



# 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.

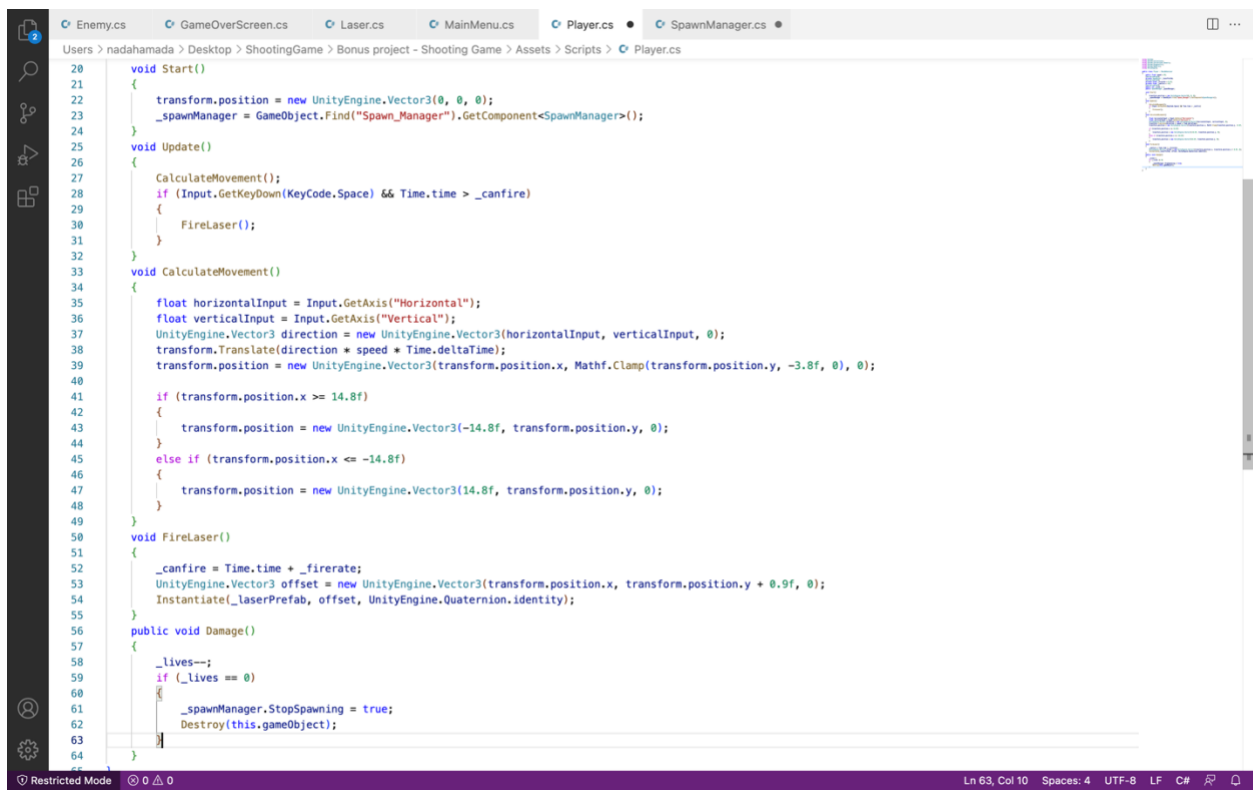
### **How to win:**

- 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:

