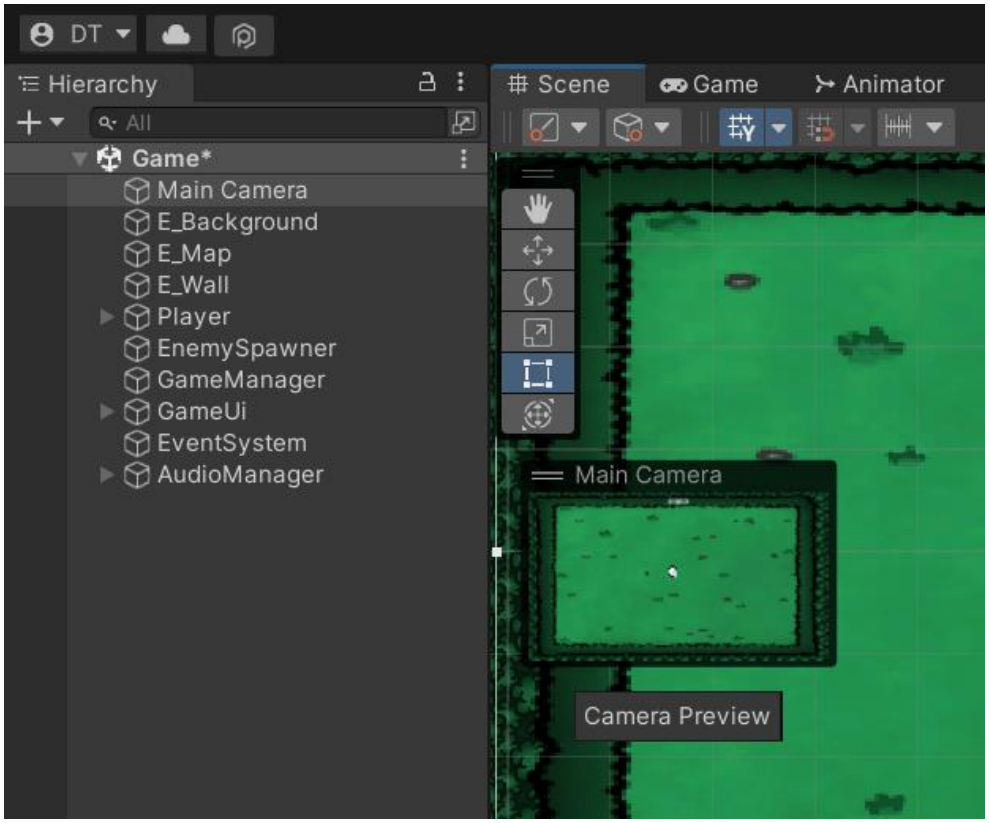
- **Rigidbody**: Áp dụng trọng lực và các lực vật lý.
- **Collider**: Xác định vùng va chạm.
- **Physics Material**: Quy định tính chất vật lý, như độ ma sát hoặc độ nảy.



Hình 2.8 Physics trong Rigidbody 2D

2.1.8. Camera

Camera là thành phần mô phỏng góc nhìn của người chơi trong Scene.



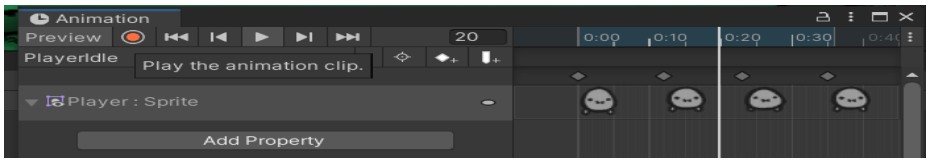
Hình 1.10 Camera

2.1.9. Animator và Animation

Unity hỗ trợ tạo các chuyển động (animation) cho đối tượng thông qua hệ thống Animator.

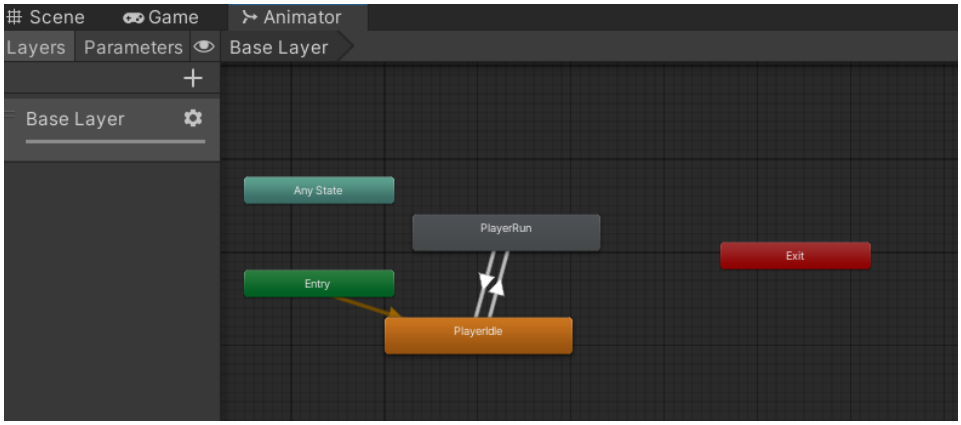
Thành phần chính:

- **Animator:** Thành phần quản lý các trạng thái chuyển động.



Hình 2.9 Cửa sổ Animation

- **Animation Clip**: Đoạn chuyển động cụ thể, chẳng hạn như đi bộ, nhảy

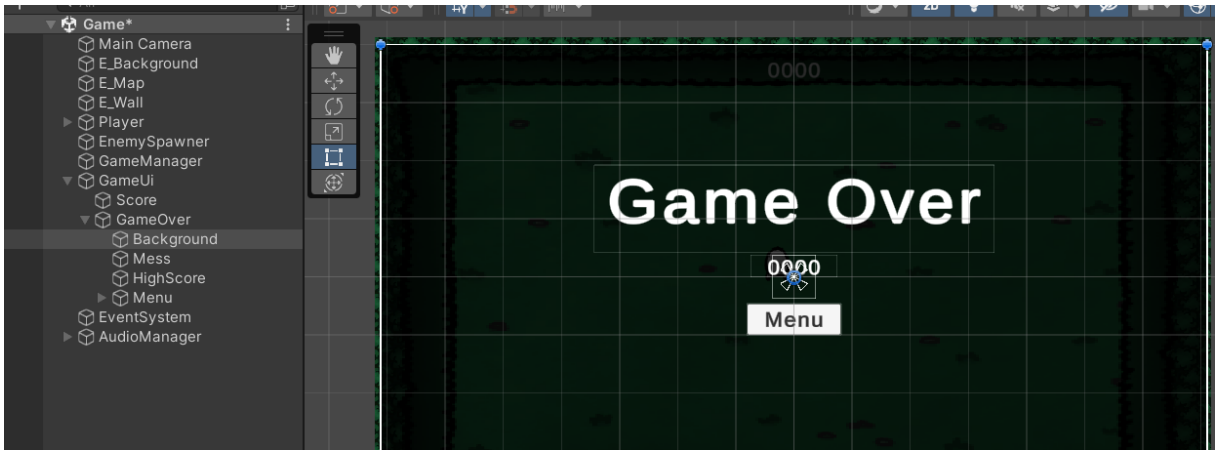


Hình 2.10 Cửa sổ Animator

2.1.10. UI (Giao diện người dùng)

Các thành phần chính:

- **Canvas**: Không gian để chứa các đối tượng UI.
- **Text**: Hiển thị văn bản.
- **Button**: Nút bấm tương tác.

