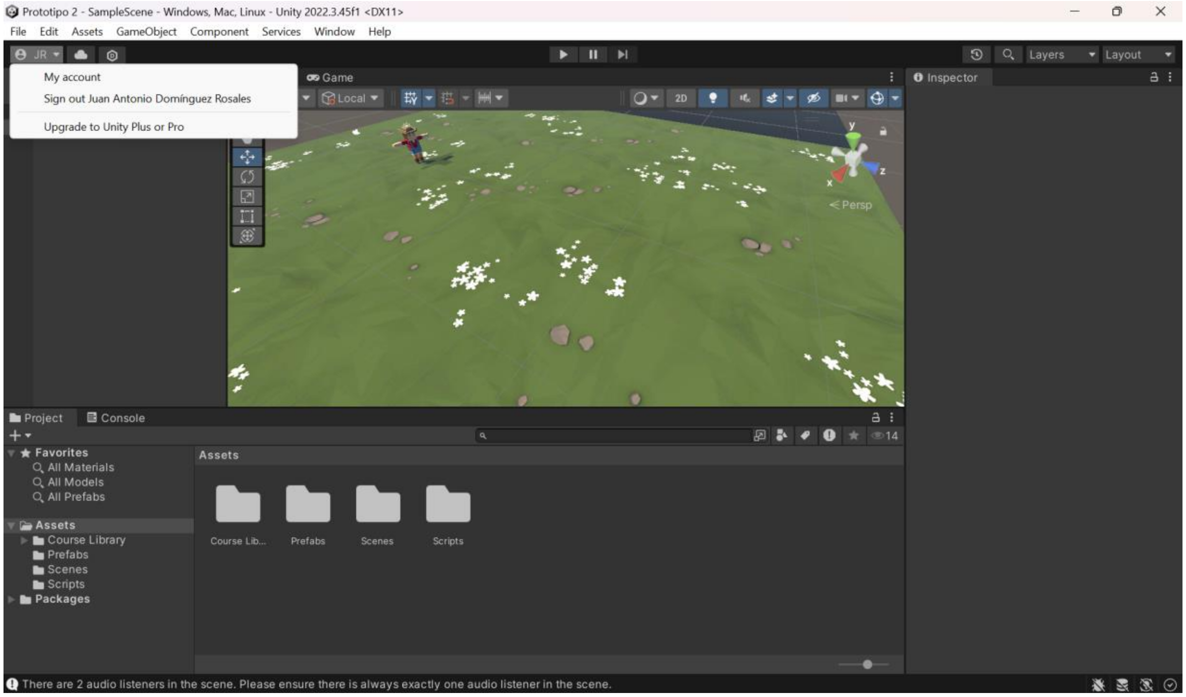
Prototipo 2

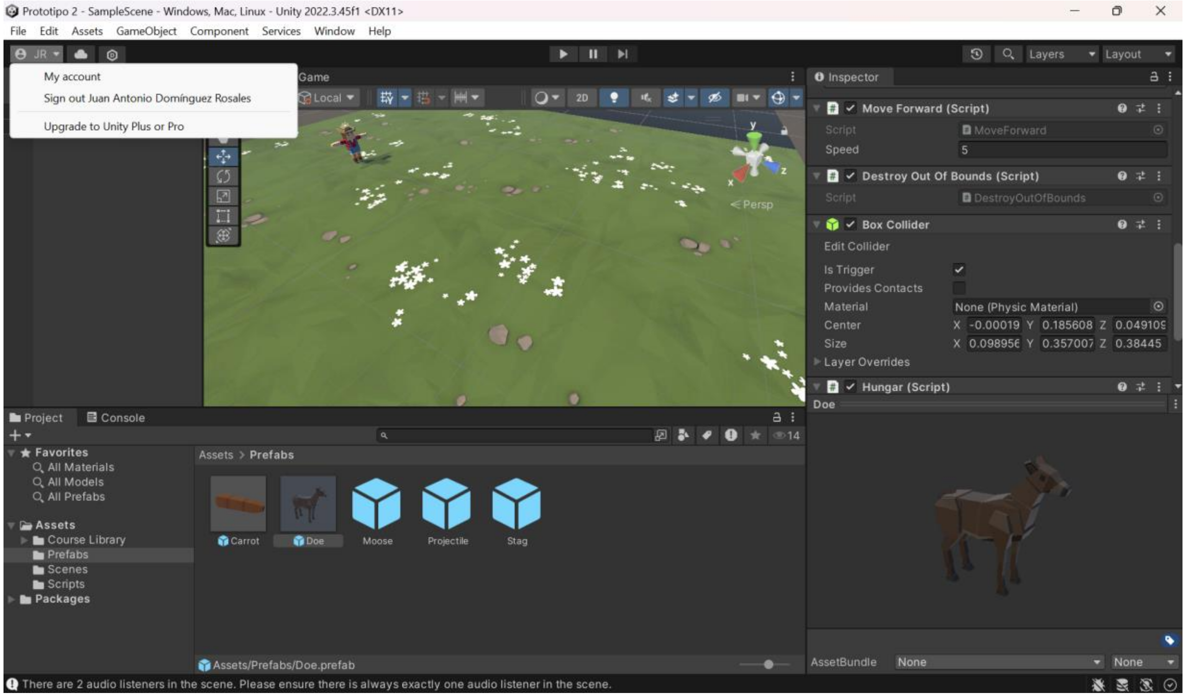
Juan Antonio Domínguez Rosales – GIDS4102

10/10/2024



Prototipo 2 - SampleScene - Windows, Mac, Linux - Unity 2022.3.45f1 <DX11>

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.



Prototipo 2 - SampleScene - Windows, Mac, Linux - Unity 2022.3.45f1 <DX11>

Move Forward (Script)
Script: MoveForward
Speed: 5

Destroy Out Of Bounds (Script)
Script: DestroyOutOfBounds

Box Collider
Is Trigger: ☑
Provides Contacts: ☐
Material: None (Physic Material)
Center: X -0.00019 Y 0.185608 Z 0.049109
Size: X 0.098956 Y 0.357007 Z 0.38445

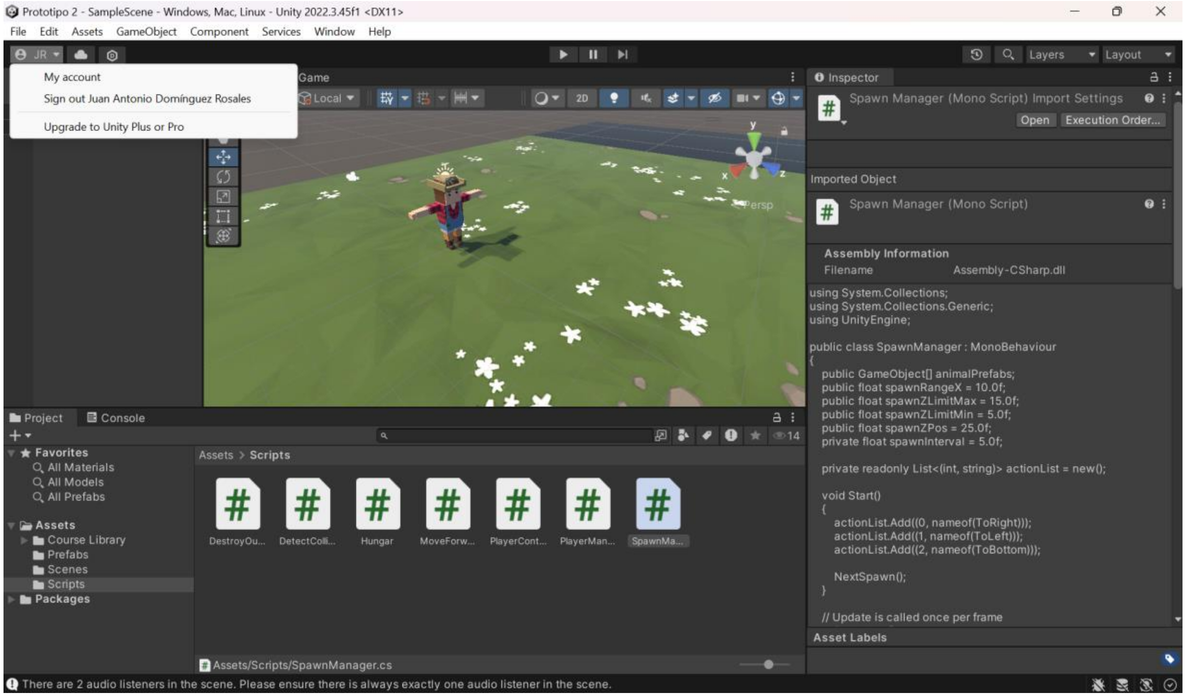
Hungar (Script)

Doe

Assets/Prefabs/Doe.prefab

AssetBundle: None None

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.



