PDF Portifolio

Name: Navod Avishka Hewawanithunga
This portfolio showcases code snippets, screenshots, and video links from my Unity projects.

Table of Contents

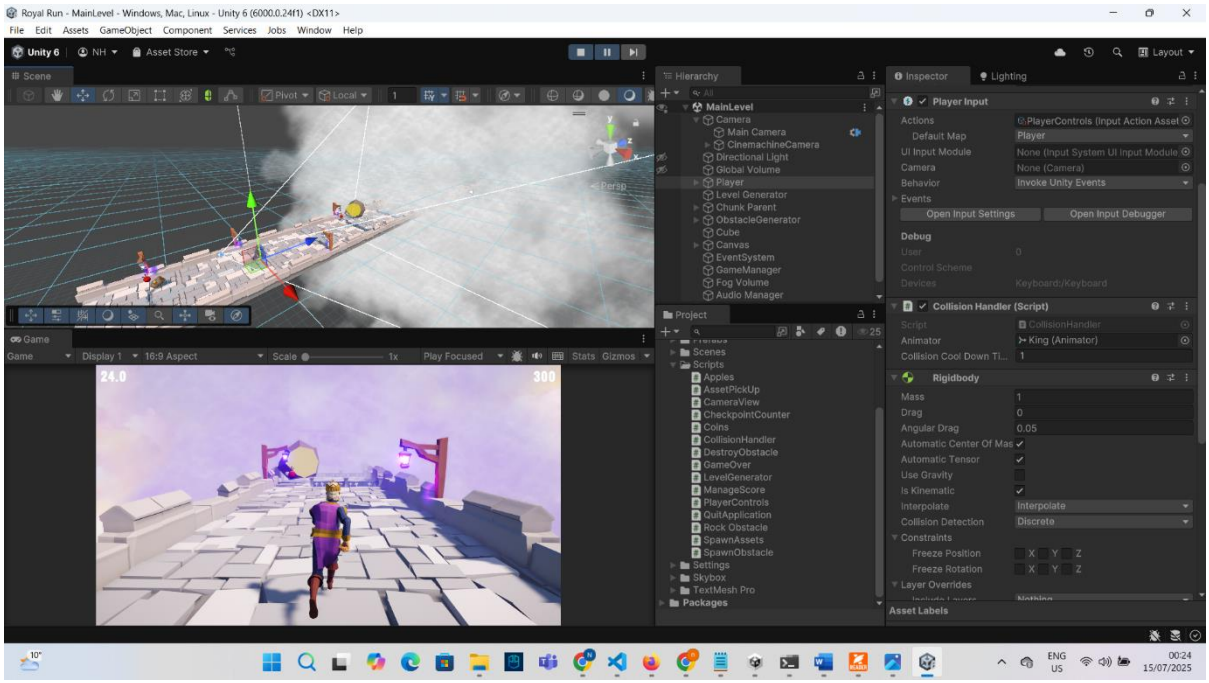
Project 1: Royal Run

Overview

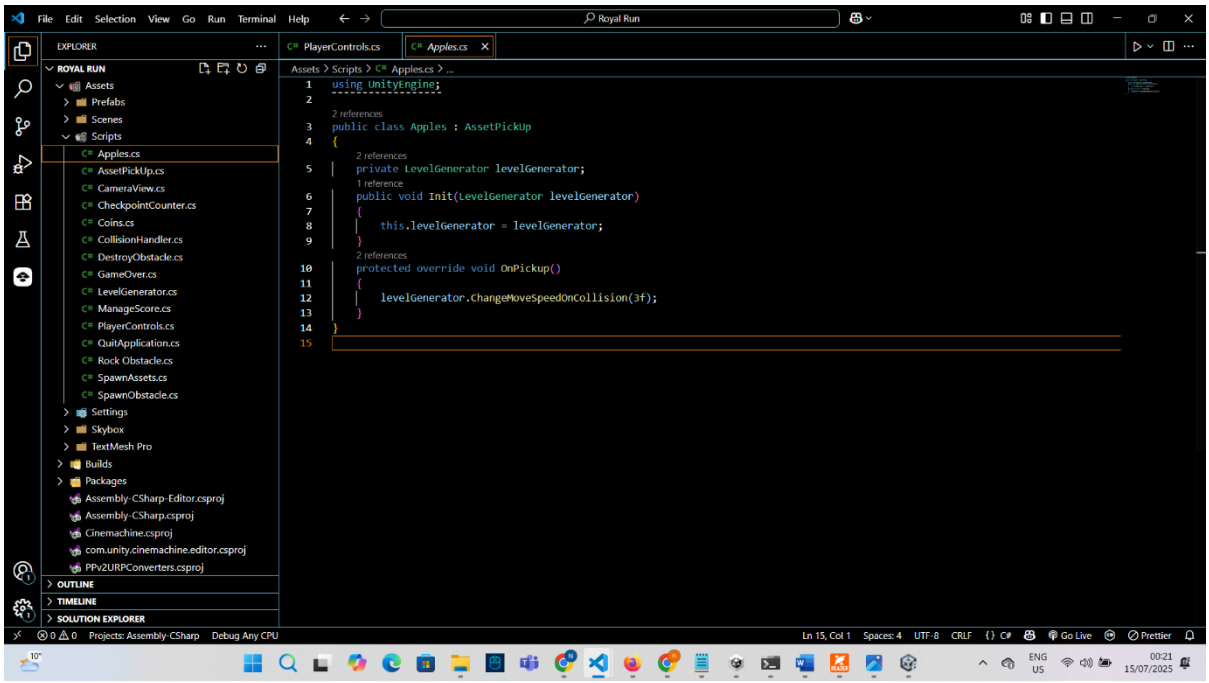
Endless runner with obstacle avoidance, pickup systems, fog lighting, Unity's new Input System, and animation events, Cinemachine and Cinemachine Impulse source. Used object-oriented concepts like Inheritance

