



RICHARD BERTHOLD

5139 MCSD GPG512 SA Screenshots



CONTENTS

Introduction	2
Screenshots.....	2
Code.....	2
Sound class.....	2
AudioManager script	3
Follow player script	5
EndTrigger script.....	5
Game Manager script	6
Score and health script.....	8
Player collision script.....	9
Player Movement script	10
Design	11
Menu	11
level 1	11
Level 2	12
Level 3	12
Game over screen when losing all three hearts.....	13
Level Complete screen when player has reached the end of the level	13
Conclusion	14

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

```
Sound.cs X AudioManager.cs FollowPlayer.cs EndTrigger.cs G
Assembly-CSharp
1  using System.Collections;
2      using System.Collections.Generic;
3      using UnityEngine;
4
5      [System.Serializable]
6      public class Sound
7      {
8          public AudioClip Clip;
9          public string name;
10
11          [Range(0f, 1f)]
12          public float volume;
13
14          [Range(.1f, 3f)]
15          public float pitch;
16
17          public bool loop;
18
19          [HideInInspector]
20          public AudioSource source;
21      }
22
```

```
AudioManager.cs X FollowPlayer.cs EndTrigger.cs Gar
ly-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();
            s.source.clip = s.Clip;

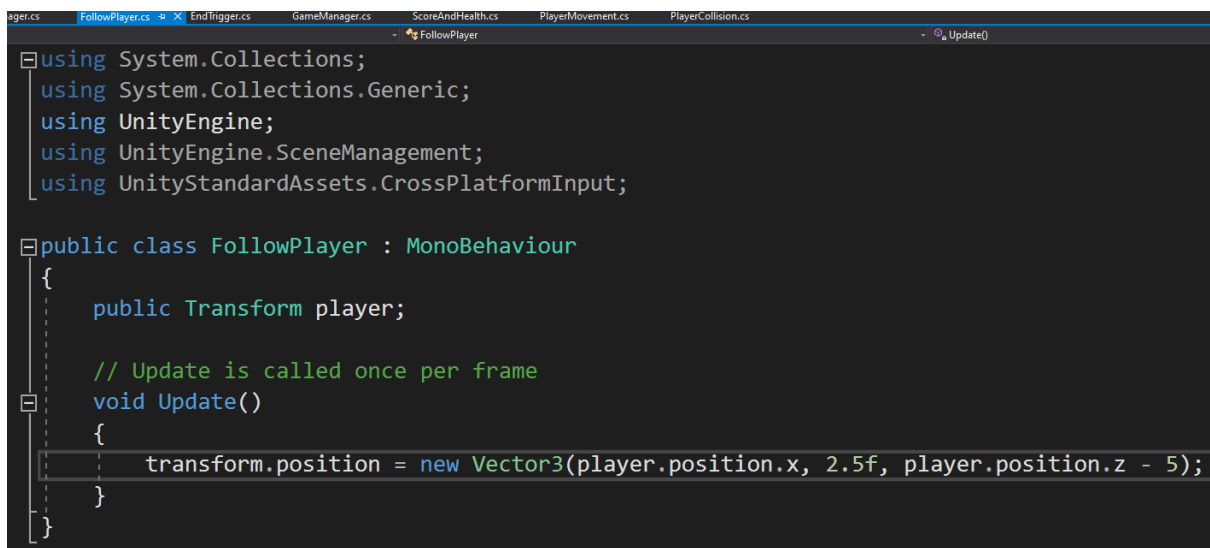
            s.source.volume = s.volume;
            s.source.pitch = s.pitch;
            s.source.loop = s.loop;