Prototipo 2 - SampleScene - Windows, Mac, Linux - Unity 2022.3.45f1 <DX11>

Spawn Manager (Mono Script) Import Settings

Imported Object

Spawn Manager (Mono Script)

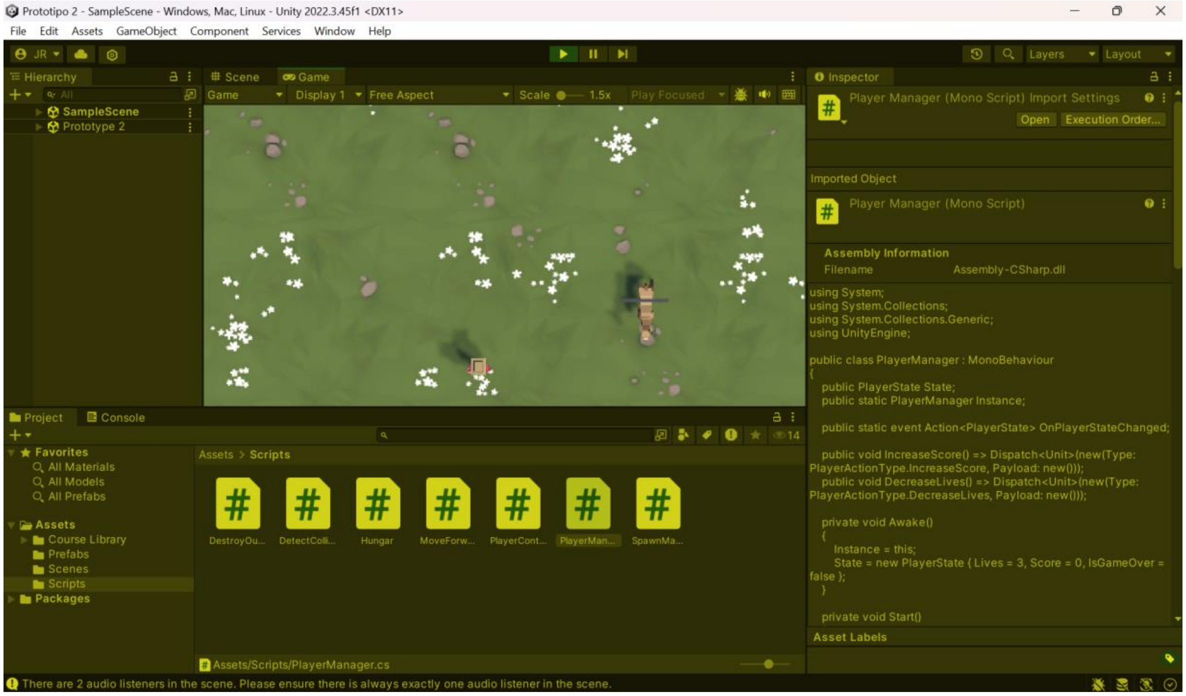
Assembly Information

Filename Assembly-CSharp.dll

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

public class SpawnManager : MonoBehaviour
{
   public GameObject[] animalPrefabs;
   public float spawnRangeX = 10.0f;
   public float spawnZLimitMax = 15.0f;
   public float spawnZLimitMin = 5.0f;
   public float spawnZPos = 25.0f;
   private float spawnInterval = 5.0f;

   private readonly List<(int, string)> actionList = new();

   void Start()
   {
      actionList.Add((0, nameof(ToRight)));
      actionList.Add((1, nameof(ToLeft)));
      actionList.Add((2, nameof(ToBottom)));

      NextSpawn();
   }

   // Update is called once per frame
```

Assets/Scripts/SpawnManager.cs

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.



Prototipo 2 - SampleScene - Windows, Mac, Linux - Unity 2022.3.45f1 <DX11>

Player Manager (Mono Script) Import Settings

Imported Object

Player Manager (Mono Script)

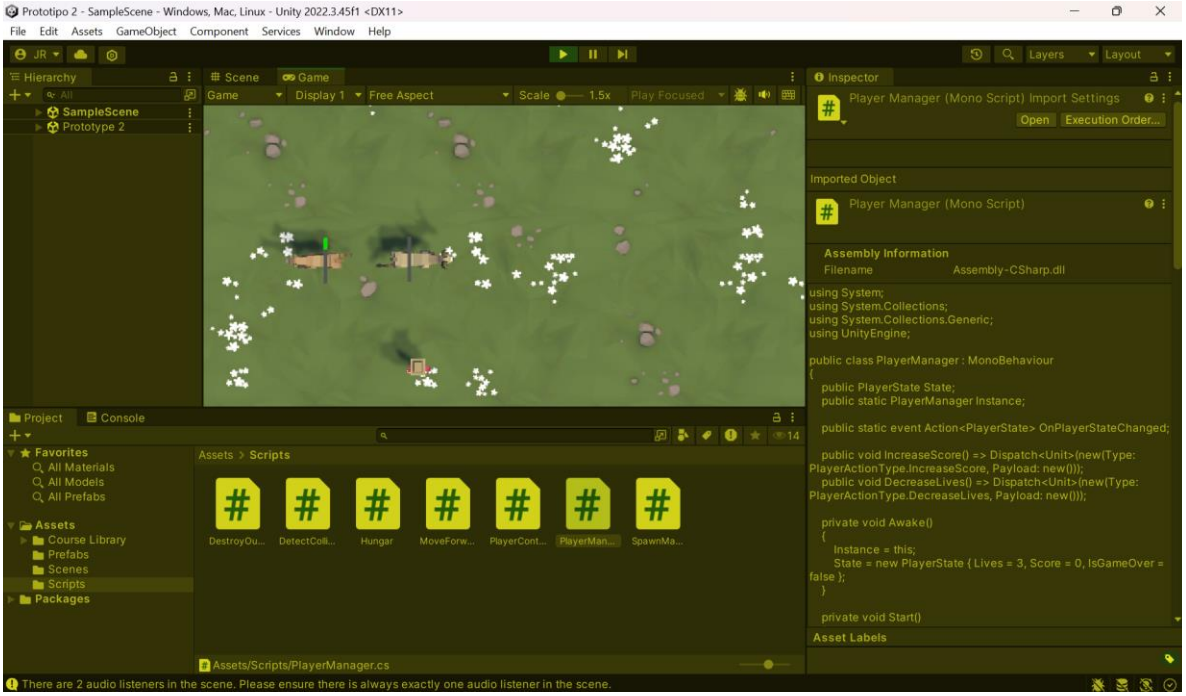
Assembly Information

Filename Assembly-CSharp.dll

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

public class PlayerManager : MonoBehaviour
{
   public PlayerState State;
   public static PlayerManager Instance;

   public static event Action<PlayerState> OnPlayerStateChanged;

   public void IncreaseScore() => Dispatch<Unit>(new(Type:
PlayerActionType.IncreaseScore, Payload: new()));
   public void DecreaseLives() => Dispatch<Unit>(new(Type:
PlayerActionType.DecreaseLives, Payload: new()));

   private void Awake()
   {
      Instance = this;
      State = new PlayerState { Lives = 3, Score = 0, IsGameOver =
false };
   }

   private void Start()
```

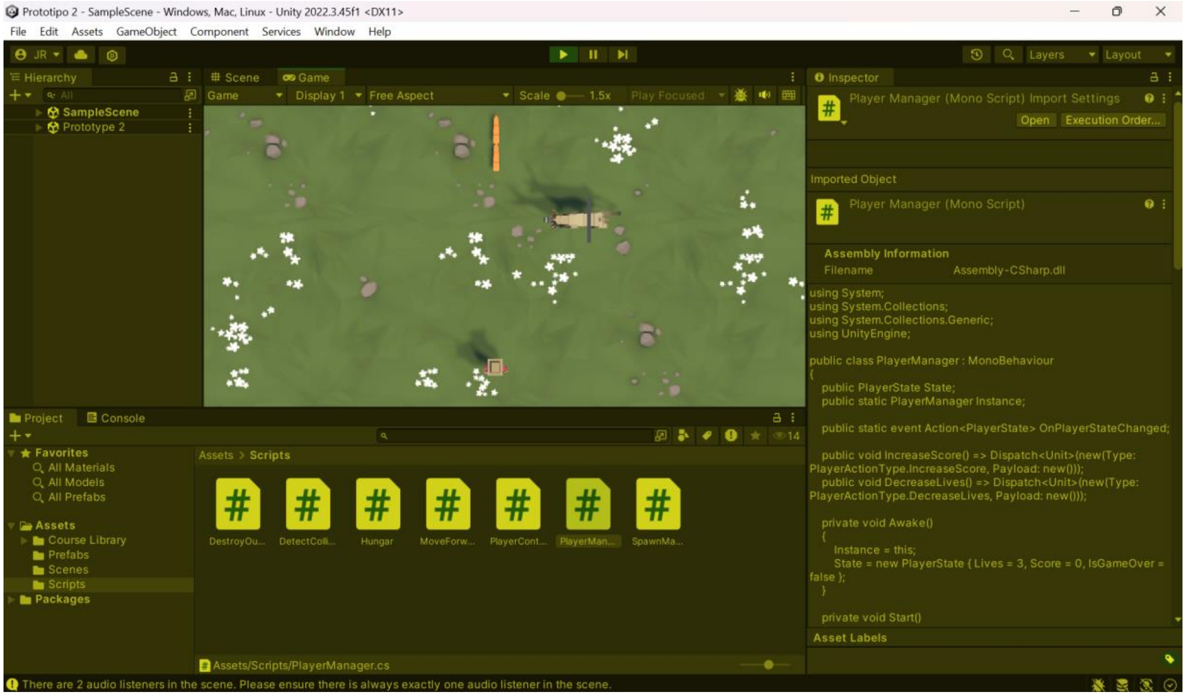
Assets/Scripts/PlayerManager.cs

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.



