Goldsmiths, University of London:

Midterm Submission

Code PDF

Important Note: All code is mine, therefore I have not included comment lines to indicate references to the work of others, as this is all my original work.

Peer-graded Assignment: 3.502 An Interactive Physics Scene

1. **BallController.cs**

```
using System;

using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;

public class BallController : MonoBehaviour
{
    bool hasTriggeredBuffer = false;
    Queue<Transform> BallPositions;
    Rigidbody rb;
    Vector3 direction;
    float multPos = 2f;
    float dirNormal = 1f;
    float powerNormal = 1f;
    float power = 0f;
    float powerMax = 255f;
    float powerMult = 250f;
    [SerializeField] GameObject lineObj;
    void Start()
    {
        BallPositions = new Queue<Transform>();
        foreach (var item in
GameManager.instance.BallPositions.GetComponentsInChildren<Transform>())
        {
            BallPositions.Enqueue(item);
        }
        //Dequeues parent transform
        BallPositions.Dequeue();

        rb = GetComponent<Rigidbody>();
        direction = new Vector3(0, 0, -14f);
        lineObj.GetComponent<LineRenderer>().enabled = false;
        lineObj.GetComponent<LineRenderer>().useWorldSpace = false;
    }

    // Update is called once per frame
    void Update()
    {
        transform.position = rb.position;
        transform.rotation = rb.rotation;
        switch (GameManager.instance.GetState())
        {
            case GameManager.gameStates.SelectPos:
                MovePosHorizontal();
                break;
            case GameManager.gameStates.SelectDir:
                SelectDirection();
                break;
            case GameManager.gameStates.SelectPower:
                SelectPower();
                break;
        }
    }

    void MovePosHorizontal()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            rb.velocity = Vector3.zero;
            rb.angularVelocity = Vector3.zero;
            rb.freezeRotation = true;

            GameManager.instance.SwitchState(GameManager.gameStates.SelectDir);
            lineObj.GetComponent<LineRenderer>().enabled = true;
        }

        float ReqZ = BallPositions.Peek().position.z;
        if ((ReqZ < 0 && transform.position.z <= ReqZ) || (ReqZ > 0 &&
transform.position.z >= ReqZ))
        {
            BallPositions.Enqueue(BallPositions.Dequeue());
            ReqZ = BallPositions.Peek().position.z;
            rb.velocity = new Vector3(0, 0, 0);
            rb.AddForce(Vector3.Normalize(new Vector3(0, 0, ReqZ)) * multPos *
multPos * rb.mass, ForceMode.Impulse);
        }
        else
        {
            rb.AddForce(Vector3.Normalize(new Vector3(0, 0, ReqZ)) * multPos *
multPos * rb.mass, ForceMode.Force);
        }
    }

    void SelectDirection()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            GameManager.instance.SwitchState(GameManager.gameStates.SelectPower);
        }
        if (direction.y >= 90f)
        {
            dirNormal = -1f;
        }
        if (direction.y <= -90f)
        {
            dirNormal = 1f;
        }
        direction = new Vector3(90f, direction.y + dirNormal * Time.deltaTime *
35f, 0);

        rb.rotation = Quaternion.Euler(direction);
    }

    void SelectPower()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            GameManager.instance.SwitchState(GameManager.gameStates.Shooting);
            Shoot();
        }
        if (power <= 0)
        {
            power = 0f;
            powerNormal = 1f;
        }
        if (power >= powerMax)
        {
            power = powerMax;
            powerNormal = -1f;
        }
        power += 1f * powerNormal;
        GameManager.instance.GetComponentInChildren<HUDManager>().SetPowerSlider(
power, powerMax, powerMult);
    }

    void Shoot()
    {
        rb.freezeRotation = false;
        lineObj.GetComponent<LineRenderer>().enabled = false;
        Vector3 shootDir = new Vector3(-1f * rb.rotation.x *
rb.transform.right.x, rb.rotation.y * rb.transform.forward.y, -2f * rb.rotation.z
* rb.transform.up.z);
        rb.AddForce(shootDir * power * powerMult, ForceMode.Force);
    }

    private void OnCollisionEnter(Collision collision)
    {
        if(GameManager.instance.GetState() == GameManager.gameStates.Shooting &&
!hasTriggeredBuffer)
        {
            if (collision.gameObject.CompareTag("Gutter"))
            {
                GameManager.instance.SwitchState(GameManager.gameStates.Gutter);
            }
            if (collision.gameObject.CompareTag("Pin"))
            {
                RegisterHit();
            }
        }

    }
    private void OnTriggerEnter(Collider other)
    {
        if (GameManager.instance.GetState() == GameManager.gameStates.Shooting &&
!hasTriggeredBuffer)
        {
            switch (other.tag)
            {
                case "Miss":
                    GameManager.instance.SwitchState(GameManager.gameStates.Miss)
;
                    break;
                case "BallBuffer":
                    RegisterHit();
                    break;
            }
        }

    }

    void RegisterHit()
    {
        hasTriggeredBuffer = true;
        GameManager.instance.GetComponentInChildren<PinManager>().CountDown();
    }
}
```