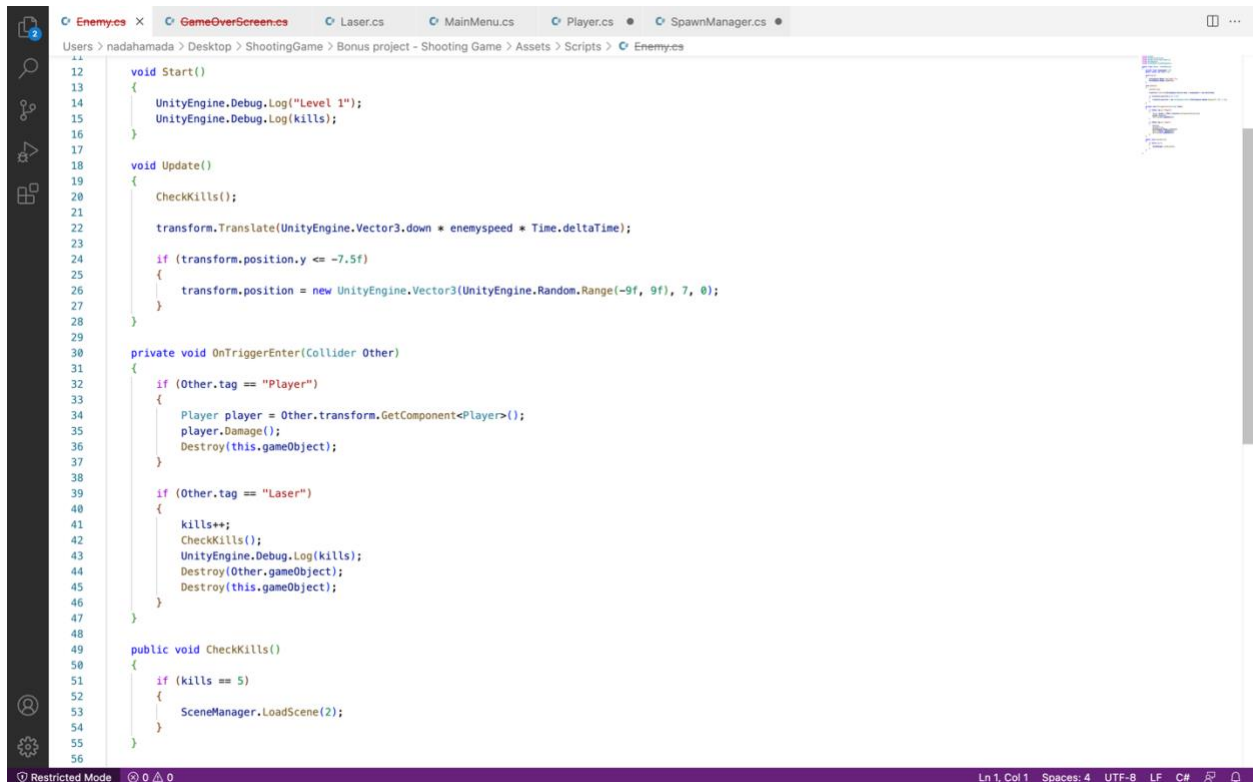
- CalculateMovement()  
Called in Update() in order to calculate the player's current (triggered per frame) position and move (within screen bounds) it upon user control.
- FireLaser()  
Gives the player ability to shoot lasers (instantiate) at a certain rate.
- Damage()  
Weaken the player's life when it collides with an enemy prefab.



```
20 void Start()
21 {
22     transform.position = new UnityEngine.Vector3(0, 0, 0);
23     _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager>();
24 }
25 void Update()
26 {
27     CalculateMovement();
28     if (Input.GetKeyDown(KeyCode.Space) && Time.time > _canfire)
29     {
30         FireLaser();
31     }
32 }
33 void CalculateMovement()
34 {
35     float horizontalInput = Input.GetAxis("Horizontal");
36     float verticalInput = Input.GetAxis("Vertical");
37     UnityEngine.Vector3 direction = new UnityEngine.Vector3(horizontalInput, verticalInput, 0);
38     transform.Translate(direction * speed * Time.deltaTime);
39     transform.position = new UnityEngine.Vector3(transform.position.x, Mathf.Clamp(transform.position.y, -3.8f, 0), 0);
40
41     if (transform.position.x >= 14.8f)
42     {
43         transform.position = new UnityEngine.Vector3(-14.8f, transform.position.y, 0);
44     }
45     else if (transform.position.x <= -14.8f)
46     {
47         transform.position = new UnityEngine.Vector3(14.8f, transform.position.y, 0);
48     }
49 }
50 void FireLaser()
51 {
52     _canfire = Time.time + _firerate;
53     UnityEngine.Vector3 offset = new UnityEngine.Vector3(transform.position.x, transform.position.y + 0.9f, 0);
54     Instantiate(_laserPrefab, offset, UnityEngine.Quaternion.identity);
55 }
56 public void Damage()
57 {
58     _lives--;
59     if (_lives == 0)
60     {
61         _spawnManager.StopSpawning = true;
62         Destroy(this.gameObject);
63     }
64 }
```