        }
    }

    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



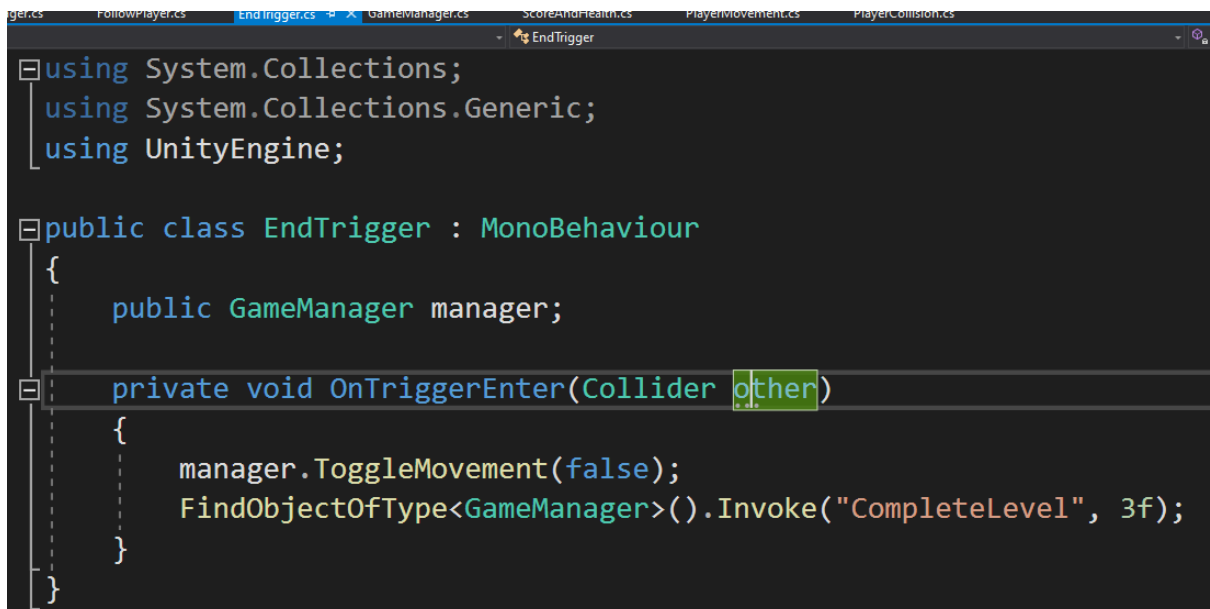
The screenshot shows the 'FollowPlayer.cs' script in a code editor. The script is a C# class that inherits from 'MonoBehaviour'. It includes several 'using' statements at the top: 'System.Collections', 'System.Collections.Generic', 'UnityEngine', 'UnityEngine.SceneManagement', and 'UnityEngine.CrossPlatformInput'. The class 'FollowPlayer' has a public 'Transform' property named 'player'. It also has an 'Update()' method that is called once per frame. Inside the 'Update()' method, the 'transform.position' is set to a new 'Vector3' with the 'player.position.x' value, a y-value of '2.5f', and the 'player.position.z' value minus 5. The code is as follows:

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



The screenshot shows the 'EndTrigger.cs' script in a code editor. The script is a C# class that inherits from 'MonoBehaviour'. It includes 'using' statements for 'System.Collections', 'System.Collections.Generic', and 'UnityEngine'. The class 'EndTrigger' has a public 'GameManager' property named 'manager'. It also has a private method 'OnTriggerEnter(Collider other)' which is called when a collider enters the trigger. Inside this method, 'manager.ToggleMovement(false)' is called, and 'FindObjectOfType<GameManager>().Invoke("CompleteLevel", 3f)' is called. The code is as follows:

```
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

```
AudioManager.cs FollowPlayer.cs EndTrigger.cs GameManager.cs ScoreAndHealth.cs PlayerMovement.cs PlayerCollision.cs
CSharp - GameManager
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;
        }
    }
}
```

```
GameManager.cs FollowPlayer.cs EndTrigger.cs GameManager.cs X ScoreAndHealth.cs P
GameManager
}
}

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

```
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

```
ent.cs  PlayerCollision.cs  Sound.cs  AudioManager.cs  FollowPlayer.cs
CSharp
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

```
PlayerMovement.cs | PlayerCollision.cs | Sound.cs | AudioManager.cs | FollowPlayer.cs | EndTrigger.cs | GameManager.cs | ScoreAndHealth.cs
Assembly-CSharp | OnCollisionEnter(Collision collision)

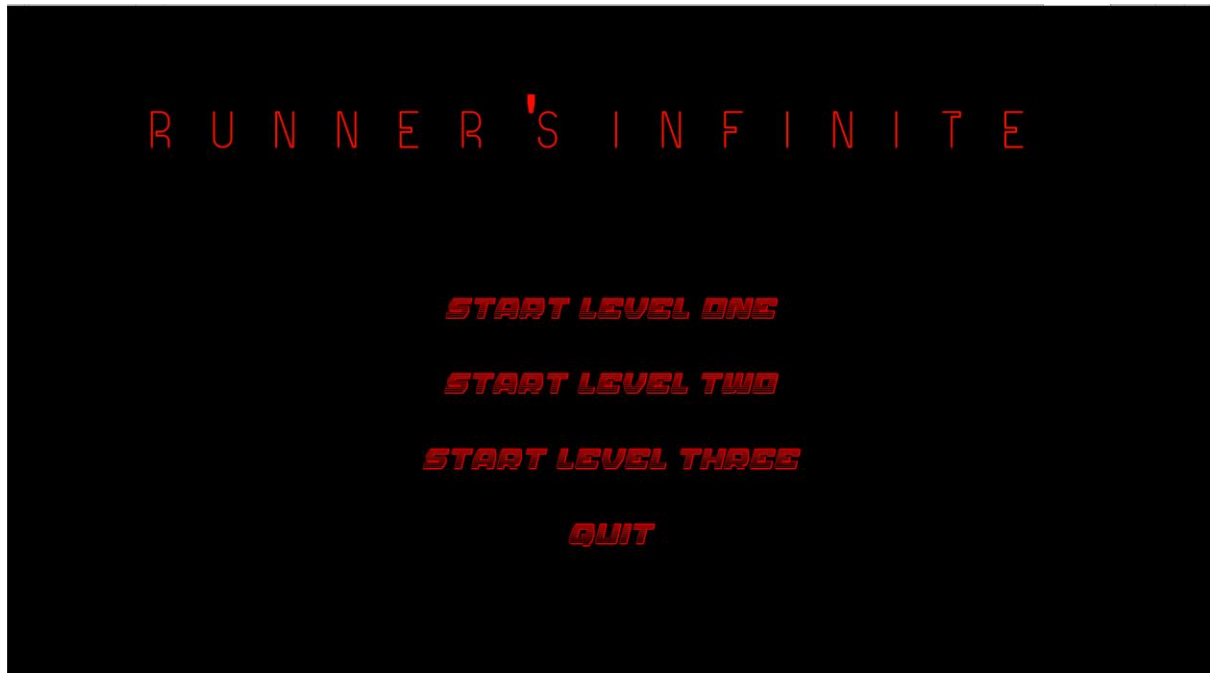
1  using System;
2  using UnityEngine;
3  using UnityEngine.SceneManagement;
4  using UnityEngine.InputSystem.CrossPlatformInput;
5
6  public class PlayerMovement : MonoBehaviour
7  {
8      public float ForwardSpeed;
9      public float StrafeSpeed;
10     public Transform env;
11     public float RotateSpeed;
12     public bool isGrounded = true;
13
14     private void FixedUpdate()
15     {
16         switch (SceneManager.GetActiveScene().buildIndex)
17         {
18             case 1:
19             case 2:
20                 GetComponent().AddForce(new Vector3(CrossPlatformInputManager.GetAxis("Horizontal") * Time.deltaTime * StrafeSpeed, 0f, ForwardSpeed * Time.deltaTime), ForceMode.VelocityChange);
21                 break;
22             case 3:
23                 GetComponent().AddForce(new Vector3(0f, 0f, ForwardSpeed * Time.deltaTime), ForceMode.VelocityChange);
24                 env.Rotate(new Vector3(0f, 0f, CrossPlatformInputManager.GetAxis("Horizontal") * RotateSpeed * Time.deltaTime));
25                 break;
26         }
27     }
28
29     private void LateUpdate()
30     {
31         if (isGrounded)
32         {
33             if (CrossPlatformInputManager.GetButtonDown("Jump"))
34             {
35                 Physics.gravity = new Vector3(0f, Physics.gravity.y * -1, 0f);
36                 if (FindObjectOfType<AudioManager>() != null)
37                 {
38                     FindObjectOfType<AudioManager>().Play("GravSwap");
39                 }
40             }
41             isGrounded = false;
42         }
43     }
44 }
```