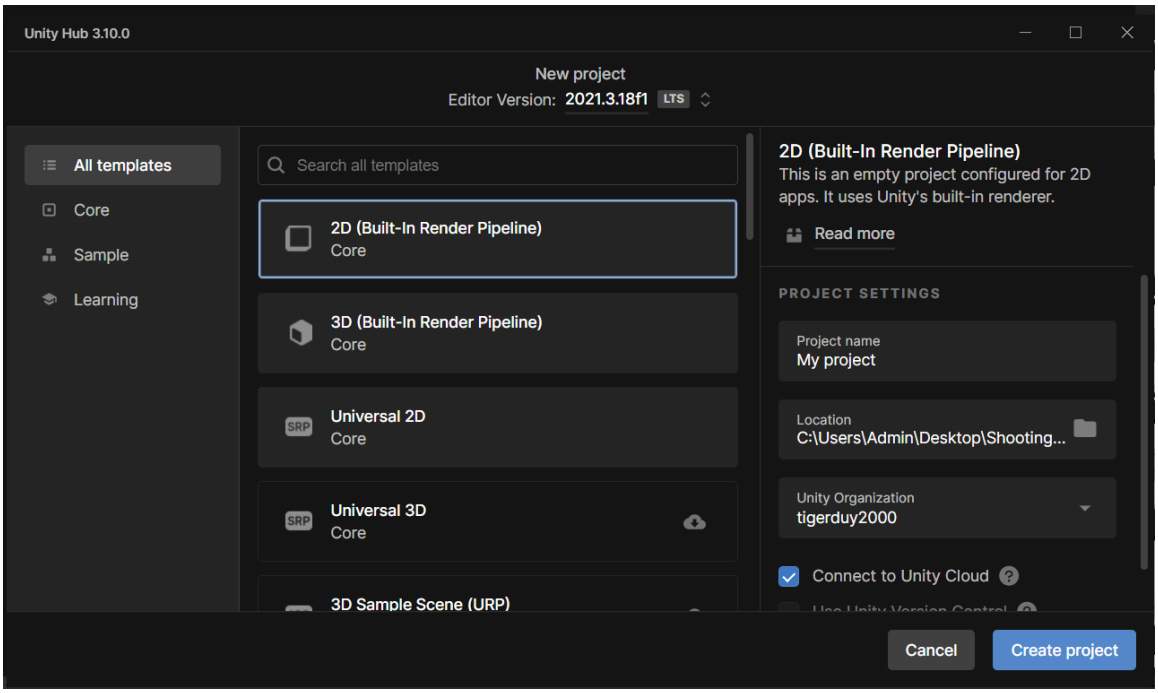


Hình 2.11 Game UI

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Tạo và cấu hình project

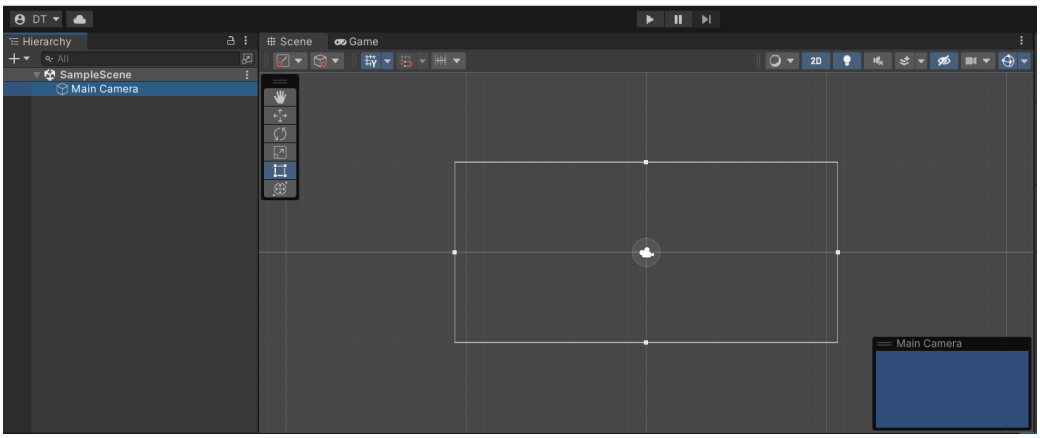
3.1.1. Bấm vào New project và chọn Templates cho project là 2D



Hình 3.1 Chọn Templates khi tạo project trong Unity Hub

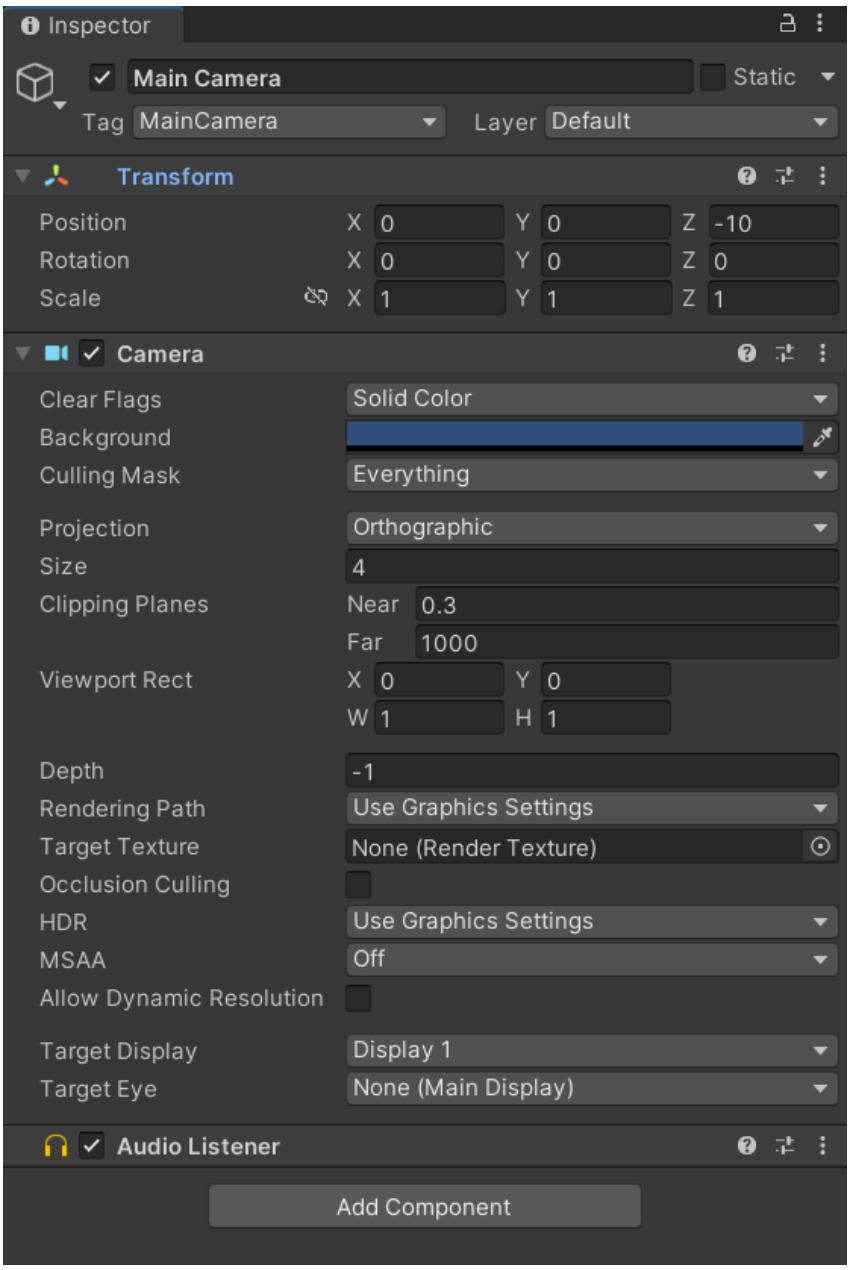
Sau khi tạo xong project sẽ được chuyển đến Unity Editor

- Main Camera là một đối tượng (GameObject) đặc biệt được tạo sẵn đại diện cho góc nhìn chính trong trò chơi