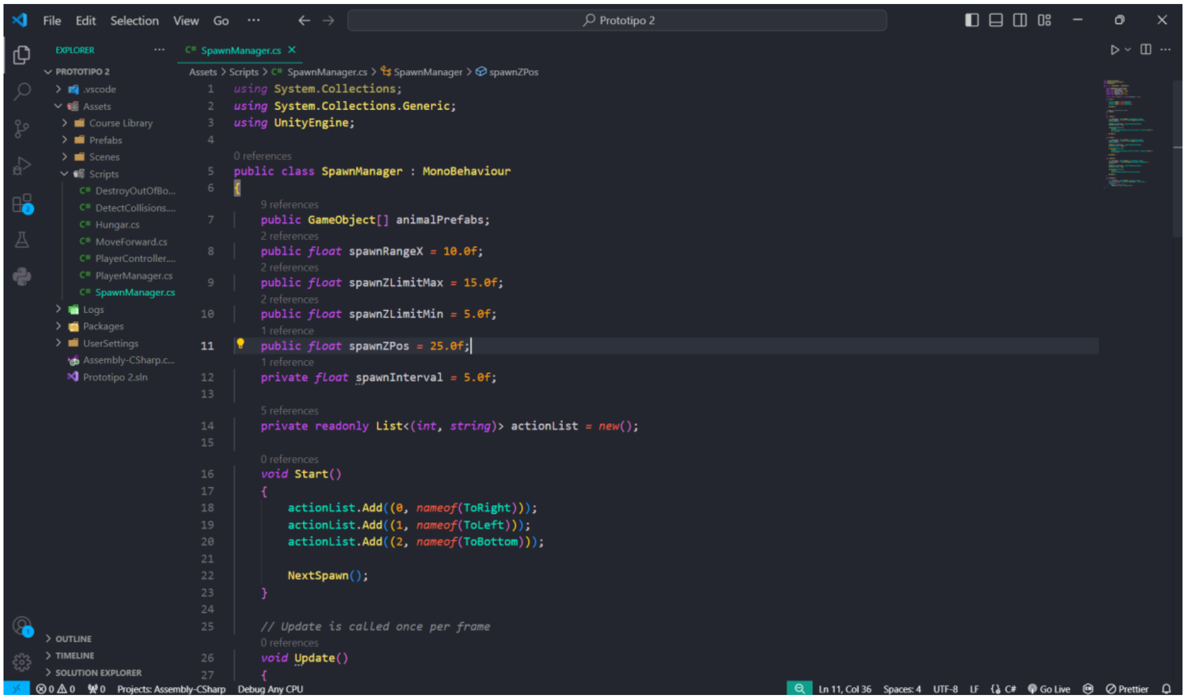
```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerManager : MonoBehaviour
{
   public PlayerState State;
   public static PlayerManager Instance;

   public static event Action<PlayerState> OnPlayerStateChanged;

   public void IncreaseScore() => Dispatch<Unit>(new(Type:
PlayerActionType.IncreaseScore, Payload: new()));
   public void DecreaseLives() => Dispatch<Unit>(new(Type:
PlayerActionType.DecreaseLives, Payload: new()));

   private void Awake()
   {
      Instance = this;
      State = new PlayerState { Lives = 3, Score = 0, IsGameOver =
false };
   }

   private void Start()
```

There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.

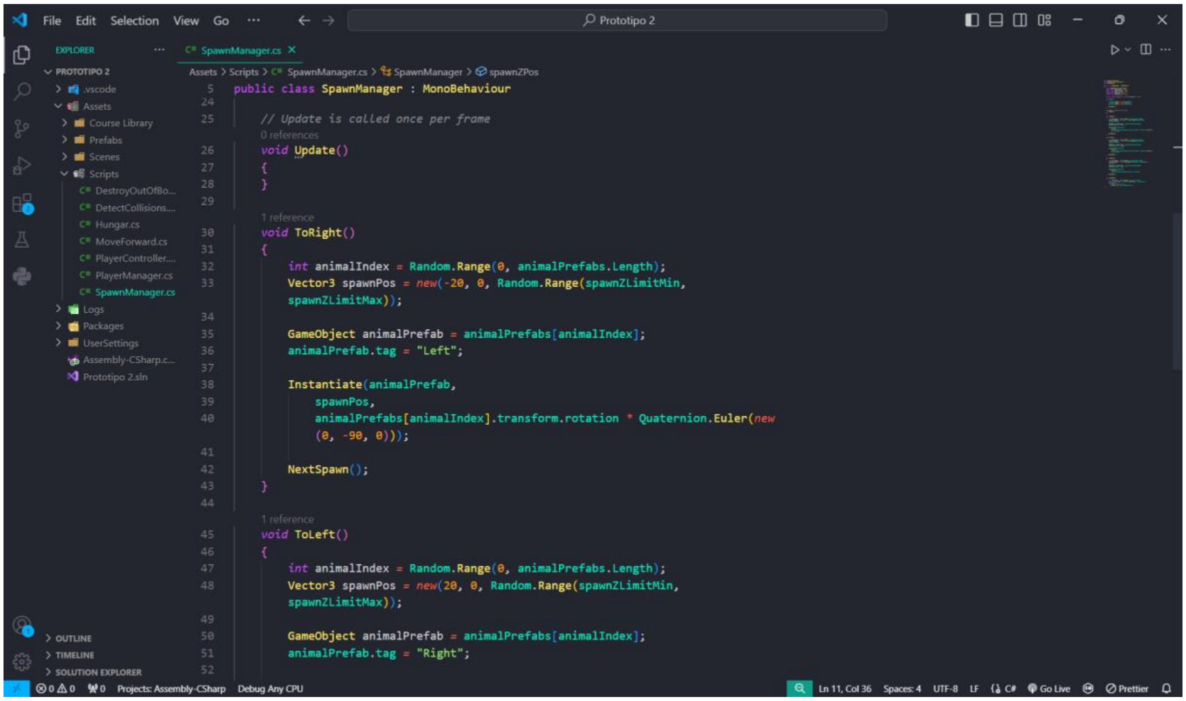


```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerManager : MonoBehaviour
{
   public PlayerState State;
   public static PlayerManager Instance;

   public static event Action<PlayerState> OnPlayerStateChanged;

   public void IncreaseScore() => Dispatch<Unit>(new(Type:
PlayerActionType.IncreaseScore, Payload: new()));
   public void DecreaseLives() => Dispatch<Unit>(new(Type:
PlayerActionType.DecreaseLives, Payload: new()));

   private void Awake()
   {
      Instance = this;
      State = new PlayerState { Lives = 3, Score = 0, IsGameOver =
false };
   }

   private void Start()
```

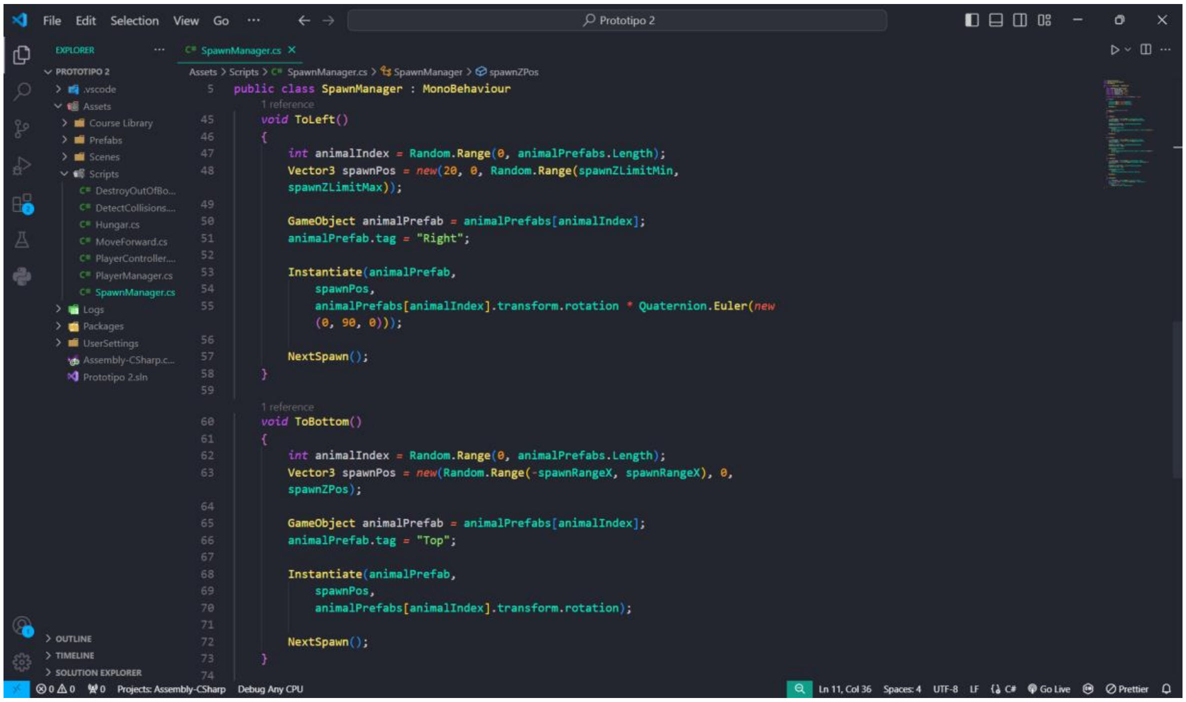
There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.