Here is the Short Youtube clip of the game: https://youtu.be/qGxxW2_Id3A

Unity Editor

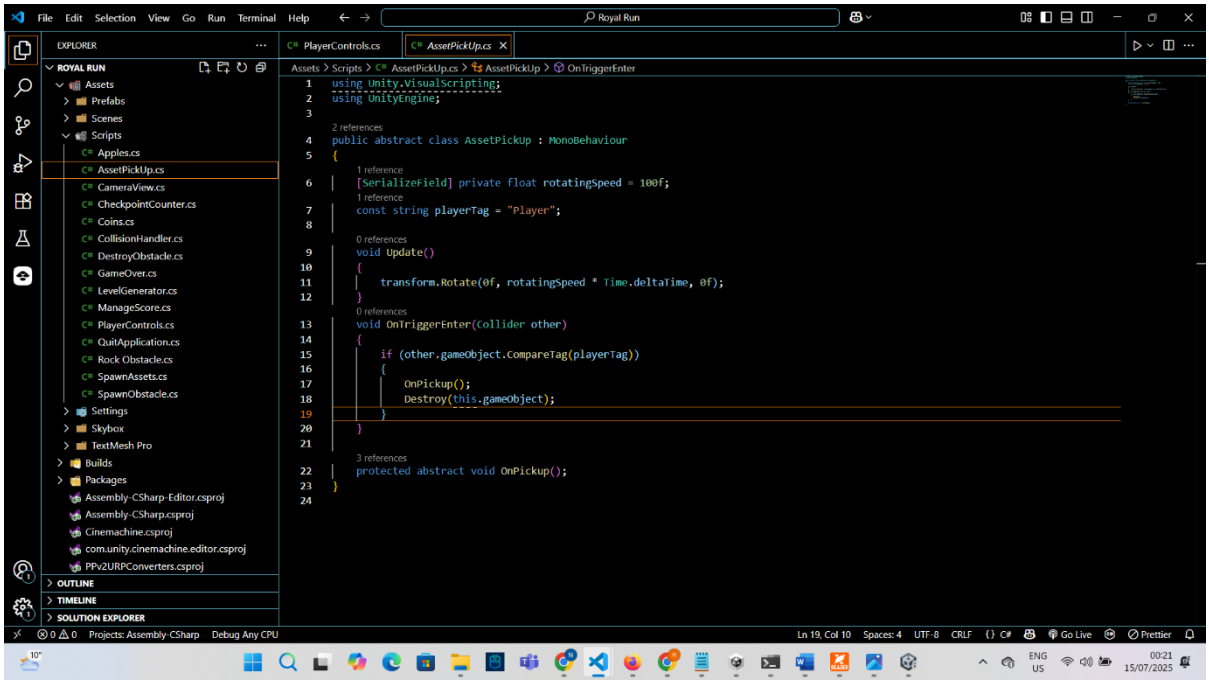


Apples.cs



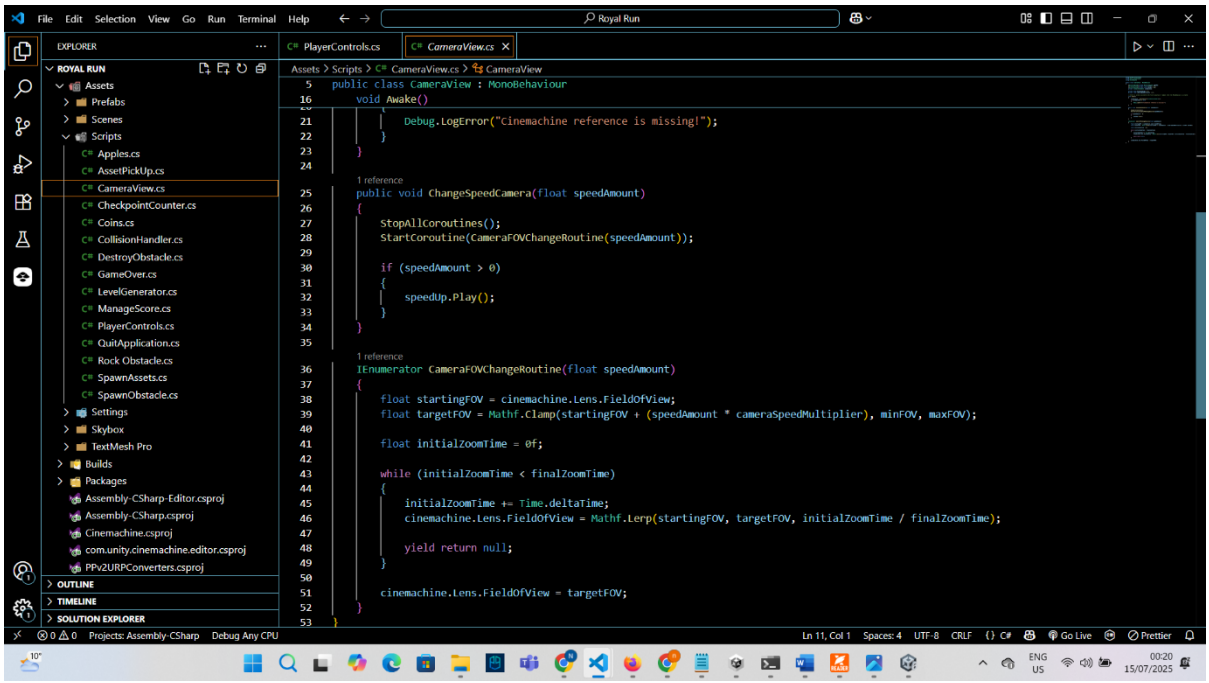
```
using UnityEngine;

2 references
public class Apples : AssetPickUp
{
    2 references
    private LevelGenerator levelGenerator;
    1 reference
    public void Init(LevelGenerator levelGenerator)
    {
        this.levelGenerator = levelGenerator;
    }
    2 references
    protected override void OnPickup()
    {
        levelGenerator.ChangeMoveSpeedOnCollision(3f);
    }
}
```

AssetPickUp.cs



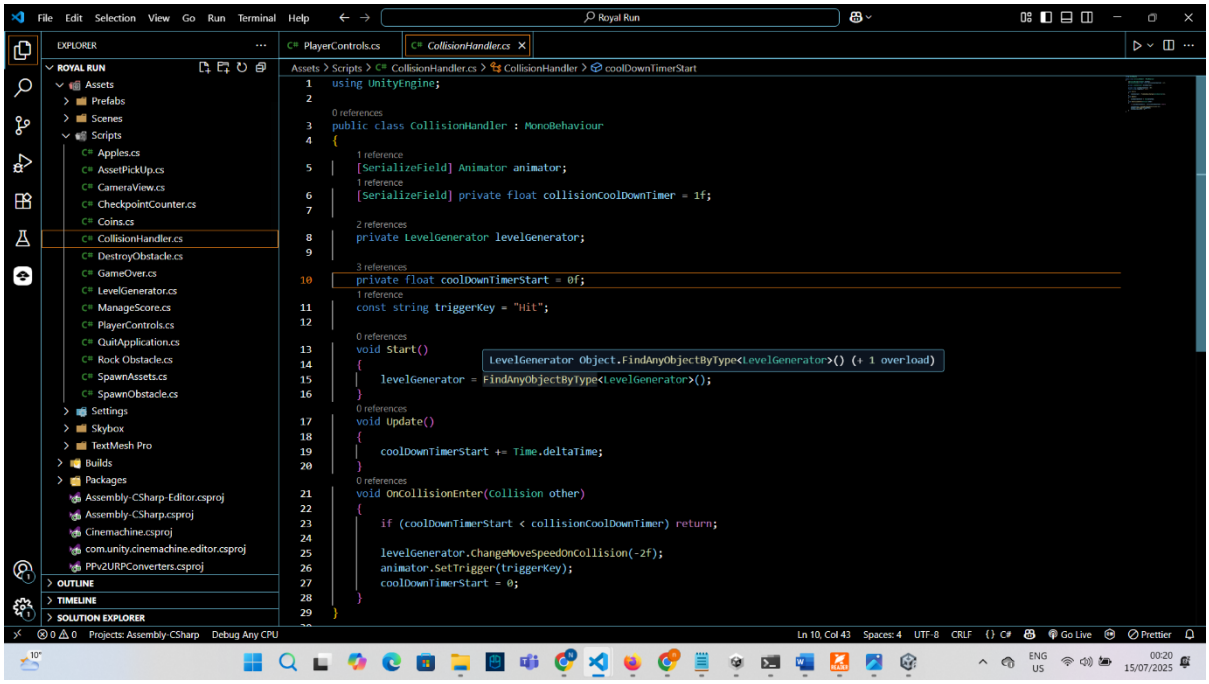
```
using Unity.VisualScripting;
using UnityEngine;

2 references
public abstract class AssetPickUp : MonoBehaviour
{
    1 reference
    [SerializeField] private float rotatingSpeed = 100f;
    1 reference
    const string playerTag = "Player";

    0 references
    void Update()
    {
        transform.Rotate(0f, rotatingSpeed * Time.deltaTime, 0f);
    }
    0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject.CompareTag(playerTag))
        {
            OnPickup();
            Destroy(this.gameObject);
        }
    }

    3 references
    protected abstract void OnPickup();
}
```

CameraView.cs



```
public class CameraView : MonoBehaviour
    void Awake()
        Debug.LogError("cinemachine reference is missing!");
    }

    1 reference
    public void ChangeSpeedCamera(float speedAmount)
    {
        StopAllCoroutines();
        StartCoroutine(CameraFOVChangeRoutine(speedAmount));

        if (speedAmount > 0)
        {
            speedUp.Play();
        }
    }

    1 reference
    IEnumerator CameraFOVChangeRoutine(float speedAmount)
    {
        float startingFOV = cinemachine.Lens.FieldOfView;
        float targetFOV = Mathf.Clamp(startingFOV + (speedAmount * cameraSpeedMultiplier), minFOV, maxFOV);

        float initialZoomTime = 0f;

        while (initialZoomTime < finalZoomTime)
        {
            initialZoomTime += Time.deltaTime;
            cinemachine.Lens.FieldOfView = Mathf.Lerp(startingFOV, targetFOV, initialZoomTime / finalZoomTime);

            yield return null;
        }

        cinemachine.Lens.FieldOfView = targetFOV;
    }
}
```

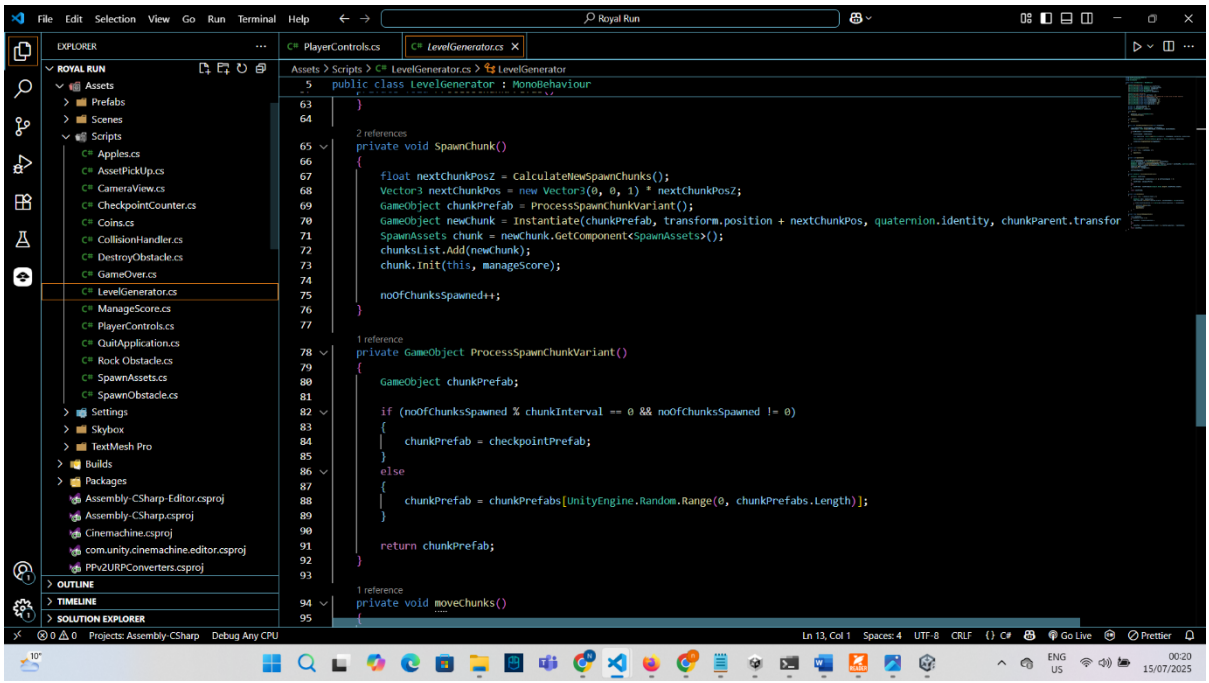
CollisionHandler.cs



```
using UnityEngine;

0 references
public class CollisionHandler : MonoBehaviour
{
    1 reference
    [SerializeField] Animator animator;
    1 reference
    [SerializeField] private float collisionCoolDownTimer = 1f;

    2 references
    private LevelGenerator levelGenerator;

    3 references
    private float coolDownTimerStart = 0f;
    1 reference
    const string triggerKey = "Hit";

    0 references
    void Start()
    {
        levelGenerator = FindAnyObjectByType<LevelGenerator>();
    }
    0 references
    void Update()
    {
        coolDownTimerStart += Time.deltaTime;
    }
    0 references
    void OnCollisionEnter(Collision other)
    {
        if (coolDownTimerStart < collisionCoolDownTimer) return;

        levelGenerator.ChangeMoveSpeedOnCollision(-2f);
        animator.SetTrigger(triggerKey);
        coolDownTimerStart = 0;
    }
}
```