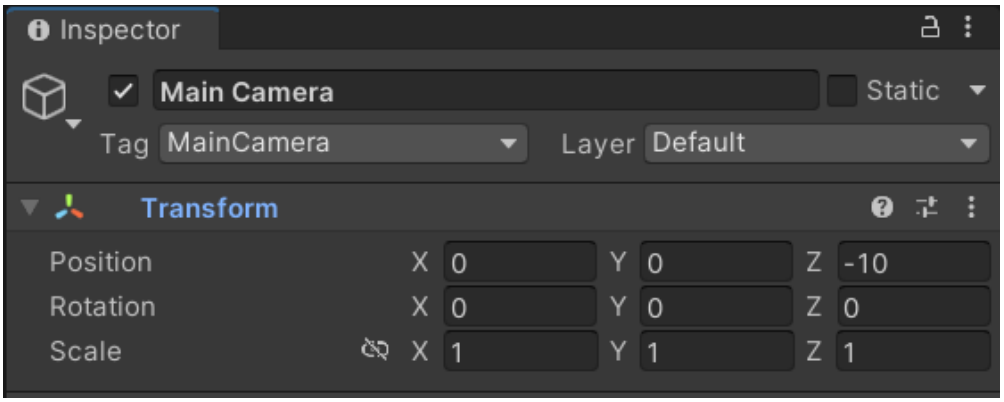
Hình 3.2 Main camera trong Unity Editor

- GameObject Main Camera có các Component sau: Transfrom, Camera, Audio Listener



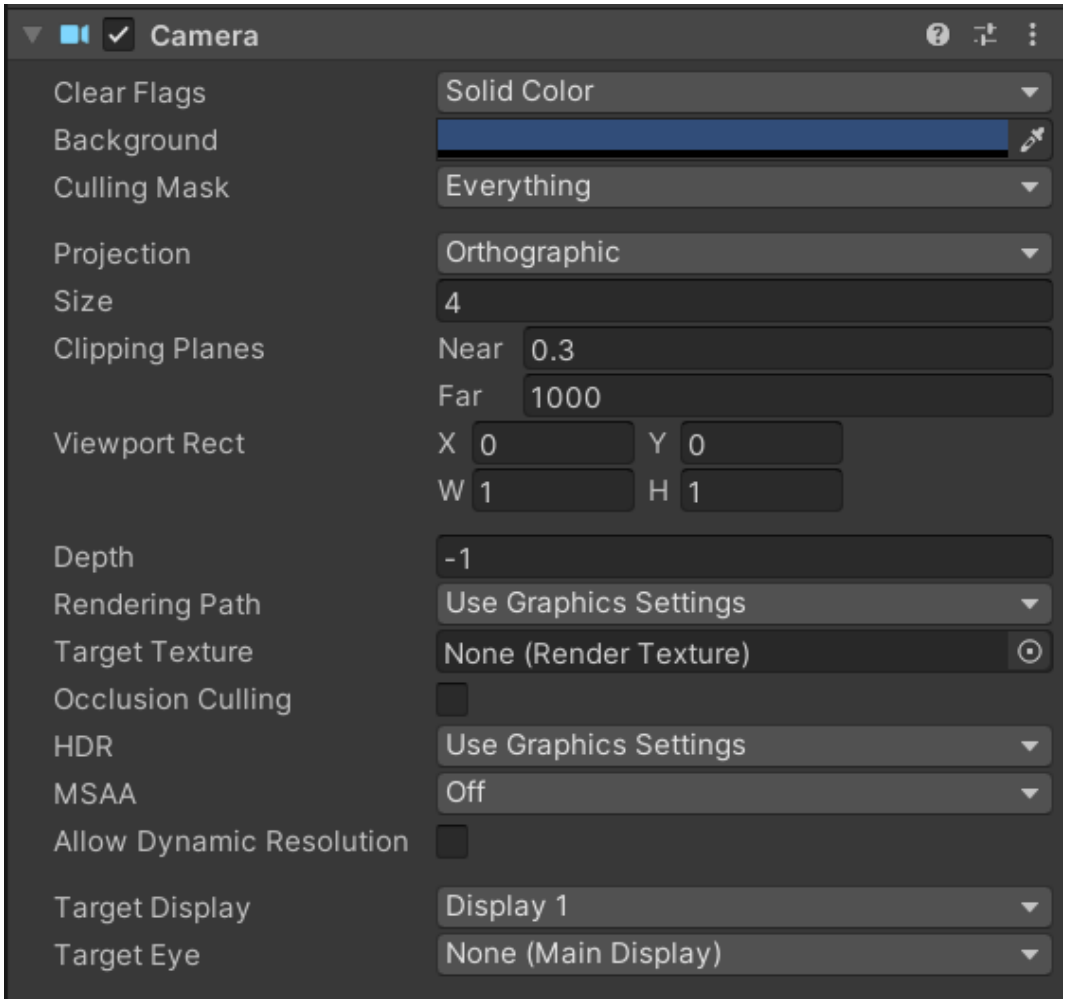
Hình 3.3 Các Component của Camera

+ **Transform** (Component quan trọng của mỗi GameObject) đại diện cho vị ví, hướng quay và tỷ lệ kích thước của một đối tượng. Mọi đối tượng trong Unity đều có một Transfrom và nó là một phần vô cùng quan trọng của mỗi đối tượng



Hình 3.4 Cấu hình Transfrom của Main Camera

+ **Camera** là thành phần chịu trách nhiệm cấu hình các thuộc tính cụ thể liên quan đến cách mà Camera hoạt động

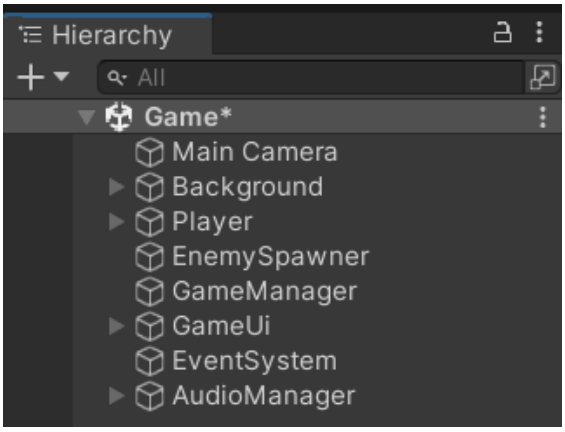


Hình 3.5 Cấu hình camera của Main Camera

+ **Audio Listener** Cho phép camera hoặc đối tượng khác nghe được âm thanh

3.2. 3.2 Game Play(Game Scenes)

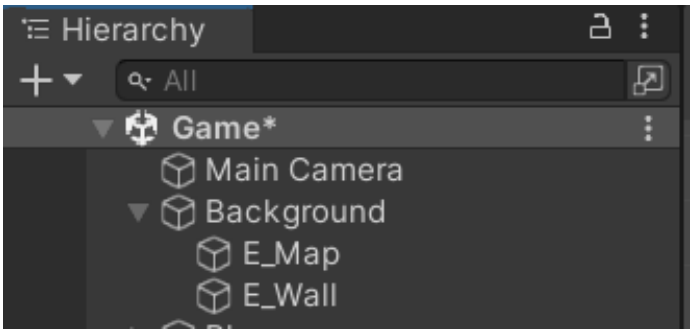
Phần này gồm các GameObject chính sau:



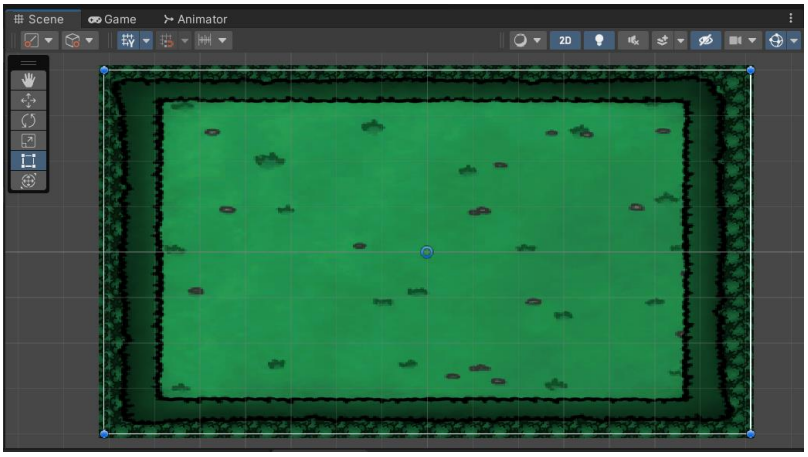
Hình 3.6 Các GameObject chính

3.2.1. Background

Trong GameOject này sẽ chứa các hình ảnh dùng để làm nền cho game



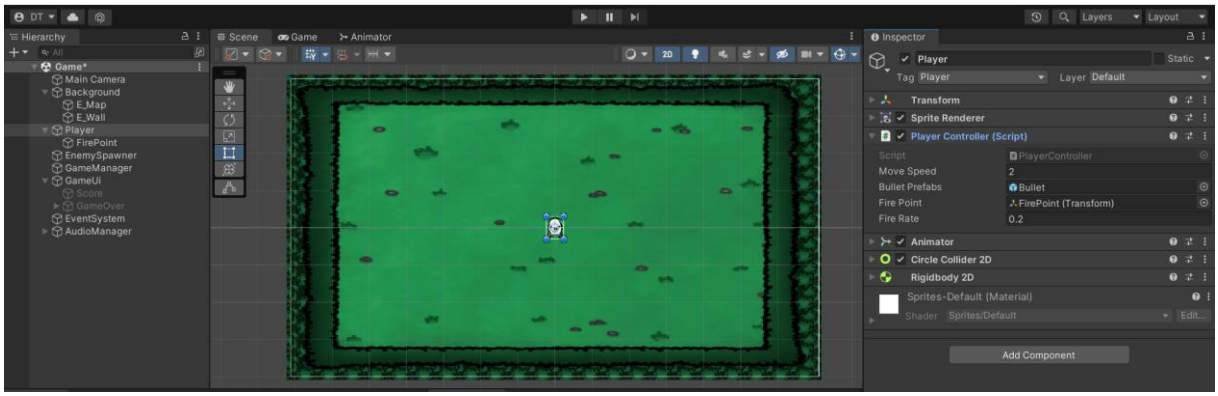
Hình 3.7 Các game object của Background



Hình 3.8 Background

3.2.2. Player

- GameObjec chứa Script Player Controller đảm nhiệm việc sử lý chuyển động các va chạm và chuyển động cũng như việc tấn công cho nhân vật. Chọn tag là Player để sử lý va chạm



