Bonus Project Report

Space Shooter Game

Nada Hamada

20101043

User Manual:

This project is 3D Shooting Game designed using Unity.

User Controls:

- Use the left and right arrow keys to hover from left to right.
- Use up and down arrow keys to fly up and down within a bounded region.
- Use the space bar to shoot lasers.

How to win:

- Each level has a set number of enemies (red blocks) being spawned, in order to win you must shoot lasers at all of the spawned enemies.
- As you progress, the number of enemies spawned increases, as well as their speed.
- You have only one life in this game. So, if an enemy hits you, you lose. You can then choose to start over from level 1, or go back to the main menu.

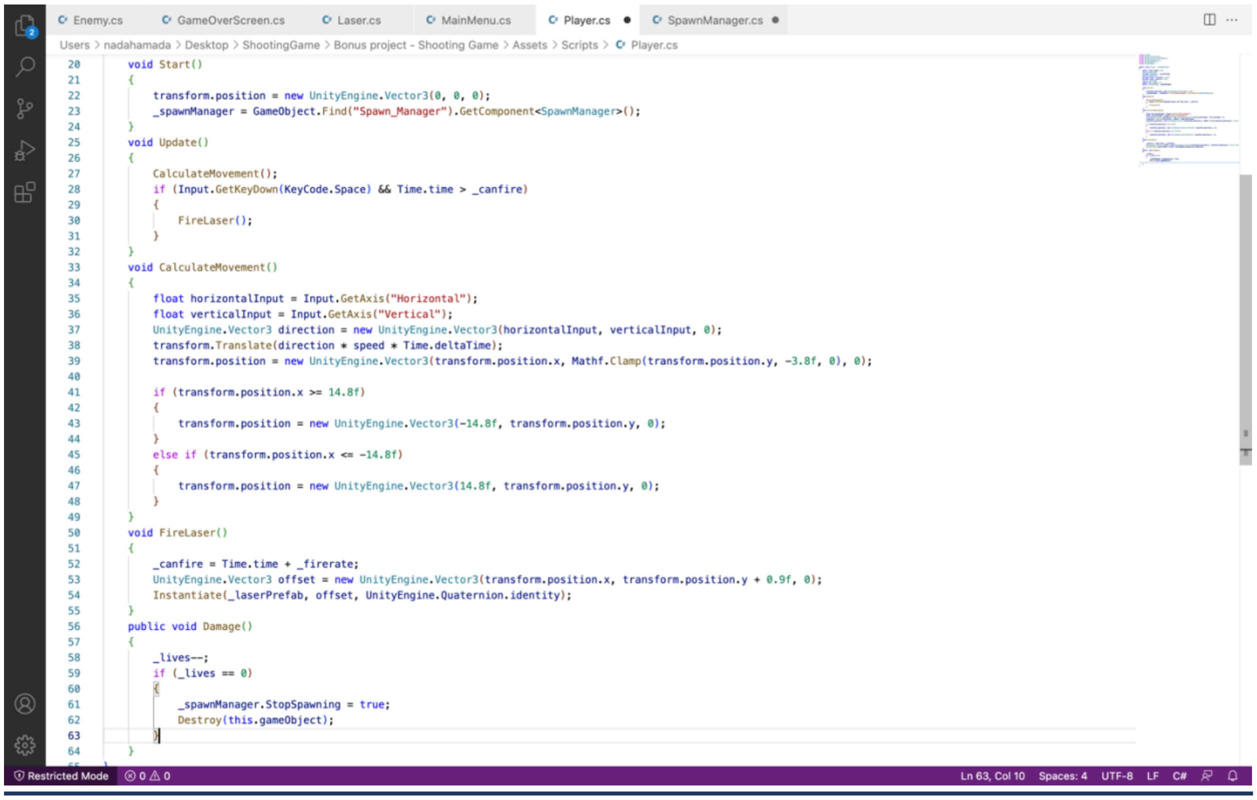
Description Report:

Player:

- CalculateMovement()
  Called in Update() in order to calculate the player's current (triggered per frame) position and move (within screen bounds) it upon user control.