```
PlayerMovement.cs | PlayerCollision.cs | Sound.cs | AudioManager.cs | FollowPlayer.cs | EndTrigger.cs
Assembly-CSharp | PlayerMovement

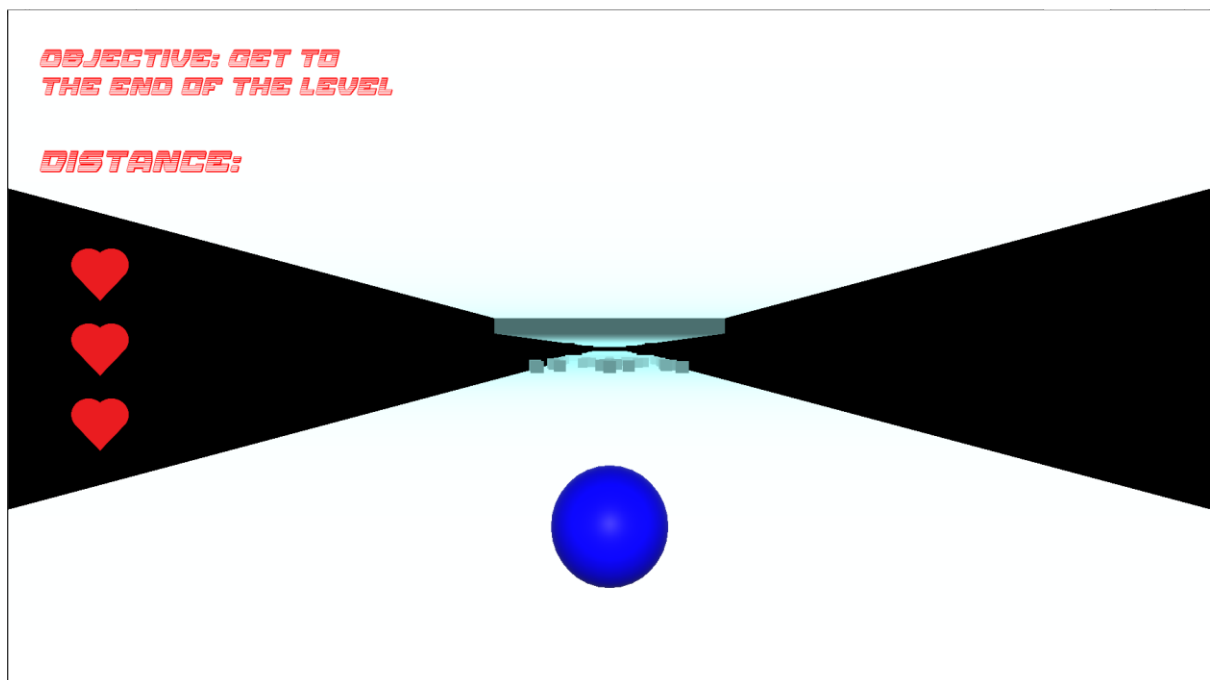
45
46     private void OnCollisionEnter(Collision collision)
47     {
48         if (collision.collider.CompareTag("Platform"))
49         {
50             isGrounded = true;
