RICHARD BERTHOLD

5139 MCSD GPG512 SA Screenshots

CONTENTS

INTRODUCTION

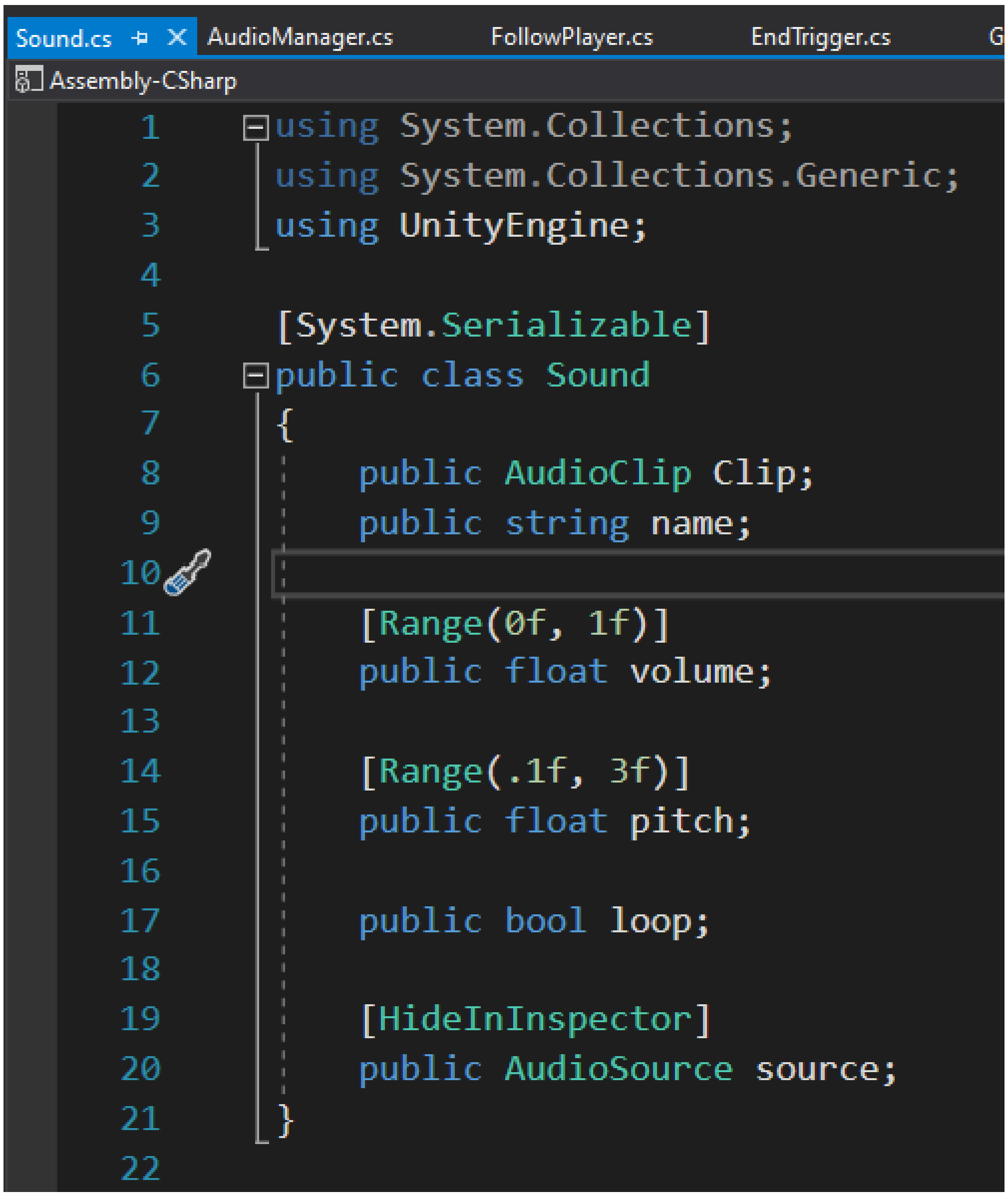
You have been tasked to create a Game that showcases your skills for a Game Development Company and have requested that the following be included in your Game:

- Must have Multiple levels/Rounds
- Must be a 3D Game
- Must have a Genre
- Must have a start Menu
- Game must have 1 Objective per level/Round
- Game Should end if Player health is 0 or if the Timer runs out
- Game must have a UI that shows the current objective
- No Payed/completed game assets may be used(Free Objects may be used but no completed Games may be used and/or altered)