2. **GameManager.cs**

```
using System;

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

public class GameManager : MonoBehaviour
{
    public static GameManager instance;
    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
            DontDestroyOnLoad(this);
        }
        else if (instance != this)
        {
            Destroy(gameObject);
        }
    }
    public GameObject BallPositions;
    [SerializeField] PinManager pinManager;
    [NonSerialized] public bool firstTurn = false;
    bool hasResed = false;
    public enum gameStates
    {
        Menu,
        SelectWeight,
        SelectPos,
        SelectDir,
        SelectPower,
        Shooting,
        Gutter,
        Miss,
        Hit,
        Spare,
        Strike,
        AITurn,
        GameOver
    }

    [SerializeField] List<GameObject> balls;
    Vector3 ballSpawnPos;

    GameObject selectedBall;
    gameStates state = gameStates.Menu;
    void Start()
    {
        Init();
    }

    void Update()
    {
        if (!hasResed)
        {
            switch (state)
            {
                case gameStates.Gutter:
                case gameStates.Miss:
                case gameStates.Hit:
                    hasResed = true;
                    if (firstTurn)
                    {
                        SoftResScene();
                    }
                    else
                    {
                        HardResScene();
                    }
                    break;
                case gameStates.Strike:
                case gameStates.Spare:
                    hasResed = true;
                    HardResScene();
                    break;
            }
        }
    }

    public void SwitchState(gameStates value)
    {
        state = value;
    }

    public void GetSelectedBall(int value)
    {
        switch (value)
        {
            case 8:
                selectedBall = balls[0];
                break;
            case 9:
                selectedBall = balls[1];
                break;
            case 10:
                selectedBall = balls[2];
                break;
            case 11:
                selectedBall = balls[3];
                break;
            case 12:
                selectedBall = balls[4];
                break;
            default:
                Debug.LogError($"Ball of value {value} not found!");
                break;
        }
        InstantiateBall();
    }

    void InstantiateBall()
    {
        selectedBall = Instantiate(selectedBall, ballSpawnPos,
Quaternion.identity);
        SwitchState(gameStates.SelectPos);
    }

    public gameStates GetState()
    {
        return state;
    }
    public void InstantiatePins()
    {
        pinManager.InstantiatePins();
    }
    public void SoftResScene()
    {
        StartCoroutine("SoftResIENUM");

    }

    public void HardResScene()
    {
        StartCoroutine("HardResIENUM");
    }

    IEnumerator SoftResIENUM()
    {
        yield return new WaitForSeconds(1.5f);
        GetComponentInChildren<HUDManager>().Init();

        SwitchState(gameStates.SelectWeight);
        GetComponentInChildren<UIManager>().Init();
        Init();

        pinManager.Init();

        SceneManager.LoadScene("SampleScene");
    }
    IEnumerator HardResIENUM()
    {
        yield return new WaitForSeconds(1.5f);
        GetComponentInChildren<HUDManager>().Init();

        SwitchState(gameStates.Menu);
        GetComponentInChildren<UIManager>().Init();
        Init();

        pinManager.HardResetAndInit();

        SceneManager.LoadScene("SampleScene");
    }

    void Init()
    {
        firstTurn = !firstTurn;
        hasResed = false;
        ballSpawnPos = new Vector3(-40, 4, 0);
        selectedBall = null;
    }
}
```