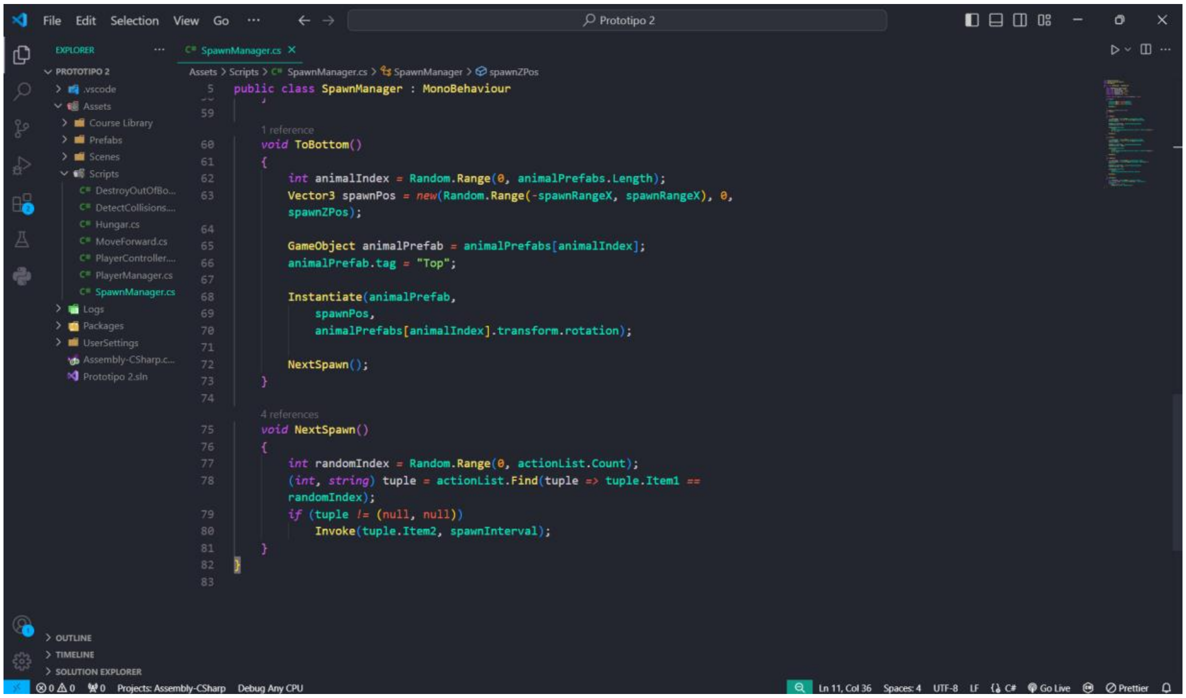
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class SpawnManager : MonoBehaviour
{
    9 references
    public GameObject[] animalPrefabs;
    2 references
    public float spawnRangeX = 10.0f;
    2 references
    public float spawnZLimitMax = 15.0f;
    2 references
    public float spawnZLimitMin = 5.0f;
    1 reference
    public float spawnZPos = 25.0f;
    1 reference
    private float spawnInterval = 5.0f;

    5 references
    private readonly List<(int, string)> actionList = new();

    0 references
    void Start()
    {
        actionList.Add((0, nameof(ToRight)));
        actionList.Add((1, nameof(ToLeft)));
        actionList.Add((2, nameof(ToBottom)));

        NextSpawn();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
```



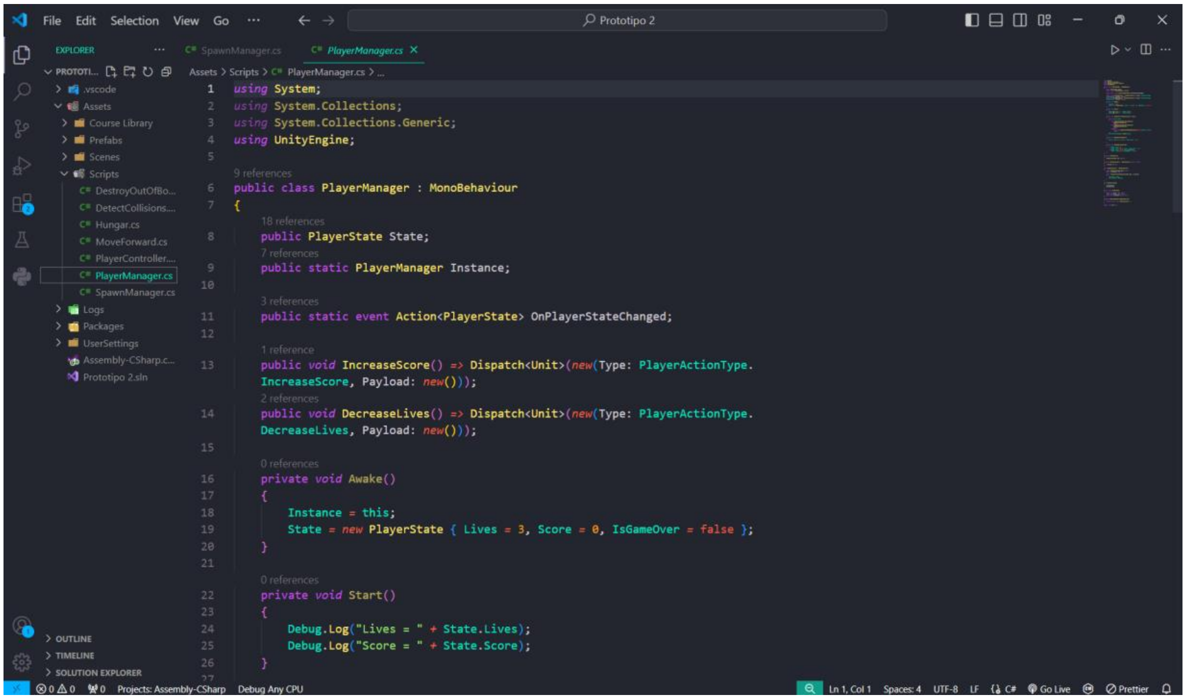
```csharp
public class SpawnManager : MonoBehaviour

    // Update is called once per frame
    0 references
    void Update()
    {
    }

    1 reference
    void ToRight()
    {
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new(-20, 0, Random.Range(spawnZLimitMin,
        spawnZLimitMax));

        GameObject animalPrefab = animalPrefabs[animalIndex];
        animalPrefab.tag = "Left";

        Instantiate(animalPrefab,
            spawnPos,
            animalPrefabs[animalIndex].transform.rotation * Quaternion.Euler(new
            (0, -90, 0)));

        NextSpawn();
    }

    1 reference
    void ToLeft()
    {
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new(20, 0, Random.Range(spawnZLimitMin,
        spawnZLimitMax));

        GameObject animalPrefab = animalPrefabs[animalIndex];
        animalPrefab.tag = "Right";

```



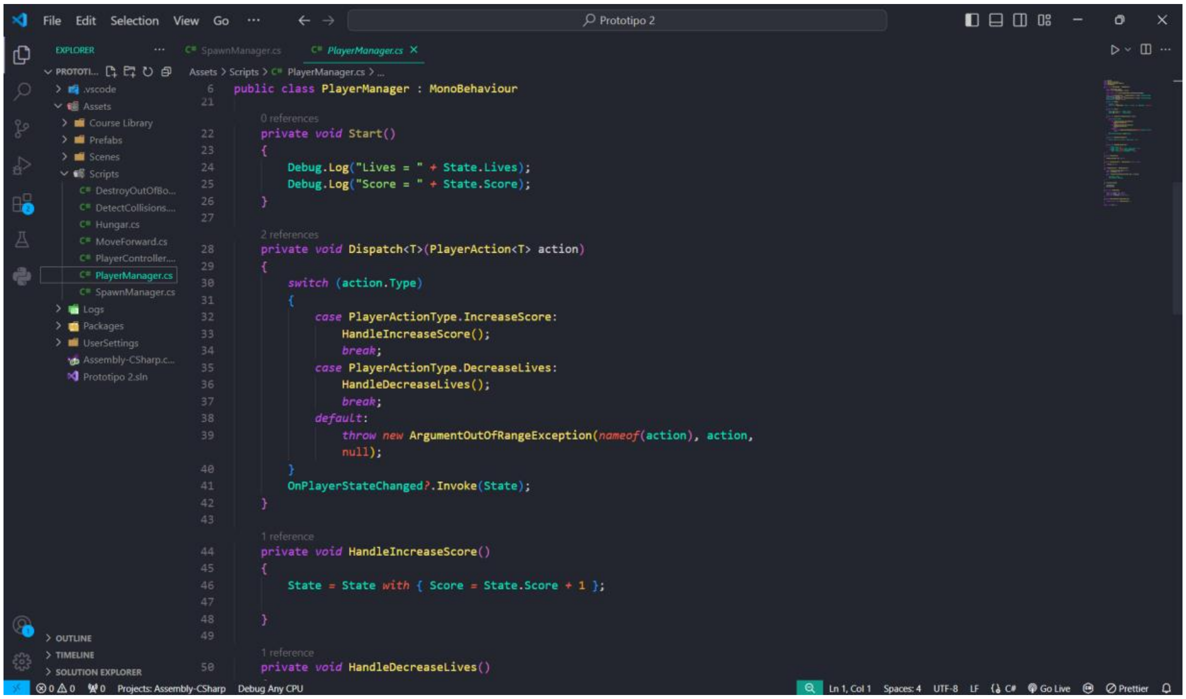
```
public class SpawnManager : MonoBehaviour
    1 reference
    void ToLeft()
    {
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new(20, 0, Random.Range(spawnZLimitMin,
        spawnZLimitMax));