SCREENSHOTS

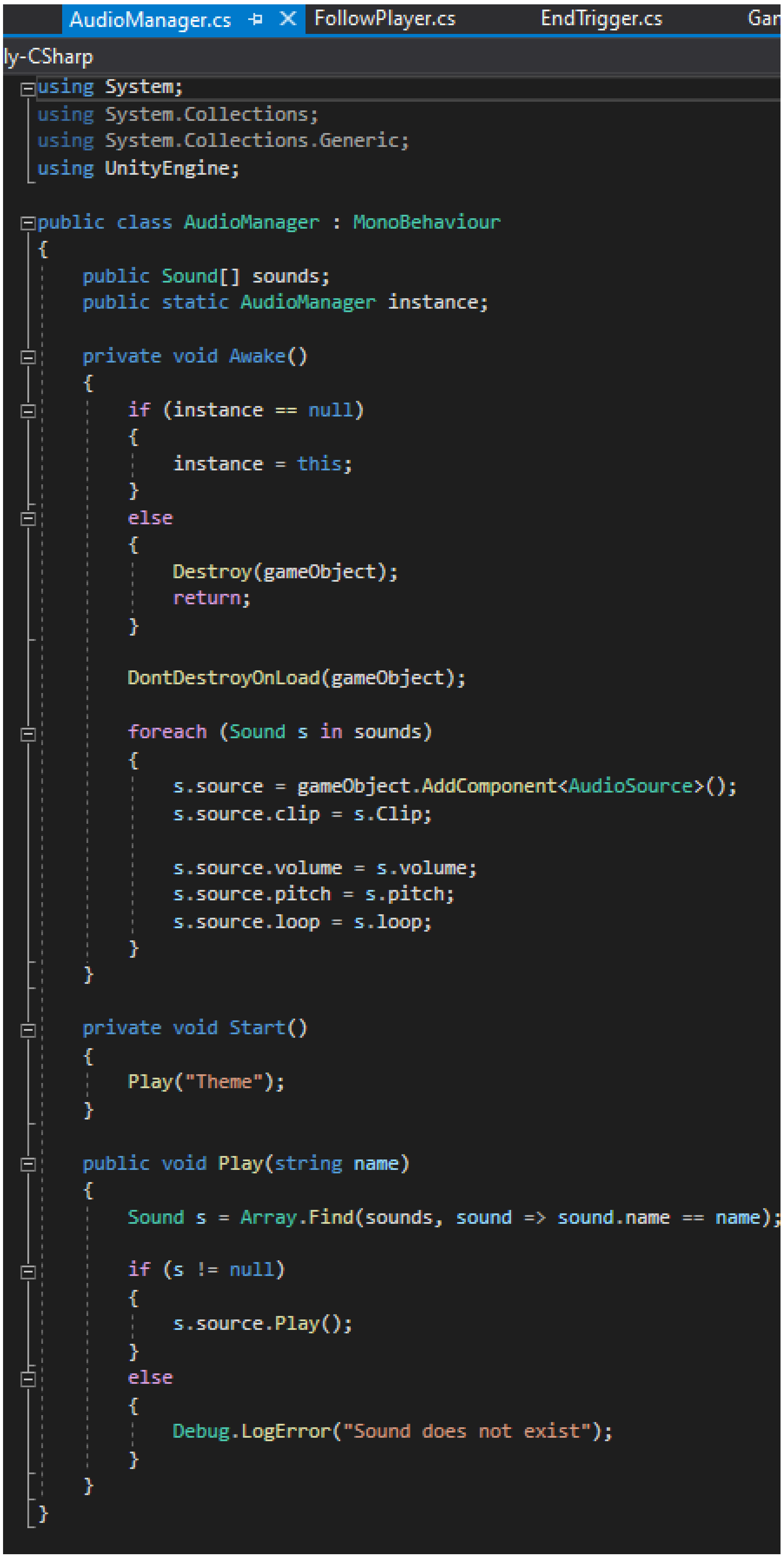
CODE

SOUND CLASS



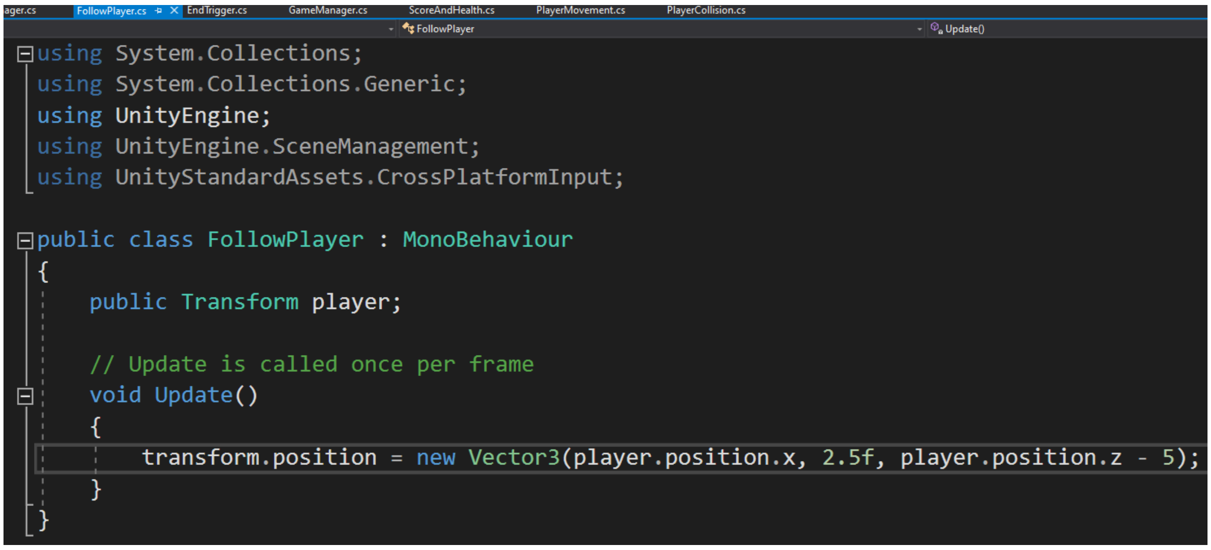
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class Sound
{
    public AudioClip Clip;
    public string name;

    [Range(0f, 1f)]
    public float volume;

    [Range(.1f, 3f)]
    public float pitch;

    public bool loop;

    [HideInInspector]
    public AudioSource source;
}
```

AUDIOMANAGER SCRIPT



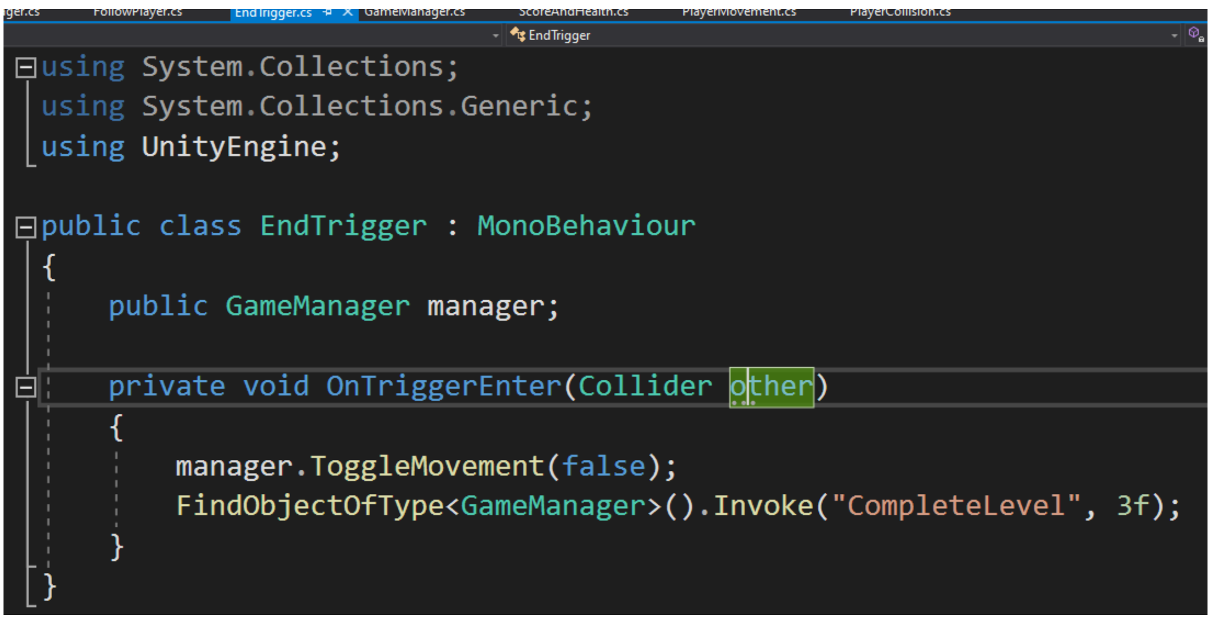
```csharp
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AudioManager : MonoBehaviour
{
    public Sound[] sounds;
    public static AudioManager instance;

    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
        }
        else
        {
            Destroy(gameObject);
            return;
        }

