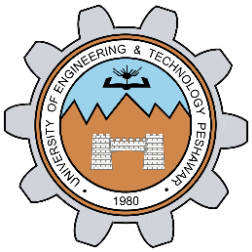
PROJECT REPORT



Spring 2022

CSE102L Computer Programming Lab

Suleman Shah (**21PWCSE1983)**

Shahzad Bangash (**21PWCSE1980)**

Ali Asghar (**21PWCSE2059)**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Engr. Abdullah Hamid

July 27, 2022

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

SOCCER PINBALL GAME

SOCCER PINBALL:

Soccer Pinball Game blends the challenges of soccer and pinball in one game. It is a hyper-casual game with fun and challenging mixed gameplay of soccer and pinball. This game truly fulfills the desire of both soccer and pinball game fans.

HYPER-CASUAL GAME:

A hyper-casual game is a lightweight game with minimal design. They are relatively easy to produce, but they guarantee to offer a high level of entertainment and enjoyment.

FRAMEWORK USED:

Unity and C#.

TOPICS USED FROM COURSE IN PROJECT:

- Objects and Classes(Object Oriented Programming)
- Basic Data types(bool, float, integer)
- Functions
- Selection Statements(if-else statements)
- Function Parameters
- Logical Operators

HOW TO PLAY:

When you hit the play button, the ball is randomly spawned in ground, then it falls down to your goal. Using the left and right flippers, save the goal by not letting the ball fall into your goal. Up ahead, there’s an enemy goal. By

scoring goal on enemy, you will earn score. Each time you score highest score, it is saved.

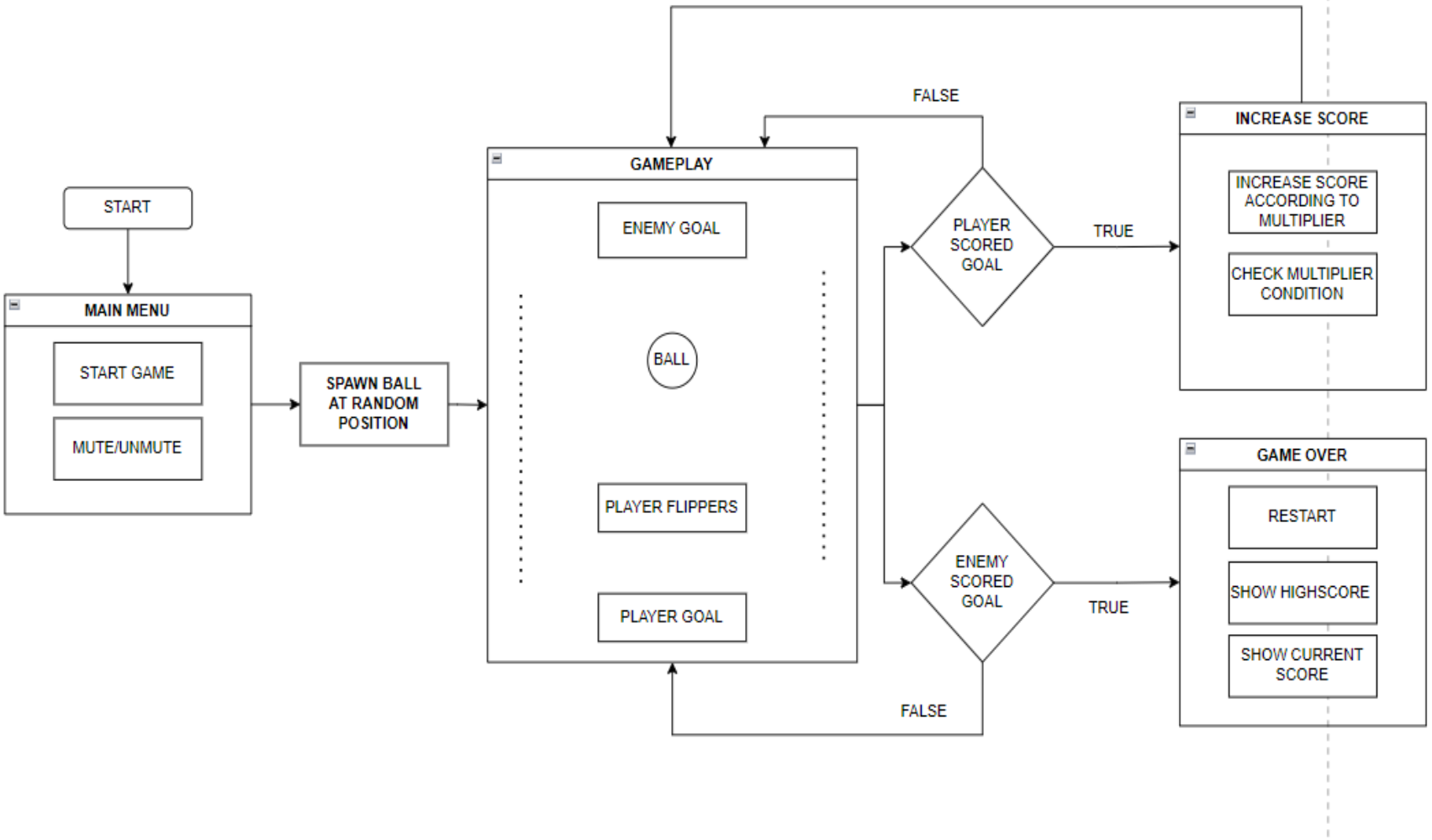
CONTROLS:

Right and left flippers can be controlled by D and A keys respectively.

APPLICATIONS:

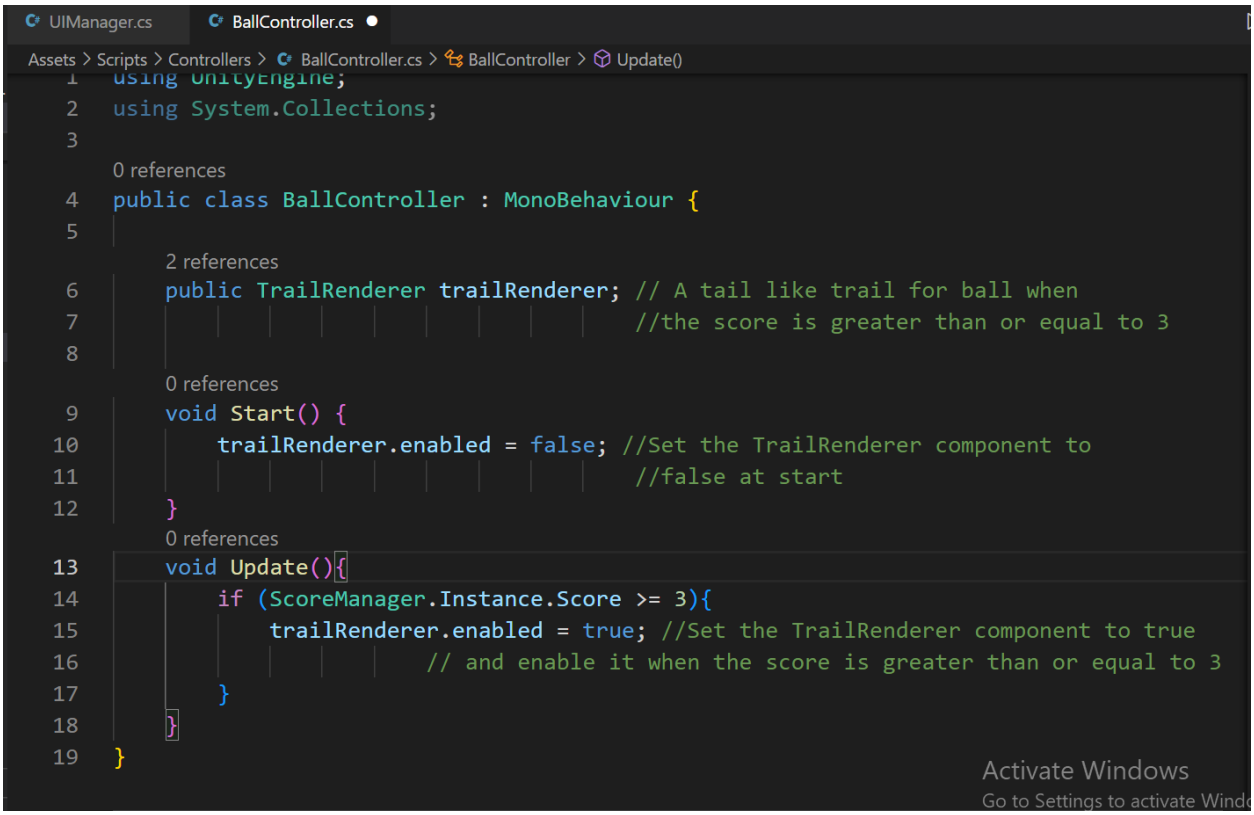
Hyper casual games market is rapidly growing day by day and the day will come when hyper casual games would rule the game market. As our project is a hyper casual game, it is mostly used for entertainment purposes. It can be a great time killer and stress releaser for gamers. Apart from this, publishers are also interested in publishing hyper casual games. Hyper casual games are easy to create and play as compared to complex RPG games.