        GameObject animalPrefab = animalPrefabs[animalIndex];
        animalPrefab.tag = "Right";

        Instantiate(animalPrefab,
            spawnPos,
            animalPrefabs[animalIndex].transform.rotation * Quaternion.Euler(new
            (0, 90, 0)));

        NextSpawn();
    }

    1 reference
    void ToBottom()
    {
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new(Random.Range(-spawnRangeX, spawnRangeX), 0,
        spawnZPos);

        GameObject animalPrefab = animalPrefabs[animalIndex];
        animalPrefab.tag = "Top";

        Instantiate(animalPrefab,
            spawnPos,
            animalPrefabs[animalIndex].transform.rotation);

        NextSpawn();
    }
```



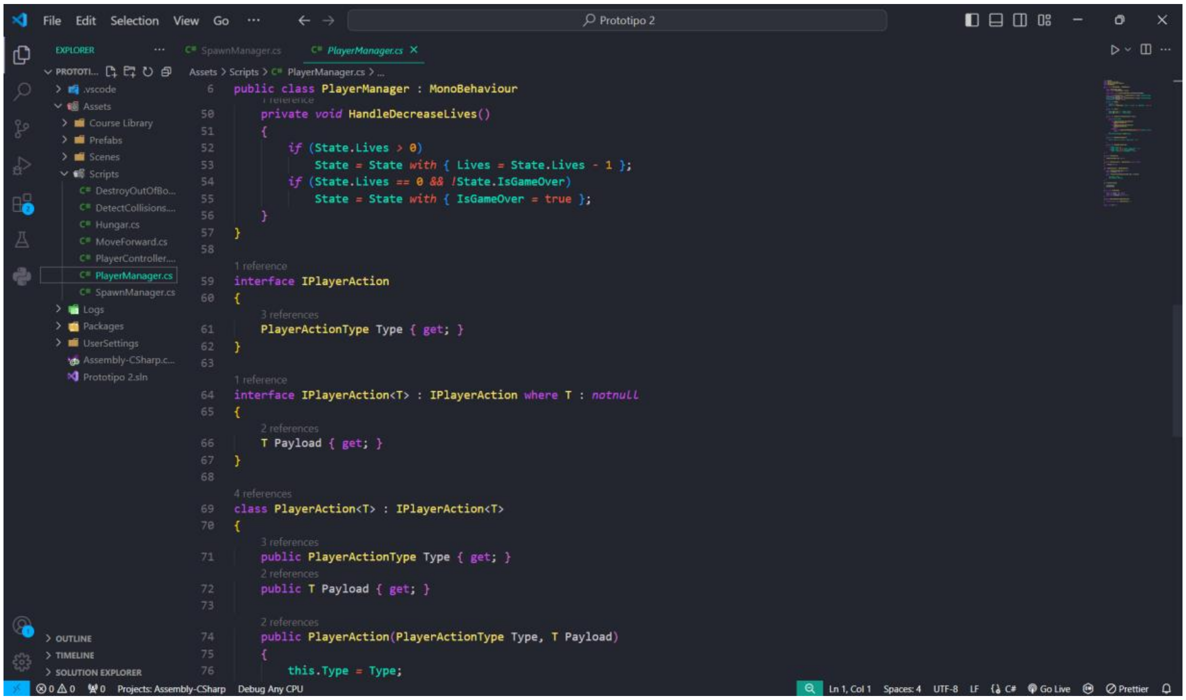
```
public class SpawnManager : MonoBehaviour

    1 reference
    void ToBottom()
    {
        int animalIndex = Random.Range(0, animalPrefabs.Length);
        Vector3 spawnPos = new(Random.Range(-spawnRangeX, spawnRangeX), 0,
        spawnZPos);

        GameObject animalPrefab = animalPrefabs[animalIndex];
        animalPrefab.tag = "Top";

        Instantiate(animalPrefab,
            spawnPos,
            animalPrefabs[animalIndex].transform.rotation);

        NextSpawn();
    }

    4 references
    void NextSpawn()
    {
        int randomIndex = Random.Range(0, actionList.Count);
        (int, string) tuple = actionList.Find(tuple => tuple.Item1 ==
        randomIndex);
        if (tuple != (null, null))
            Invoke(tuple.Item2, spawnInterval);
    }
}
```



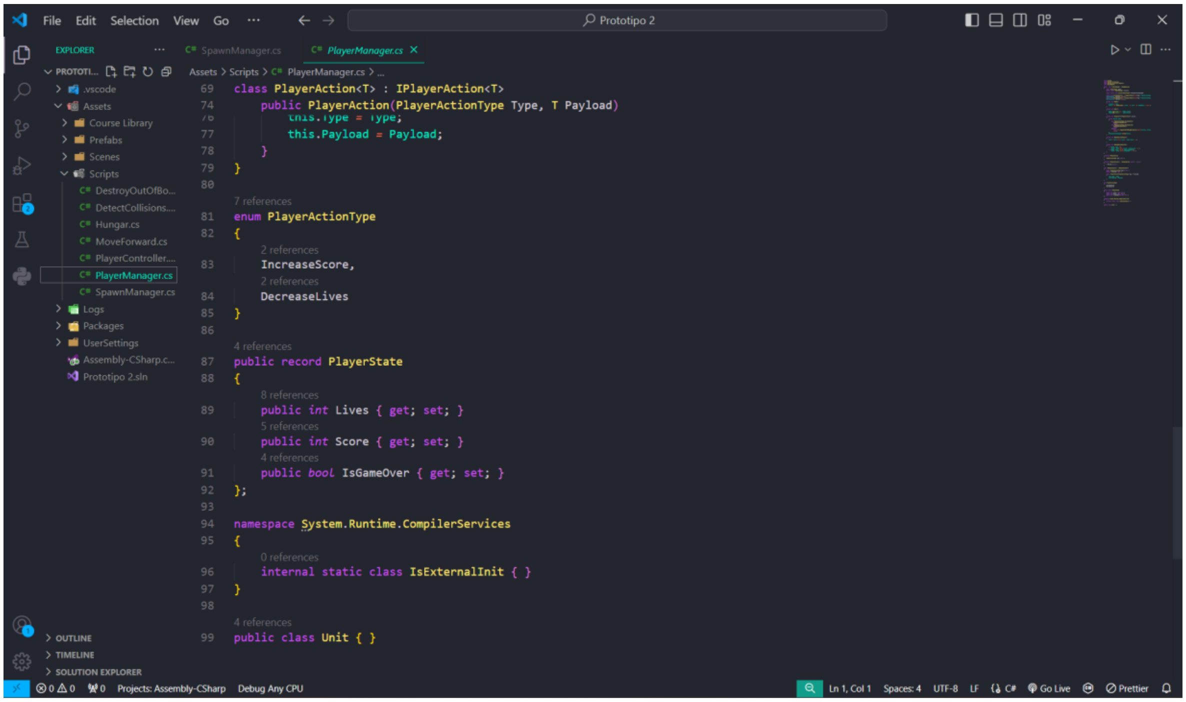
```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

9 references
public class PlayerManager : MonoBehaviour
{
    18 references
    public PlayerState State;
    7 references
    public static PlayerManager Instance;

    3 references
    public static event Action<PlayerState> OnPlayerStateChanged;

    1 reference
    public void IncreaseScore() => Dispatch<Unit>(new(Type: PlayerActionType.
    IncreaseScore, Payload: new()));
    2 references
    public void DecreaseLives() => Dispatch<Unit>(new(Type: PlayerActionType.
    DecreaseLives, Payload: new()));

    0 references
    private void Awake()
    {
        Instance = this;
        State = new PlayerState { Lives = 3, Score = 0, IsGameOver = false };
    }