        DontDestroyOnLoad(gameObject);

        foreach (Sound s in sounds)
        {
            s.source = gameObject.AddComponent<AudioSource>();
            s.source.clip = s.Clip;

            s.source.volume = s.volume;
            s.source.pitch = s.pitch;
            s.source.loop = s.loop;
        }
    }

    private void Start()
    {
        Play("Theme");
    }

    public void Play(string name)
    {
        Sound s = Array.Find(sounds, sound => sound.name == name);

        if (s != null)
        {
            s.source.Play();
        }
        else
        {
            Debug.LogError("Sound does not exist");
        }
    }
}
```

FOLLOW PLAYER SCRIPT



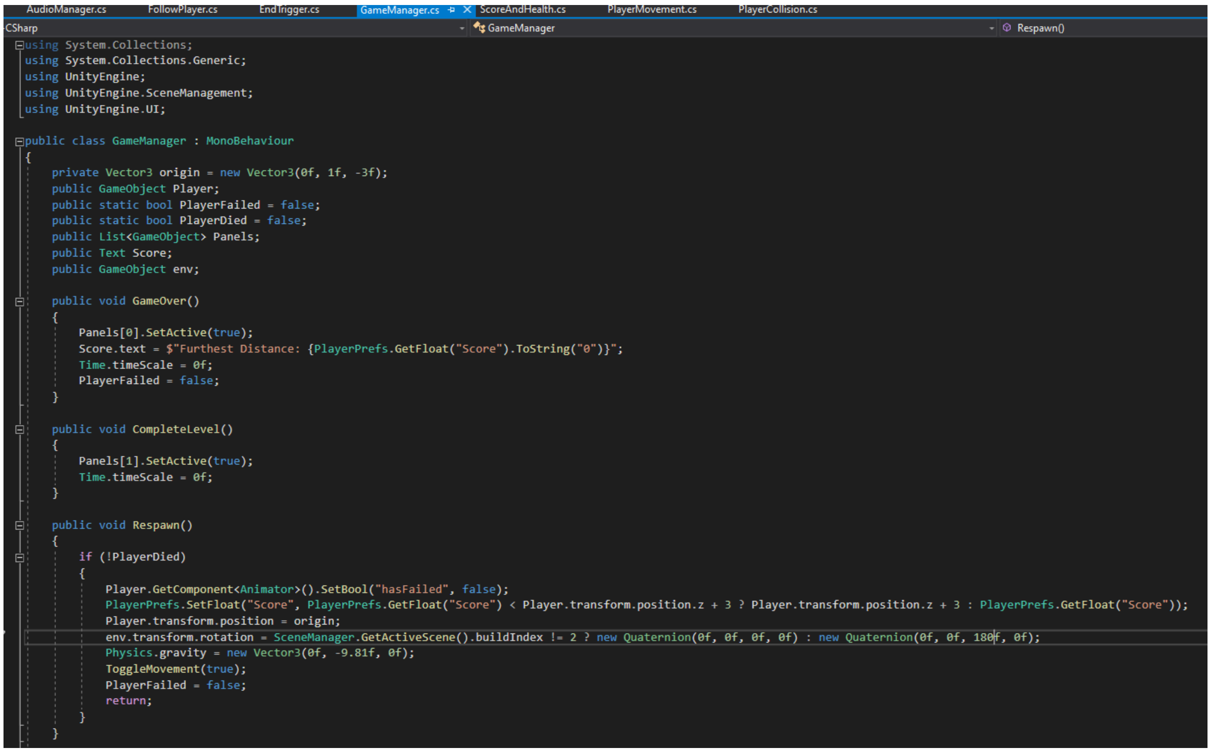
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityStandardAssets.CrossPlatformInput;

public class FollowPlayer : MonoBehaviour
{
    public Transform player;

    // Update is called once per frame
    void Update()
    {
        transform.position = new Vector3(player.position.x, 2.5f, player.position.z - 5);
    }
}
```

ENDTRIGGER SCRIPT

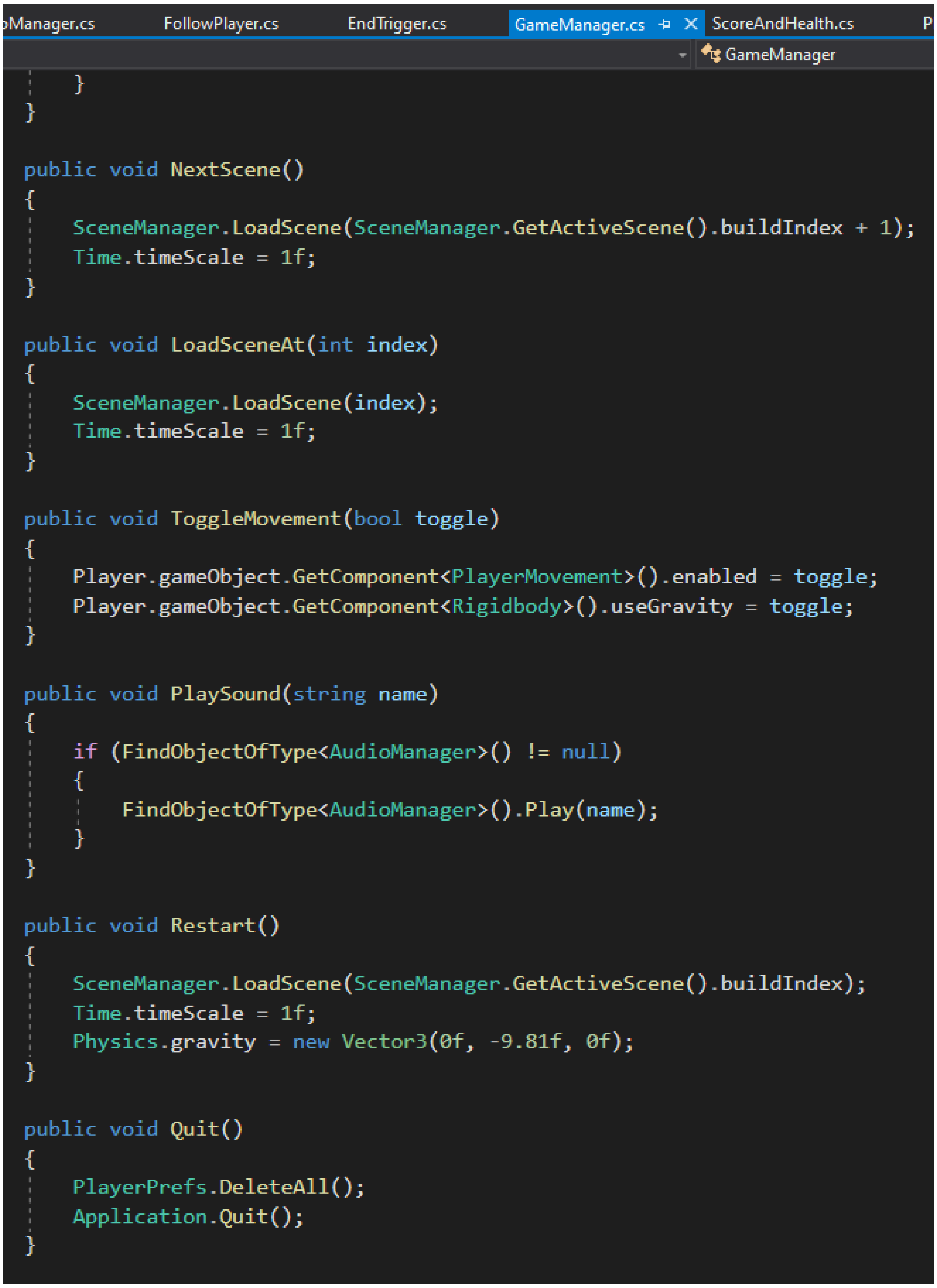


```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EndTrigger : MonoBehaviour
{
    public GameManager manager;

    private void OnTriggerEnter(Collider other)
    {
        manager.ToggleMovement(false);
        FindObjectOfType<GameManager>().Invoke("CompleteLevel", 3f);
    }
}
```