## 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.

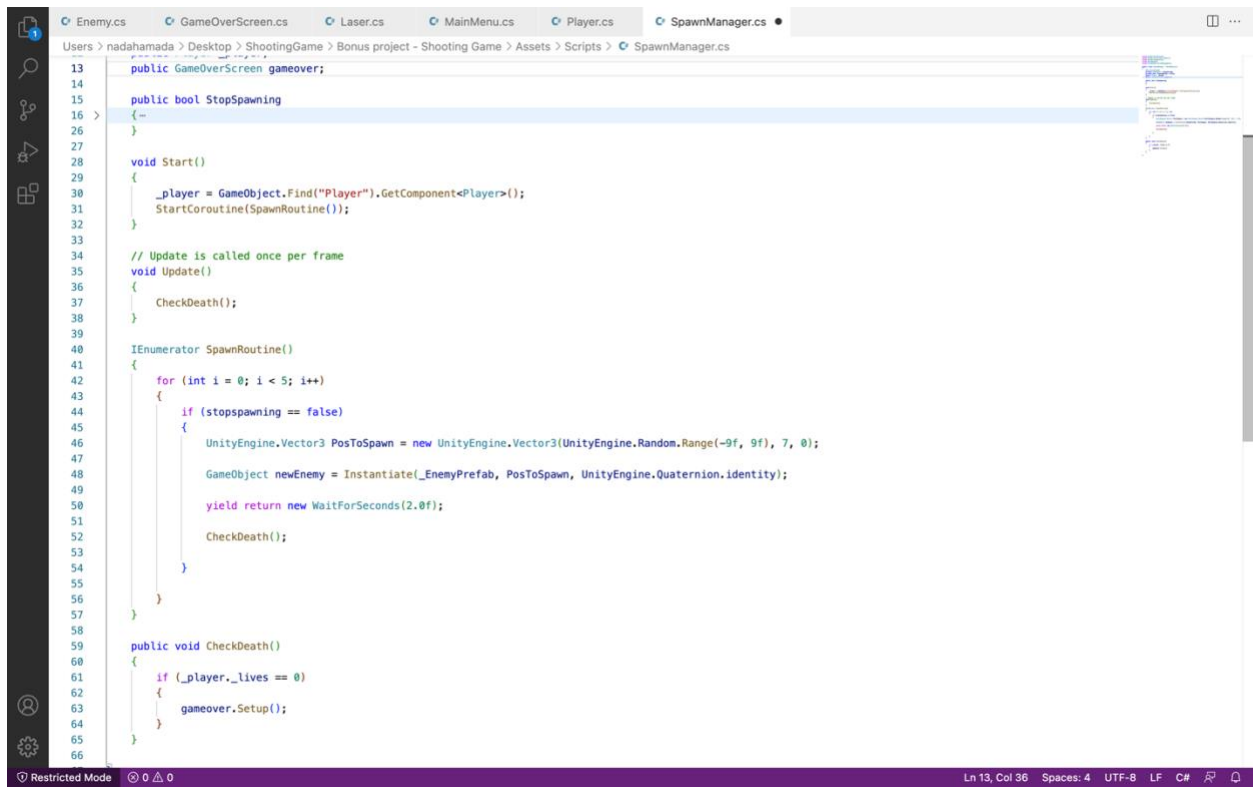
A screenshot of a Unity C# script editor showing the 'Enemy.cs' script. The script is located at 'Assets > Scripts > Enemy.cs'. It contains three main methods: 'Start()', 'Update()', and 'OnTriggerEnter(Collider Other)'. The 'Start()' method logs 'Level 1' and the current number of kills. The 'Update()' method calls 'CheckKills()', moves the enemy down, and resets its position if it goes off-screen. The 'OnTriggerEnter()' method checks for collisions with 'Player' (damaging the player and destroying the enemy) or 'Laser' (increasing kills, logging, and destroying both). A 'CheckKills()' method is also shown, which triggers a scene change when 5 kills are reached.

```
12 void Start()
13 {
14     UnityEngine.Debug.Log("Level 1");
15     UnityEngine.Debug.Log(kills);
16 }
17
18 void Update()
19 {
20     CheckKills();
21
22     transform.Translate(UnityEngine.Vector3.down * enemyspeed * Time.deltaTime);
23
24     if (transform.position.y <= -7.5f)
25     {
26         transform.position = new UnityEngine.Vector3(UnityEngine.Random.Range(-9f, 9f), 7, 0);
27     }
28 }
29
30 private void OnTriggerEnter(Collider Other)
31 {
32     if (Other.tag == "Player")
33     {
34         Player player = Other.transform.GetComponent<Player>();
35         player.Damage();
36         Destroy(this.gameObject);
37     }
38
39     if (Other.tag == "Laser")
40     {
41         kills++;
42         CheckKills();
43         UnityEngine.Debug.Log(kills);
44         Destroy(Other.gameObject);
45         Destroy(this.gameObject);
46     }
47 }
48
49 public void CheckKills()
50 {
51     if (kills == 5)
52     {
53         SceneManager.LoadScene(2);
54     }
55 }
56
```