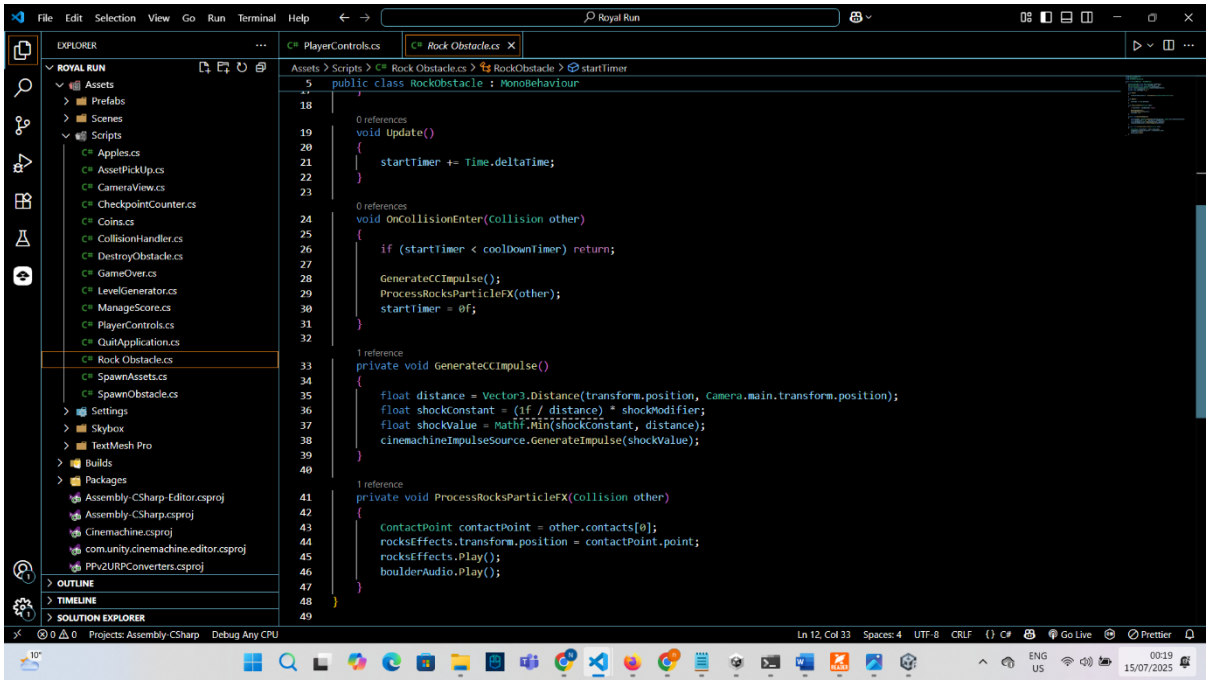
LevelGenerator Object.FindAnyObjectByType<LevelGenerator>() (+ 1 overload)

LevelGenerator.cs



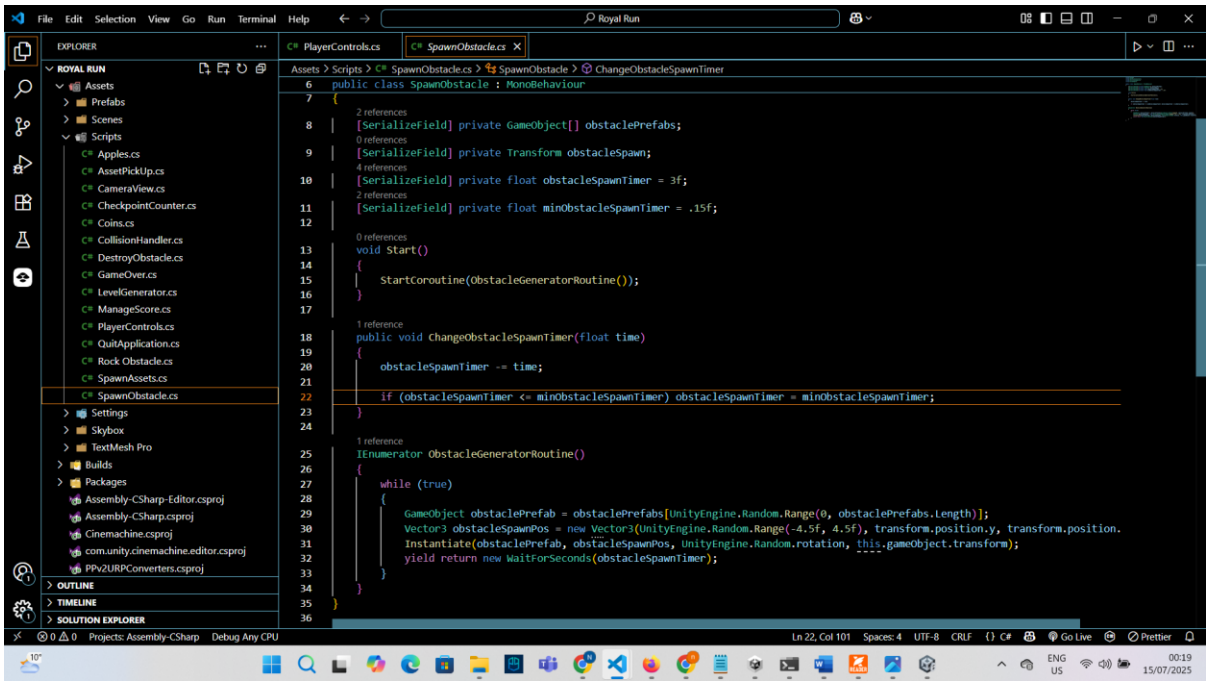
```
public class LevelGenerator : MonoBehaviour
{
}

2 references
private void SpawnChunk()
{
    float nextChunkPosZ = CalculateNewSpawnChunks();
    Vector3 nextChunkPos = new Vector3(0, 0, 1) * nextChunkPosZ;
    GameObject chunkPrefab = ProcessSpawnChunkVariant();
    GameObject newChunk = Instantiate(chunkPrefab, transform.position + nextChunkPos, quaternion.identity, chunkParent.transfor
    SpawnAssets chunk = newChunk.GetComponent<SpawnAssets>();
    chunksList.Add(newChunk);
    chunk.Init(this, manageScore);

    noOfChunksSpawned++;
}

1 reference
private GameObject ProcessSpawnChunkVariant()
{
    GameObject chunkPrefab;

    if (noOfChunksSpawned % chunkInterval == 0 && noOfChunksSpawned != 0)
    {
        chunkPrefab = checkpointPrefab;
    }
    else
    {
        chunkPrefab = chunkPrefabs[UnityEngine.Random.Range(0, chunkPrefabs.Length)];
    }

    return chunkPrefab;
}

1 reference
private void moveChunks()
{
```

Rock Obstacle.cs



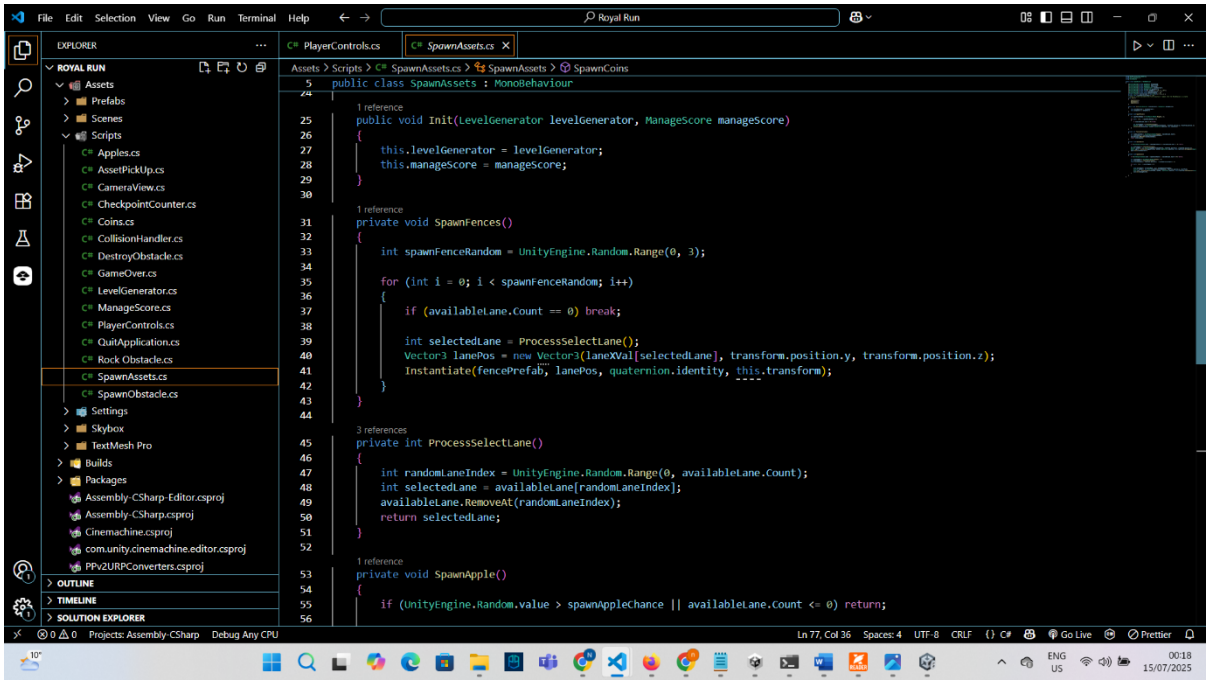
```
public class RockObstacle : MonoBehaviour
{

0 references
void Update()
{
    startTimer += Time.deltaTime;
}

0 references
void OnCollisionEnter(Collision other)
{
    if (startTimer < coolDownTimer) return;

    GenerateCCImpulse();
    ProcessRocksParticleFX(other);
    startTimer = 0f;
}

1 reference
private void GenerateCCImpulse()
{
    float distance = Vector3.Distance(transform.position, Camera.main.transform.position);
    float shockConstant = (1f / distance) * shockModifier;
    float shockValue = Mathf.Min(shockConstant, distance);
    cinemachineImpulseSource.GenerateImpulse(shockValue);
}

1 reference
private void ProcessRocksParticleFX(Collision other)
{
    ContactPoint contactPoint = other.contacts[0];
    rocksEffects.transform.position = contactPoint.point;
    rocksEffects.Play();
    boulderAudio.Play();
}
}
```