- FireLaser()
  Gives the player ability to shoot lasers (instantiate) at a certain rate.

- Damage()
  Weaken the player's life when it collides with an enemy prefab.


Enemy.cs
GameOverScreen.cs
Laser.cs
MainMenu.cs
Player.cs
SpawnManager.cs
Users > nadahamada > Desktop > ShootingGame > Bonus project - Shooting Game > Assets > Scripts > Player.cs
Restricted Mode
0 0
Ln 63, Col 10 Spaces: 4 UTF-8 LF C#

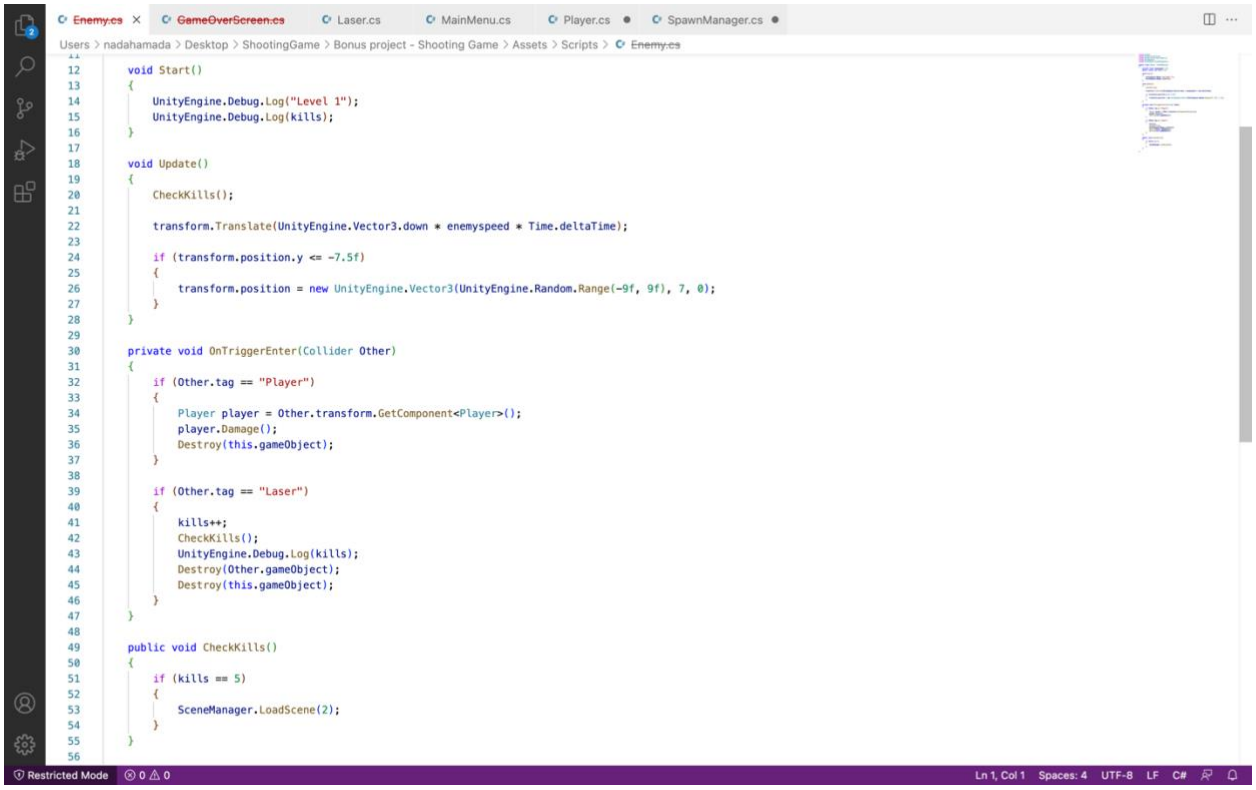
```
void Start()
{
    transform.position = new UnityEngine.Vector3(0, 0, 0);
    _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager>();
}
void Update()
{
    CalculateMovement();
    if (Input.GetKeyDown(KeyCode.Space) && Time.time > _canfire)
    {
        FireLaser();
    }
}
void CalculateMovement()
{
    float horizontalInput = Input.GetAxis("Horizontal");
    float verticalInput = Input.GetAxis("Vertical");
    UnityEngine.Vector3 direction = new UnityEngine.Vector3(horizontalInput, verticalInput, 0);
    transform.Translate(direction * speed * Time.deltaTime);
    transform.position = new UnityEngine.Vector3(transform.position.x, Mathf.Clamp(transform.position.y, -3.8f, 0), 0);

    if (transform.position.x >= 14.8f)
    {
        transform.position = new UnityEngine.Vector3(-14.8f, transform.position.y, 0);
    }
    else if (transform.position.x <= -14.8f)
    {
        transform.position = new UnityEngine.Vector3(14.8f, transform.position.y, 0);
    }
}
void FireLaser()
{
    _canfire = Time.time + _firerate;
    UnityEngine.Vector3 offset = new UnityEngine.Vector3(transform.position.x, transform.position.y + 0.9f, 0);
    Instantiate(_laserPrefab, offset, UnityEngine.Quaternion.identity);
}
public void Damage()
{
    _lives--;
    if (_lives == 0)
    {
        _spawnManager.StopSpawning = true;
        Destroy(this.gameObject);
    }
}
```