3. **HUDManager.cs**

```
using System.Collections;

using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class HUDManager : MonoBehaviour
{
    [SerializeField] GameObject botPanel;
    [SerializeField] Slider powerSlider;
    [SerializeField] Image sliderBG;
    [SerializeField] TextMeshProUGUI anouncements;
    void Start()
    {
        Init();
    }

    void Update()
    {
        if (GameManager.instance.GetComponentInChildren<UIManager>().HUDEnabled)
        {
            switch (GameManager.instance.GetState())
            {
                case GameManager.gameStates.SelectPos:
                    botPanel.GetComponentInChildren<TextMeshProUGUI>().text =
"Set Position: Spacebar";
                    break;

                case GameManager.gameStates.SelectDir:
                    botPanel.GetComponentInChildren<TextMeshProUGUI>().text =
"Set Direction: Spacebar";
                    break;

                case GameManager.gameStates.SelectPower:
                    botPanel.GetComponentInChildren<TextMeshProUGUI>().text =
"Set Power: Spacebar";

                    if (!powerSlider.gameObject.activeSelf)
                    {
                        powerSlider.gameObject.SetActive(true);
                    }
                    break;
                case GameManager.gameStates.Shooting:
                    if (powerSlider.gameObject.activeSelf)
                    {
                        powerSlider.gameObject.SetActive(false);
                    }
                    botPanel.GetComponentInChildren<TextMeshProUGUI>().text =
"Forfeit Shot: R";
                    if (Input.GetKeyDown(KeyCode.R))
                    {
                        GameManager.instance.SwitchState(GameManager.gameStates.M
iss);
                    }
                    break;
                case GameManager.gameStates.Gutter:
                case GameManager.gameStates.Miss:
                case GameManager.gameStates.Hit:
                case GameManager.gameStates.Strike:
                case GameManager.gameStates.Spare:
                    botPanel.GetComponent<Image>().CrossFadeAlpha(0f, .5f, true);
                    botPanel.GetComponentInChildren<TextMeshProUGUI>().CrossFadeA
lpha(0f, .5f, true);
                    anouncements.text =
GameManager.instance.GetState().ToString() + "!";
                    anouncements.CrossFadeAlpha(0f, .5f, true);
                    break;
            }
        }
    }

    public void SetPowerSlider(float value, float max, float mult)
    {
        Color color = new Color(value / 255, (255 - value) / 255, 0);
        powerSlider.maxValue = max;
        sliderBG.color = color;
        powerSlider.value = value;

        int textNum = Mathf.RoundToInt(value * mult);
        powerSlider.GetComponentInChildren<TextMeshProUGUI>().text =
$"{textNum}";
    }
    public void Init()
    {
        anouncements.text = "";
        powerSlider.gameObject.SetActive(false);
    }
}
```

4. **Pin.cs**

```
using System;

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

public class Pin : MonoBehaviour
{
    [NonSerialized] public bool Standing;
    Rigidbody rb;
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        Standing = true;
    }

    // Update is called once per frame
    void Update()
    {
        CheckIfStanding();
    }

    private void CheckIfStanding()
    {
        if (Standing)
        {
            if (rb.rotation.x.ToString("0.0") != "-0.7")
            {
                Standing = false;
            }
        }

    }
}
```