GAME
FLOWCHART

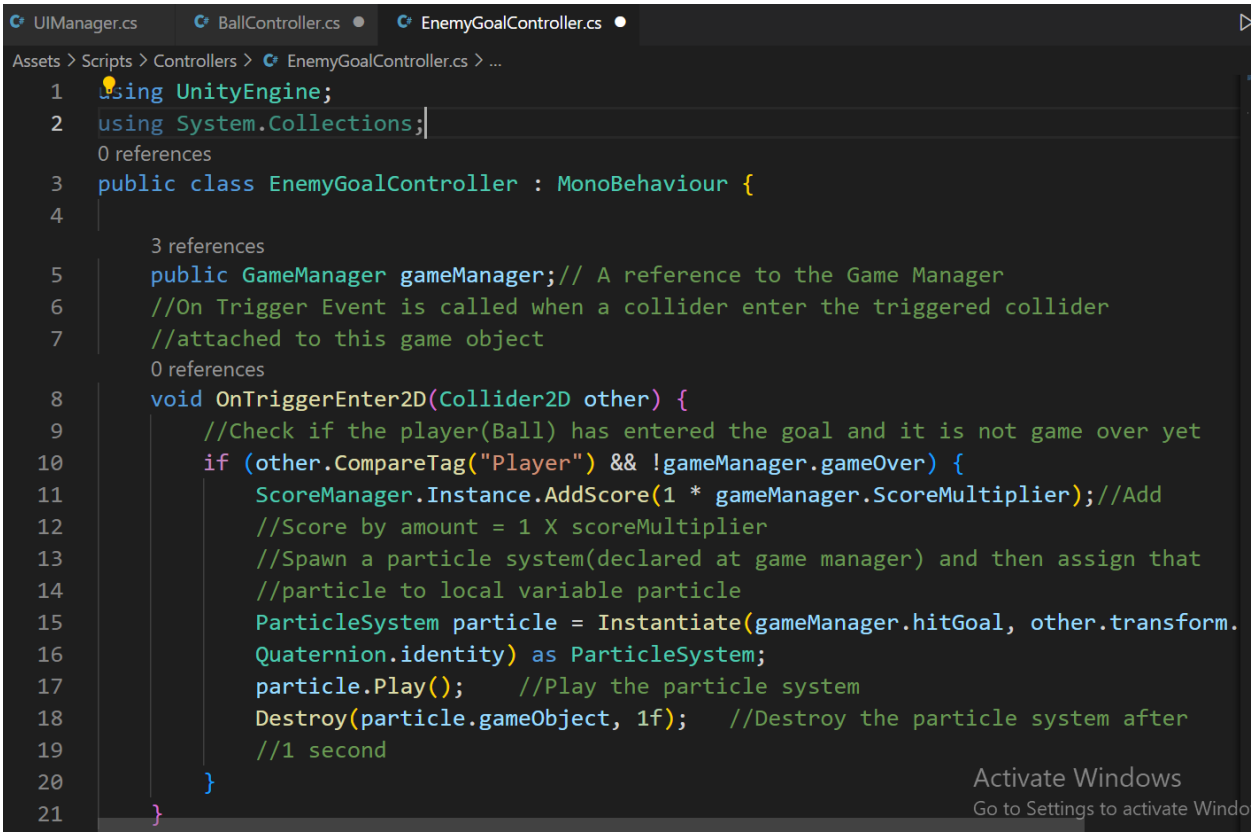


CODE SCREENSHOTS:

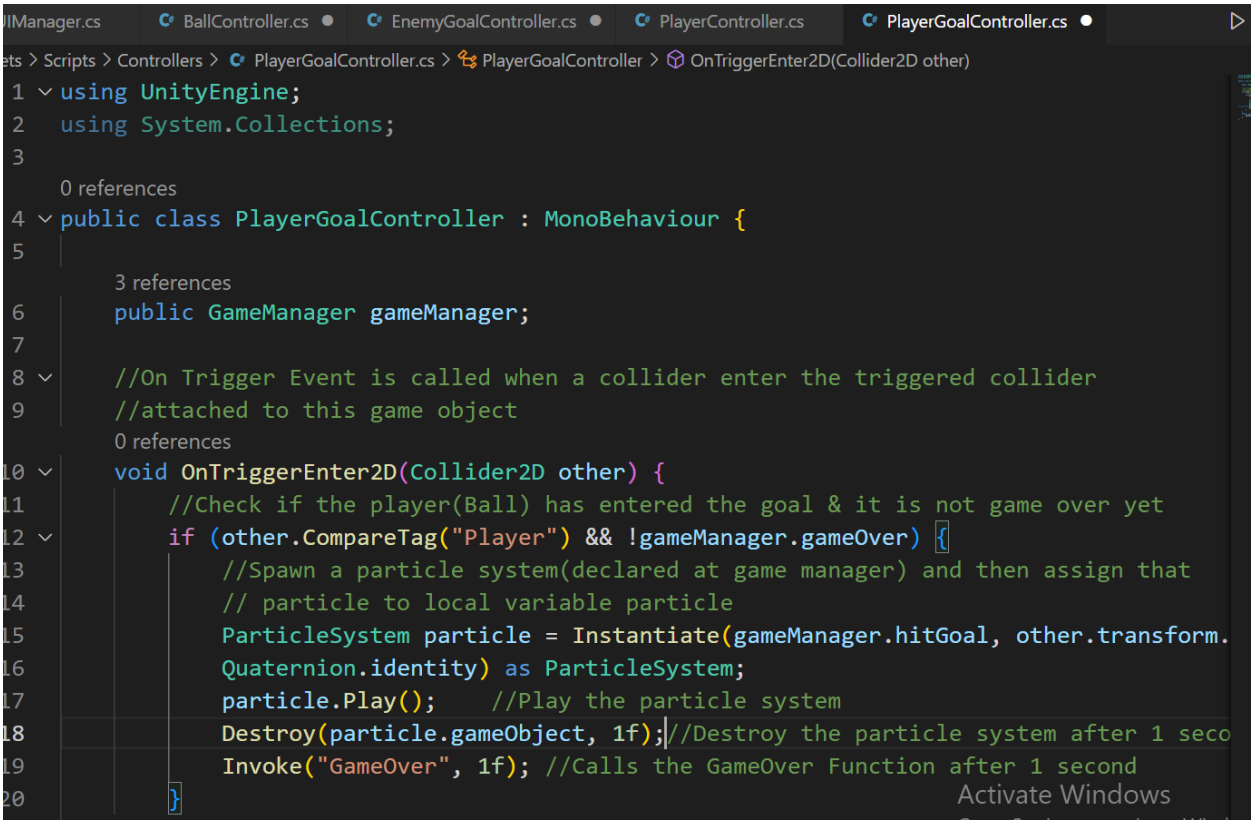
Ball Controller



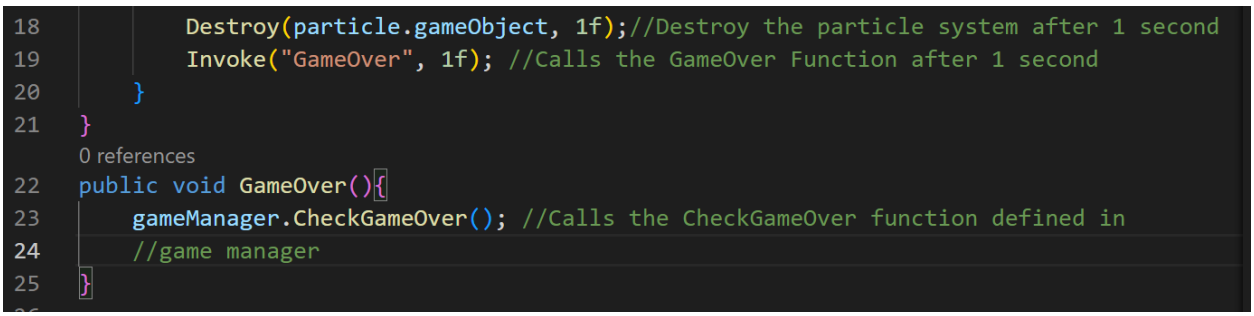
Enemy Goal Controller



Player Goal Controller



```
UIManager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | PlayerGoalController.cs
ets > Scripts > Controllers > PlayerGoalController.cs > PlayerGoalController > OnTriggerEnter2D(Collider2D other)
1  using UnityEngine;
2  using System.Collections;
3
   0 references
4  public class PlayerGoalController : MonoBehaviour {
5
       3 references
6      public GameManager gameManager;
7
8      //On Trigger Event is called when a collider enter the triggered collider
9      //attached to this game object
       0 references
10     void OnTriggerEnter2D(Collider2D other) {
11         //Check if the player(Ball) has entered the goal & it is not game over yet
12         if (other.CompareTag("Player") && !gameManager.gameOver) {
13             //Spawn a particle system(declared at game manager) and then assign that
14             // particle to local variable particle
15             ParticleSystem particle = Instantiate(gameManager.hitGoal, other.transform.
16             Quaternion.identity) as ParticleSystem;
17             particle.Play();    //Play the particle system
18             Destroy(particle.gameObject, 1f);//Destroy the particle system after 1 seco
19             Invoke("GameOver", 1f); //Calls the GameOver Function after 1 second
20         }
```

```
18             Destroy(particle.gameObject, 1f);//Destroy the particle system after 1 second
19             Invoke("GameOver", 1f); //Calls the GameOver Function after 1 second
20         }
21     }
       0 references
22     public void GameOver(){
23         gameManager.CheckGameOver(); //Calls the CheckGameOver function defined in
24         //game manager
25     }
```

Player Controller

Manager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | PlayerGoalController.cs

s > Scripts > Controllers > PlayerController.cs > PlayerController

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

0 references
public class PlayerController : MonoBehaviour {

    1 reference
    public Rigidbody2D leftFlipperRigid; //A reference to the rigidbody of left flipper
    1 reference
    public Rigidbody2D rightFlipperRigid; //A reference to the rigidbody of right flipp
    2 references
    public float torqueForce; //A variable of type float to define the amount of
    //torque to apply

    // Update is called once per frame
    0 references
    void Update()
    {
        //Check if player presses A button on Keyboard and then add torque
        //to rigidbody of left flipper in anti clockwise direction
        if (Input.GetKey(KeyCode.A))
            AddTorqueToFlipper(leftFlipperRigid, torqueForce);
```


UIManager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | PlayerGoalController.cs

sets > Scripts > Controllers > PlayerController.cs > PlayerController

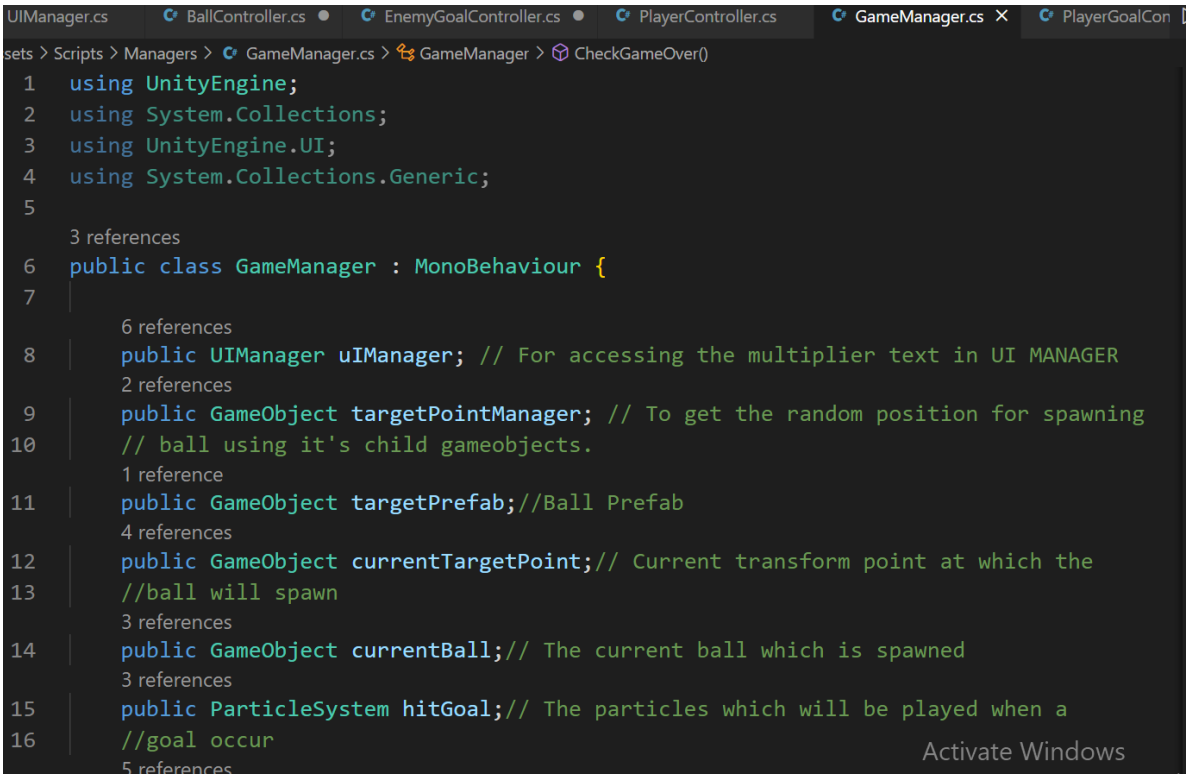
```
    void Update()
    {
        //Check if player presses A button on Keyboard and then add torque
        //to rigidbody of left flipper in anti clockwise direction
        if (Input.GetKey(KeyCode.A))
            AddTorqueToFlipper(leftFlipperRigid, torqueForce);