Enemy:

- CheckKills()
  Check if number of kills sufficient for level up.

- UpdateKills()
  Increases number of kills upon laser-enemy collision.

- OnTriggerEnter()
  Checks the kind of GamObject the enemy collided with. If it collided with laser, the enemy dies. If it collided with the player, the player's life decreases.


Enemy.cs
GameOverScreen.cs
Laser.cs
MainMenu.cs
Player.cs
SpawnManager.cs
Users > nadahamada > Desktop > ShootingGame > Bonus project - Shooting Game > Assets > Scripts > Enemy.cs
Restricted Mode
Ln 1, Col 1   Spaces: 4   UTF-8   LF   C#

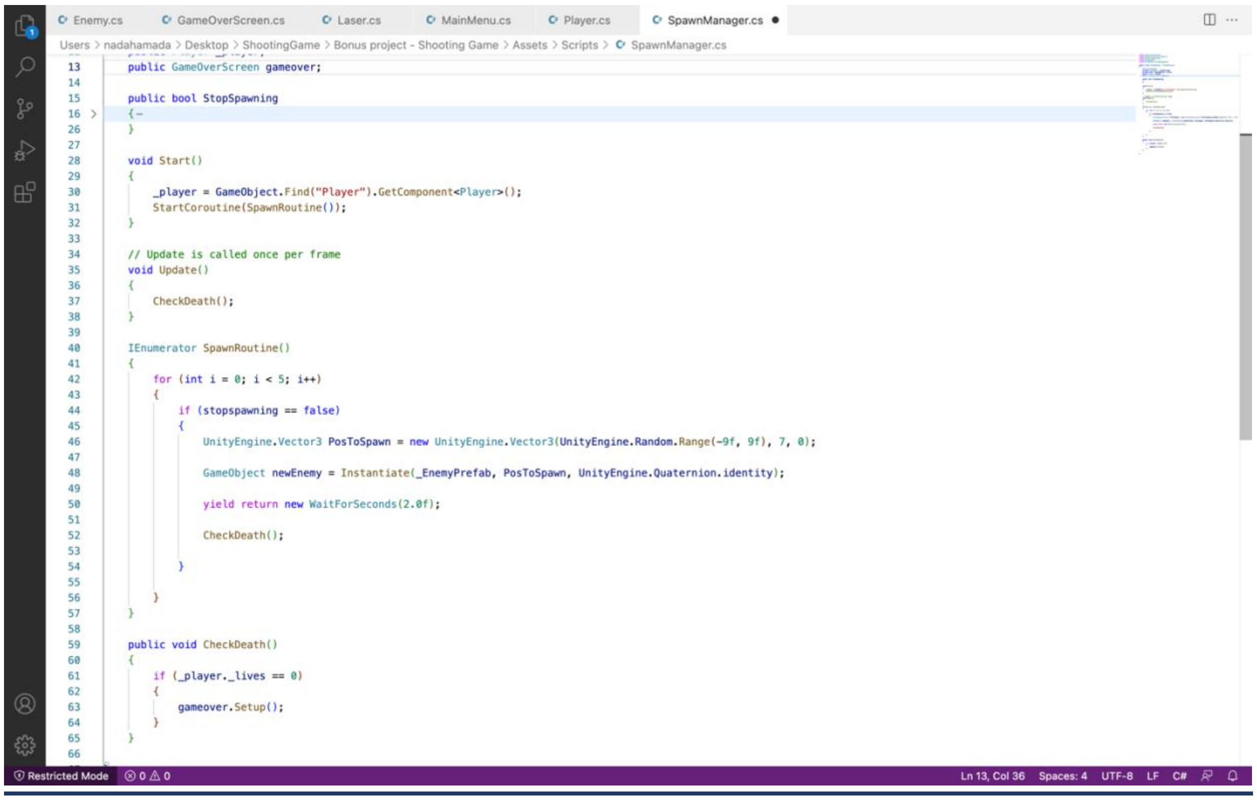
```
    void Start()
    {
        UnityEngine.Debug.Log("Level 1");
        UnityEngine.Debug.Log(kills);
    }

    void Update()
    {
        CheckKills();

        transform.Translate(UnityEngine.Vector3.down * enemyspeed * Time.deltaTime);

        if (transform.position.y <= -7.5f)
        {
            transform.position = new UnityEngine.Vector3(UnityEngine.Random.Range(-9f, 9f), 7, 0);
        }
    }

    private void OnTriggerEnter(Collider Other)
    {
        if (Other.tag == "Player")
        {
            Player player = Other.transform.GetComponent<Player>();
            player.Damage();
            Destroy(this.gameObject);
        }

        if (Other.tag == "Laser")
        {
            kills++;
            CheckKills();
            UnityEngine.Debug.Log(kills);
            Destroy(Other.gameObject);
            Destroy(this.gameObject);
        }
    }

    public void CheckKills()
    {
        if (kills == 5)
        {
            SceneManager.LoadScene(2);
        }
    }

```

Note: The difference between Enemy, EnemyLevel2 and EnemyLevel3 is that each script checks for a different number of total kill upon laser-enemy collision.

Spawn Manager:

- IEnumerator SpawnRoutine()
  Instantiates fixed number of enemies for each level at random positions in the screen with 2 second time interval between each spawn.

- CheckDeath()
  Checks for GameOverPage trigger (player death)