5. **PinManager.cs**

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

public class PinManager : MonoBehaviour
{
    List<Vector3> PinSpawnPositions;
    List<GameObject> pins;
    List<bool> pushedPins;
    bool canSpare;
    [SerializeField] GameObject pin;
    void Start()
    {
        PinSpawnPositions = new List<Vector3>();
        float posX = -425f;
        //first row
        PinSpawnPositions.Add(new Vector3(posX, 5, 0));
        //second row
        PinSpawnPositions.Add(new Vector3(posX-3, 5, -2));
        PinSpawnPositions.Add(new Vector3(posX-3, 5, 2));
        //third row
        PinSpawnPositions.Add(new Vector3(posX-6, 5, 0));
        PinSpawnPositions.Add(new Vector3(posX-6, 5, -4));
        PinSpawnPositions.Add(new Vector3(posX-6, 5, 4));
        //fourth row
        PinSpawnPositions.Add(new Vector3(posX-9, 5, -2));
        PinSpawnPositions.Add(new Vector3(posX-9, 5, 2));
        PinSpawnPositions.Add(new Vector3(posX - 9, 5, 6));
        PinSpawnPositions.Add(new Vector3(posX - 9, 5, -6));


        Init();
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void InstantiatePins()
    {
        if (pushedPins.Any())
        {
            SpawnStandingPins();
        }
        else
        {
            foreach (Vector3 pos in PinSpawnPositions)
            {
                pins.Add(Instantiate(pin, pos, pin.transform.rotation));
            }
        }
    }

    public void CountDown()
    {
        //-0.7071068 is uprigt X rotation
        StartCoroutine("WaitForPins");
    }

    IEnumerator WaitForPins()
    {
        int prevPushed = pushedPins.Where(n => n==true).Count();
        yield return new WaitForSeconds(3f);

        CheckPushedPins();

        canSpare = prevPushed != 0;
        if (pushedPins.Where(n => n==true).Count() == 10)
        {
            if (GameManager.instance.firstTurn)
            {
                GameManager.instance.SwitchState(GameManager.gameStates.Strike);
            }
            else
            {
                if (canSpare)
                {
                    GameManager.instance.SwitchState(GameManager.gameStates.Spare
);
                }
                else
                {
                    GameManager.instance.SwitchState(GameManager.gameStates.Strik
e);
                }
            }
        }
        else if((GameManager.instance.firstTurn && pushedPins.Where(n => n ==
true).Any()) ||
            (!GameManager.instance.firstTurn && pushedPins.Where(n => n ==
true).Count()>prevPushed))
        {
            GameManager.instance.SwitchState(GameManager.gameStates.Hit);
        }
        else
        {
            GameManager.instance.SwitchState(GameManager.gameStates.Miss);
        }
    }

    void CheckPushedPins()
    {
        if (pushedPins.Any())
        {
            Debug.Log($"pushed before:{pushedPins.Where(n => n ==
true).Count()}");
            for (int i = 0; i < pins.Count; i++)
            {
                if (pins[i] != null)
                {
                    if (!pins[i].GetComponent<Pin>().Standing)
                    {
                        pushedPins[i] = true;
                    }
                }
            }
            Debug.Log($"pushed:{pushedPins.Where(n => n==true).Count()}");
        }
        else
        {
            for (int i = 0; i < pins.Count; i++)
            {
                if (!pins[i].GetComponent<Pin>().Standing)
                {
                    pushedPins.Add(true);
                }
                else
                {
                    pushedPins.Add(false);
                }
            }
        }
    }
    public void Init()
    {
        if (pushedPins == null)
        {
            pushedPins = new List<bool>();
        }
        pins = new List<GameObject>();
    }

    public void HardResetAndInit()
    {
        pins = null;
        pushedPins = null;
        Init();
    }

    void SpawnStandingPins()
    {
        for (int i = 0; i < PinSpawnPositions.Count; i++)
        {
            if (!pushedPins[i])
            {
                pins.Add(Instantiate(pin, PinSpawnPositions[i],
pin.transform.rotation));
            }
            else
            {
                pins.Add(null);
            }
        }
    }
}
```

6. UIManager.cs