**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)



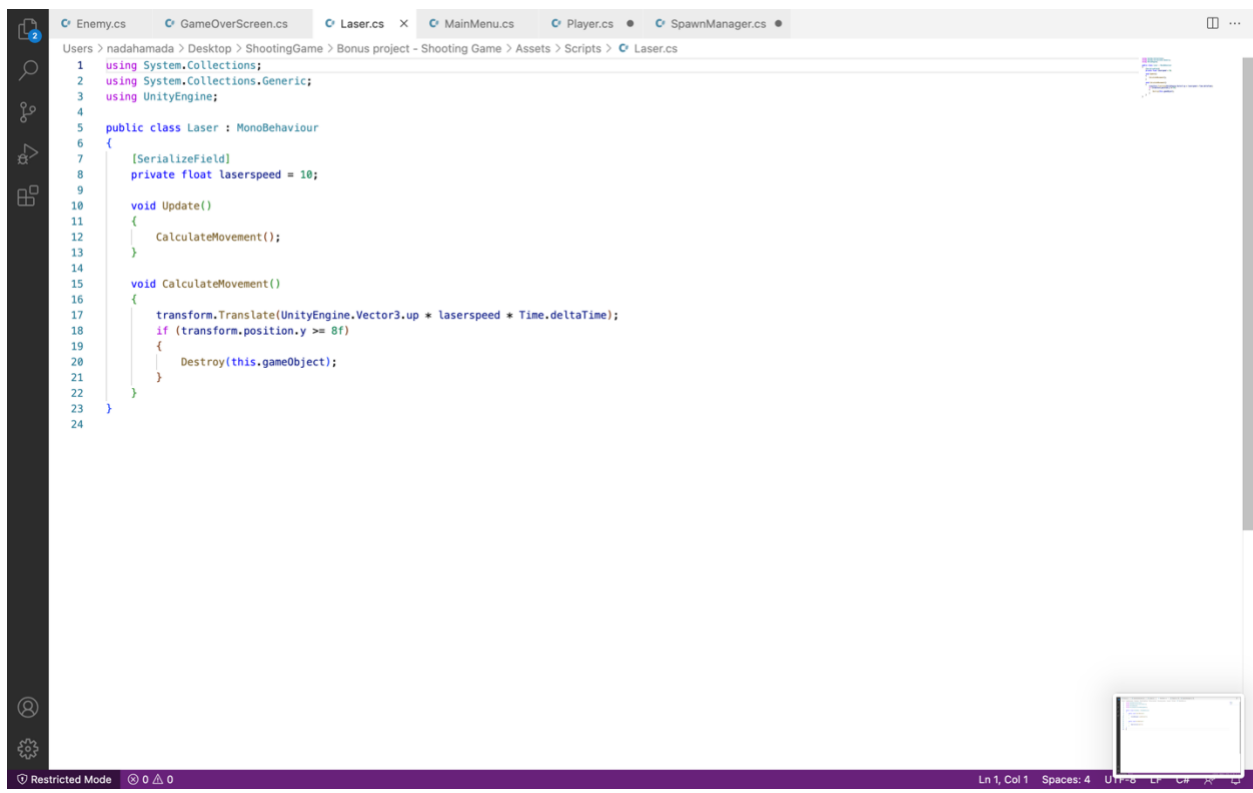
```
13 public GameOverScreen gameover;
14
15 public bool StopSpawning
16 {
17     get { return false; }
18     set { }
19 }
20
21 void Start()
22 {
23     _player = GameObject.Find("Player").GetComponent<Player>();
24     StartCoroutine(SpawnRoutine());
25 }
26
27 // Update is called once per frame
28 void Update()
29 {
30     CheckDeath();
31 }
32
33 IEnumerator SpawnRoutine()
34 {
35     for (int i = 0; i < 5; i++)
36     {
37         if (stopspawning == false)
38         {
39             UnityEngine.Vector3 PosToSpawn = new UnityEngine.Vector3(UnityEngine.Random.Range(-9f, 9f), 7, 0);
40
41             GameObject newEnemy = Instantiate(_EnemyPrefab, PosToSpawn, UnityEngine.Quaternion.identity);
42
43             yield return new WaitForSeconds(2.0f);
44
45             CheckDeath();
46         }
47     }
48 }
49
50 public void CheckDeath()
51 {
52     if (_player._lives == 0)
53     {
54         gameover.Setup();
55     }
56 }
57
58
59
60
61
62
63
64
65
66
```