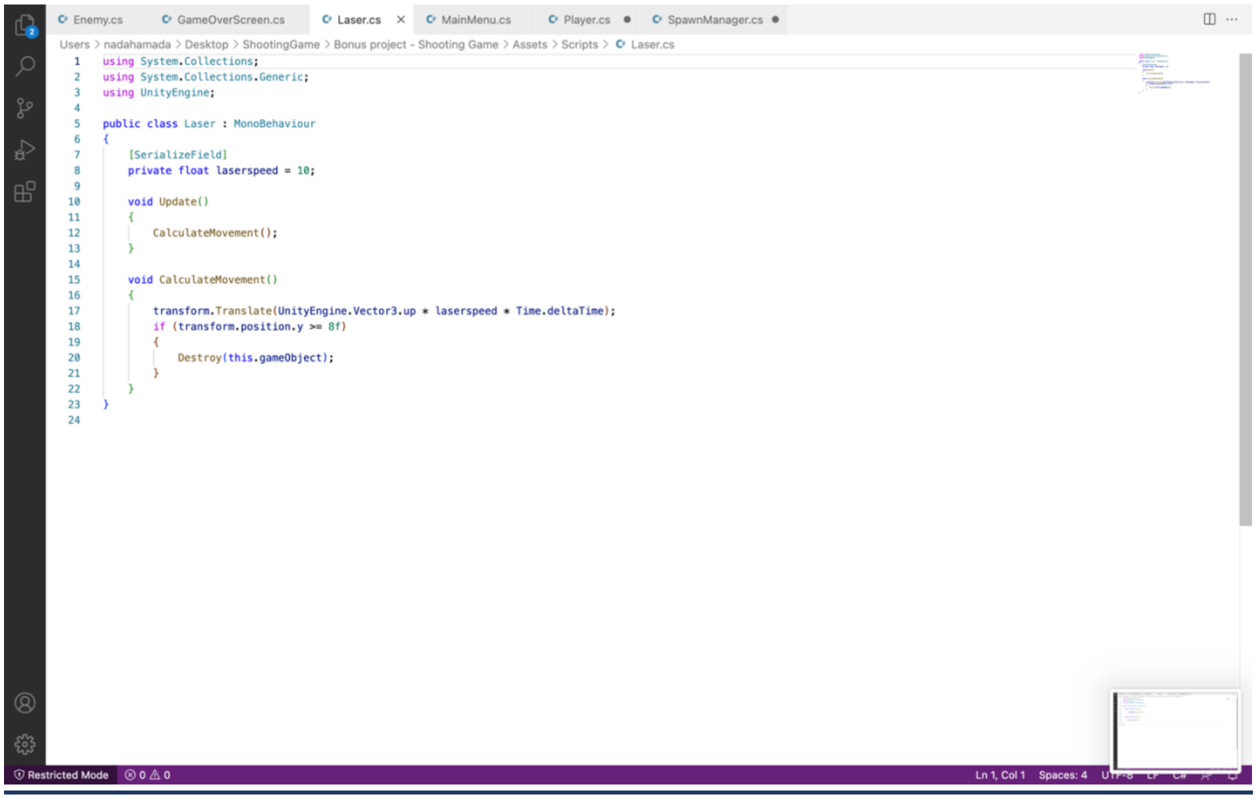


```
    public GameOverScreen gameover;

    public bool StopSpawning
    {…
    }

    void Start()
    {
        _player = GameObject.Find("Player").GetComponent<Player>();
        StartCoroutine(SpawnRoutine());
    }

    // Update is called once per frame
    void Update()
    {
        CheckDeath();
    }

    IEnumerator SpawnRoutine()
    {
        for (int i = 0; i < 5; i++)
        {
            if (stopspawning == false)
            {
                UnityEngine.Vector3 PosToSpawn = new UnityEngine.Vector3(UnityEngine.Random.Range(-9f, 9f), 7, 0);

                GameObject newEnemy = Instantiate(_EnemyPrefab, PosToSpawn, UnityEngine.Quaternion.identity);

                yield return new WaitForSeconds(2.0f);

                CheckDeath();

            }

        }
    }

    public void CheckDeath()
    {
        if (_player._lives == 0)
        {
            gameover.Setup();
        }
    }

```

Note: The difference between SpawnManager, SpawnManagerLevel2 and SpawnManagerLevel3 is that each script has a different (increasing) number of enemies to be spawned, with different (faster) speeds.

Laser:

- CalculateMovement()
  Sets the laser movement upwards at a fixed speed. When the laser gets to a point where it is no longer visible in the scene, it is destroyed in order not to be "flying on its own" and taking up space.