```
using System;

using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.UI;

public class UIManager : MonoBehaviour
{
    public List<GameObject> Menus;
    TextMeshProUGUI sliderValue;
    [NonSerialized] public bool HUDEnabled;
    void Start()
    {
        sliderValue =
Menus[1].GetComponentInChildren<Slider>().gameObject.GetComponentInChildren<TextM
eshProUGUI>();
        Init();

    }

    // Update is called once per frame
    void Update()
    {

    }

    void ActivateMenuDeactivateAllOthers(int menuIndex)
    {
        Menus.ForEach(menu => menu.SetActive(false));
        Menus[menuIndex].SetActive(true);
    }

    public void SwitchToMain()
    {
        ActivateMenuDeactivateAllOthers(0);
    }

    public void SwitchToWeightSelect()
    {
        ActivateMenuDeactivateAllOthers(1);
        GameManager.instance.SwitchState(GameManager.gameStates.SelectWeight);
    }

    public void SwitchToHUD()
    {
        HUDEnabled = true;
        ActivateMenuDeactivateAllOthers(2);
    }

    public void ChangeWeight(float value)
    {
        sliderValue.text = $"{value} Kg";
    }
    public void PassSelectedBalltoGM()
    {
        int selected = int.Parse(sliderValue.text.Split(' ')[0]);
        GameManager.instance.GetSelectedBall(selected);
    }

    public void Init()
    {
        HUDEnabled = false;
        switch (GameManager.instance.GetState())
        {
            case GameManager.gameStates.Menu:
                SwitchToMain();
                break;
            case GameManager.gameStates.SelectWeight:
                SwitchToWeightSelect();
                break;
            case GameManager.gameStates.SelectPos:
                break;
            default:
                break;
        }
    }
}
```

Peer-graded Assignment: 4.502 Keyframe Animation

I. Enemies

1. Enemy.cs

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class Enemy : MonoBehaviour
{
    public GameObject player;
    [SerializeField] float speed;
    [SerializeField] float moveDelay;
    [SerializeField] GameObject parent;
    [SerializeField] GameObject particles;
    public Rigidbody rb;
    public int tier;
    public UnityEvent OnDeath;
    public bool canDie { get; private set; }
    private void Awake()
    {
        canDie = false;
        StartCoroutine(DisableSpawnInvincibility());
    }
    void Start()
    {
        OnDeath.AddListener(delegate {
GameObject.FindGameObjectWithTag("Respawn").GetComponent<SlimeSpawner>().OnSlimeD
eathSpawn(parent); });
        OnDeath.AddListener(delegate { GameManager.instance.AddScore(tier); });
        StartCoroutine(FollowPlayerRecursive());
        float scale = tier * 0.4f;
        parent.GetComponent<Transform>().localScale =
parent.GetComponent<Transform>().localScale * scale;
    }

    private void FixedUpdate()
    {
        rb.rotation = transform.rotation;
    }
    void Update()
    {
        speed = 2.5f;
        speed += GameManager.instance.Wave * 0.1f;
        transform.LookAt(player.transform.position);
        if (transform.position.y < -10f)
        {
            Die();
        }

    }

    void MoveTowardsPlayer()
    {
        rb.AddForce(transform.forward * speed, ForceMode.Impulse);
    }

    IEnumerator FollowPlayerRecursive()
    {
        yield return new WaitForSeconds(moveDelay);
        MoveTowardsPlayer();
        StartCoroutine(FollowPlayerRecursive());
    }
    IEnumerator DisableSpawnInvincibility()
    {
        yield return new WaitForSeconds(1.5f);
        canDie = true;
    }
    public void Die()
    {
        OnDeath.Invoke();
        Instantiate(particles, transform.position, Quaternion.identity);
        Destroy(parent);
    }

    public void PushInDir(Vector3 dir)
    {
        if (rb == null) rb = GetComponent<Rigidbody>();
        rb.mass = tier;
        rb.AddForce(dir * 5f * rb.mass, ForceMode.Impulse);
    }
}
```