SpawnObstacle.cs



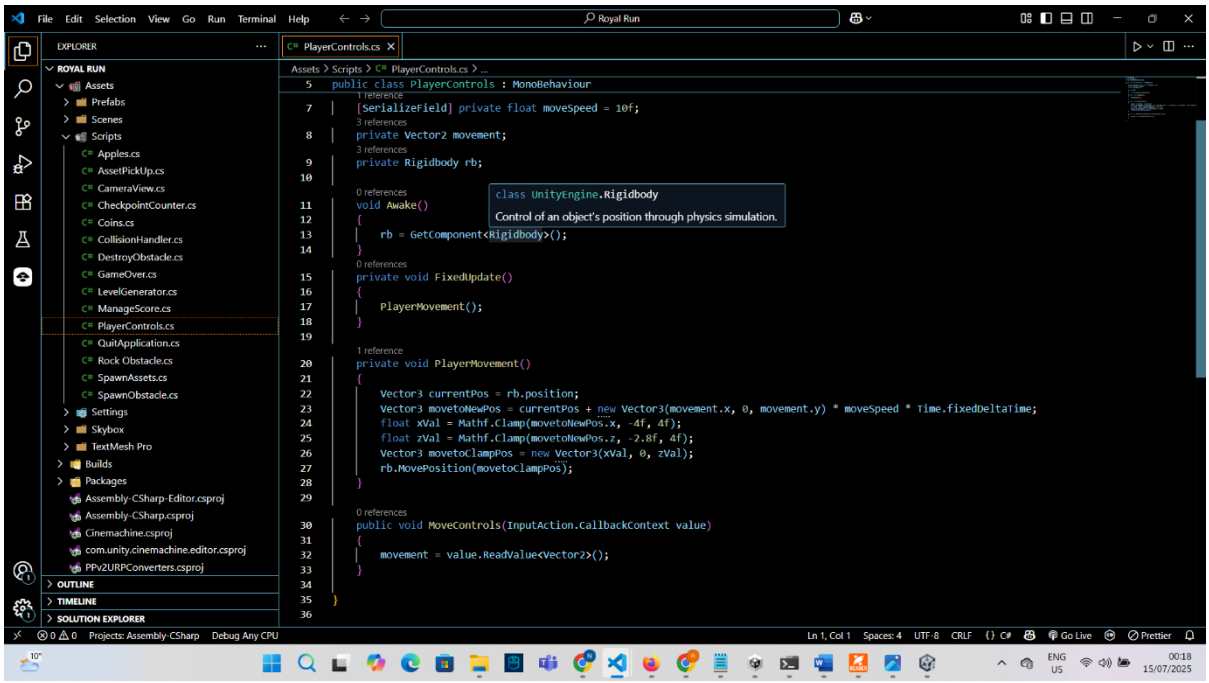
```
public class SpawnObstacle : MonoBehaviour
{
    [SerializeField] private GameObject[] obstaclePrefabs;
    [SerializeField] private Transform obstacleSpawn;
    [SerializeField] private float obstacleSpawnTimer = 3f;
    [SerializeField] private float minObstacleSpawnTimer = .15f;

    void Start()
    {
        StartCoroutine(ObstacleGeneratorRoutine());
    }

    public void ChangeObstacleSpawnTimer(float time)
    {
        obstacleSpawnTimer -= time;

        if (obstacleSpawnTimer <= minObstacleSpawnTimer) obstacleSpawnTimer = minObstacleSpawnTimer;
    }

    IEnumerator ObstacleGeneratorRoutine()
    {
        while (true)
        {
            GameObject obstaclePrefab = obstaclePrefabs[UnityEngine.Random.Range(0, obstaclePrefabs.Length)];
            Vector3 obstacleSpawnPos = new Vector3(UnityEngine.Random.Range(-4.5f, 4.5f), transform.position.y, transform.position.
            Instantiate(obstaclePrefab, obstacleSpawnPos, UnityEngine.Random.rotation, this.gameObject.transform);
            yield return new WaitForSeconds(obstacleSpawnTimer);
        }
    }
}
```

SpawnAssets.cs



```
public class SpawnAssets : MonoBehaviour

    public void Init(LevelGenerator levelGenerator, ManageScore manageScore)
    {
        this.levelGenerator = levelGenerator;
        this.manageScore = manageScore;
    }

    private void SpawnFences()
    {
        int spawnFenceRandom = UnityEngine.Random.Range(0, 3);

        for (int i = 0; i < spawnFenceRandom; i++)
        {
            if (availableLane.Count == 0) break;

            int selectedLane = ProcessSelectLane();
            Vector3 lanePos = new Vector3(laneXVal[selectedLane], transform.position.y, transform.position.z);
            Instantiate(fencePrefab, lanePos, quaternion.identity, this.transform);
        }
    }

    private int ProcessSelectLane()
    {
        int randomLaneIndex = UnityEngine.Random.Range(0, availableLane.Count);
        int selectedLane = availableLane[randomLaneIndex];
        availableLane.RemoveAt(randomLaneIndex);
        return selectedLane;
    }

    private void SpawnApple()
    {
        if (UnityEngine.Random.value > spawnAppleChance || availableLane.Count <= 0) return;
```

PlayerControls.cs

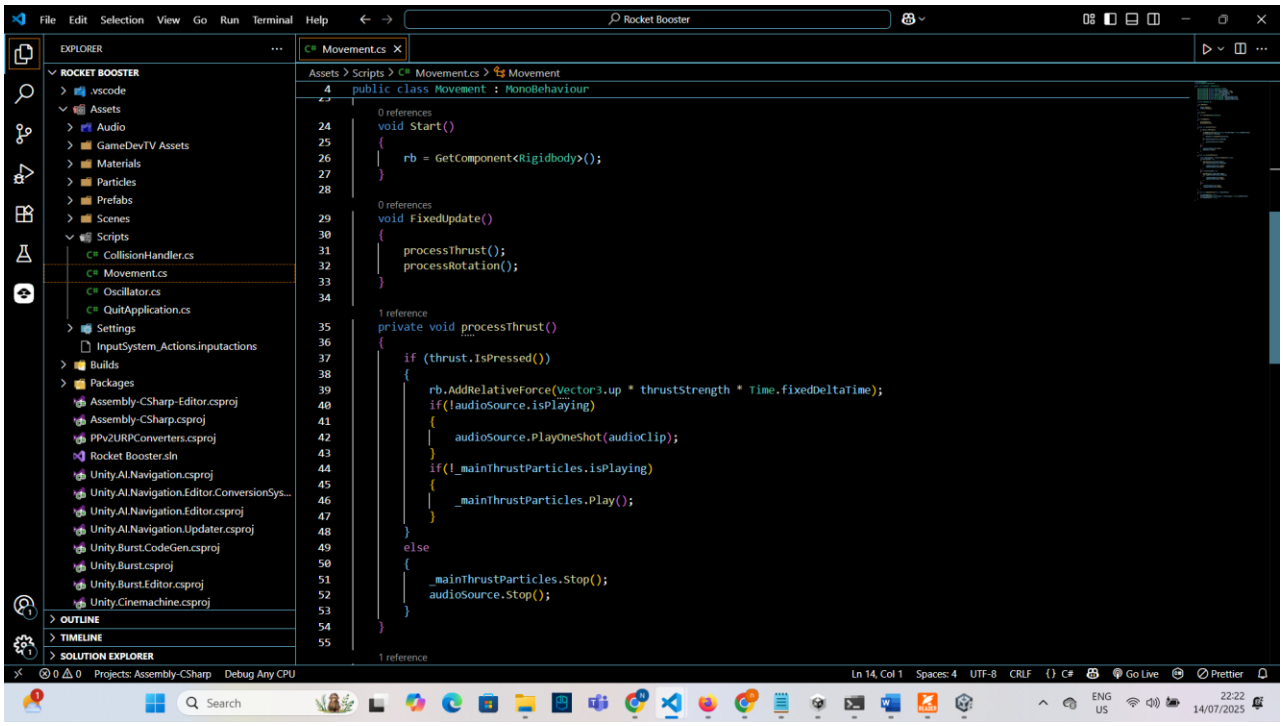


Project 2: Rocket Booster

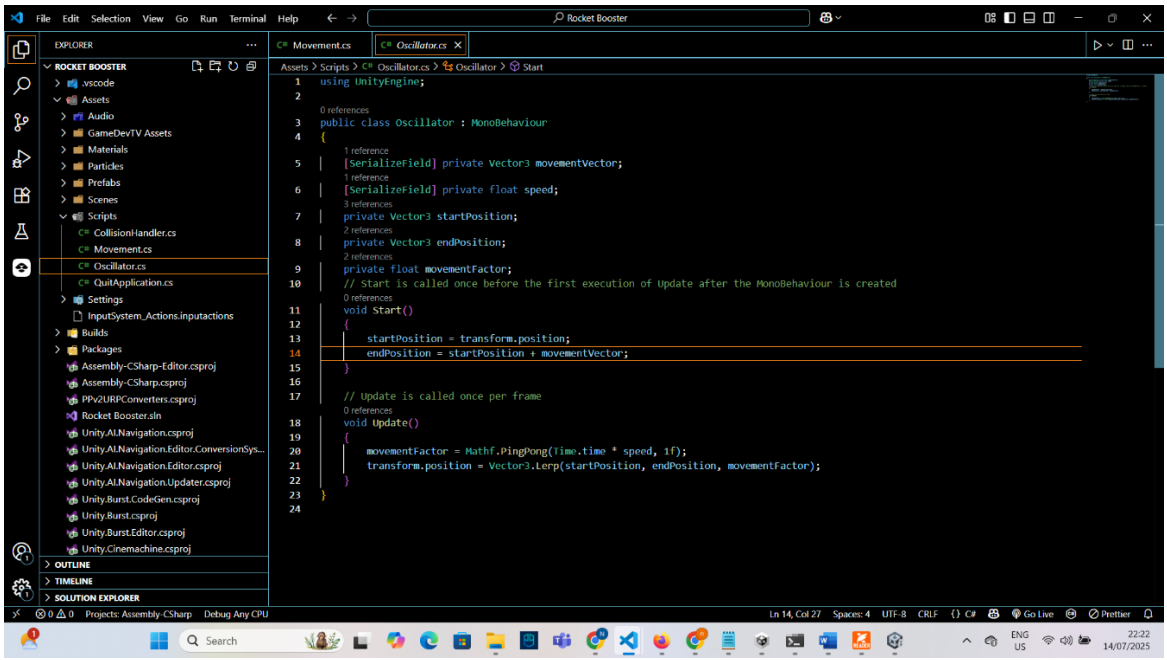
Overview: A 3D rocket navigation game using Rigidbody physics, Cinemachine, particle systems, and level progression logic.