    0 references
    private void Start()
    {
        Debug.Log("Lives = " + State.Lives);
        Debug.Log("Score = " + State.Score);
    }
```



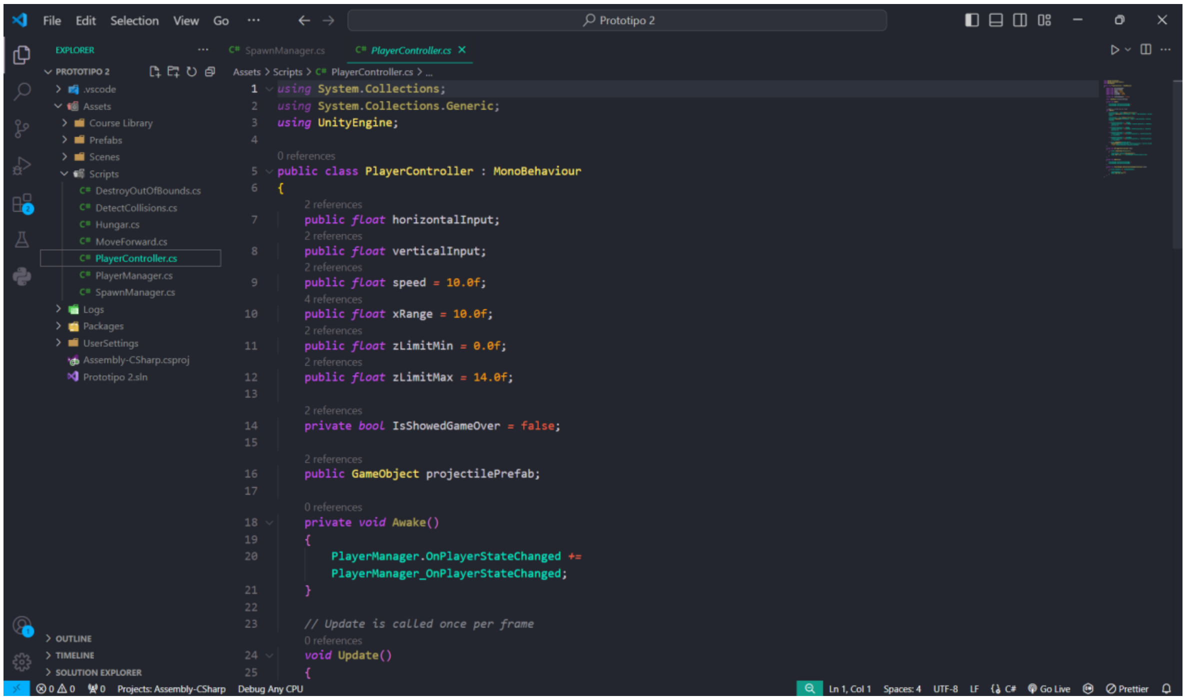
```
public class PlayerManager : MonoBehaviour

    0 references
    private void Start()
    {
        Debug.Log("Lives = " + State.Lives);
        Debug.Log("Score = " + State.Score);
    }

    2 references
    private void Dispatch<T>(PlayerAction<T> action)
    {
        switch (action.Type)
        {
            case PlayerActionType.IncreaseScore:
                HandleIncreaseScore();
                break;
            case PlayerActionType.DecreaseLives:
                HandleDecreaseLives();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(action), action,
                null);
        }
        OnPlayerStateChanged?.Invoke(State);
    }

    1 reference
    private void HandleIncreaseScore()
    {
        State = State with { Score = State.Score + 1 };

    }

    1 reference
    private void HandleDecreaseLives()
```



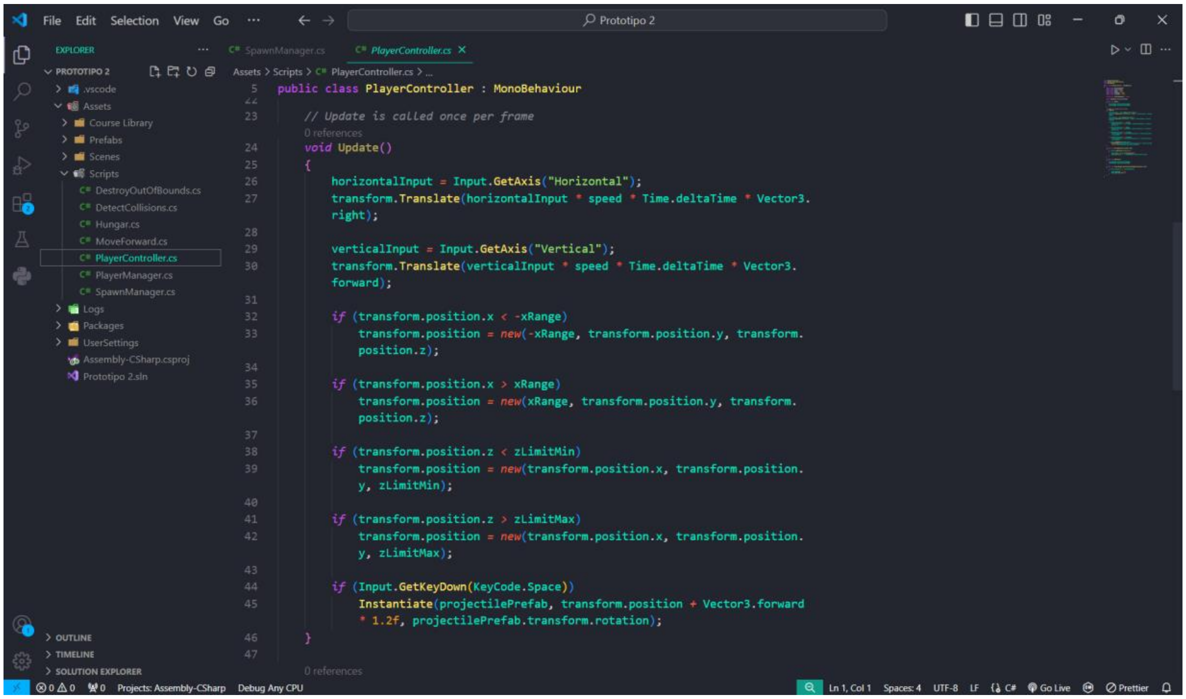
```
public class PlayerManager : MonoBehaviour
    1 reference
    private void HandleDecreaseLives()
    {
        if (State.Lives > 0)
            State = State with { Lives = State.Lives - 1 };
        if (State.Lives == 0 && !State.IsGameOver)
            State = State with { IsGameOver = true };
    }
}

1 reference
interface IPlayerAction
{
    3 references
    PlayerActionType Type { get; }
}

1 reference
interface IPlayerAction<T> : IPlayerAction where T : notnull
{
    2 references
    T Payload { get; }
}

4 references
class PlayerAction<T> : IPlayerAction<T>
{
    3 references
    public PlayerActionType Type { get; }
    2 references
    public T Payload { get; }

    2 references
    public PlayerAction(PlayerActionType Type, T Payload)
    {
        this.Type = Type;
```



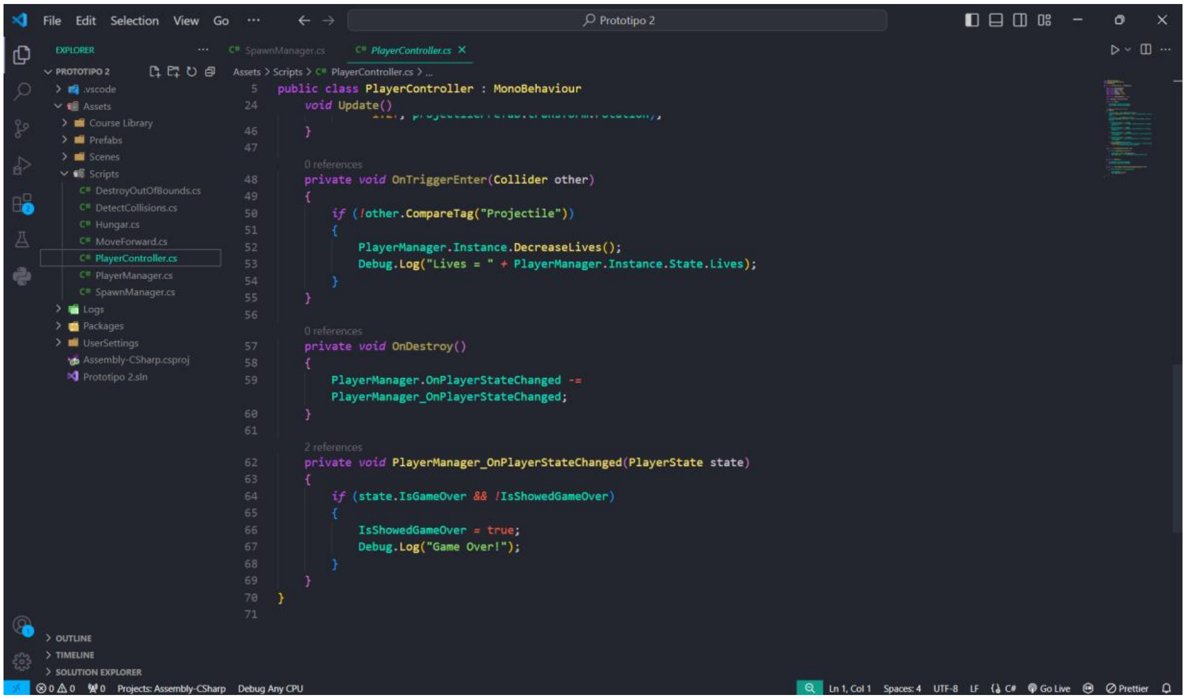
```
class PlayerAction<T> : IPlayerAction<T>
    public PlayerAction(PlayerActionType Type, T Payload)
        this.Type = Type;
        this.Payload = Payload;
    }
}

7 references
enum PlayerActionType
{
    2 references
    IncreaseScore,
    2 references
    DecreaseLives
}

4 references
public record PlayerState
{
    8 references
    public int Lives { get; set; }
    5 references
    public int Score { get; set; }
    4 references
    public bool IsGameOver { get; set; }
};

namespace System.Runtime.CompilerServices
{
    0 references
    internal static class IsExternalInit { }
}