2. **SlimeSpawner.cs**

```
using System;

using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.Timeline;

public class SlimeSpawner : MonoBehaviour
{
    [SerializeField] GameObject spawnPoints;
    [SerializeField] GameObject player;
    List<Transform> spawnTransforms;
    [SerializeField] GameObject slimePrefab;
    List<Enemy> enemies;
    int spawnCount;
    public UnityEvent OnSpawn;
    void Start()
    {
        enemies = new List<Enemy>();
        spawnTransforms =
spawnPoints.GetComponentsInChildren<Transform>().ToList();
        spawnTransforms.RemoveAt(0);
        GenerateSpawns();
    }

    // Update is called once per frame
    void Update()
    {
        CleanSlimes();

        if (!enemies.Any())
        {
            int diffMult = Mathf.RoundToInt(Mathf.Clamp(GameManager.instance.Wave
/ 10, 0, spawnTransforms.Count - 1));
            GenerateSpawns(diffMult);
        }
    }

    void CleanSlimes()
    {
        if (enemies.Any(n => n == null))
        {
            enemies.RemoveAll(n => n == null);
        }
    }
    void GenerateSpawns(int difficultyMult = 0)
    {
        spawnCount = UnityEngine.Random.Range(1 + difficultyMult,
spawnTransforms.Count + 1);
        List<int> spawnIndexes = GenerateSpawnIndexes();
        spawnIndexes.ForEach(index => enemies.Add(Instantiate(slimePrefab,
spawnTransforms[index].position,
Quaternion.identity).GetComponentInChildren<Enemy>()));
        foreach (Enemy enemy in enemies)
        {
            enemy.player = player;
            enemy.tier = UnityEngine.Random.Range(1, 4);
            enemy.rb.mass = enemy.tier;
        }
        OnSpawn.Invoke();
    }
    List<int> GenerateSpawnIndexes()
    {
        List<int> indexes = new List<int>();

        for (int i = 0; i < spawnCount; i++)
        {
            int toAdd = UnityEngine.Random.Range(0, spawnTransforms.Count);
            int compare = toAdd;
            while (indexes.Contains(toAdd))
            {
                toAdd++;
                if (toAdd == spawnTransforms.Count)
                {
                    toAdd = 0;
                }
                if (toAdd == compare)
                {
                    break;
                }
            }
            indexes.Add(toAdd);
        }
        return indexes;
    }

    public void OnSlimeDeathSpawn(GameObject slime)
    {
        if (slime.GetComponentInChildren<Enemy>().tier != 1)
        {
            Transform t =
slime.GetComponentInChildren<Enemy>().gameObject.transform;
            t.position = new Vector3(t.position.x, 4f, t.position.z);
            enemies.Add(Instantiate(slime, t.position,
Quaternion.identity).GetComponentInChildren<Enemy>());
            enemies.Add(Instantiate(slime, t.position,
Quaternion.identity).GetComponentInChildren<Enemy>());
            enemies[enemies.Count - 1].tier -= 1;
            enemies[enemies.Count - 2].tier -= 1;
            enemies[enemies.Count - 1].PushInDir(t.right * -1);
            enemies[enemies.Count - 2].PushInDir(t.right);
        }
    }
}
```

II. Misc

1. GameManager.cs

```
using System.Collections;

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

```
public class GameManager : MonoBehaviour
{
    public static GameManager instance;
    public int Score { get; private set; }
    public int Wave { get; private set; }

    public UnityEvent ScoreChanged;
    public UnityEvent WaveChanged;
    public UnityEvent OnPause;
    public UnityEvent OnGameOver;

    bool paused = false;
    private void Awake()
    {
        instance = this;
    }
    void Start()
    {
        Time.timeScale = 1f;
        Score = 0;
        Wave = 0;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetAxisRaw("Cancel") == 1)
        {
            PauseGame();
        }
    }