Here is the youtube video clip of the game: https://youtu.be/1MA5DufkdFA

Movement.cs

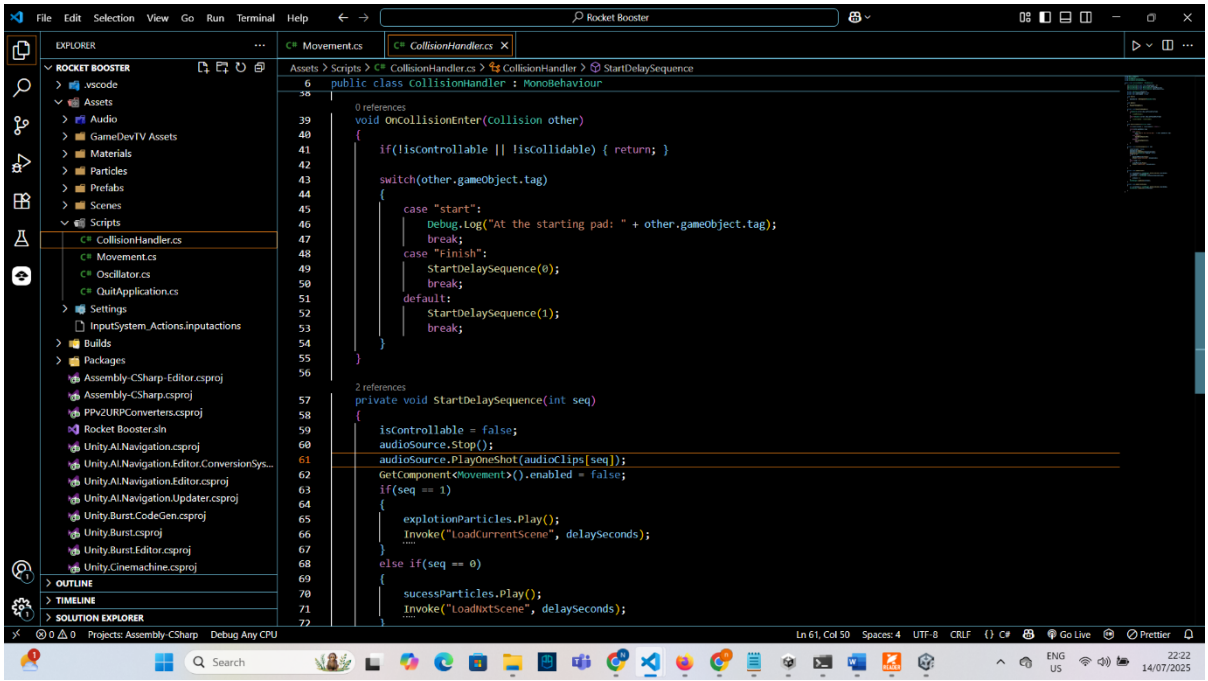


Oscillator.cs



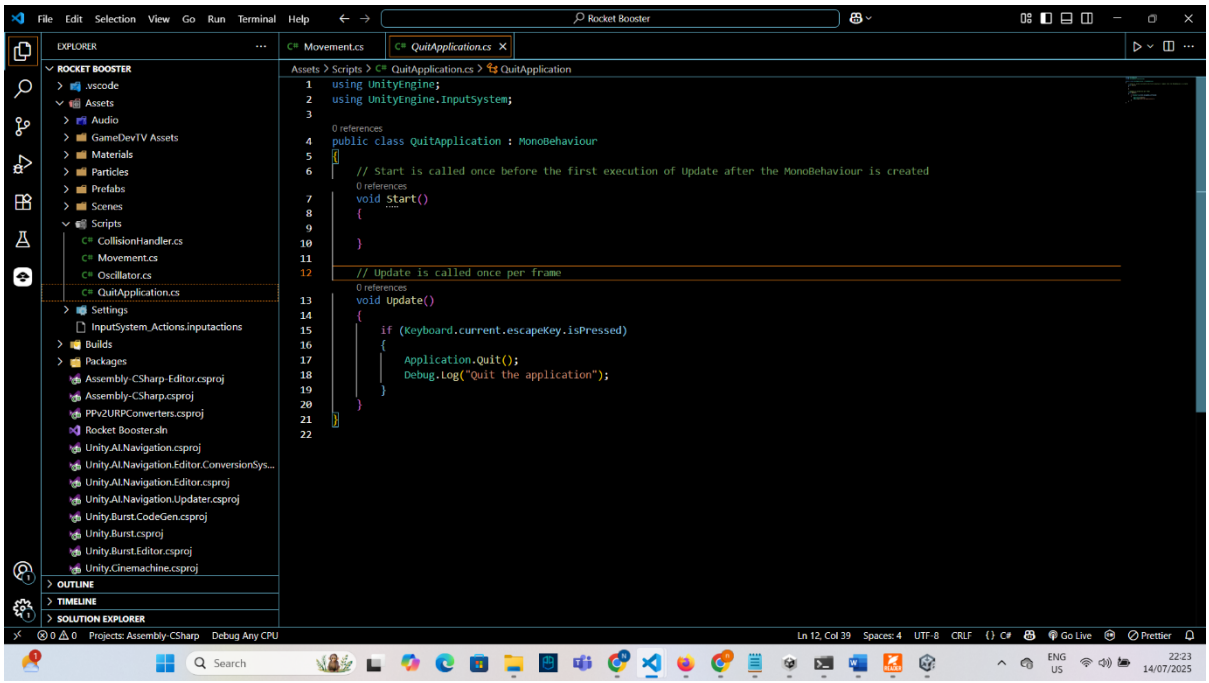
```
using UnityEngine;

public class Oscillator : MonoBehaviour
{
    [SerializeField] private Vector3 movementVector;
    [SerializeField] private float speed;
    private Vector3 startPosition;
    private Vector3 endPosition;
    private float movementFactor;
    // Start is called once before the first execution of Update after the MonoBehaviour is created
    void Start()
    {
        startPosition = transform.position;
        endPosition = startPosition + movementVector;
    }

    // Update is called once per frame
    void Update()
    {
        movementFactor = Mathf.PingPong(Time.time * speed, 1f);
        transform.position = Vector3.Lerp(startPosition, endPosition, movementFactor);
    }
}
```

CollisionHandler.cs



```
public class CollisionHandler : MonoBehaviour
{

    void OnCollisionEnter(Collision other)
    {
        if(!isControllable || !isCollidable) { return; }

        switch(other.gameObject.tag)
        {
            case "start":
                Debug.Log("At the starting pad: " + other.gameObject.tag);
                break;
            case "Finish":
                StartDelaySequence(0);
                break;
            default:
                StartDelaySequence(1);
                break;
        }
    }

    private void StartDelaySequence(int seq)
    {
        isControllable = false;
        audioSource.Stop();
        audioSource.PlayOneShot(audioClips[seq]);
        GetComponent<Movement>().enabled = false;
        if(seq == 1)
        {
            explotionParticles.Play();
            Invoke("LoadCurrentScene", delaySeconds);
        }
        else if(seq == 0)
        {
            sucessParticles.Play();
            Invoke("LoadNxtScene", delaySeconds);
        }
```