Hình 3.9 GameObject Player và các Component

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class PlayerController : MonoBehaviour
{
    public float moveSpeed = 2f;
    private Rigidbody2D rb;
    private Vector2 moveInput;
    private Animator animator;


    private Camera mainCamera;
    //Bullet
    public GameObject BulletPrefabs;
    public Transform firePoint;
```

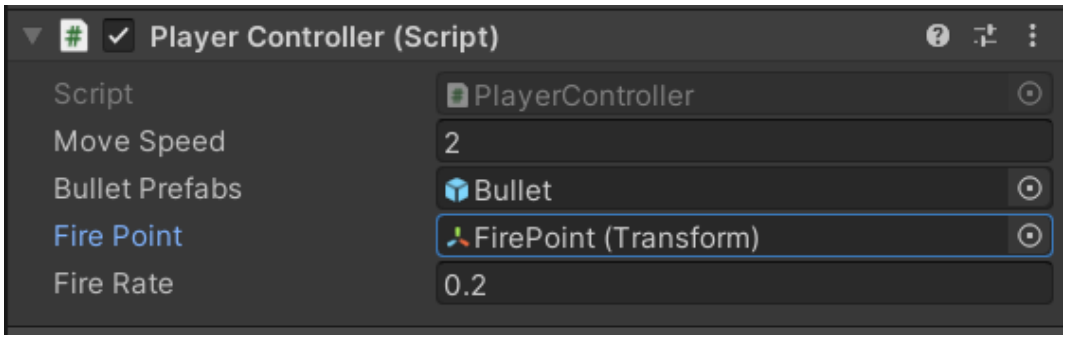
```
    public float fireRate = 0.5f;
    private float nextFireTime;

    // Start is called before the first frame update
    void Start()
    {
        // Lấy thành phần Rigidbody2D từ nhân vật
        rb = GetComponent<Rigidbody2D>();

        //Animtions
        animator = GetComponent<Animator>();

        //Camera
        mainCamera = Camera.main;
    }

    // Update is called once per frame
    void Update()
    {
        // Nhận input từ bàn phím hoặc tay cầm
        moveInput = new Vector2(Input.GetAxisRaw("Horizontal"),
Input.GetAxisRaw("Vertical")).normalized;

        //update animations
        UpdateAnimationState();

        //Bullet
        if(Input.GetMouseButton(0)&&Time.time>=nextFireTime)
```

```
        {
            Shoot();
            nextFireTime = Time.time + fireRate;
        }
    }

     void FixedUpdate()
    {
        // Áp dụng lực để di chuyển nhân vật
        rb.velocity = moveInput * moveSpeed;

        rotatePlayer();
    }

    void UpdateAnimationState()
    {
        if(moveInput.magnitude>0)
        {
            animator.SetBool("isRunning", true);
        }
        else
        {
            animator.SetBool("isRunning", false);
        }
    }

    void rotatePlayer()
    {
```

```
        Vector2 mousePosition =
mainCamera.ScreenToWorldPoint(Input.mousePosition);
        Vector2 lookDirection = mousePosition - rb.position;
        float angle = Mathf.Atan2(lookDirection.y,
lookDirection.x) * Mathf.Rad2Deg;
        rb.rotation = angle;


        Quaternion rotation = Quaternion.Euler(0, 0, angle);
        transform.rotation = rotation;


        if (transform.eulerAngles.z > 90 &&
transform.eulerAngles.z < 270)
            transform.localScale = new Vector3(2, -2, 0);
        else
            transform.localScale = new Vector3(2, 2, 0);
    }
    void Shoot()
    {
        Instantiate(BulletPrefabs, firePoint.position,
firePoint.rotation);
        AudioManager.instance.PlayShootingClip();
    }
}
```

Hình 3.10 Mã của Script Player Controller

- **FirePoint(Transfrom)** là vị trí đòn tấn công phát ra, được đặt nằm cạnh player



Hình 3.11 FirePoint

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class Bullet : MonoBehaviour

{

    public float speed = 10f;

    public float lifeTime = 2f;

    private Rigidbody2D rb;

    public GameObject Effect;


    void Start()

    {

        rb = GetComponent<Rigidbody2D>();

        rb.velocity = transform.right * speed;

        Destroy(gameObject, lifeTime);

    }


    private void OnTriggerEnter2D(Collider2D collision)

    {

        if (collision.tag == "Enemy")
```

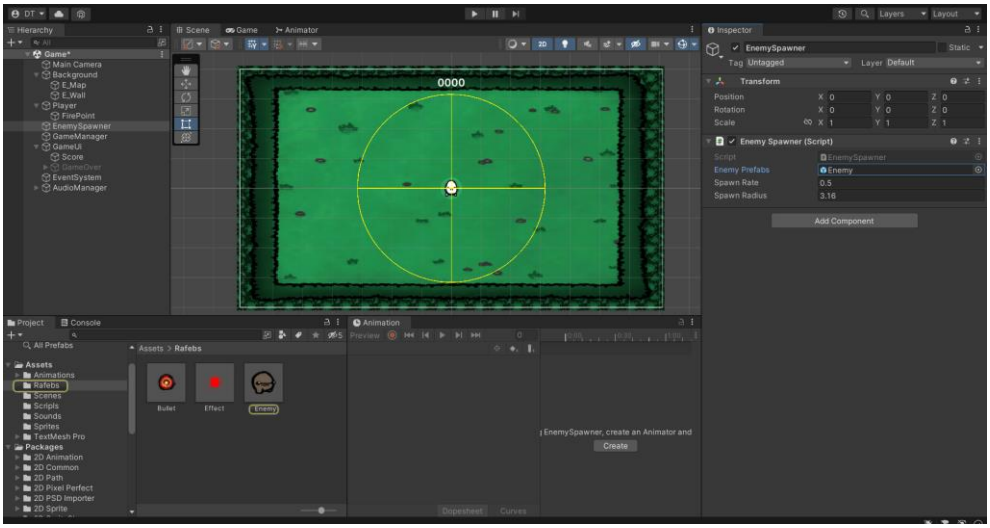
```
        {
            Instantiate(Effect, transform.position,
Quaternion.identity);
            if (collision.tag == "Enemy")
            {
                Destroy(gameObject);
            }
        }
    }
}
```

Hình 3.12 Mã của Script Bullet

-**Bullet** sau khi đưa lên cửa sổ Hierarchy để tùy chỉnh cho GameObject sau đó kéo xuống lại khung Assets sẽ trở thành một Prefabs để không hiển thị trong Camera. Bullet được kéo thả vào Script Player Cotroller để sinh sản Bullet

3.2.3. EnemySpawner

GameObject đảm nhiệm việc sinh sản ra các Enemy 



Hình 3.13 GameObject EnemySpawner và Component

- **Enemy** sau khi đưa lên cửa sổ Hierarchy để tùy chỉnh cho GameObject sau đó kéo xuống lại khung Assets sẽ trở thành một Prefabs để không hiển thị trong Camera. Enemy được kéo thả vào Script EnemySpawn để sinh sản