4 references
public class Unit { }
```



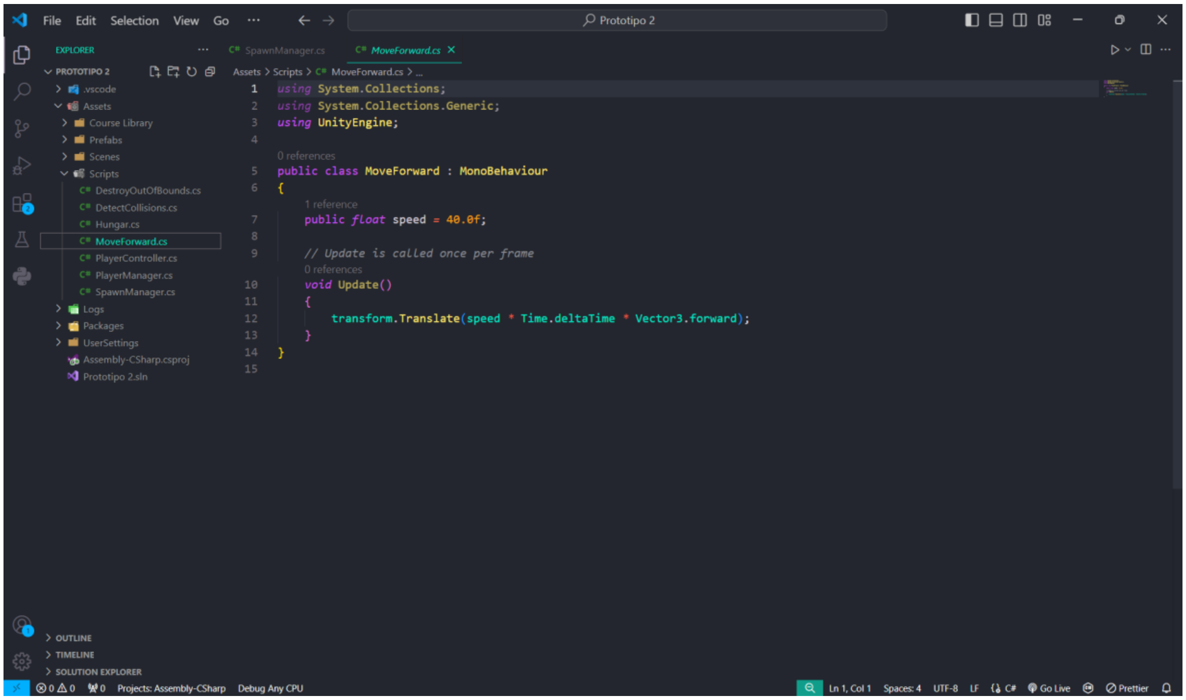
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class PlayerController : MonoBehaviour
{
    2 references
    public float horizontalInput;
    2 references
    public float verticalInput;
    2 references
    public float speed = 10.0f;
    4 references
    public float xRange = 10.0f;
    2 references
    public float zLimitMin = 0.0f;
    2 references
    public float zLimitMax = 14.0f;

    2 references
    private bool IsShowedGameOver = false;

    2 references
    public GameObject projectilePrefab;

    0 references
    private void Awake()
    {
        PlayerManager.OnPlayerStateChanged +=
        PlayerManager_OnPlayerStateChanged;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
```



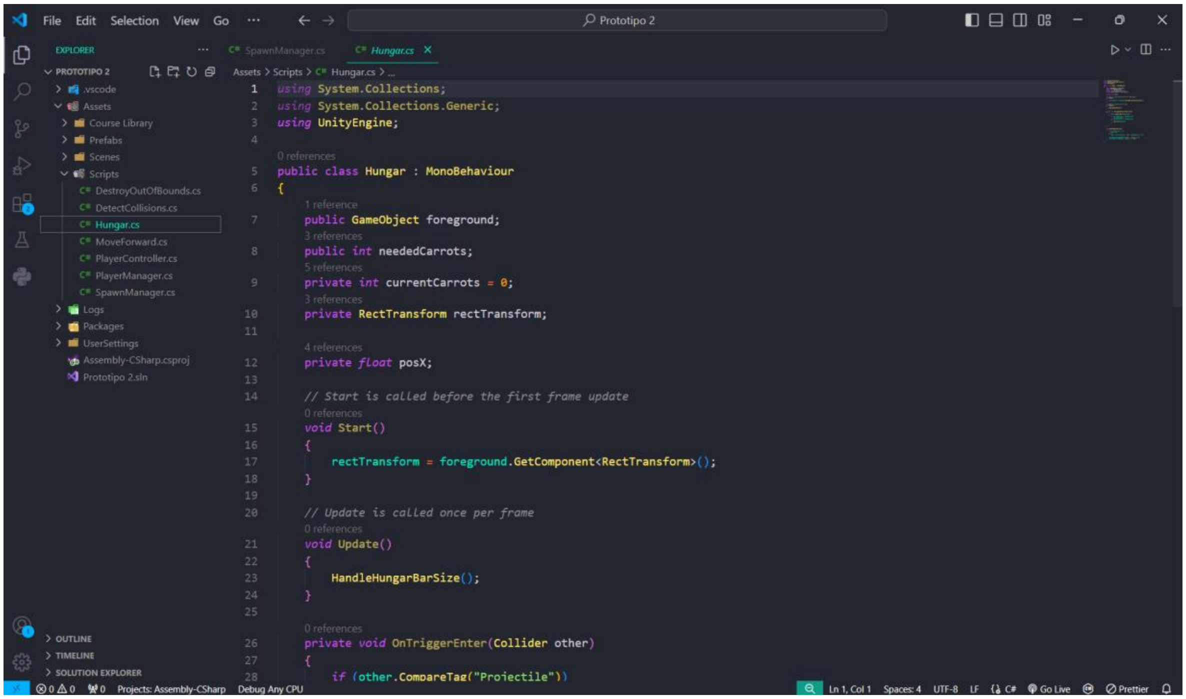
```
public class PlayerController : MonoBehaviour

    // Update is called once per frame
    0 references
    void Update()
    {
        horizontalInput = Input.GetAxis("Horizontal");
        transform.Translate(horizontalInput * speed * Time.deltaTime * Vector3.
        right);

        verticalInput = Input.GetAxis("Vertical");
        transform.Translate(verticalInput * speed * Time.deltaTime * Vector3.
        forward);

        if (transform.position.x < -xRange)
            transform.position = new(-xRange, transform.position.y, transform.
            position.z);

        if (transform.position.x > xRange)
            transform.position = new(xRange, transform.position.y, transform.
            position.z);

        if (transform.position.z < zLimitMin)
            transform.position = new(transform.position.x, transform.position.
            y, zLimitMin);

        if (transform.position.z > zLimitMax)
            transform.position = new(transform.position.x, transform.position.
            y, zLimitMax);

        if (Input.GetKeyDown(KeyCode.Space))
            Instantiate(projectilePrefab, transform.position + Vector3.forward
            * 1.2f, projectilePrefab.transform.rotation);
    }

    0 references
```



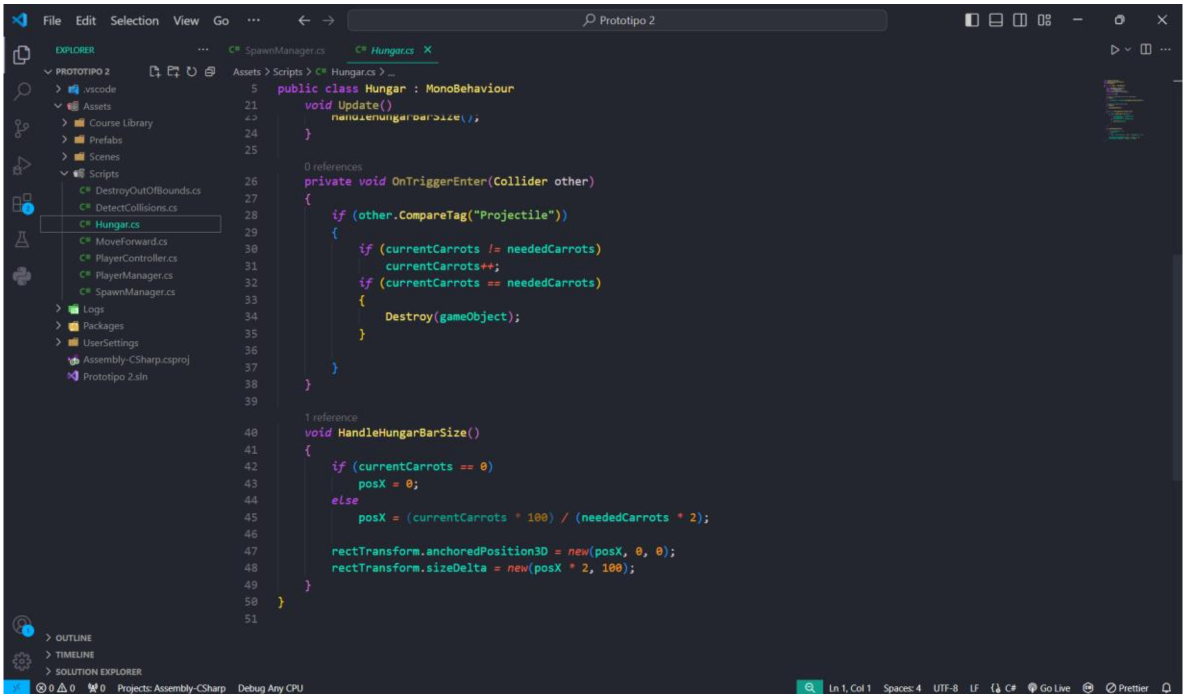
```
public class PlayerController : MonoBehaviour
    void Update()
                1.2f), projectilePrefab.transform.rotation);
    }

    0 references
    private void OnTriggerEnter(Collider other)
    {
        if (!other.CompareTag("Projectile"))
        {
            PlayerManager.Instance.DecreaseLives();
            Debug.Log("Lives = " + PlayerManager.Instance.State.Lives);
        }
    }

    0 references
    private void OnDestroy()
    {
        PlayerManager.OnPlayerStateChanged -=
        PlayerManager_OnPlayerStateChanged;
    }

    2 references
    private void PlayerManager_OnPlayerStateChanged(PlayerState state)
    {
        if (state.IsGameOver && !IsShowedGameOver)
        {
            IsShowedGameOver = true;
            Debug.Log("Game Over!");
        }
    }
}
```