QuitApplication.cs



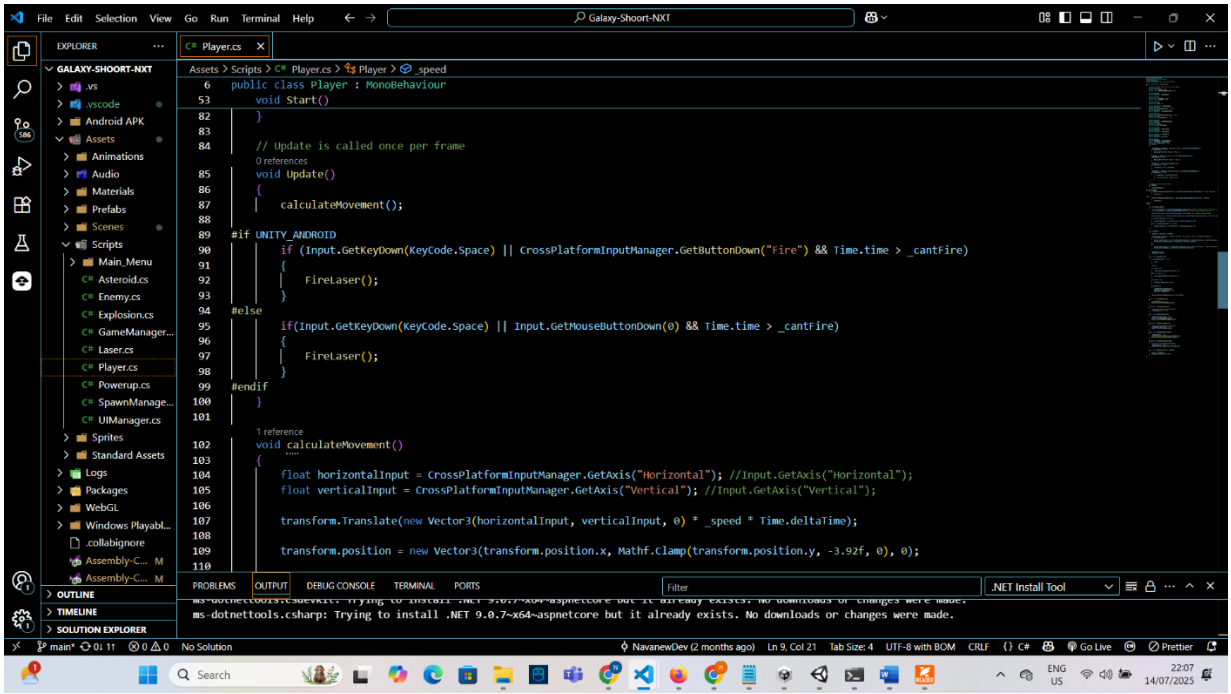
Project 3: Space Shooter NXT

Overview

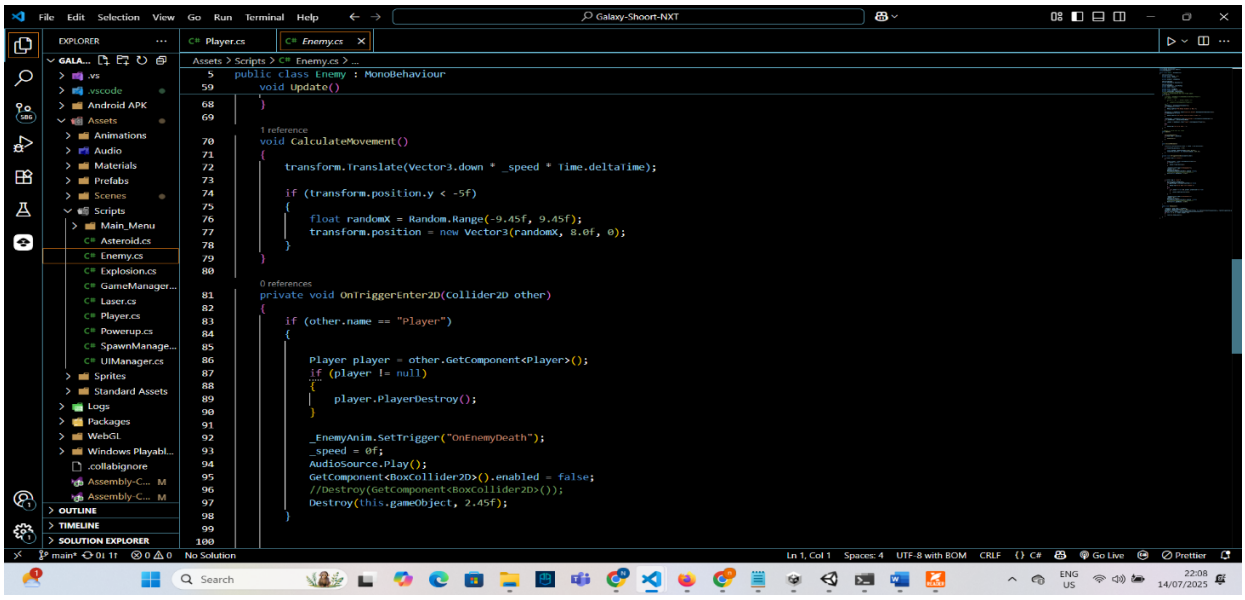
A 2d Game developed using Unity (2019.1.5f1). A spaceship moving right to left while killing enemies using lasers, while having clamped movement up and down.

Here is the youtube clip of the game: https://youtu.be/q08RrHWISYQ

Player.cs

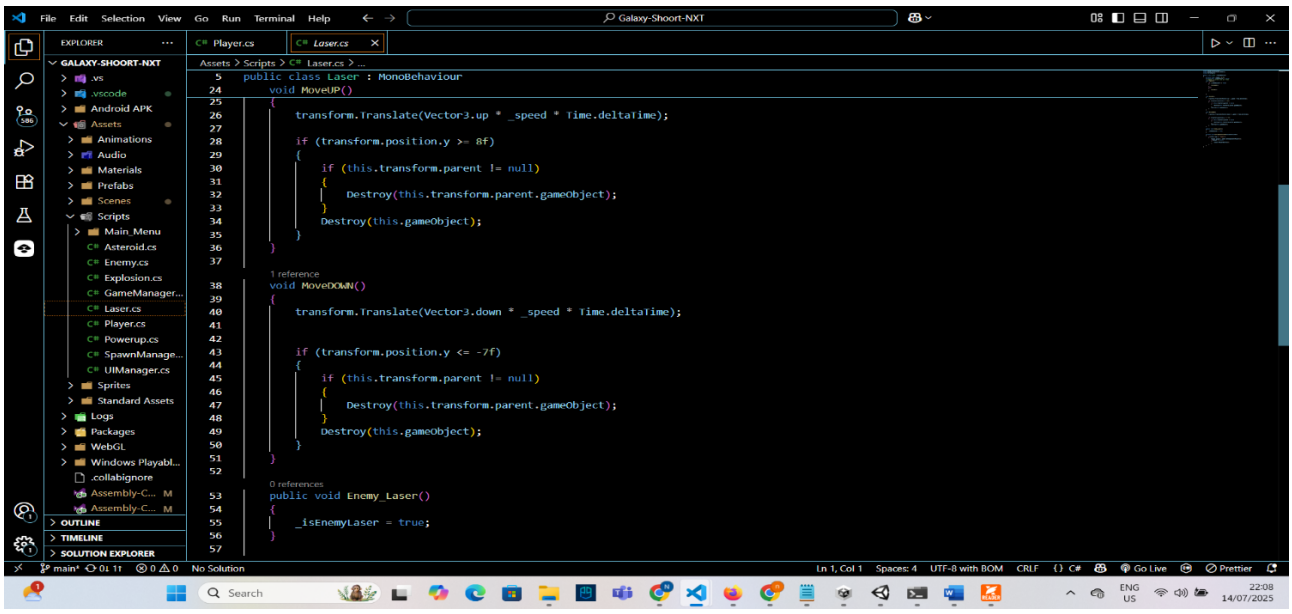


Enemy.cs



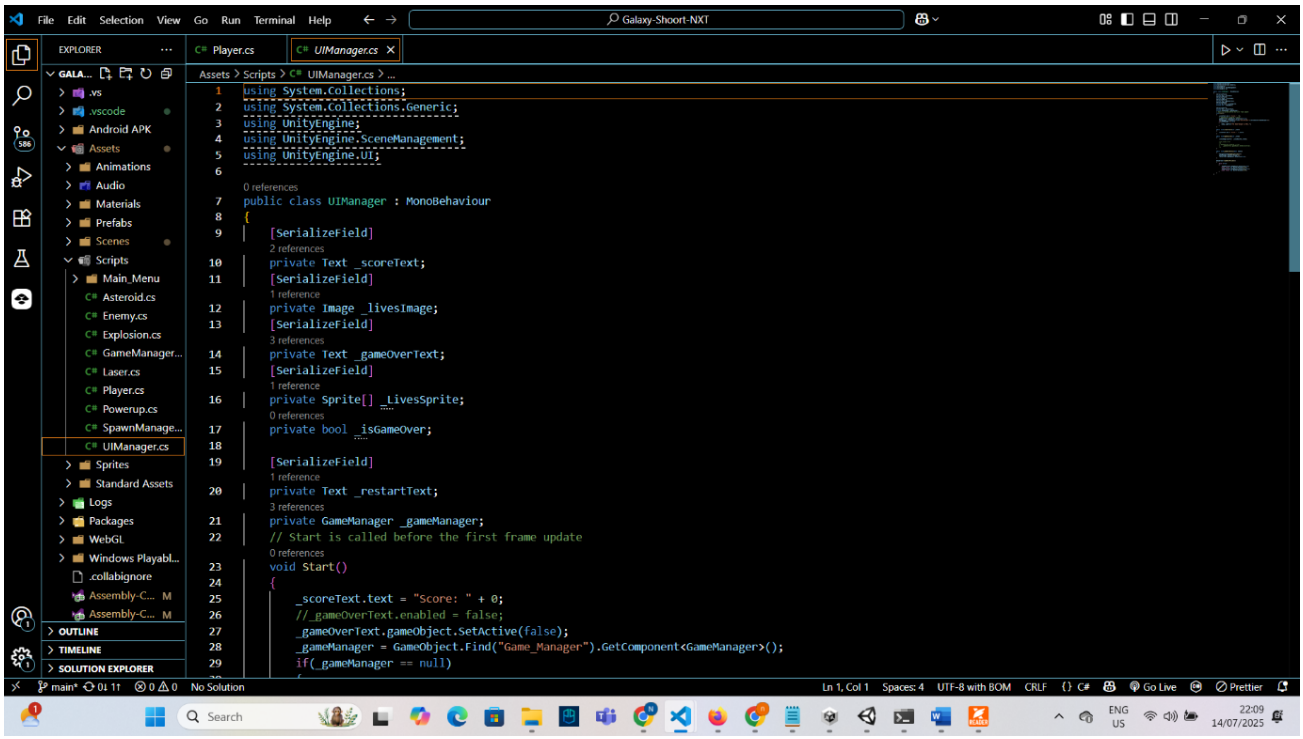
```
public class Enemy : MonoBehaviour
    void Update()
    }

    1 reference
    void CalculateMovement()
    {
        transform.Translate(Vector3.down * _speed * Time.deltaTime);

        if (transform.position.y < -5f)
        {
            float randomX = Random.Range(-9.45f, 9.45f);
            transform.position = new Vector3(randomX, 8.0f, 0);
        }
    }

    0 references
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.name == "Player")
        {

            Player player = other.GetComponent<Player>();
            if (player != null)
            {
                player.PlayerDestroy();
            }

            _EnemyAnim.SetTrigger("OnEnemyDeath");
            _speed = 0f;
            AudioSource.Play();
            GetComponent<BoxCollider2D>().enabled = false;
            //Destroy(GetComponent<BoxCollider2D>());
            Destroy(this.gameObject, 2.45f);
        }
```