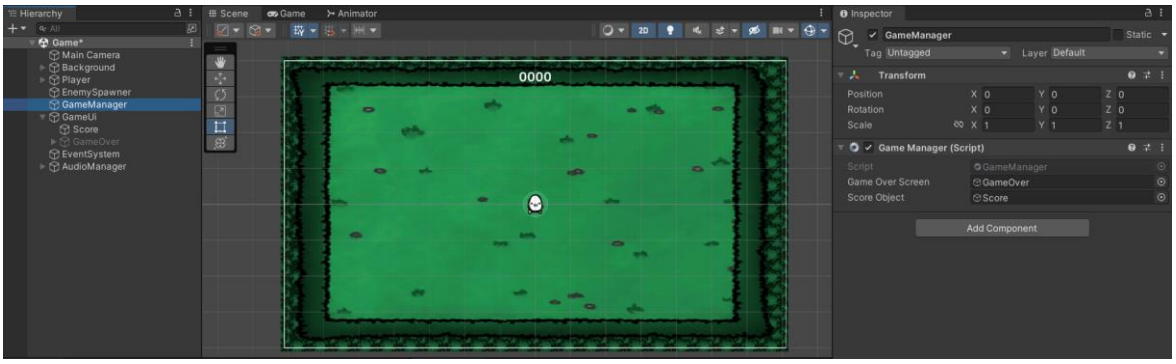
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemySpawner : MonoBehaviour
{
    public GameObject enemyPrefabs;
    public float spawnRate = 1.5f;
    public Vector3 spawnSize = new(5f, 5f, 5f);
    private float spawnTimer = 0f;



    // Start is called before the first frame update
    void Start()
    {


    }


    // Update is called once per frame
    void Update()
    {
        if (!GameManager.instance.GameOver())
        {
            spawnTimer += Time.deltaTime;
```

```
            if (spawnTimer >= spawnRate)
            {
                SpawnEnemy();
                spawnTimer = 0f;
            }
        }
    }

    private void OnDrawGizmosSelected()
    {
        Gizmos.color = Color.yellow;
        Gizmos.DrawWireCube(transform.position, spawnSize);
    }

    void SpawnEnemy()
    {
        Vector2 randomPosition = (Vector2)transform.position +
(Random.insideUnitCircle.normalized * spawnSize);

        Instantiate(enemyPrefabs, randomPosition,
Quaternion.identity);
    }
}
```

Hình 3.14 Mã của EnemySpawner

3.2.4. GameMannager

GameObject dùng để chứa các sự kiện trong game như: score (điểm số) và GameOver



Hình 3.15 GameManager và Componet

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class GameManager : MonoBehaviour

{

    public static GameManager instance;

    private int score = 0;

    private bool isGameOver = false;

    public GameObject GameOverScreen;

    public GameObject ScoreObject;


    private void Awake()

    {

        if(instance==null)

        {

            instance = this;

        }
```

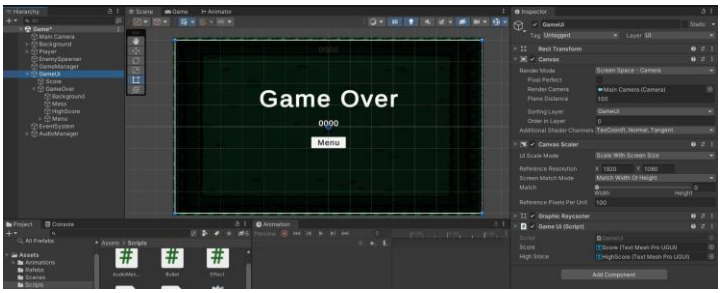
```
        else
        {
            Destroy(gameObject);
        }
    }

    // Start is called before the first frame update
    void Start()
    {

    }

    void Update()
    {
        if(isGameOver)
        {
            PlayerDie();
        }
    }

    public void AddScore(int scoreValue)
    {
        score += scoreValue;
    }

    public int Getscore()
    {
        return score;
```

```
    }


    public void EnemySkillPlayer()
    {
        isGameOver = true;
    }


    public void PlayerDie()
    {
        GameOverScreen.SetActive(true);
        ScoreObject.SetActive(false);
    }


    public bool GameOver()
    {
        return isGameOver;
    }
}
```

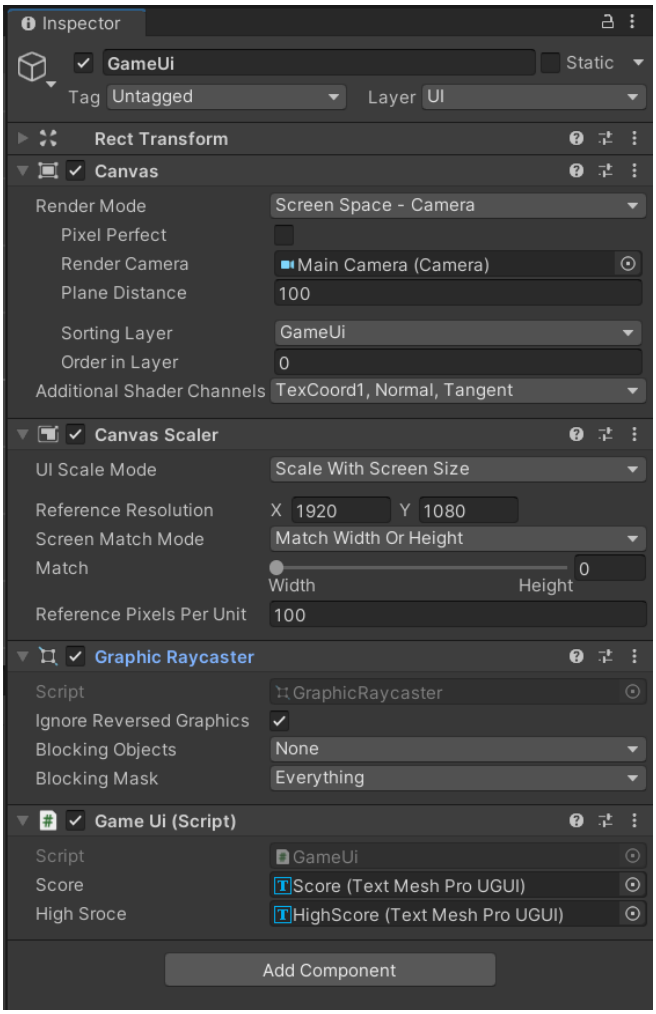
Hình 3.16 Mã của GameManager

3.2.5. GameUi

- GameObject đảm nhận việc hiển thị các sự kiện như Menu, điểm số và thông báo kết thúc game



Hình 3.17 GameUi và các Component



Hình 3.18 Cấu hình các Componet của GameUi

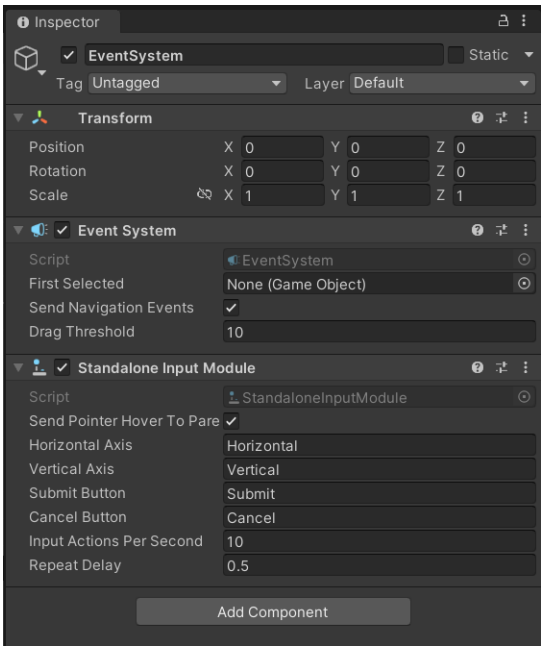
```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.SceneManagement;


public class GameUi : MonoBehaviour

{

    public TextMeshProUGUI Score;

    public TextMeshProUGUI HighSroce;
```

```
    // Update is called once per frame
    void Update()
    {
        Score.SetText(GameManager.instance.Getscore().ToString());

HighSroce.SetText(GameManager.instance.Getscore().ToString());
    }

    public void Menu()
    {
        SceneManager.LoadScene("Menu");
    }
}
```

Hình 3.19 Mã của GameUi

3.2.6. EventSystem

- GameObject dùng để phát âm thanh cho game



Hình 3.20 Cấu hình của EventSystem

3.2.7. AudioManager

- GameObject dùng để chứa các file âm thanh như nhạc nền và âm thanh hiệu ứng được sử dụng trong game