    public void AddScore(int score)
    {
        Score += score;
        ScoreChanged.Invoke();
    }

    public void AddWave()
    {
        Wave++;
        WaveChanged.Invoke();
    }
    public void PauseGame()
    {
        if (!paused)
        {
            Time.timeScale = 0f;
            paused = true;
            OnPause.Invoke();
        }
    }

    public void UnpauseGame()
    {
        paused = false;
        Time.timeScale = 1f;
    }

    public void EndGame()
    {
        StartCoroutine(GameOverInSeconds(2f));
    }
    IEnumerator GameOverInSeconds(float seconds)
    {
        yield return new WaitForSeconds(seconds);
        PlayerPrefsManager.SetHighScore(Score);
        Time.timeScale = 0f;
        OnGameOver.Invoke();
    }
    public void LoadMainMenu()
    {
        SceneManager.LoadScene("MainMenu");
    }
    public void LoadGame()
    {
        SceneManager.LoadScene("Level");
    }
}
```

2. **MainMenu.cs**

```
using System.Collections;

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

public class MainMenu : MonoBehaviour
{
    [SerializeField] TextMeshProUGUI highscoreText;
    void Start()
    {
        Time.timeScale = 1.0f;
        int hs = PlayerPrefsManager.GetHighScore();
        highscoreText.text = $"High Score: {hs}";
    }
    public void LoadGame()
    {
        SceneManager.LoadScene("Level");
    }
}
```

3. **OutOfBoundsBuffer.cs**

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class OutOfBoundsBuffer : MonoBehaviour
{
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Enemy"))
        {
            Debug.Log("Enemy Caught");
            if (other.GetComponent<Enemy>().canDie)
            {
                other.GetComponent<Enemy>().Die();
            }
        }
    }
}
```

4. **PlayerPrefsManager.cs**

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class PlayerPrefsManager : MonoBehaviour
{
    public static int GetHighScore()
    {
        if (!PlayerPrefs.HasKey("HighScore"))
        {
            PlayerPrefs.SetInt("HighScore", 0);
        }
        return PlayerPrefs.GetInt("HighScore");

    }

    public static void SetHighScore(int value)
    {
        int hs = GetHighScore();
        if (value > hs)
        {
            PlayerPrefs.SetInt("HighScore", value);
        }
    }
}
```

5. UIManager.cs

```
using System.Collections;

using System.Collections.Generic;
using TMPro;
using UnityEngine;

public class UIManager : MonoBehaviour
{
    [SerializeField] TextMeshProUGUI scoreText;
    [SerializeField] TextMeshProUGUI GOScoreText;
    [SerializeField] TextMeshProUGUI GOHighScoreText;
    [SerializeField] TextMeshProUGUI WaveText;
    [SerializeField] List<GameObject> Menus;
    void Start()
    {
        ChangeScoreText();
        ChangeWaveText();
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void ChangeGOScoreText()
    {
        GOScoreText.text = $"Current Score: {GameManager.instance.Score}";
    }
    public void ChangeGOHighScoreText()
    {
        int hs = PlayerPrefsManager.GetHighScore();
        GOHighScoreText.text = $"High Score: {hs}";
    }
    public void ChangeScoreText()
    {
        scoreText.text = $"Score: {GameManager.instance.Score}";
    }
    public void ChangeWaveText()
    {
        StartCoroutine(FadeWaveTXTAlphaForTime());
    }
    IEnumerator FadeWaveTXTAlphaForTime()
    {
        float seconds = 1.2f;
        WaveText.CrossFadeAlpha(1f, seconds, true);
        WaveText.text = $"Wave {GameManager.instance.Wave}";
        yield return new WaitForSeconds(seconds);
        WaveText.CrossFadeAlpha(0f, seconds, true);
    }