Laser.cs



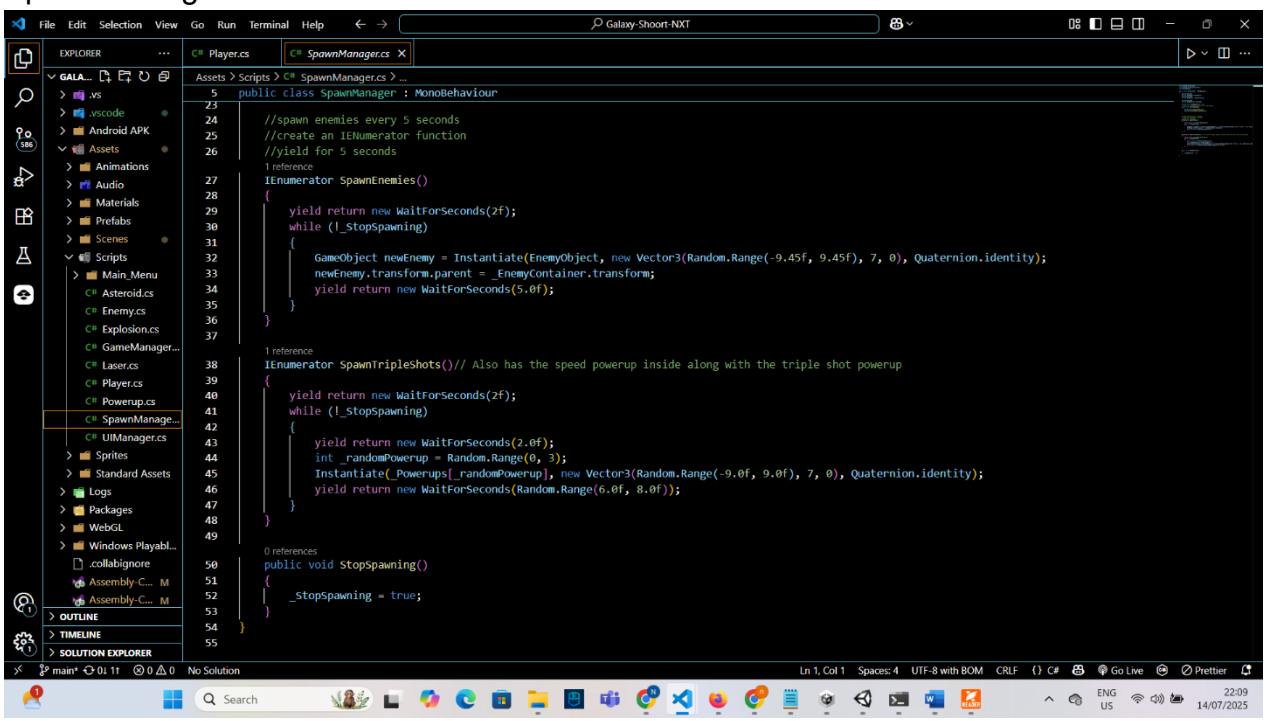
```
public class Laser : MonoBehaviour
    void MoveUP()
    {
        transform.Translate(Vector3.up * _speed * Time.deltaTime);

        if (transform.position.y >= 8f)
        {
            if (this.transform.parent != null)
            {
                Destroy(this.transform.parent.gameObject);
            }
            Destroy(this.gameObject);
        }
    }

    1 reference
    void MoveDOWN()
    {
        transform.Translate(Vector3.down * _speed * Time.deltaTime);


        if (transform.position.y <= -7f)
        {
            if (this.transform.parent != null)
            {
                Destroy(this.transform.parent.gameObject);
            }
            Destroy(this.gameObject);
        }
    }

    0 references
    public void Enemy_Laser()
    {
        _isEnemyLaser = true;
    }
```

UIManager.cs



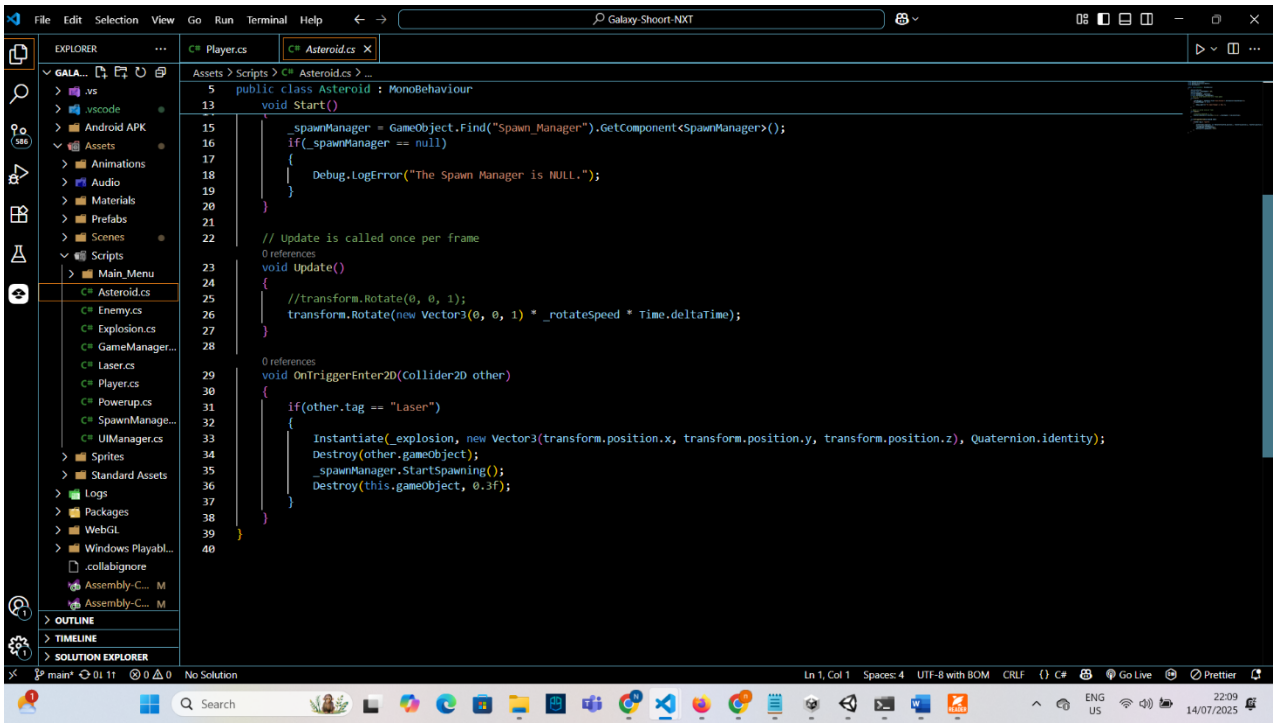
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class UIManager : MonoBehaviour
{
    [SerializeField]
    private Text _scoreText;
    [SerializeField]
    private Image _livesImage;
    [SerializeField]
    private Text _gameOverText;
    [SerializeField]
    private Sprite[] _LivesSprite;
    private bool _isGameOver;

    [SerializeField]
    private Text _restartText;
    private GameManager _gameManager;
    // Start is called before the first frame update
    void Start()
    {
        _scoreText.text = "Score: " + 0;
        //_gameOverText.enabled = false;
        _gameOverText.gameObject.SetActive(false);
        _gameManager = GameObject.Find("Game_Manager").GetComponent<GameManager>();
        if(_gameManager == null)
```

SpawnManager.cs



```
public class SpawnManager : MonoBehaviour

    //spawn enemies every 5 seconds
    //create an IENumerator function
    //yield for 5 seconds
    IEnumerator SpawnEnemies()
    {
        yield return new WaitForSeconds(2f);
        while (!_StopSpawning)
        {
            GameObject newEnemy = Instantiate(EnemyObject, new Vector3(Random.Range(-9.45f, 9.45f), 7, 0), Quaternion.identity);
            newEnemy.transform.parent = _EnemyContainer.transform;
            yield return new WaitForSeconds(5.0f);
        }
    }

    IEnumerator SpawnTripleShots()// Also has the speed powerup inside along with the triple shot powerup
    {
        yield return new WaitForSeconds(2f);
        while (!_StopSpawning)
        {
            yield return new WaitForSeconds(2.0f);
            int _randomPowerup = Random.Range(0, 3);
            Instantiate(_Powerups[_randomPowerup], new Vector3(Random.Range(-9.0f, 9.0f), 7, 0), Quaternion.identity);
            yield return new WaitForSeconds(Random.Range(6.0f, 8.0f));
        }
    }

    public void StopSpawning()
    {
        _StopSpawning = true;
    }
}
```

Asteroid.cs