GAME MANAGER SCRIPT

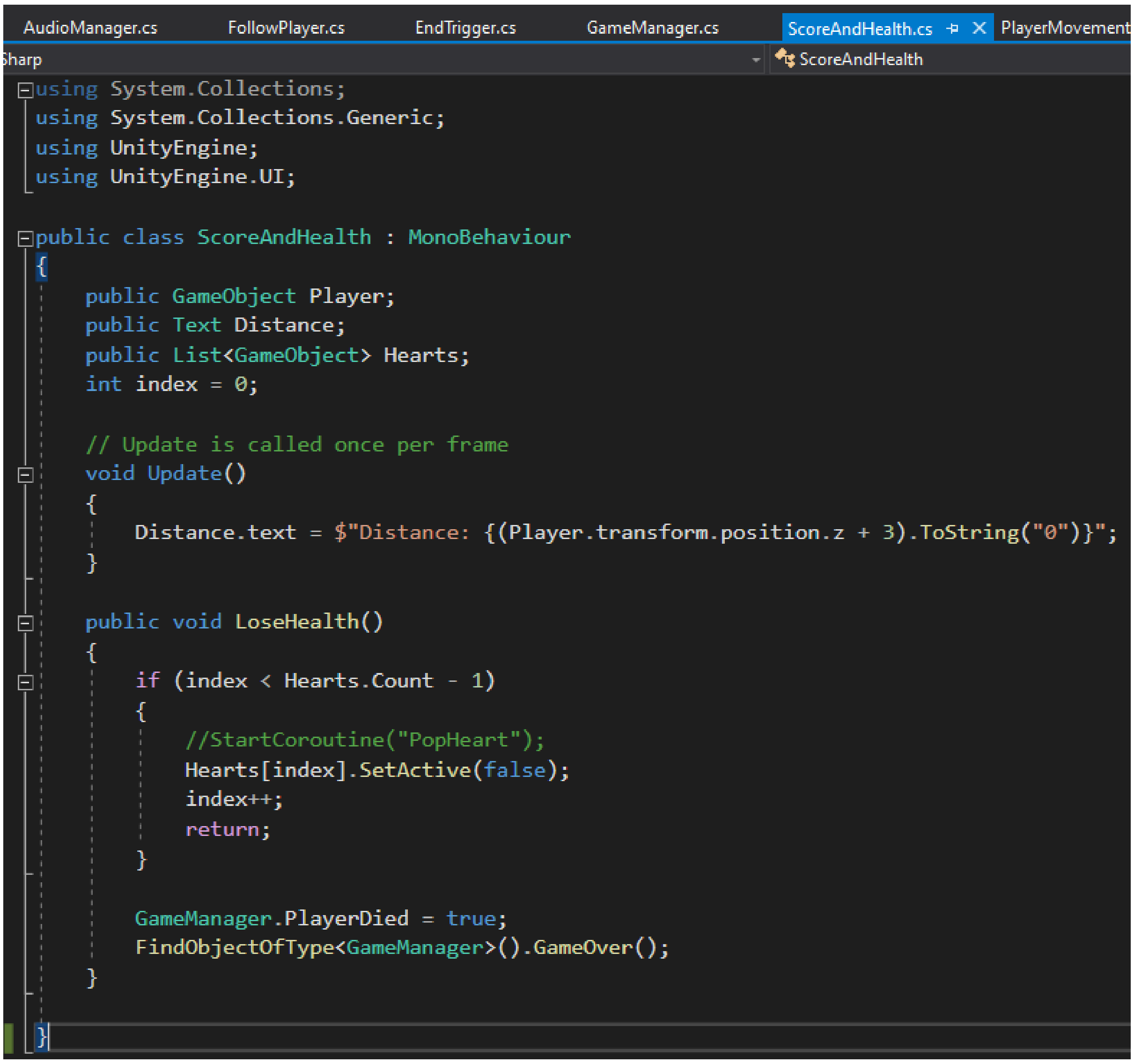


```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    private Vector3 origin = new Vector3(0f, 1f, -3f);
    public GameObject Player;
    public static bool PlayerFailed = false;
    public static bool PlayerDied = false;
    public List<GameObject> Panels;
    public Text Score;
    public GameObject env;

    public void GameOver()
    {
        Panels[0].SetActive(true);
        Score.text = $"Furthest Distance: {PlayerPrefs.GetFloat("Score").ToString("0")}";
        Time.timeScale = 0f;
        PlayerFailed = false;
    }

    public void CompleteLevel()
    {
        Panels[1].SetActive(true);
        Time.timeScale = 0f;
    }

    public void Respawn()
    {
        if (!PlayerDied)
        {
            Player.GetComponent<Animator>().SetBool("hasFailed", false);
            PlayerPrefs.SetFloat("Score", PlayerPrefs.GetFloat("Score") < Player.transform.position.z + 3 ? Player.transform.position.z + 3 : PlayerPrefs.GetFloat("Score"));
            Player.transform.position = origin;
            env.transform.rotation = SceneManager.GetActiveScene().buildIndex != 2 ? new Quaternion(0f, 0f, 0f, 0f) : new Quaternion(0f, 0f, 180f, 0f);
            Physics.gravity = new Vector3(0f, -9.81f, 0f);
            ToggleMovement(true);
            PlayerFailed = false;
            return;
        }
    }
```