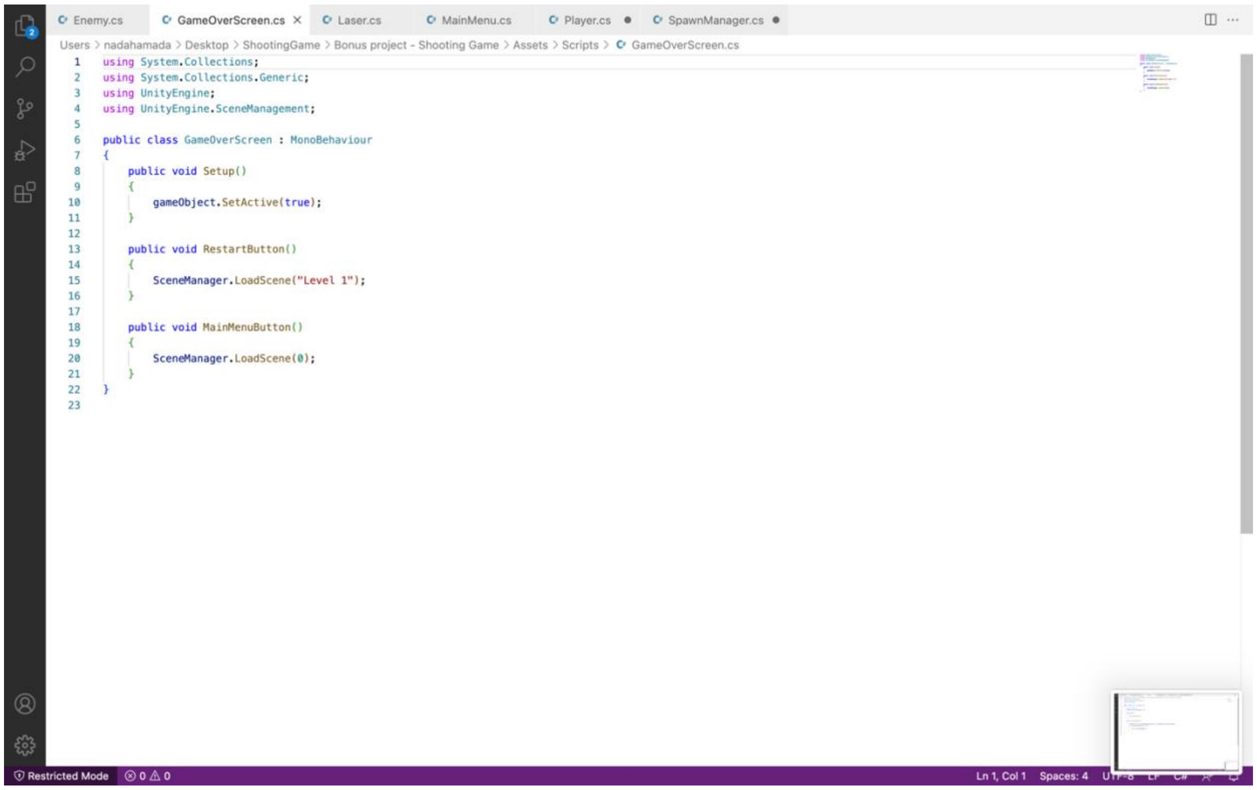


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Laser : MonoBehaviour
{
    [SerializeField]
    private float laserspeed = 10;