        //Check if player presses D button on Keyboard and then add torque
        //to rigidbody of right flipper in clockwise direction
        if (Input.GetKey(KeyCode.D))
            AddTorqueToFlipper(rightFlipperRigid, -torqueForce);

    }

    //A function to add torque to flipper by some amount force
    2 references
    void AddTorqueToFlipper(Rigidbody2D rigid, float force)
    {
        //Adds torque to the parameter rigidbody using a built in
        //function called AddTorque
        rigid.AddTorque(force);
    }
```

Game Manager

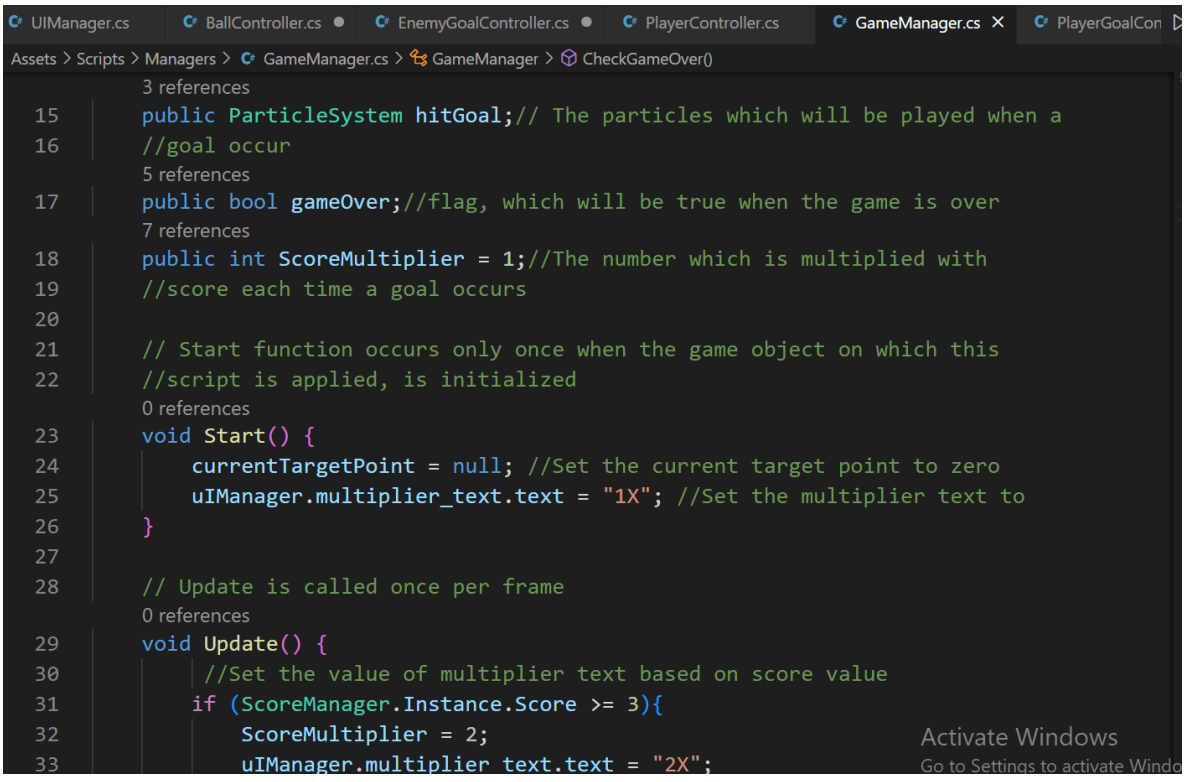


UIManager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | GameManager.cs | PlayerGoalCon

sets > Scripts > Managers > GameManager.cs > GameManager > CheckGameOver()

```
1   using UnityEngine;
2   using System.Collections;
3   using UnityEngine.UI;
4   using System.Collections.Generic;
5
    3 references
6   public class GameManager : MonoBehaviour {
7
        6 references
8       public UIManager uIManager; // For accessing the multiplier text in UI MANAGER
        2 references
9       public GameObject targetPointManager; // To get the random position for spawning
10      // ball using it's child gameobjects.
        1 reference
11      public GameObject targetPrefab;//Ball Prefab
        4 references
12      public GameObject currentTargetPoint;// Current transform point at which the
13      //ball will spawn
        3 references
14      public GameObject currentBall;// The current ball which is spawned
        3 references
15      public ParticleSystem hitGoal;// The particles which will be played when a
16      //goal occur
        5 references
```

Activate Windows



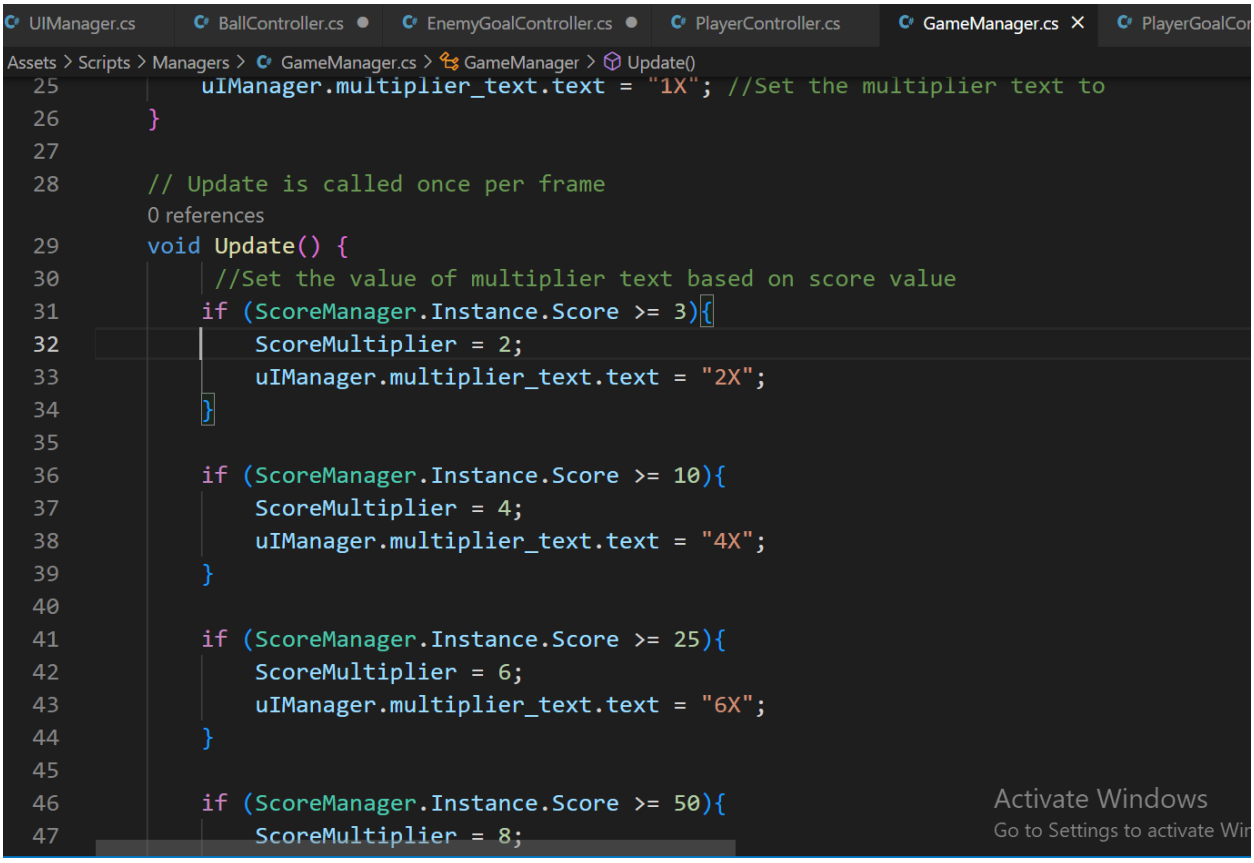
UIManager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | GameManager.cs | PlayerGoalCon

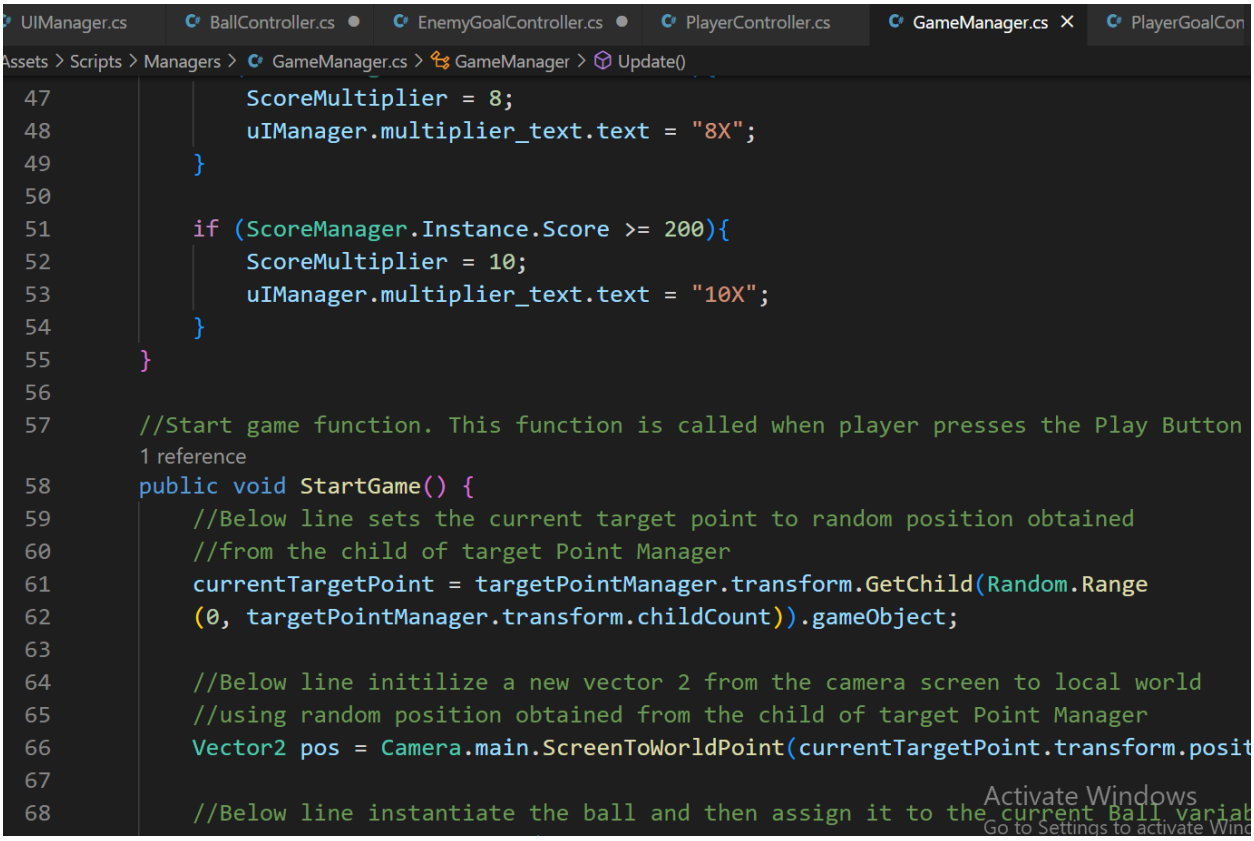
Assets > Scripts > Managers > GameManager.cs > GameManager > CheckGameOver()