```
public class Asteroid : MonoBehaviour
    void Start()
        _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager>();
        if(_spawnManager == null)
            Debug.LogError("The Spawn Manager is NULL.");
    // Update is called once per frame
    void Update()
        //transform.Rotate(0, 0, 1);
        transform.Rotate(new Vector3(0, 0, 1) * _rotateSpeed * Time.deltaTime);
    void OnTriggerEnter2D(Collider2D other)
        if(other.tag == "Laser")
            Instantiate(_explosion, new Vector3(transform.position.x, transform.position.y, transform.position.z), Quaternion.identity);
            Destroy(other.gameObject);
            _spawnManager.StartSpawning();
            Destroy(this.gameObject, 0.3f);
```

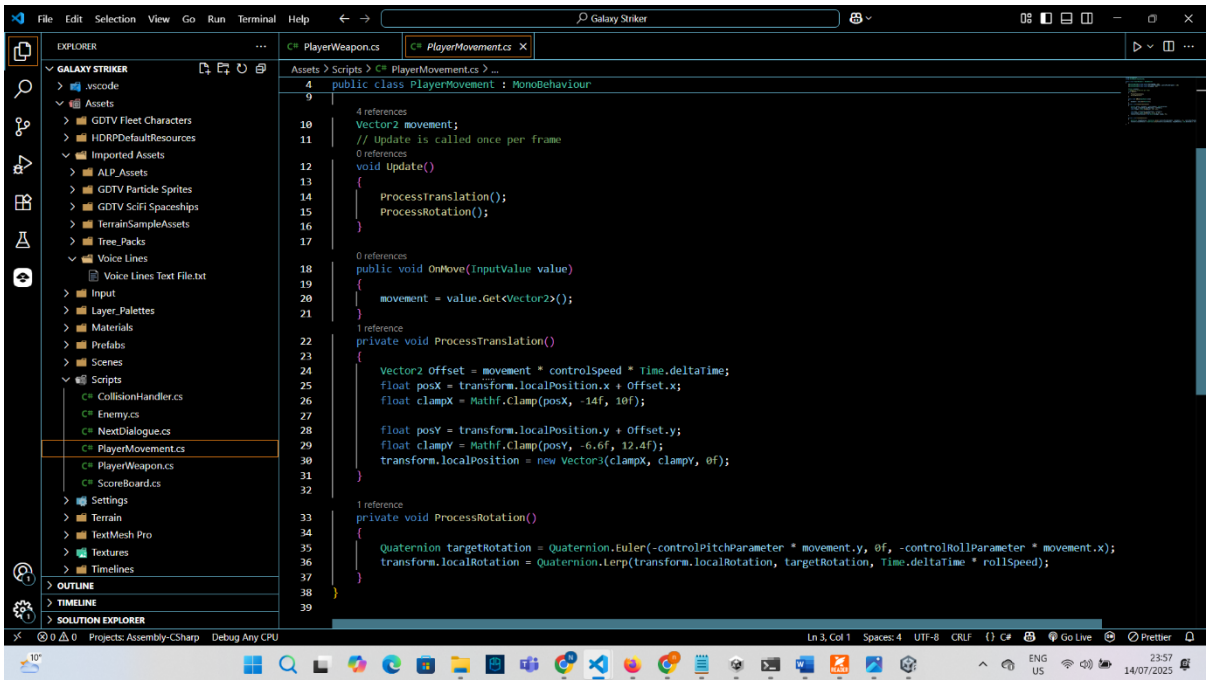
Project 4: Galaxy Striker

Overview

Space combat game featuring timeline-based movement, fog effects, prefab variants, and terrain tool usage.

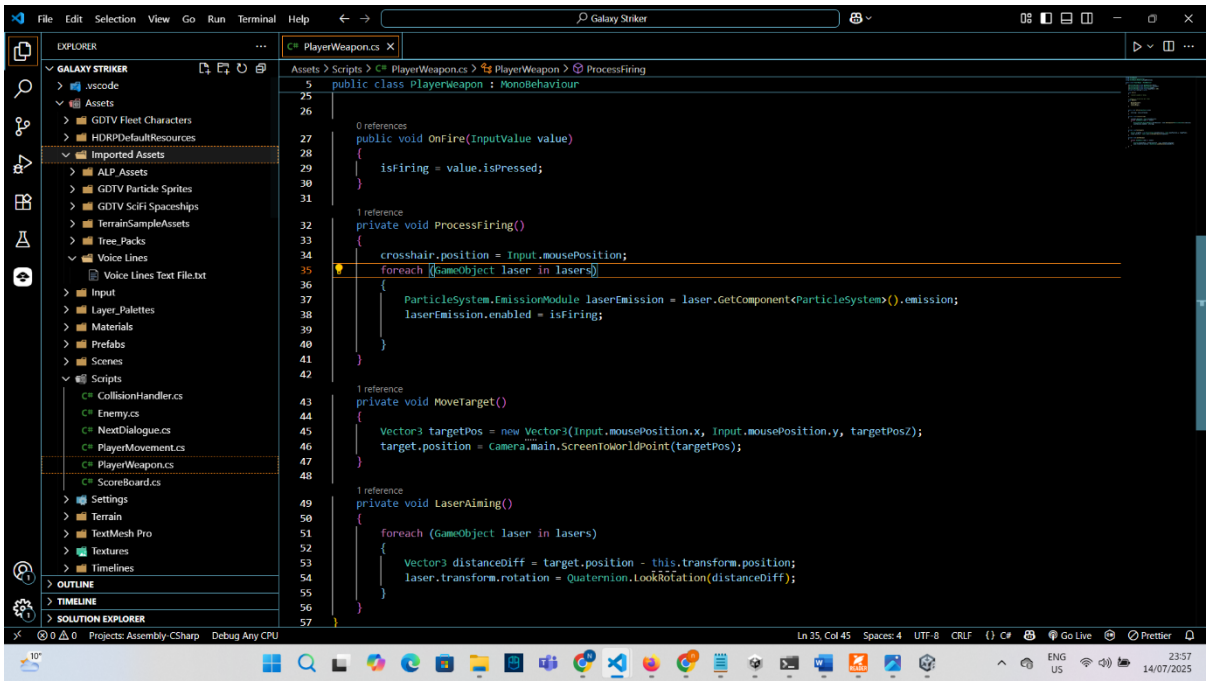
Here is the youtube clip of the game: https://youtu.be/09QRfdK5YKA

PlayerMovement.cs



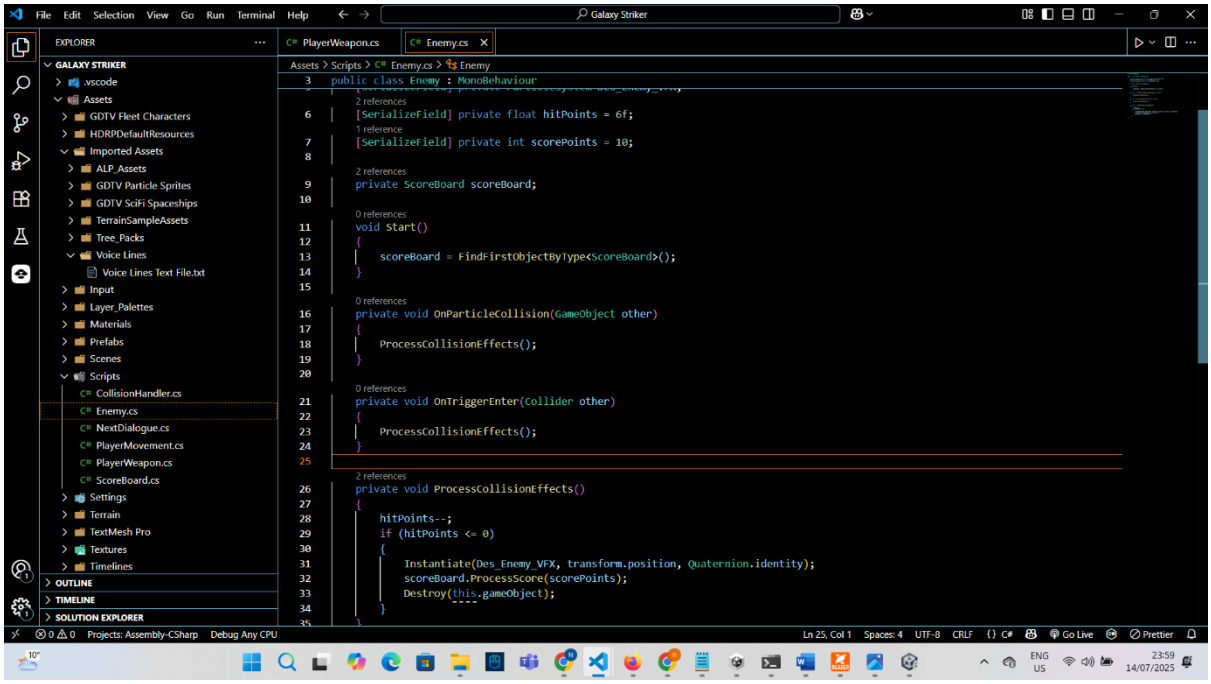
```
public class PlayerMovement : MonoBehaviour
    Vector2 movement;
    // Update is called once per frame
    void Update()
        ProcessTranslation();
        ProcessRotation();
    public void OnMove(InputValue value)
        movement = value.Get<Vector2>();
    private void ProcessTranslation()
        Vector2 Offset = movement * controlSpeed * Time.deltaTime;
        float posX = transform.localPosition.x + Offset.x;
        float clampX = Mathf.Clamp(posX, -14f, 10f);
        float posY = transform.localPosition.y + Offset.y;
        float clampY = Mathf.Clamp(posY, -6.6f, 12.4f);
        transform.localPosition = new Vector3(clampX, clampY, 0f);
    private void ProcessRotation()
        Quaternion targetRotation = Quaternion.Euler(-controlPitchParameter * movement.y, 0f, -controlRollParameter * movement.x);
        transform.localRotation = Quaternion.Lerp(transform.localRotation, targetRotation, Time.deltaTime * rollSpeed);
```

PlayerWeapon.cs



```
public class PlayerWeapon : MonoBehaviour

    0 references
    public void OnFire(InputValue value)
    {
        isFiring = value.isPressed;
    }

    1 reference
    private void ProcessFiring()
    {
        crosshair.position = Input.mousePosition;
        foreach (GameObject laser in lasers)
        {
            ParticleSystem.EmissionModule laserEmission = laser.GetComponent<ParticleSystem>().emission;
            laserEmission.enabled = isFiring;

        }
    }

    1 reference
    private void MoveTarget()
    {
        Vector3 targetPos = new Vector3(Input.mousePosition.x, Input.mousePosition.y, targetPosZ);
        target.position = Camera.main.ScreenToWorldPoint(targetPos);
    }

    1 reference
    private void LaserAiming()
    {
        foreach (GameObject laser in lasers)
        {
            Vector3 distanceDiff = target.position - this.transform.position;
            laser.transform.rotation = Quaternion.LookRotation(distanceDiff);
        }
    }
}
```

Enemy.cs



```
public class Enemy : MonoBehaviour

    2 references
    [SerializeField] private float hitPoints = 6f;
    1 reference
    [SerializeField] private int scorePoints = 10;

    2 references
    private ScoreBoard scoreBoard;

    0 references
    void Start()
    {
        scoreBoard = FindFirstObjectByType<ScoreBoard>();
    }

    0 references
    private void OnParticleCollision(GameObject other)
    {
        ProcessCollisionEffects();
    }

    0 references
    private void OnTriggerEnter(Collider other)
    {
        ProcessCollisionEffects();
    }

    2 references
    private void ProcessCollisionEffects()
    {
        hitPoints--;
        if (hitPoints <= 0)
        {
            Instantiate(Des_Enemy_VFX, transform.position, Quaternion.identity);
            scoreBoard.ProcessScore(scorePoints);
            Destroy(this.gameObject);
        }
    }
```

Unity Editor