Hình 3.21 AudioManager và Component

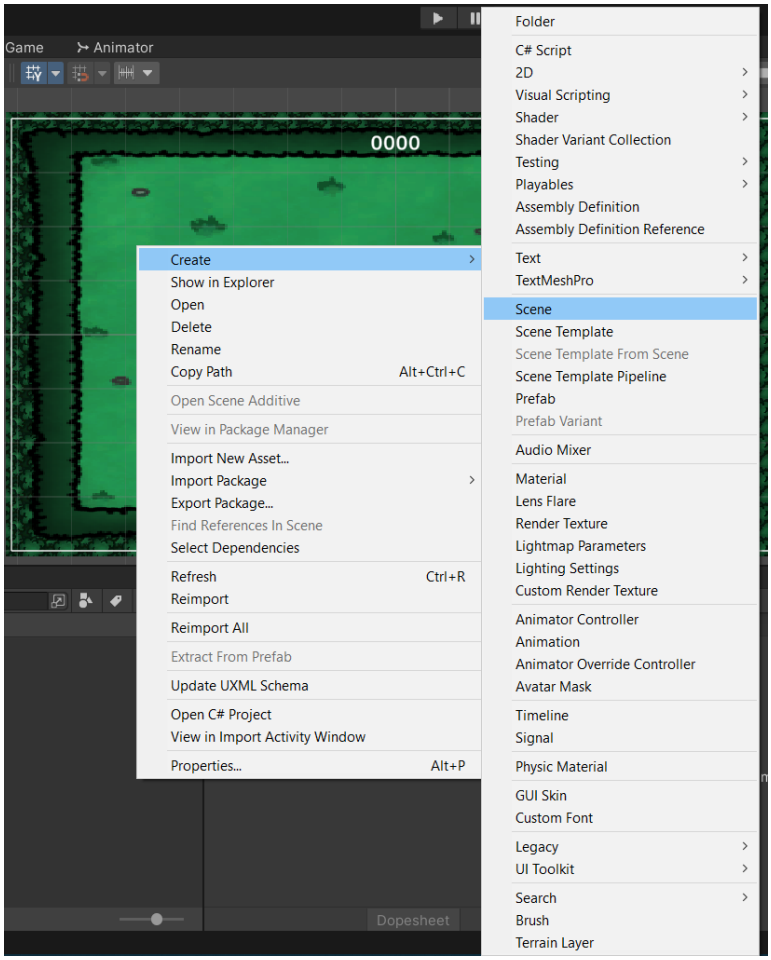
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class AudioManager : MonoBehaviour
{
    public static AudioManager instance;
    public AudioSource BackGroundsource;
    public AudioSource sfxSource;


    public AudioClip ShootClip;
    public AudioClip deadClip;
    public AudioClip backgroundClip;


    private void Awake()
    {
        if(instance==null)
```

```
        {
            instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
        PhatNhacNen();
    }
    void PhatNhacNen()
    {
        BackGroundsource.clip = backgroundClip;
        BackGroundsource.Play();
    }