```
        3 references
15      public ParticleSystem hitGoal;// The particles which will be played when a
16      //goal occur
        5 references
17      public bool gameOver;//flag, which will be true when the game is over
        7 references
18      public int ScoreMultiplier = 1;//The number which is multiplied with
19      //score each time a goal occurs
20
21      // Start function occurs only once when the game object on which this
22      //script is applied, is initialized
        0 references
23      void Start() {
24          currentTargetPoint = null; //Set the current target point to zero
25          uIManager.multiplier_text.text = "1X"; //Set the multiplier text to
26      }
27
28      // Update is called once per frame
        0 references
29      void Update() {
30          //Set the value of multiplier text based on score value
31          if (ScoreManager.Instance.Score >= 3){
32              ScoreMultiplier = 2;
33              uIManager.multiplier_text.text = "2X";
```

Activate Windows

Go to Settings to activate Windo

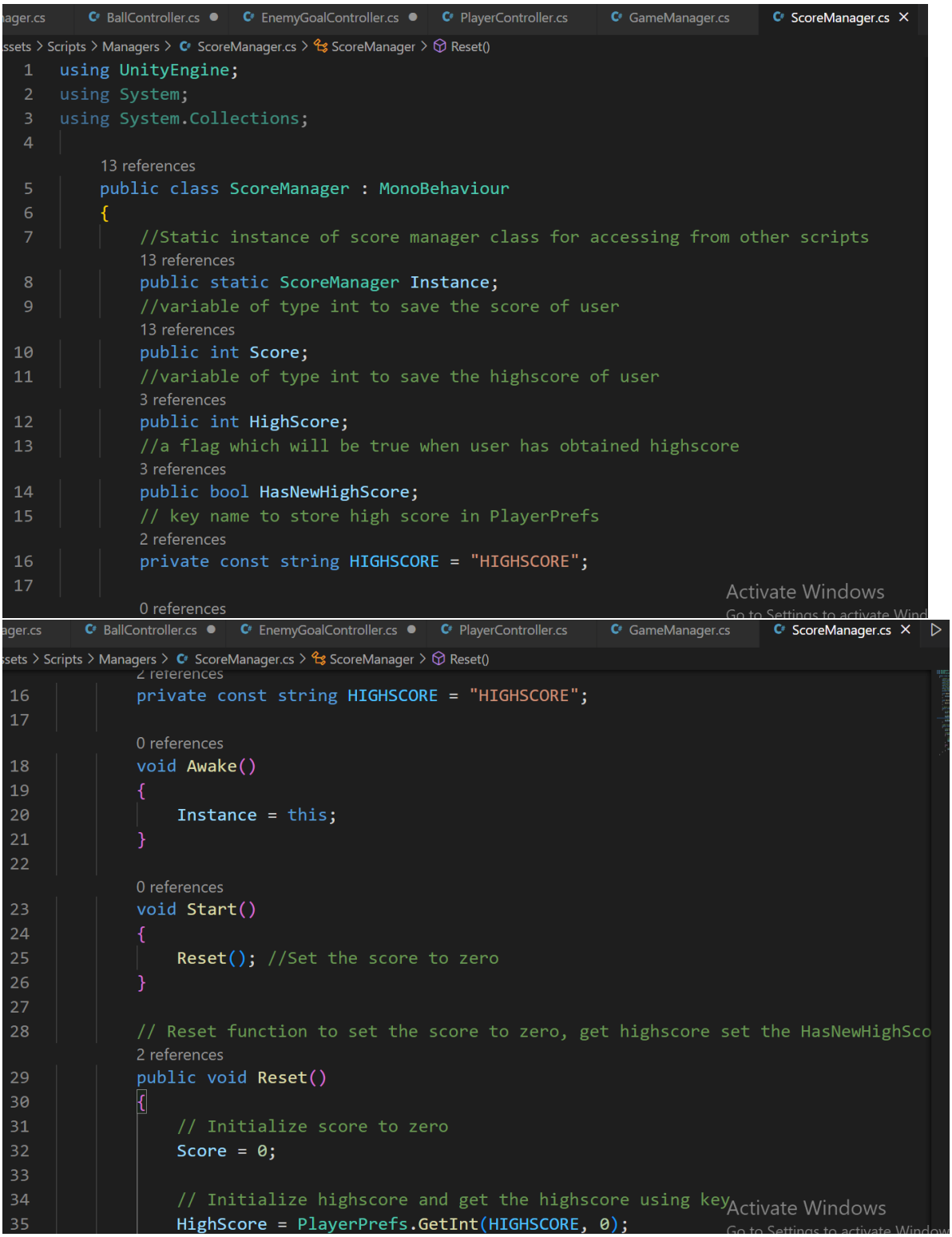

UIManager.cs
BallController.cs
EnemyGoalController.cs
PlayerController.cs
GameManager.cs
PlayerGoalCon
Assets > Scripts > Managers > GameManager.cs > GameManager > Update()
uIManager.multiplier_text.text = "1X"; //Set the multiplier text to
}
// Update is called once per frame
0 references
void Update() {
//Set the value of multiplier text based on score value
if (ScoreManager.Instance.Score >= 3){
ScoreMultiplier = 2;
uIManager.multiplier_text.text = "2X";
}
if (ScoreManager.Instance.Score >= 10){
ScoreMultiplier = 4;
uIManager.multiplier_text.text = "4X";
}
if (ScoreManager.Instance.Score >= 25){
ScoreMultiplier = 6;
uIManager.multiplier_text.text = "6X";
}
if (ScoreManager.Instance.Score >= 50){
ScoreMultiplier = 8;
Activate Windows
Go to Settings to activate Wir


UIManager.cs
BallController.cs
EnemyGoalController.cs
PlayerController.cs
GameManager.cs
PlayerGoalCon
Assets > Scripts > Managers > GameManager.cs > GameManager > Update()
ScoreMultiplier = 8;
uIManager.multiplier_text.text = "8X";
}
if (ScoreManager.Instance.Score >= 200){
ScoreMultiplier = 10;
uIManager.multiplier_text.text = "10X";
}
}
//Start game function. This function is called when player presses the Play Button
1 reference
public void StartGame() {
//Below line sets the current target point to random position obtained
//from the child of target Point Manager
currentTargetPoint = targetPointManager.transform.GetChild(Random.Range
(0, targetPointManager.transform.childCount)).gameObject;
//Below line initilize a new vector 2 from the camera screen to local world
//using random position obtained from the child of target Point Manager
Vector2 pos = Camera.main.ScreenToWorldPoint(currentTargetPoint.transform.posit
//Below line instantiate the ball and then assign it to the current Ball variab
Activate Windows
Go to Settings to activate Wind

ssets > Scripts > Managers > GameManager.cs > GameManager > Update()