    void Update()
    {
        CalculateMovement();
    }

    void CalculateMovement()
    {
        transform.Translate(UnityEngine.Vector3.up * laserspeed * Time.deltaTime);
        if (transform.position.y >= 8f)
        {
            Destroy(this.gameObject);
        }
    }
}
```

GameOver:

- RestartButton()
  Takes you back to Level 1.

- MainMenuButton()
  Takes you to the MainMenu.


Enemy.cs
GameOverScreen.cs
Laser.cs
MainMenu.cs
Player.cs
SpawnManager.cs
Users > nadahamada > Desktop > ShootingGame > Bonus project - Shooting Game > Assets > Scripts > GameOverScreen.cs

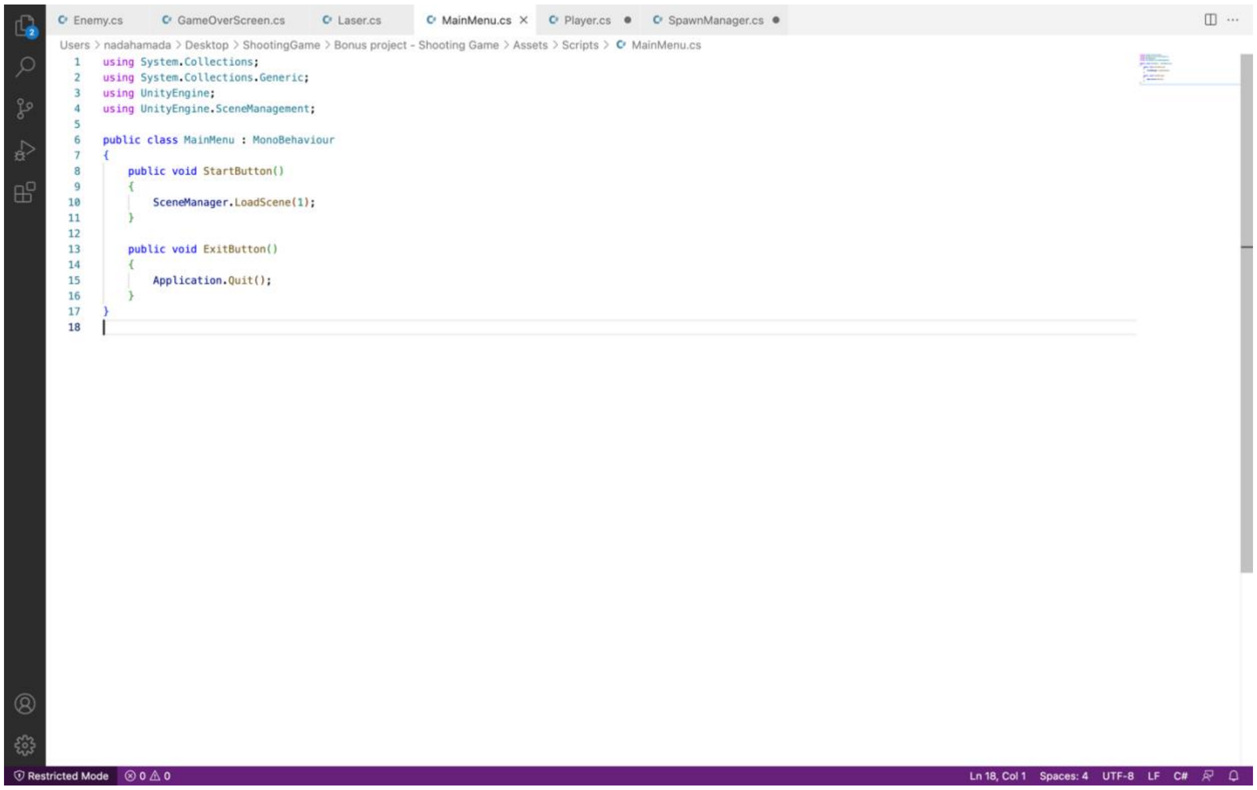
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GameOverScreen : MonoBehaviour
{
    public void Setup()
    {
        gameObject.SetActive(true);
    }

    public void RestartButton()
    {
        SceneManager.LoadScene("Level 1");
    }

    public void MainMenuButton()
    {
        SceneManager.LoadScene(0);
    }
}

```

Restricted Mode
⊗ 0 ⚠ 0
Ln 1, Col 1
Spaces: 4

Main Menu:

- StartButton()
  Takes you to Level 1.

- ExitButton()
  Closes the game.


Enemy.cs
GameOverScreen.cs
Laser.cs
MainMenu.cs
Player.cs
SpawnManager.cs
Users > nadahamada > Desktop > ShootingGame > Bonus project - Shooting Game > Assets > Scripts > MainMenu.cs

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

public class MainMenu : MonoBehaviour
{
    public void StartButton()
    {
        SceneManager.LoadScene(1);
    }

    public void ExitButton()
    {
        Application.Quit();
    }
}

```