**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.

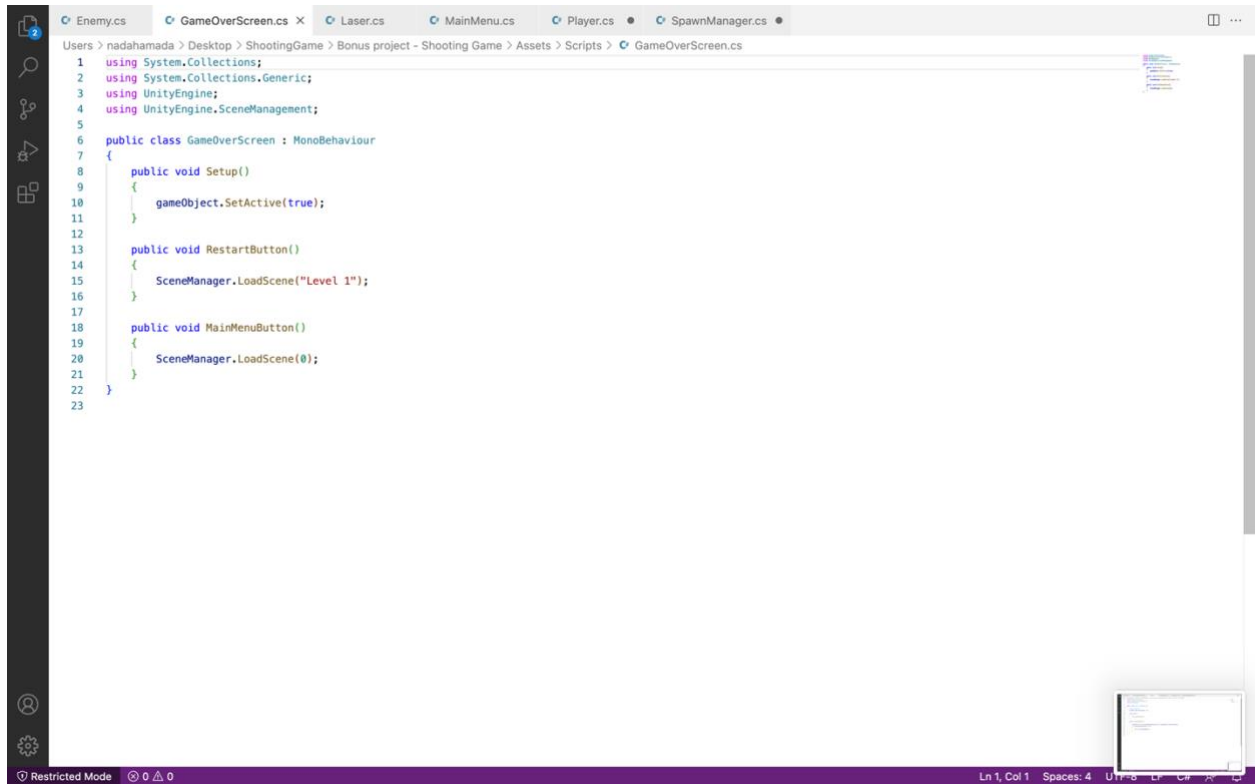


The screenshot shows the Unity IDE with the 'Laser.cs' script open. The script is a C# class that inherits from 'MonoBehaviour'. It contains a private float variable 'laserspeed' set to 10. The 'Update()' method calls 'CalculateMovement()'. The 'CalculateMovement()' method moves the laser upwards by a fixed speed multiplied by 'Time.deltaTime'. If the laser's position.y is greater than or equal to 8f, it is destroyed using 'Destroy(this.gameObject)'. The status bar at the bottom indicates 'Restricted Mode' and 'Ln 1, Col 1 Spaces: 4'.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Laser : MonoBehaviour
6 {
7     [SerializeField]
8     private float laserspeed = 10;
9
10    void Update()
11    {
12        CalculateMovement();
13    }
14
15    void CalculateMovement()
16    {
17        transform.Translate(UnityEngine.Vector3.up * laserspeed * Time.deltaTime);