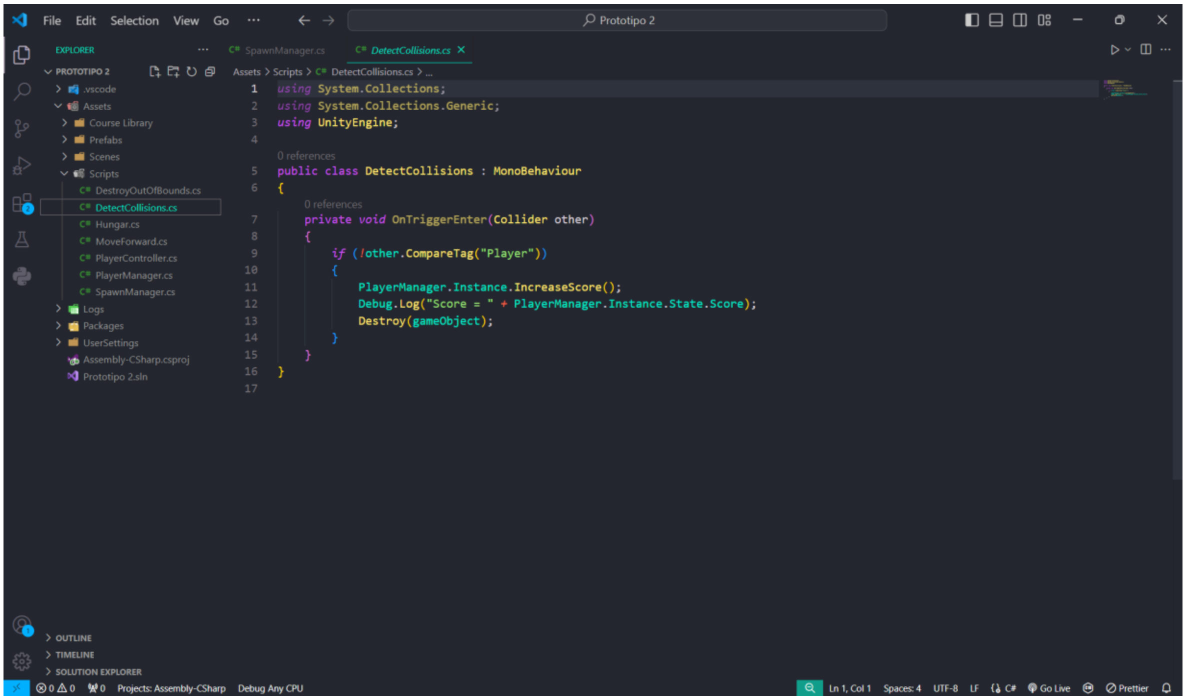
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class MoveForward : MonoBehaviour
{
    1 reference
    public float speed = 40.0f;

    // Update is called once per frame
    0 references
    void Update()
    {
        transform.Translate(speed * Time.deltaTime * Vector3.forward);
    }
}
```



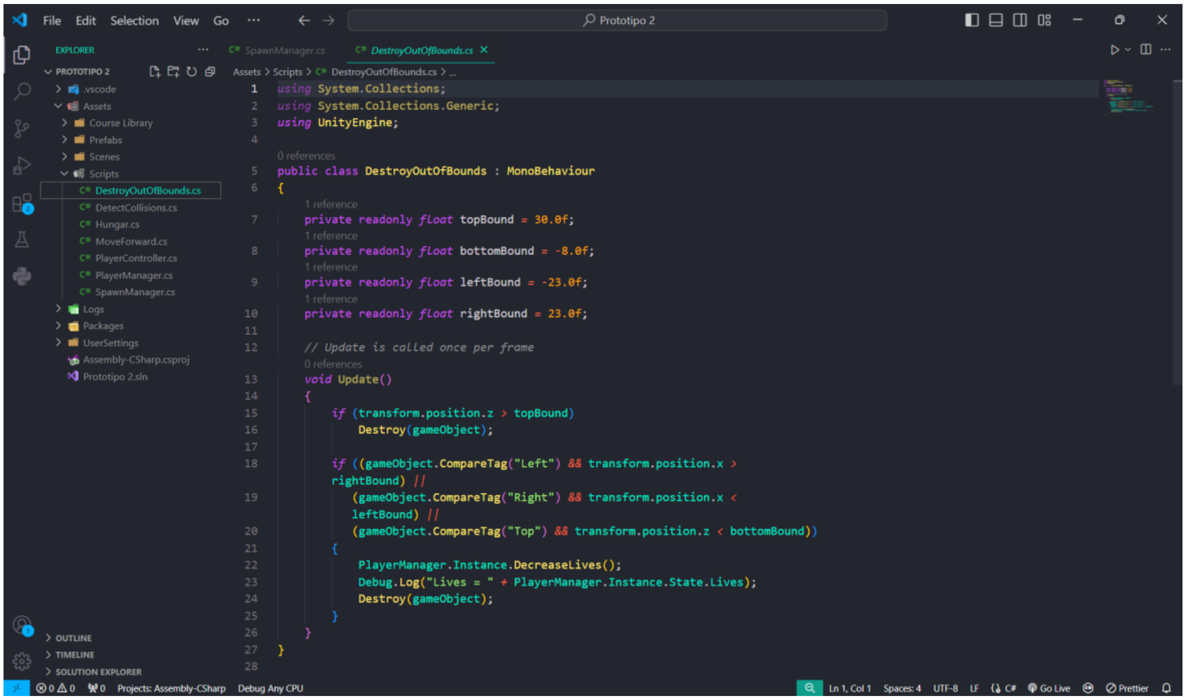
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

0 references
public class Hungar : MonoBehaviour
{
    1 reference
    public GameObject foreground;
    3 references
    public int neededCarrots;
    5 references
    private int currentCarrots = 0;
    3 references
    private RectTransform rectTransform;

    4 references
    private float posX;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        rectTransform = foreground.GetComponent<RectTransform>();
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        HandleHungarBarSize();
    }

    0 references
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Projectile"))
```



```
public class Hungar : MonoBehaviour
    void Update()
        HandleHungarBarSize();
    }

    0 references
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Projectile"))
        {
            if (currentCarrots != neededCarrots)
                currentCarrots++;
            if (currentCarrots == neededCarrots)
            {
                Destroy(gameObject);
            }

        }
    }

    1 reference
    void HandleHungarBarSize()
    {
        if (currentCarrots == 0)
            posX = 0;
        else
            posX = (currentCarrots * 100) / (neededCarrots * 2);

        rectTransform.anchoredPosition3D = new(posX, 0, 0);
        rectTransform.sizeDelta = new(posX * 2, 100);
    }
}
```



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

0 references
public class DetectCollisions : MonoBehaviour
{
    0 references
    private void OnTriggerEnter(Collider other)
    {
        if (!other.CompareTag("Player"))
        {
            PlayerManager.Instance.IncreaseScore();
            Debug.Log("Score = " + PlayerManager.Instance.State.Score);
            Destroy(gameObject);
        }
    }
}
```



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

0 references
public class DestroyOutOfBounds : MonoBehaviour
{
    1 reference
    private readonly float topBound = 30.0f;
    1 reference
    private readonly float bottomBound = -8.0f;
    1 reference
    private readonly float leftBound = -23.0f;
    1 reference
    private readonly float rightBound = 23.0f;

    // Update is called once per frame
    0 references
    void Update()
    {
        if (transform.position.z > topBound)
            Destroy(gameObject);

        if ((gameObject.CompareTag("Left") && transform.position.x >
        rightBound) ||
           (gameObject.CompareTag("Right") && transform.position.x <
           leftBound) ||
           (gameObject.CompareTag("Top") && transform.position.z < bottomBound))
        {
            PlayerManager.Instance.DecreaseLives();
            Debug.Log("Lives = " + PlayerManager.Instance.State.Lives);
            Destroy(gameObject);
        }
    }
}
```