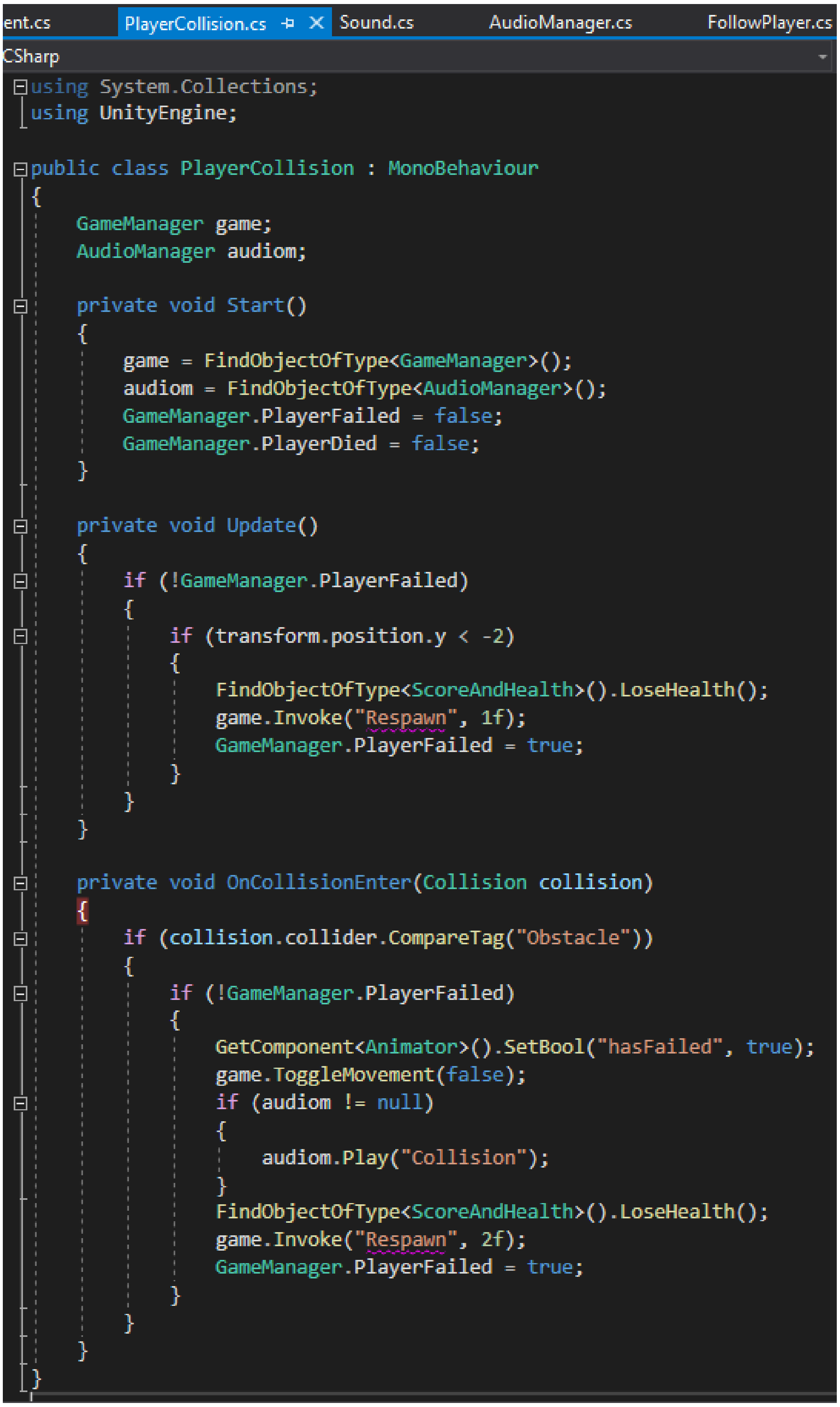
```
        }
    }

    public void NextScene()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        Time.timeScale = 1f;
    }

    public void LoadSceneAt(int index)
    {
        SceneManager.LoadScene(index);
        Time.timeScale = 1f;
    }

    public void ToggleMovement(bool toggle)
    {
        Player.gameObject.GetComponent<PlayerMovement>().enabled = toggle;
        Player.gameObject.GetComponent<Rigidbody>().useGravity = toggle;
    }

    public void PlaySound(string name)
    {
        if (FindObjectOfType<AudioManager>() != null)
        {
            FindObjectOfType<AudioManager>().Play(name);
        }
    }

    public void Restart()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
        Time.timeScale = 1f;
        Physics.gravity = new Vector3(0f, -9.81f, 0f);
    }

    public void Quit()
    {
        PlayerPrefs.DeleteAll();
        Application.Quit();
    }
```

SCORE AND HEALTH SCRIPT



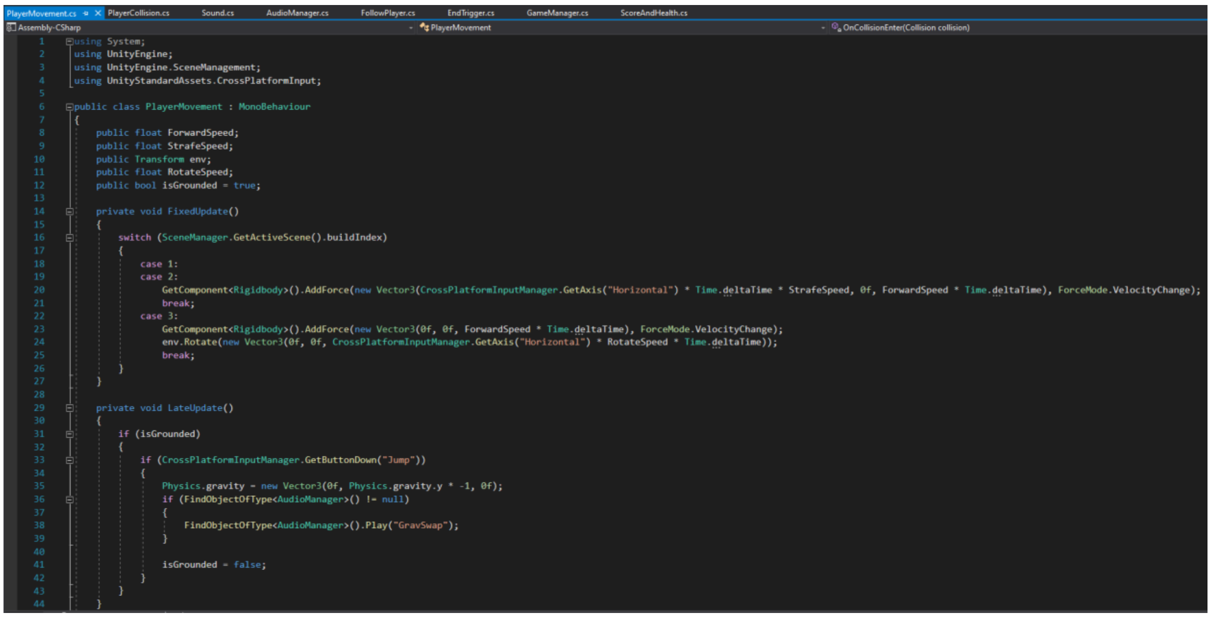
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ScoreAndHealth : MonoBehaviour
{
    public GameObject Player;
    public Text Distance;
    public List<GameObject> Hearts;
    int index = 0;

    // Update is called once per frame
    void Update()
    {
        Distance.text = $"Distance: {(Player.transform.position.z + 3).ToString("0")}";
    }

    public void LoseHealth()
    {
        if (index < Hearts.Count - 1)
        {
            //StartCoroutine("PopHeart");
            Hearts[index].SetActive(false);
            index++;
            return;
        }

        GameManager.PlayerDied = true;
        FindObjectOfType<GameManager>().GameOver();
    }