```
68          //Below line instantiate the ball and then assign it to the current Ball variab
69          currentBall = Instantiate(targetPrefab, pos, Quaternion.identity) as GameObject
70
71          gameOver = false; //Set the gameOver flag to false at start
72          ScoreMultiplier = 1; //Set the multiplier text to 1 at start
73      }
74
75      //Game Over function which is called when the ball enters player goal
        1 reference
76      public void CheckGameOver() {
77              gameOver = true; //Set the gameOver flag to true
78              currentTargetPoint.SetActive(false); //Disable the current target point
79
80              //Below line instantiate the particle system and then assign it to local
81              //variable particle
82              ParticleSystem particle = Instantiate(hitGoal, currentBall.transform.positi
83               Quaternion.identity) as ParticleSystem;
84              particle.Play(); //Plays the particle
85              Destroy(particle.gameObject, 1f);//Destroys the particle after 1 second
86              Destroy(currentBall.gameObject);//Destroy the current ball
87
88      }
89  }
```

Score Manager

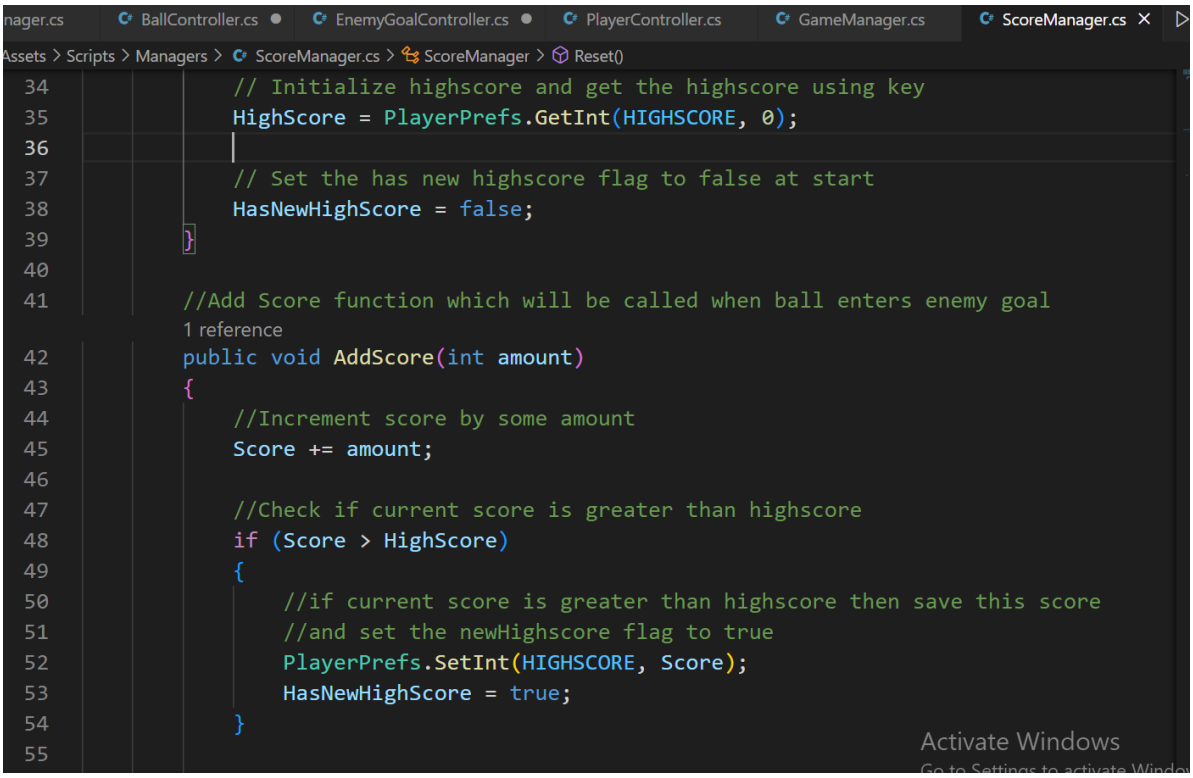


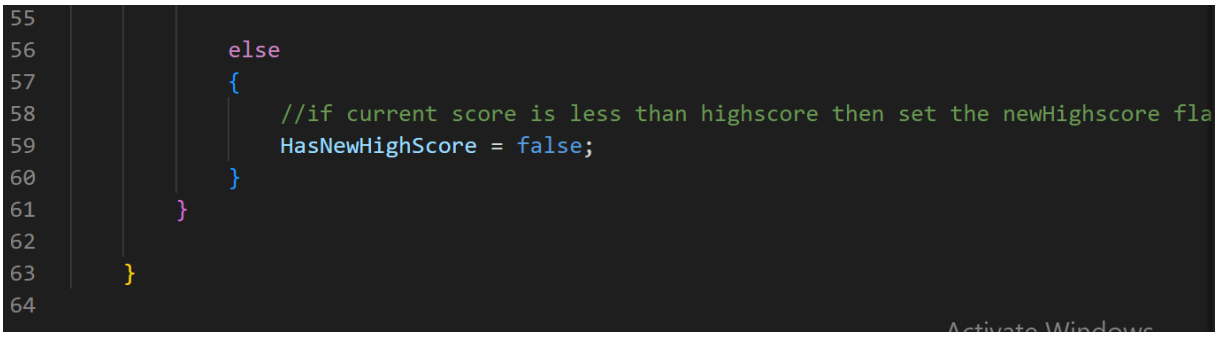
```
ager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | GameManager.cs | ScoreManager.cs
ssets > Scripts > Managers > ScoreManager.cs > ScoreManager > Reset()
1   using UnityEngine;
2   using System;
3   using System.Collections;
4
        13 references
5       public class ScoreManager : MonoBehaviour
6       {
7           //Static instance of score manager class for accessing from other scripts
                13 references
8           public static ScoreManager Instance;
9           //variable of type int to save the score of user
                13 references
10          public int Score;
11          //variable of type int to save the highscore of user
                3 references
12          public int HighScore;
13          //a flag which will be true when user has obtained highscore
                3 references
14          public bool HasNewHighScore;
15          // key name to store high score in PlayerPrefs