18        if (transform.position.y >= 8f)
19        {
20            Destroy(this.gameObject);
21        }
22    }
23 }
24
```

## GameOver:

- **RestartButton()**  
Takes you back to Level 1.
- **MainMenuButton()**  
Takes you to the MainMenu.



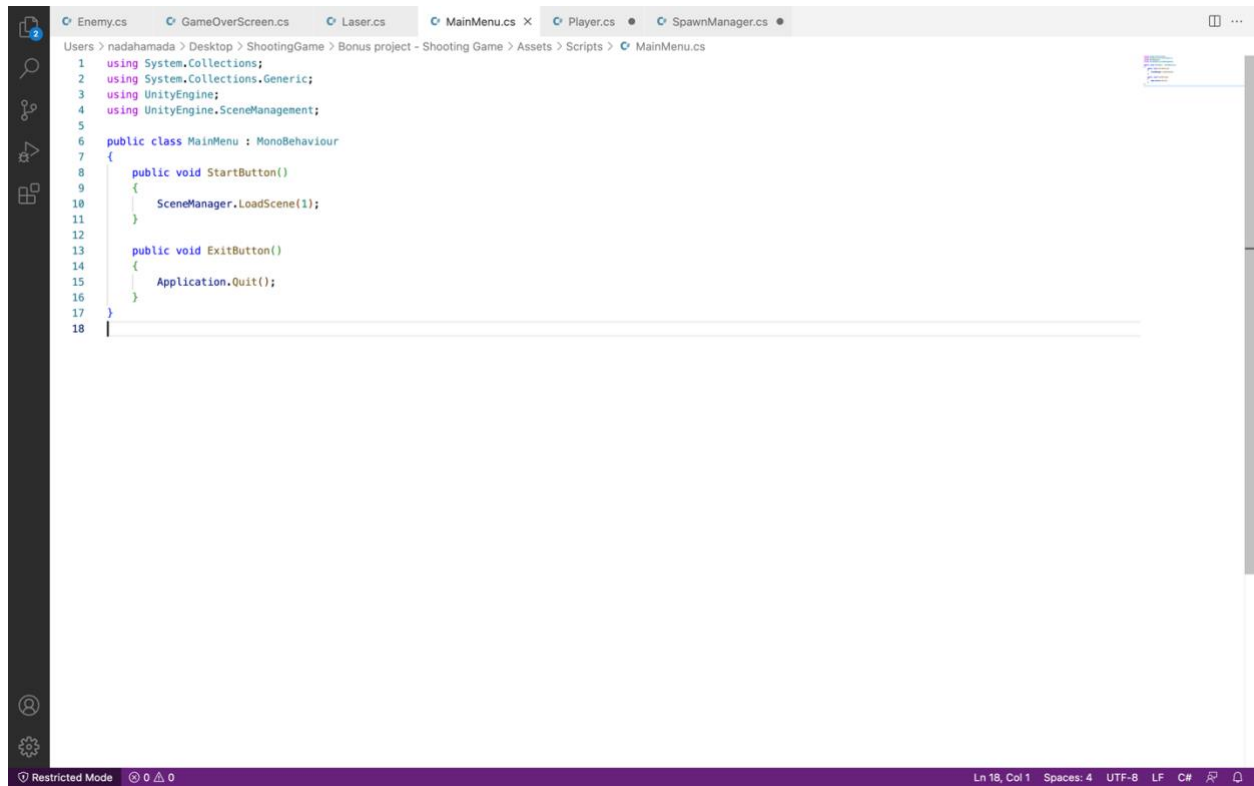
The screenshot shows a Unity script editor with the following code in `GameOverScreen.cs`:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class GameOverScreen : MonoBehaviour
7 {
8     public void Setup()
9     {
10         gameObject.SetActive(true);
11     }
12
13     public void RestartButton()
14     {
15         SceneManager.LoadScene("Level 1");
16     }
17
18     public void MainMenuButton()
19     {
20         SceneManager.LoadScene(0);
21     }
22 }
23
```

The editor interface includes a top toolbar with icons for file operations, a left sidebar with icons for the Hierarchy, Console, and other Unity tools, and a bottom status bar showing "Restricted Mode" and "Ln 1, Col 1 Spaces: 4 UTF-8".

## Main Menu:

- StartButton()  
Takes you to Level 1.
- ExitButton()  
Closes the game.



The screenshot shows the Unity IDE with the MainMenu.cs script open. The script is a C# class that inherits from MonoBehaviour and contains two public void methods: StartButton() and ExitButton(). The StartButton() method calls SceneManager.LoadScene(1), and the ExitButton() method calls Application.Quit(). The IDE interface includes a top toolbar with icons for file operations, a left sidebar with icons for the Hierarchy, Console, and other tools, and a bottom status bar showing 'Restricted Mode' and file encoding information.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class MainMenu : MonoBehaviour
7 {
8     public void StartButton()
9     {
10         SceneManager.LoadScene(1);
11     }
12
13     public void ExitButton()
14     {
15         Application.Quit();
16     }
17 }
18
```