}
```

PLAYER COLLISION SCRIPT

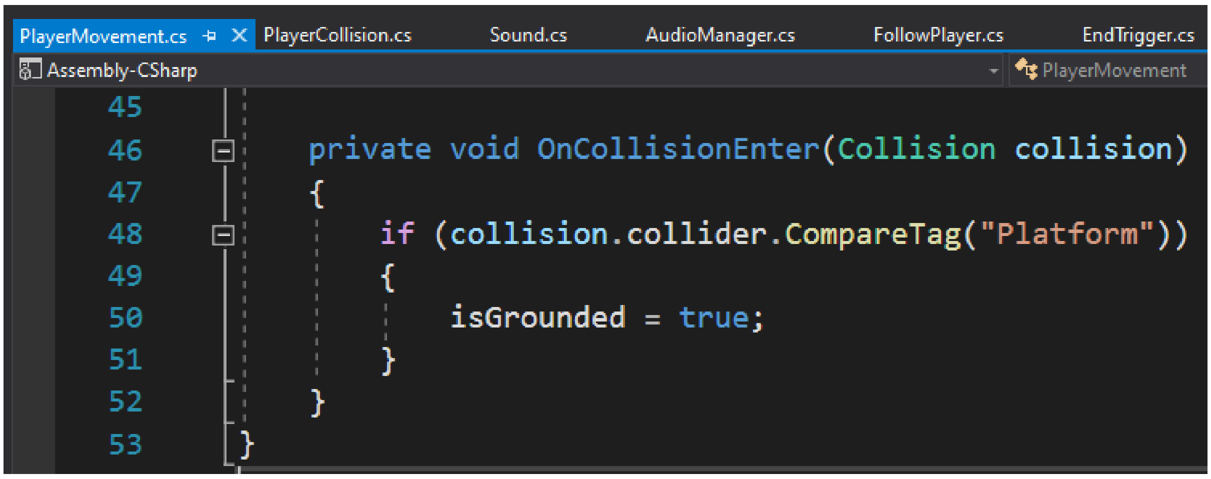


```
using System.Collections;
using UnityEngine;

public class PlayerCollision : MonoBehaviour
{
    GameManager game;
    AudioManager audiom;

    private void Start()
    {
        game = FindObjectOfType<GameManager>();
        audiom = FindObjectOfType<AudioManager>();
        GameManager.PlayerFailed = false;
        GameManager.PlayerDied = false;
    }

    private void Update()
    {
        if (!GameManager.PlayerFailed)
        {
            if (transform.position.y < -2)
            {
                FindObjectOfType<ScoreAndHealth>().LoseHealth();
                game.Invoke("Respawn", 1f);
                GameManager.PlayerFailed = true;
            }
        }
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.collider.CompareTag("Obstacle"))
        {
            if (!GameManager.PlayerFailed)
            {
                GetComponent<Animator>().SetBool("hasFailed", true);
                game.ToggleMovement(false);
                if (audiom != null)
                {
                    audiom.Play("Collision");
                }
                FindObjectOfType<ScoreAndHealth>().LoseHealth();
                game.Invoke("Respawn", 2f);
                GameManager.PlayerFailed = true;
            }
        }
    }
}
```

PLAYER MOVEMENT SCRIPT



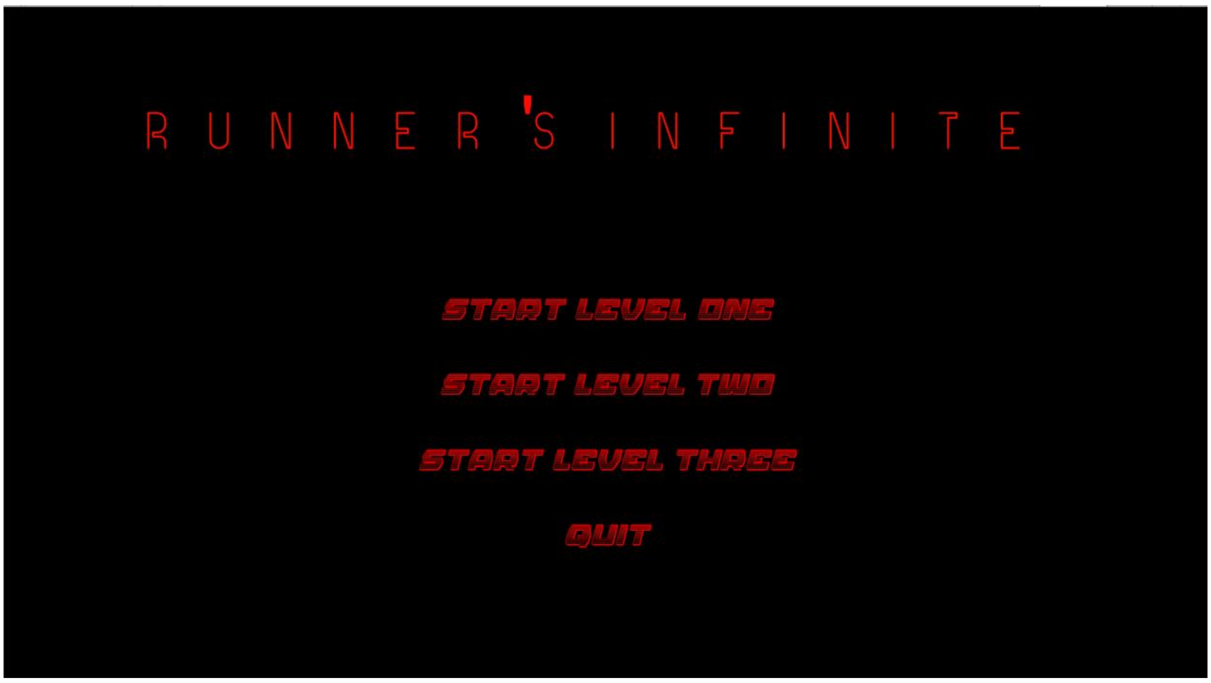
```csharp
using System;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityStandardAssets.CrossPlatformInput;

public class PlayerMovement : MonoBehaviour
{
    public float ForwardSpeed;
    public float StrafeSpeed;
    public Transform env;
    public float RotateSpeed;
    public bool isGrounded = true;

    private void FixedUpdate()
    {
        switch (SceneManager.GetActiveScene().buildIndex)
        {
            case 1:
            case 2:
                GetComponent<Rigidbody>().AddForce(new Vector3(CrossPlatformInputManager.GetAxis("Horizontal") * Time.deltaTime * StrafeSpeed, 0f, ForwardSpeed * Time.deltaTime), ForceMode.VelocityChange);
                break;
            case 3:
                GetComponent<Rigidbody>().AddForce(new Vector3(0f, 0f, ForwardSpeed * Time.deltaTime), ForceMode.VelocityChange);
                env.Rotate(new Vector3(0f, 0f, CrossPlatformInputManager.GetAxis("Horizontal") * RotateSpeed * Time.deltaTime));
                break;
        }
    }

    private void LateUpdate()
    {
        if (isGrounded)
        {
            if (CrossPlatformInputManager.GetButtonDown("Jump"))
            {
                Physics.gravity = new Vector3(0f, Physics.gravity.y * -1, 0f);
                if (FindObjectOfType<AudioManager>() != null)
                {
                    FindObjectOfType<AudioManager>().Play("GravSwap");
                }