    public void PlayShootingClip()
    {
        sfxSource.PlayOneShot(ShootClip);
    }
    public void PlayDeadClip()
    {
        sfxSource.PlayOneShot(deadClip);
    }
}
```

Hình 3.22 Mã của AudioManager

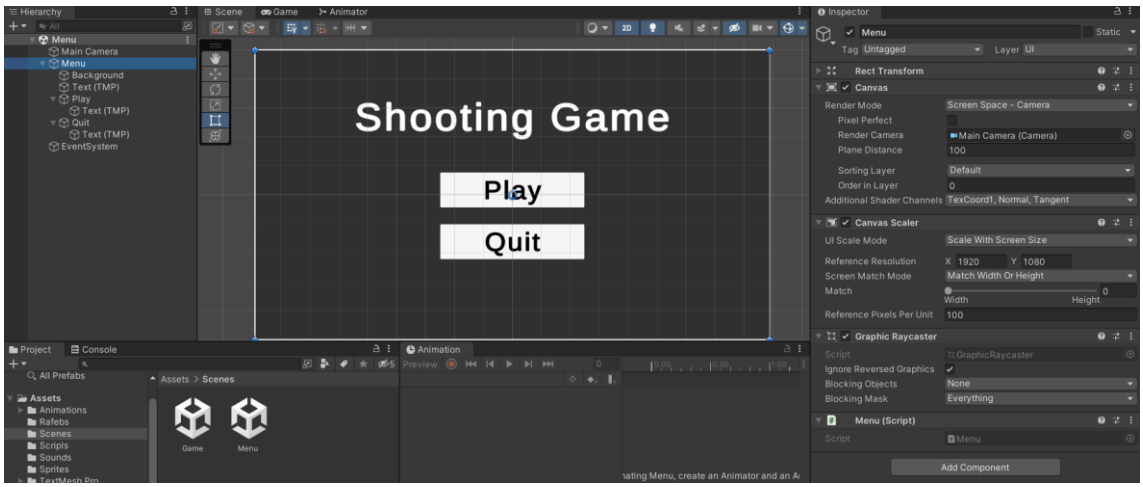
3.3. Menu(Menu Scenes)

Tạo Scenes mới để chứa màn hình Menu



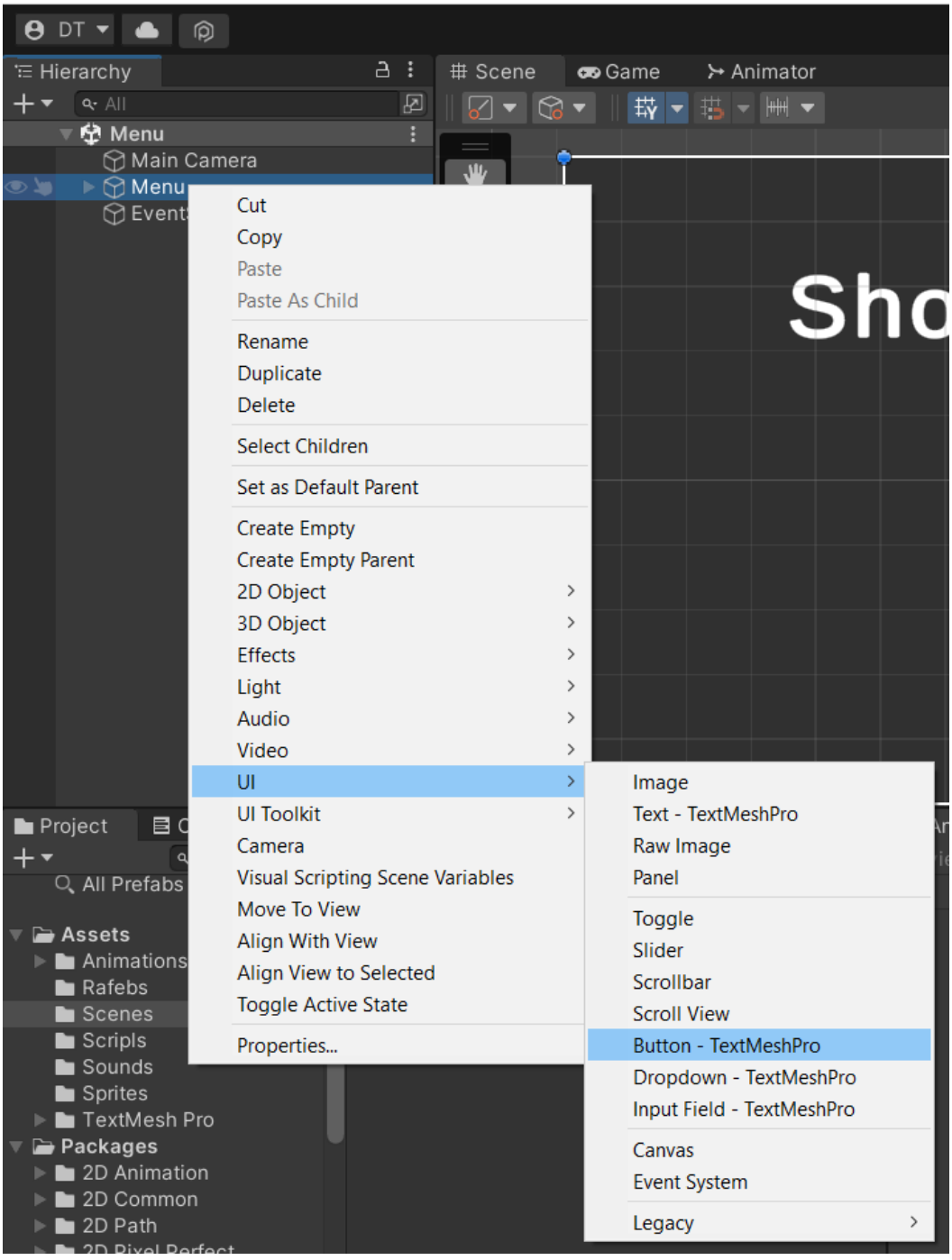
Hình 3.23 Tạo Scenes Menu

- GameObject Menu được dùng để chứa các GameObject khác



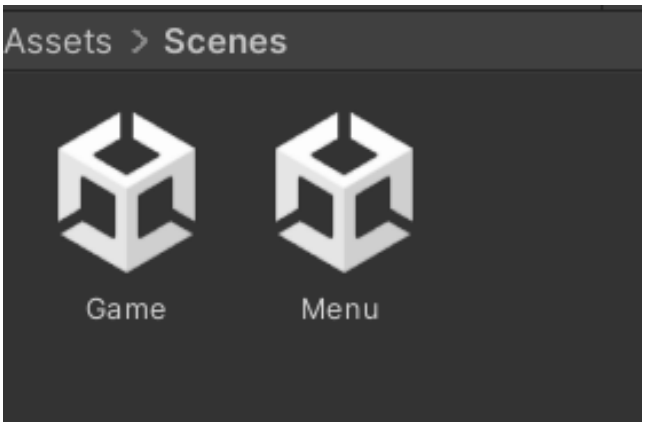
Hình 3.24 Menu và các Component

- Tạo Các nút bấm Play và Quit

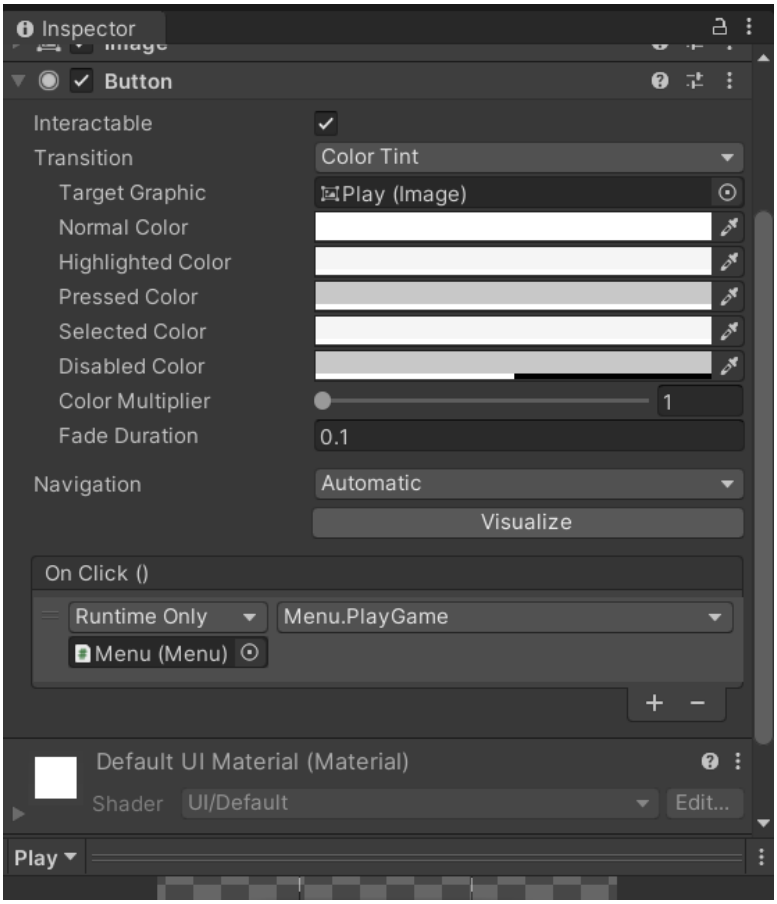


Hình 3.25 Tạo nút Play và Quit

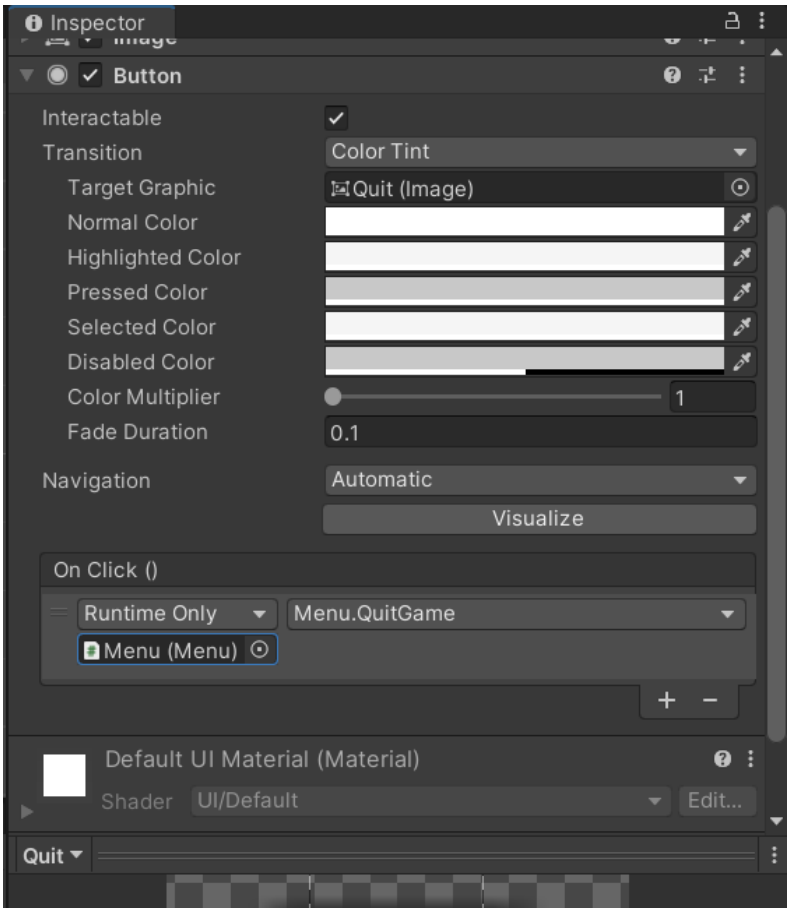
+ Nút Play và Quit đượchoạt động thông qua việc thay đổi Scenes.



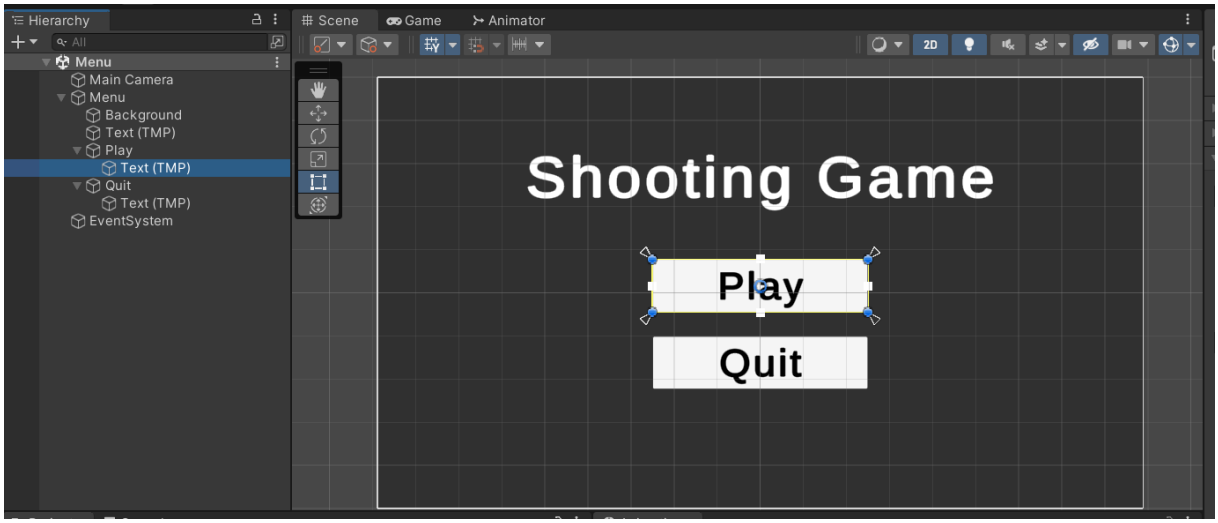
Hình 3.26 Scens



Hình 3.27 Cấu hình Click của nút Play



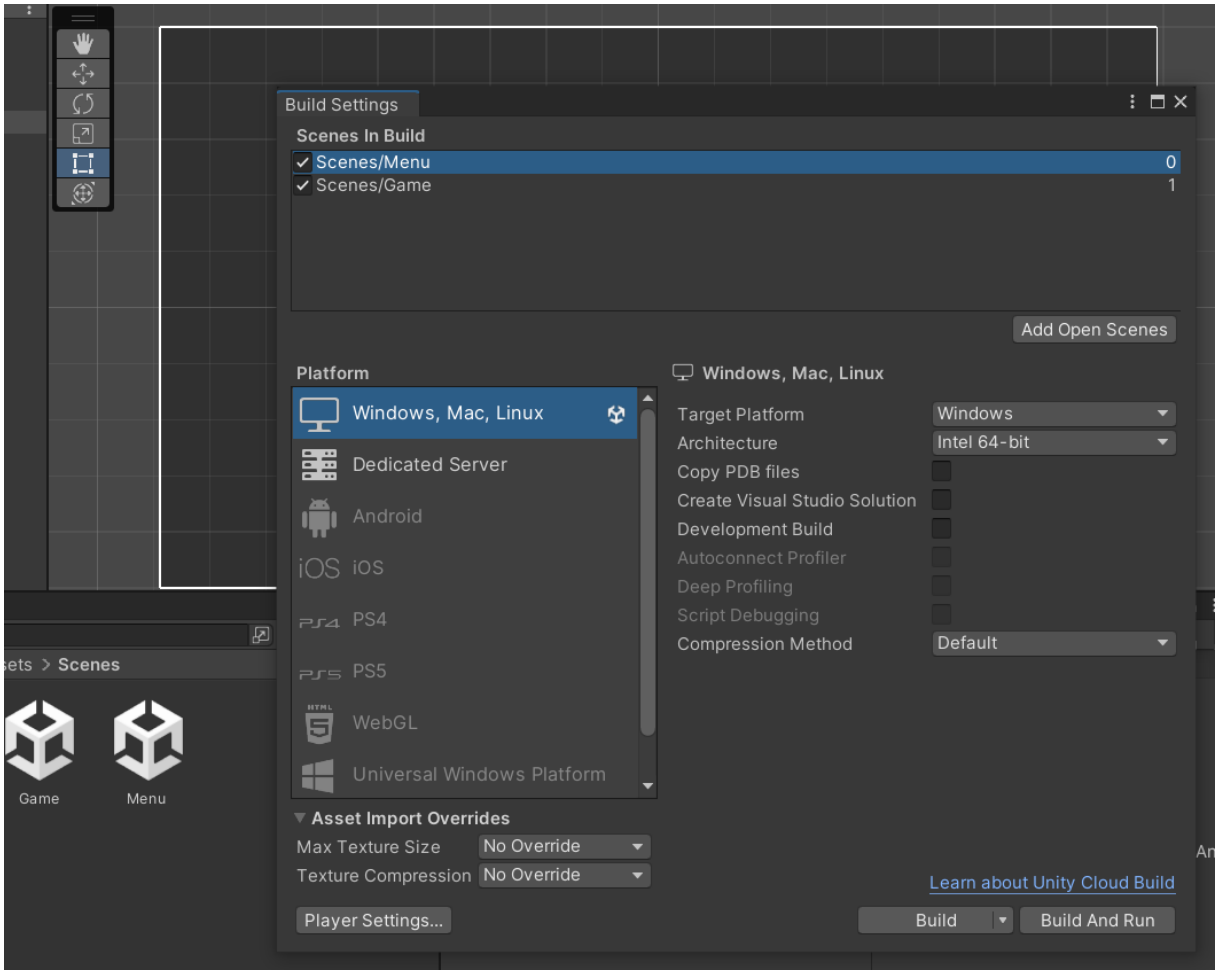
Hình 3.28 Cấu hình Click của nút Quit



Hình 3.29 Nút Play và Quit sau khi tạo

3.4. Cấu hình khi bulid

Vào File → Bulid Settings : Chỉnh sửa thứ tự Scenes bằng cách xóa dòng Scenes có sẵn trong khung Scenes In Bulid và thay thế bằng cách kéo thả Scenes mới vào như hình. Tiếp tục vào Player Settings để điền các thông tin và thêm Icon nếu muốn.