                2 references
16          private const string HIGHSCORE = "HIGHSCORE";
17
                0 references
```

Activate Windows
Go to Settings to activate Wind

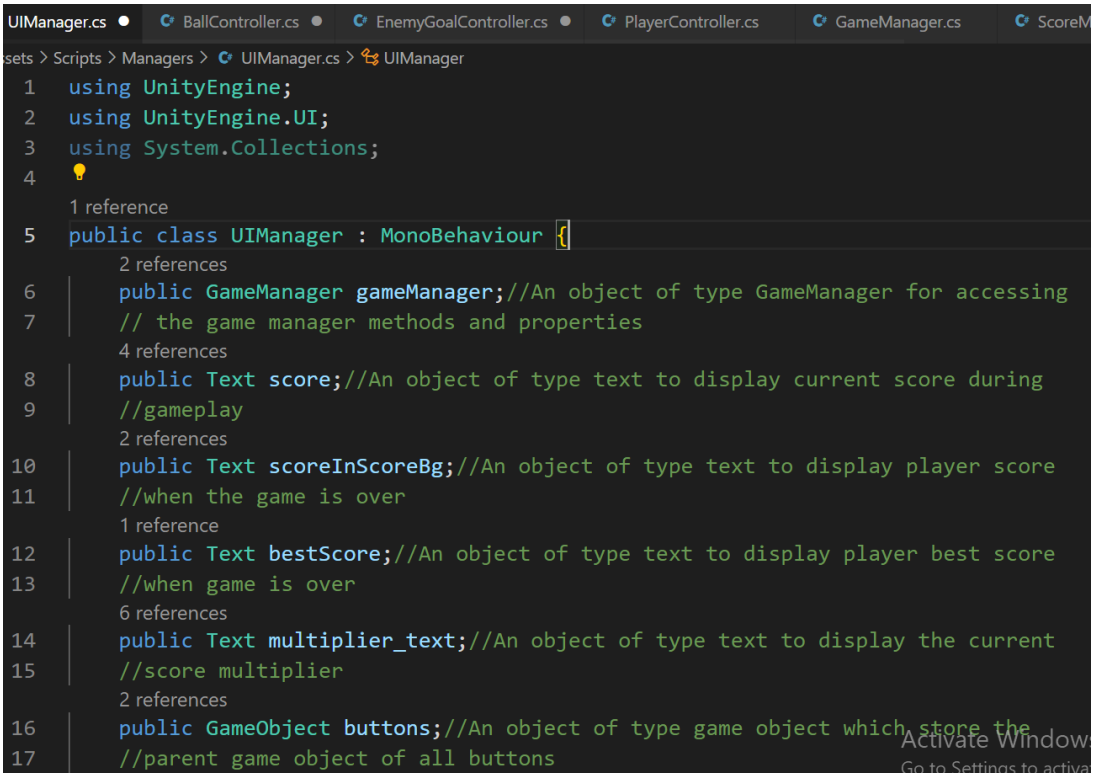
```
ager.cs | BallController.cs | EnemyGoalController.cs | PlayerController.cs | GameManager.cs | ScoreManager.cs
ssets > Scripts > Managers > ScoreManager.cs > ScoreManager > Reset()
                2 references
16          private const string HIGHSCORE = "HIGHSCORE";
17
                0 references
18          void Awake()
19          {
20              Instance = this;
21          }
22
                0 references
23          void Start()
24          {
25              Reset(); //Set the score to zero
26          }
27
28          // Reset function to set the score to zero, get highscore set the HasNewHighSco
                2 references
29          public void Reset()
30          {
31              // Initialize score to zero
32              Score = 0;
33
34              // Initialize highscore and get the highscore using key
35              HighScore = PlayerPrefs.GetInt(HIGHSCORE, 0);
```

Activate Windows
Go to Settings to activate Windows

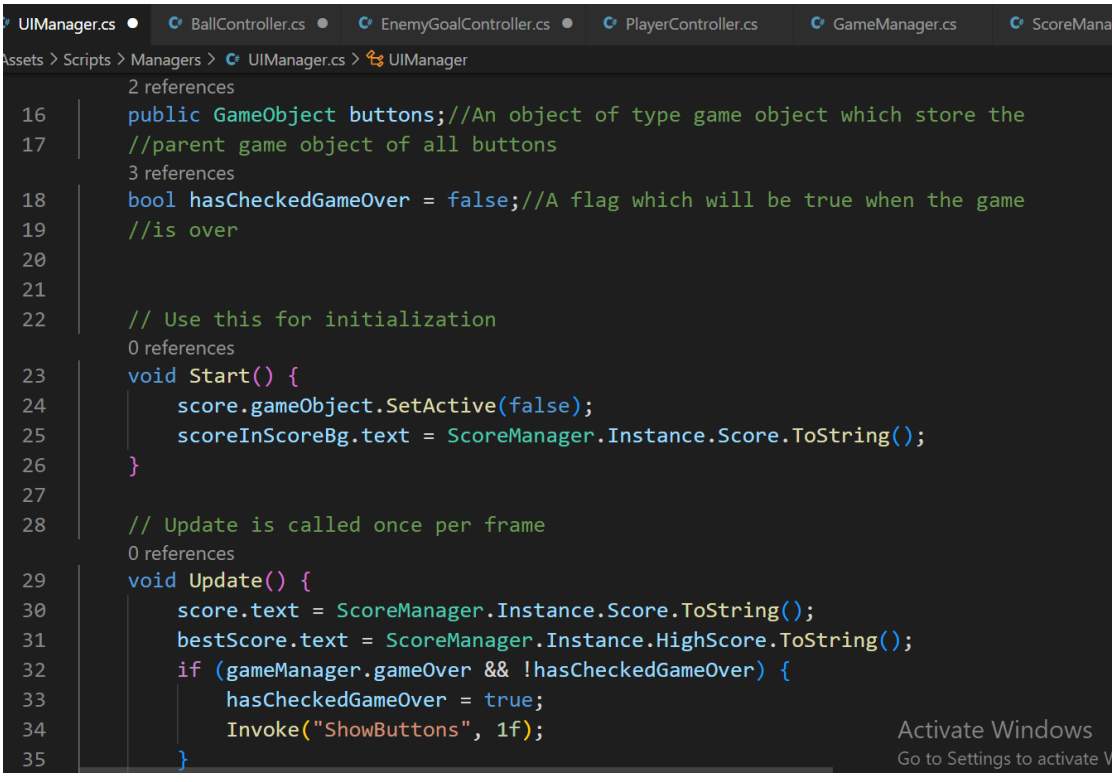

nager.cs
BallController.cs
EnemyGoalController.cs
PlayerController.cs
GameManager.cs
ScoreManager.cs
Assets > Scripts > Managers > ScoreManager.cs > ScoreManager > Reset()
// Initialize highscore and get the highscore using key
HighScore = PlayerPrefs.GetInt(HIGHSCORE, 0);
// Set the has new highscore flag to false at start
HasNewHighScore = false;
}
//Add Score function which will be called when ball enters enemy goal
1 reference
public void AddScore(int amount)
{
//Increment score by some amount
Score += amount;
//Check if current score is greater than highscore
if (Score > HighScore)
{
//if current score is greater than highscore then save this score
//and set the newHighscore flag to true
PlayerPrefs.SetInt(HIGHSCORE, Score);
HasNewHighScore = true;
}
Activate Windows


else
{
//if current score is less than highscore then set the newHighscore fla
HasNewHighScore = false;
}
}
}

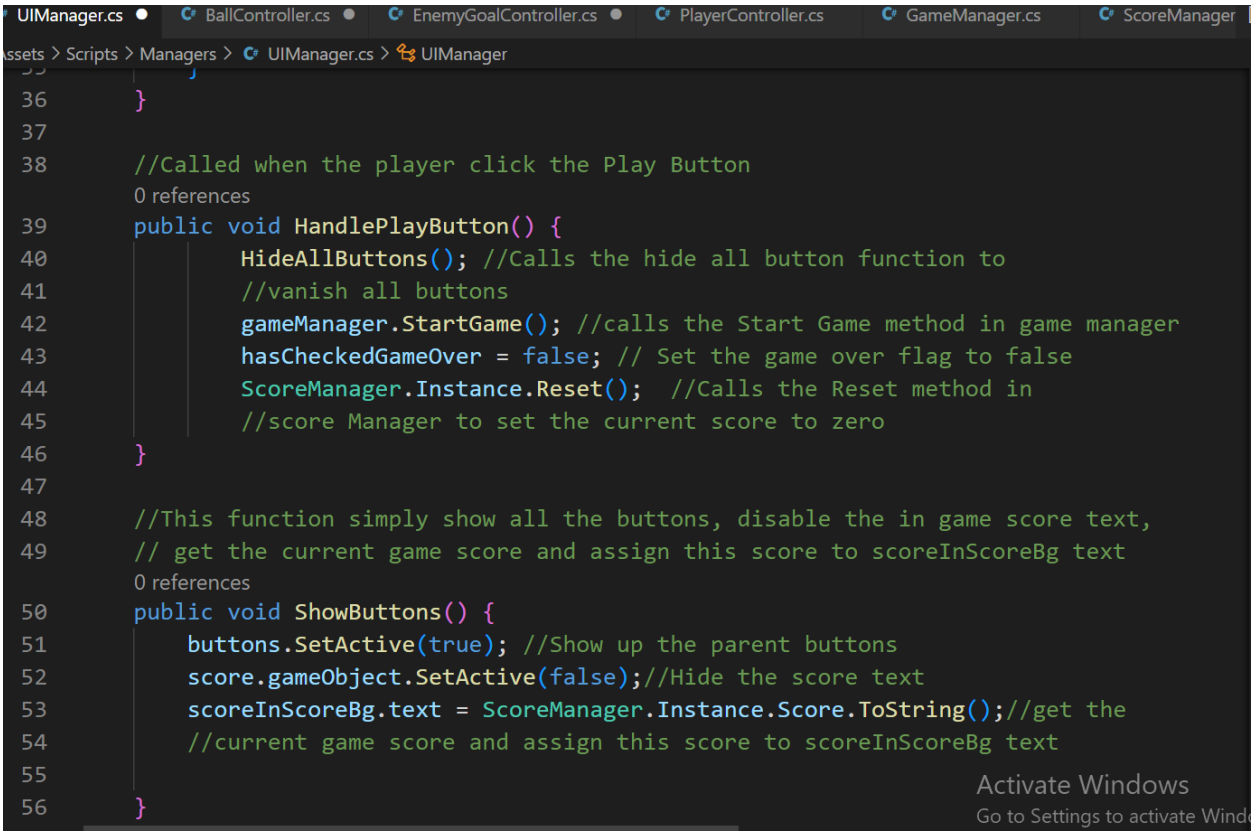
UI Manager