                isGrounded = false;
            }
        }
    }
```



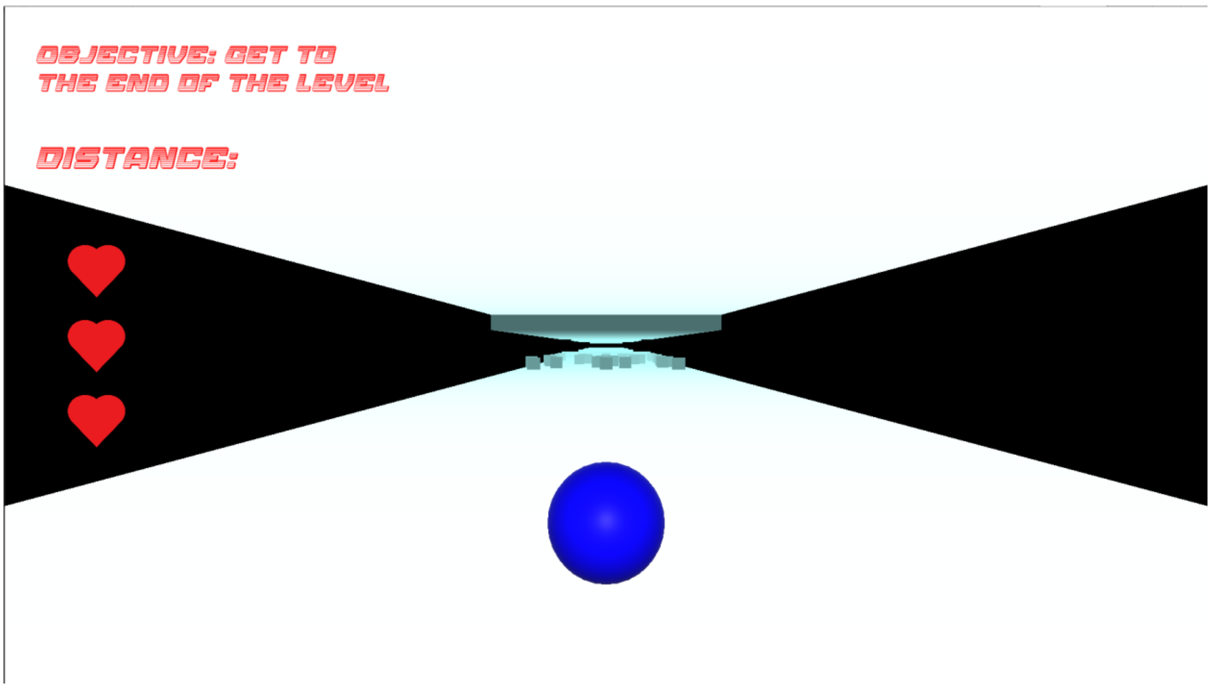
```csharp