    public void SelectMenu(string menuName)
    {
        foreach (var menu in Menus)
        {
            menu.SetActive(false);
        }
        switch (menuName)
        {
            case "HUD":
                Menus[0].SetActive(true);
                break;
            case "GameOver":
                Menus[1].SetActive(true);
                break;
            case "Pause":
                Menus[2].SetActive(true);
                break;
        }
    }
}
```

III. Player

1. PlayerController.cs

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class PlayerController : MonoBehaviour
{
    Animator animator;
    GameObject hitbox;
    public enum playerStates
    {
        Idle,
        Moving,
        Attacking,
        Dead
    }

    Rigidbody rb;
    Vector2 moveInput;
    [SerializeField] playerStates state;
    [SerializeField] float speed;
    public UnityEvent OnDeath;
    void Start()
    {
        Physics.IgnoreLayerCollision(9, 8,false);
        animator = GetComponent<Animator>();
        rb = GetComponent<Rigidbody>();
        hitbox = GetComponentInChildren<BoxCollider>().gameObject;
        hitbox.SetActive(false);
    }
    void FixedUpdate()
    {
        if (state != playerStates.Dead)
        {
            transform.position = new Vector3(rb.position.x, -.1f, rb.position.z);
        }
    }
    // Update is called once per frame
    void Update()
    {
        moveInput = new Vector2(Input.GetAxisRaw("Horizontal"),
            Input.GetAxisRaw("Vertical"));
        SetStateOnInput();
        ActOnState();
    }

    void SetStateOnInput()
    {
        if (state != playerStates.Dead)
        {
            if (state != playerStates.Attacking)
            {
                if (moveInput.magnitude == 0)
                {
                    state = playerStates.Idle;
                }
                else
                {
                    state = playerStates.Moving;
                }
            }


            if (Input.GetAxisRaw("Fire1") == 1)
            {
                state = playerStates.Attacking;
            }
        }

        animator.SetInteger("State", (int)state);
    }

    void ActOnState()
    {
        if(state == playerStates.Moving)
        {
            Move();
            RotateOnIpnut();
        }
    }

    void Move()
    {
        float moveMag = 1f;
        if (moveInput.magnitude == Mathf.Sqrt(2))
        {
            moveMag = 0.75f;
        }
        Vector3 dir = new Vector3(moveInput.x, 0, moveInput.y);
        rb.AddForce(dir * speed * moveMag, ForceMode.Impulse);
    }

    void RotateOnIpnut()
    {
        if (moveInput.magnitude != 0 && Time.timeScale==1f)
        {
            float yRot = 90f * moveInput.x;
            if(yRot == 0 && moveInput.y == -1)
            {
                yRot = 180f;
            }
            else if (yRot > 0)
            {
                yRot -= 45f * moveInput.y;
            }
            else if(yRot < 0)
            {
                yRot += 45f * moveInput.y;
            }
            rb.transform.rotation = Quaternion.Euler(0f, yRot, 0f);
        }

    }

    public void EnableHitbox()
    {
        hitbox.SetActive(true);
    }

    public void DisableHitbox()
    {
        hitbox.SetActive(false);
    }

    public void FinishedAttack()
    {
        state = playerStates.Idle;
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Enemy"))
        {
            if (state != playerStates.Dead &&
                collision.gameObject.GetComponent<Enemy>().canDie &&
                collision.transform.position.y <= 1f)
            {
                OnDeath.Invoke();
            }
        }
    }

    public void WhenDead()
    {
        state = playerStates.Dead;
        Physics.IgnoreLayerCollision(9, 8);
        rb.freezeRotation = false;
        rb.AddForce(transform.forward * -1000f, ForceMode.Impulse);
        rb.AddTorque(transform.right * -300f, ForceMode.Impulse);
    }
}
```

2. PlayerHitbox.cs

```
using System.Collections;

using System.Collections.Generic;
using UnityEngine;

public class PlayerHitbox : MonoBehaviour
{
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Enemy"))
        {
            if(other.GetComponent<Enemy>().canDie)
            {
                other.GetComponent<Enemy>().Die();
            }
        }
    }
}
```