```
UIManager.cs ● | BallController.cs ● | EnemyGoalController.cs ● | PlayerController.cs | GameManager.cs | ScoreM
sets > Scripts > Managers > UIManager.cs > UIManager
1  using UnityEngine;
2  using UnityEngine.UI;
3  using System.Collections;
4
   1 reference
5  public class UIManager : MonoBehaviour {
       2 references
6      public GameManager gameManager;//An object of type GameManager for accessing
7      // the game manager methods and properties
       4 references
8      public Text score;//An object of type text to display current score during
9      //gameplay
       2 references
10     public Text scoreInScoreBg;//An object of type text to display player score
11     //when the game is over
       1 reference
12     public Text bestScore;//An object of type text to display player best score
13     //when game is over
       6 references
14     public Text multiplier_text;//An object of type text to display the current
15     //score multiplier
       2 references
16     public GameObject buttons;//An object of type game object which store the
17     //parent game object of all buttons
```



```
UIManager.cs ● | BallController.cs ● | EnemyGoalController.cs ● | PlayerController.cs | GameManager.cs | ScoreMana
Assets > Scripts > Managers > UIManager.cs > UIManager
       2 references
16     public GameObject buttons;//An object of type game object which store the
17     //parent game object of all buttons
       3 references
18     bool hasCheckedGameOver = false;//A flag which will be true when the game
19     //is over
20
21
22     // Use this for initialization
       0 references
23     void Start() {
24         score.gameObject.SetActive(false);
25         scoreInScoreBg.text = ScoreManager.Instance.Score.ToString();
26     }
27
28     // Update is called once per frame
       0 references
29     void Update() {
30         score.text = ScoreManager.Instance.Score.ToString();
31         bestScore.text = ScoreManager.Instance.HighScore.ToString();
32         if (gameManager.gameOver && !hasCheckedGameOver) {
33             hasCheckedGameOver = true;
34             Invoke("ShowButtons", 1f);
35         }
```


UIManager.cs
BallController.cs
EnemyGoalController.cs
PlayerController.cs
GameManager.cs
ScoreManager
ssets > Scripts > Managers > UIManager.cs > UIManager
36 }
37
38 //Called when the player click the Play Button
0 references
39 public void HandlePlayButton() {
40 HideAllButtons(); //Calls the hide all button function to
41 //vanish all buttons
42 gameManager.StartGame(); //calls the Start Game method in game manager
43 hasCheckedGameOver = false; // Set the game over flag to false
44 ScoreManager.Instance.Reset(); //Calls the Reset method in
45 //score Manager to set the current score to zero
46 }
47
48 //This function simply show all the buttons, disable the in game score text,
49 // get the current game score and assign this score to scoreInScoreBg text
0 references
50 public void ShowButtons() {
51 buttons.SetActive(true); //Show up the parent buttons
52 score.gameObject.SetActive(false);//Hide the score text
53 scoreInScoreBg.text = ScoreManager.Instance.Score.ToString();//get the
54 //current game score and assign this score to scoreInScoreBg text
55
56 }
Activate Windows
Go to Settings to activate Wind

GAME SCREENSHOTS:


2X
3