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.collider.CompareTag("Platform"))
        {
            isGrounded = true;
        }
    }
}
```

DESIGN

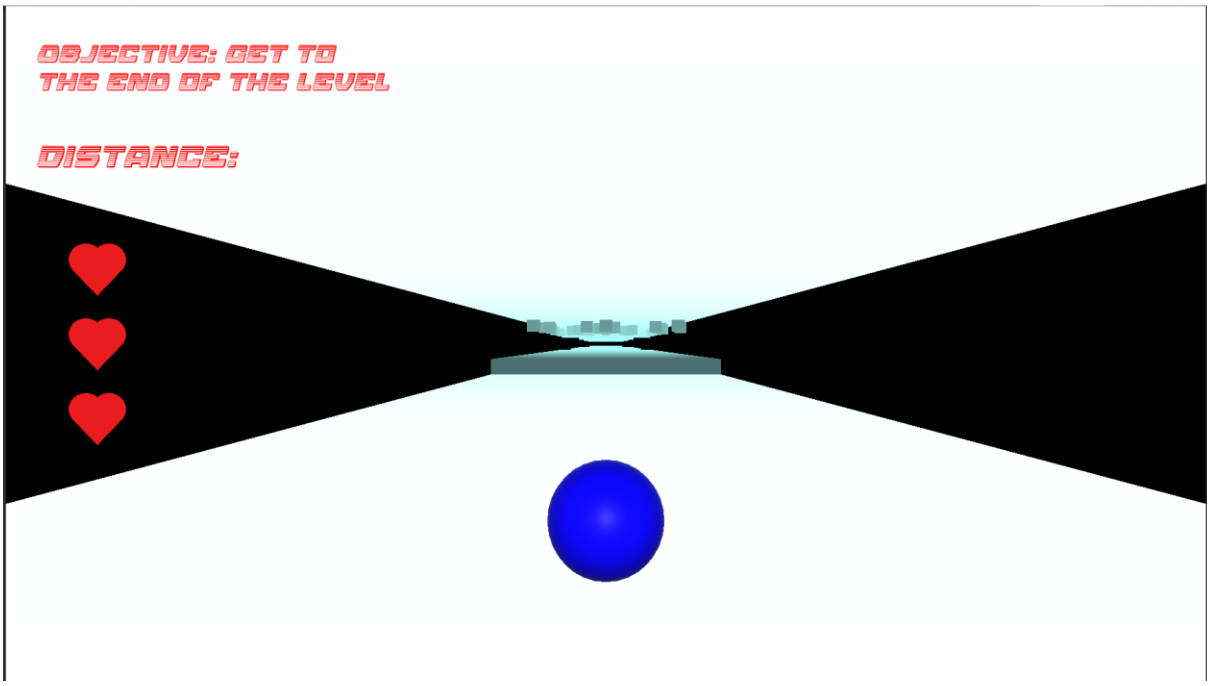
MENU



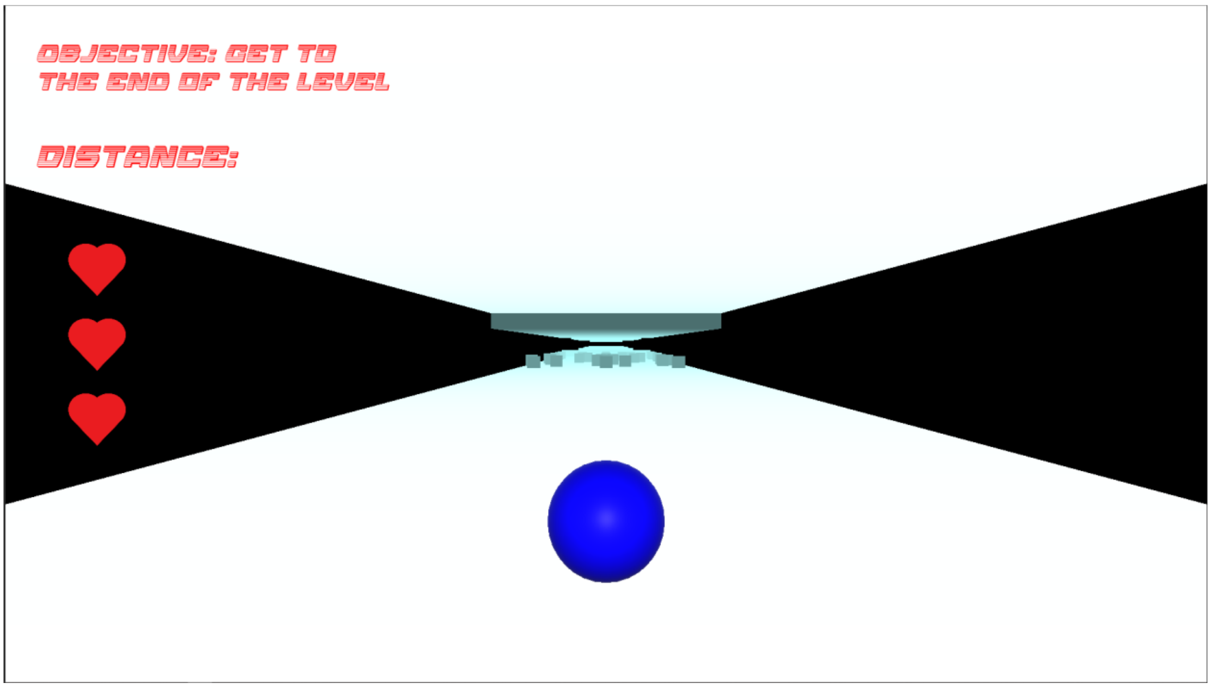
LEVEL 1



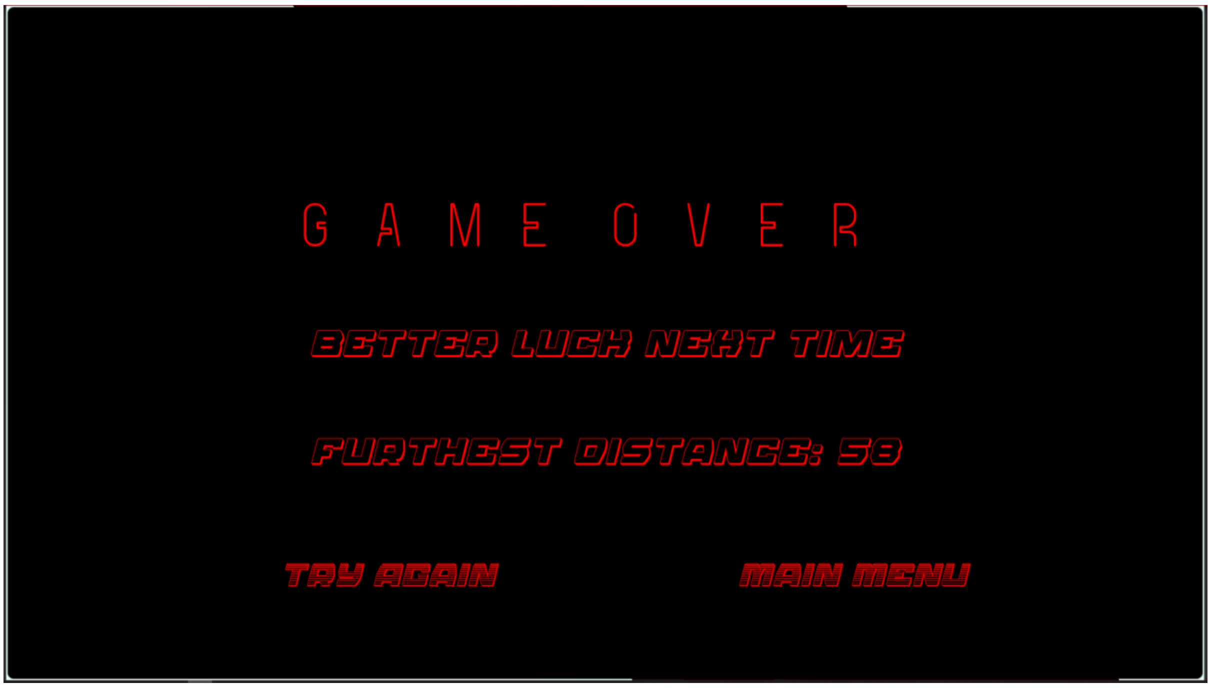
LEVEL 2



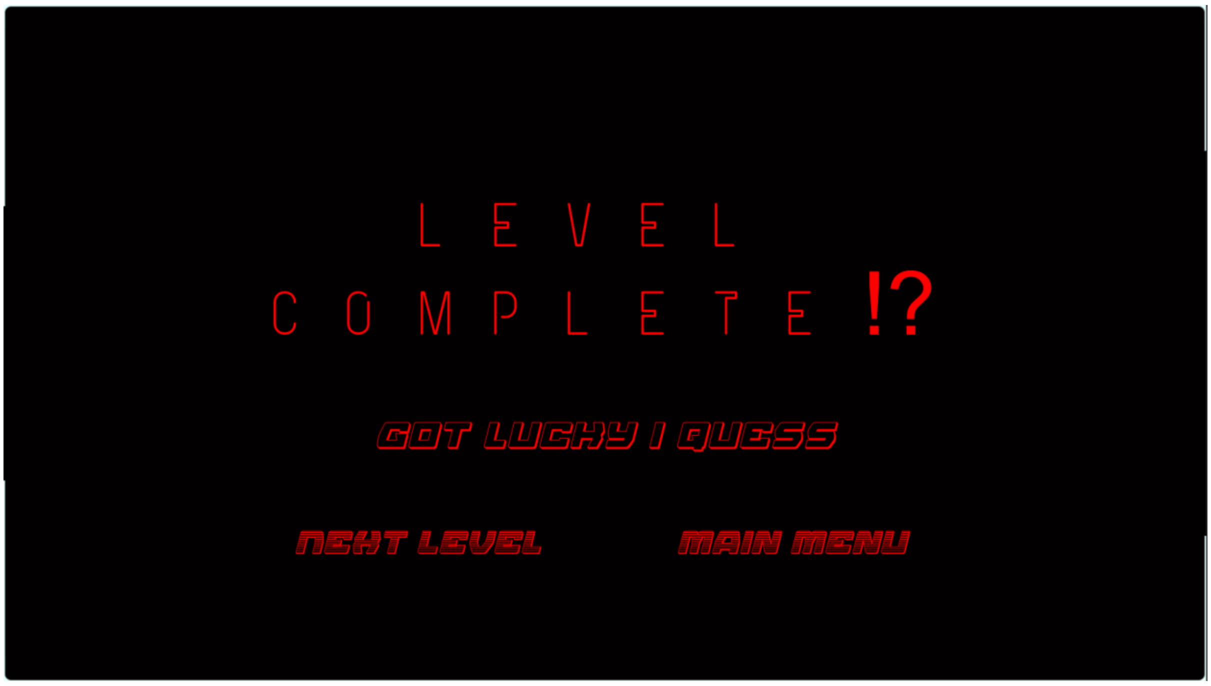
LEVEL 3



GAME OVER SCREEN WHEN LOSING ALL THREE HEARTS



LEVEL COMPLETE SCREEN WHEN PLAYER HAS REACHED THE END OF THE LEVEL



CONCLUSION

- My game has multiple levels 1, 2 and 3
- My game is a 3d game
- My game has a genre (levelled runner)
- My game has a start menu
- My game has 1 objective per level
- The game ends when the player's health reaches 0
- My game has an UI that displays the level objective
- No payed/completed assets where used in the game only unity's primitive objects where used.