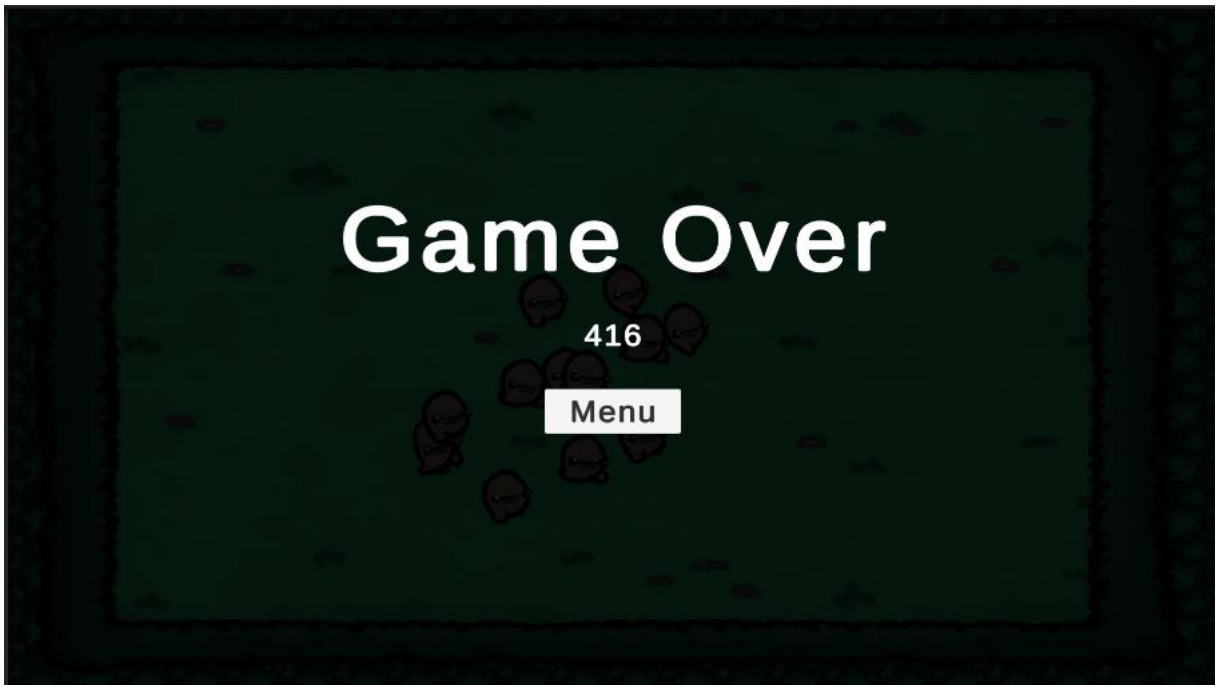
Hình 3.30 Cấu hình khi bulid

CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

Thành công tạo và hoàn thiện Game thể loại Top-dowm Shooting và build thành công với 3 cảnh chính như sau:

Hình 4.1 Menu game

Hình 4.2 Cảnh trong game



Hình5.6 Game Over

Link Github bài nộp :

https://github.com/TigerDuy/csn_da22ttd_nguyenthanhduy_110122062_lamgametrenunity

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận:

Hoàn thành thiết kế và hoàn thiện một trò chơi.

Tiếp thu được nhiều kiến thức và kinh nghiệm trong quá trình thiết kế và hoàn thiện trò chơi

Nâng cao kỹ năng lập trình game và kỹ năng thiết kế game, đồ họa, âm thanh,…

Hướng phát triển:

Tích hợp thêm một số chức năng như: Setting, nâng cấp, chọn nhân vật, xếp hạng điểm số,…

Thêm nhiều chế độ, số lượng nhân vật và nhiều loại enemy hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

https://viblo.asia/newest

https://www.wikipedia.org/

https://www.youtube.com/user/Brackeys

https://gamedevacademy.org/

https://learn.microsoft.com/en-us/dotnet/csharp/

https://docs.unity3d.com/ScriptReference/