51         }
52     }
53 }
```

DESIGN

MENU



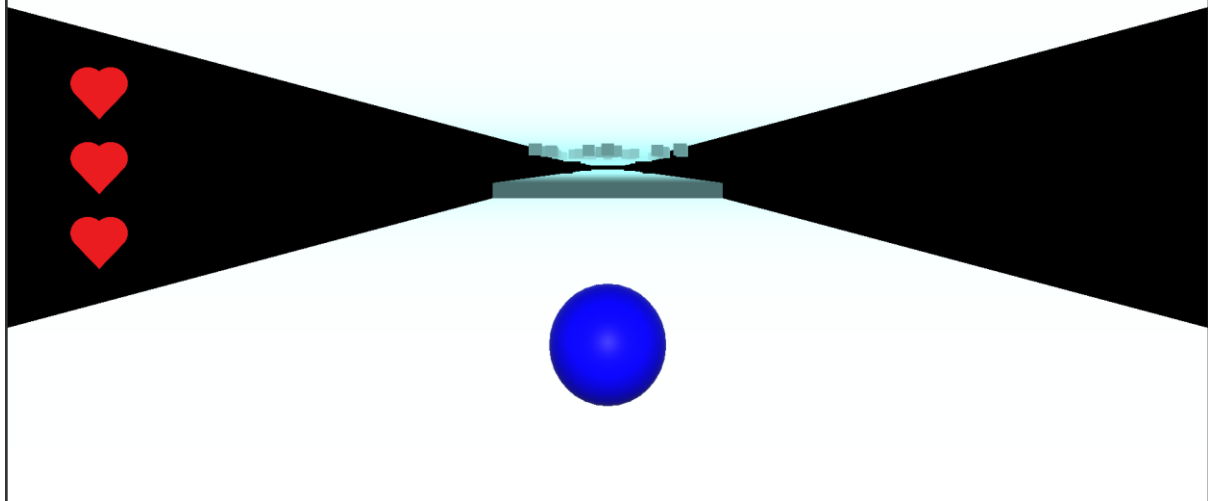
LEVEL 1



LEVEL 2

**OBJECTIVE: GET TO
THE END OF THE LEVEL**

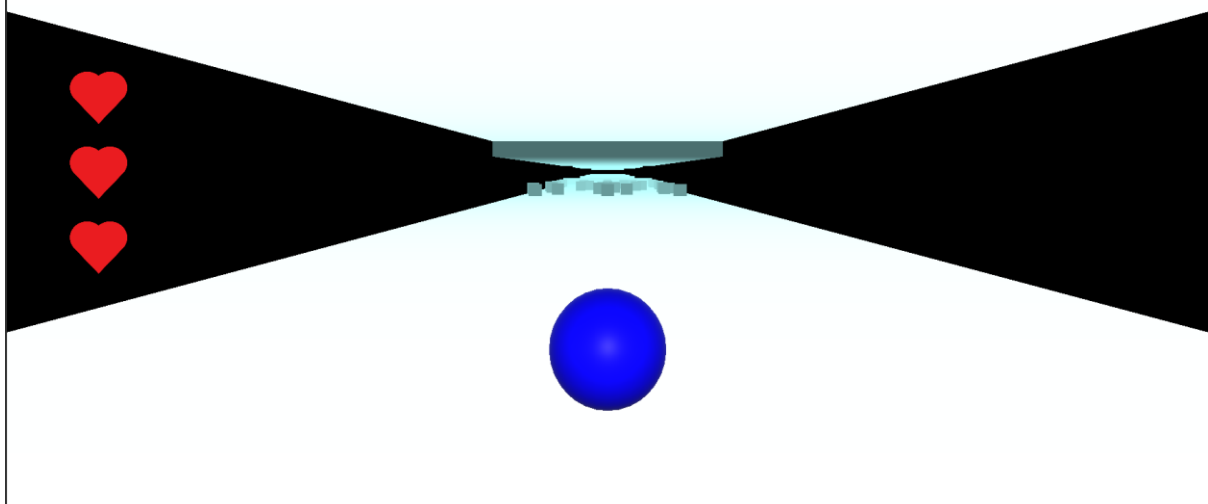
DISTANCE:



LEVEL 3

**OBJECTIVE: GET TO
THE END OF THE LEVEL**

DISTANCE:



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.