Project Report

Spring 2025

CSE-411L Intro to Game Development

Submitted by: **Noor-ul-Huda**

Registration No.: **22PWCSE2117**

Class Section: **B**

Student Signature: Noor

Submitted to:

Engr. Abdullah Hamid

December 15, 2025

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Mini Project 1

Project Title: Crescent Isle

1. Introduction

This project, titled **"Crescent Isle"**, is a small 3D exploration game developed using the **Unity Game Engine**. The objective of the game is to allow the player to explore an island environment, collect **10 resource objects**, and avoid a hostile enemy called the **Hunter**.

The project is designed to test practical understanding of core Unity concepts such as **terrain creation, physics-based movement, prefabs, scripting, UI handling, collision detection, and basic AI behavior**, as covered in Lectures 1 to 6.

2. Objective of the Project

The main objectives of this project are:

- To design a 3D island environment using Unity Terrain tools
- To implement a controllable player using physics-based movement
- To create collectible resources that spawn one by one
- To design an AI-controlled enemy that chases the player
- To implement win and loss conditions
- To apply concepts learned in lectures through practical implementation

3. Tools & Technologies Used

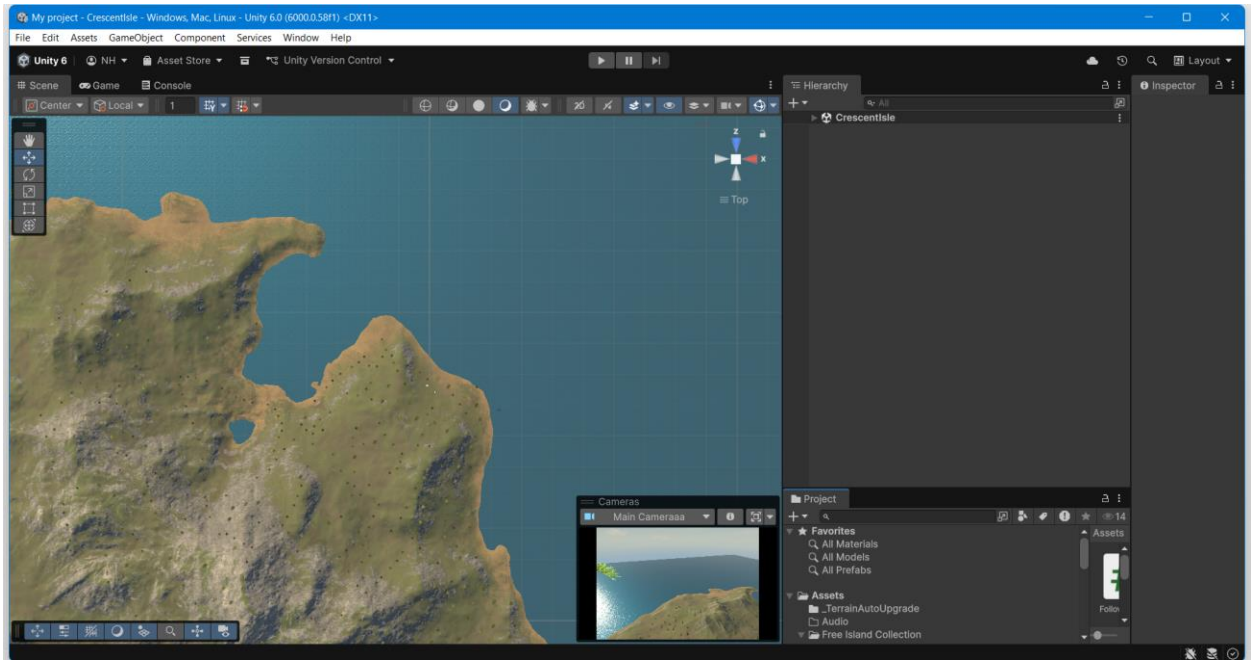
- Unity Game Engine
- C# Programming Language
- Unity Terrain System
- Unity UI System
- Physics System (Rigidbody, Colliders)
- Prefabs & GameObjects

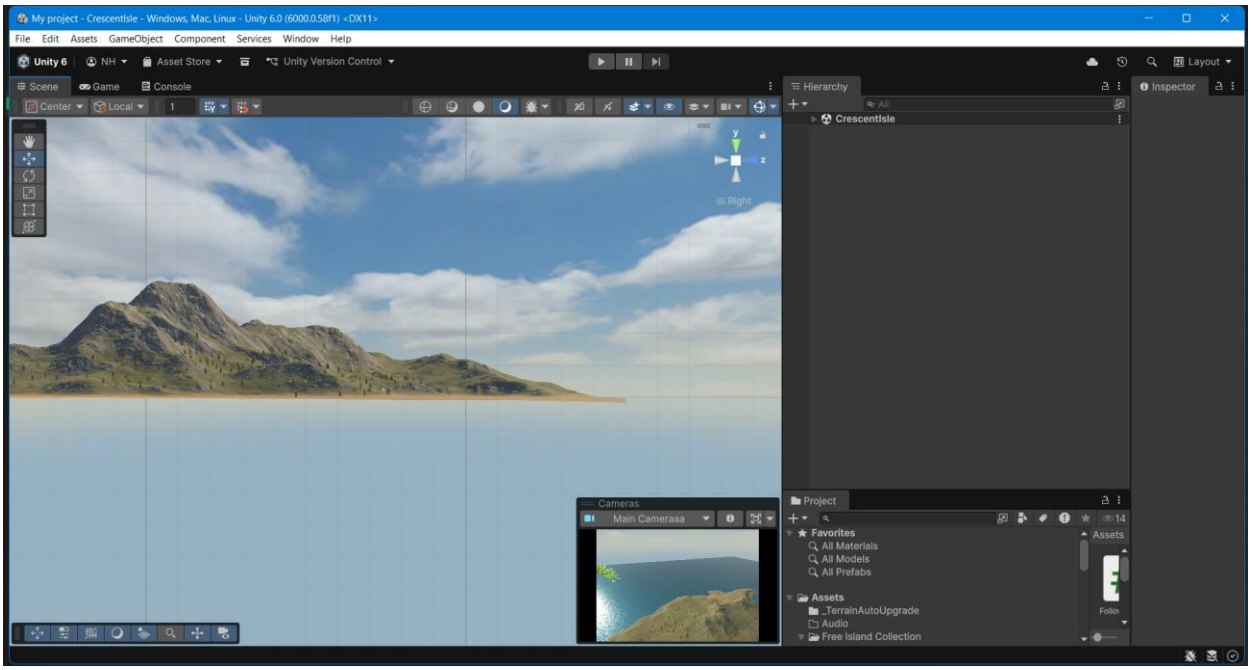
4. World Design (Terrain & Environment)

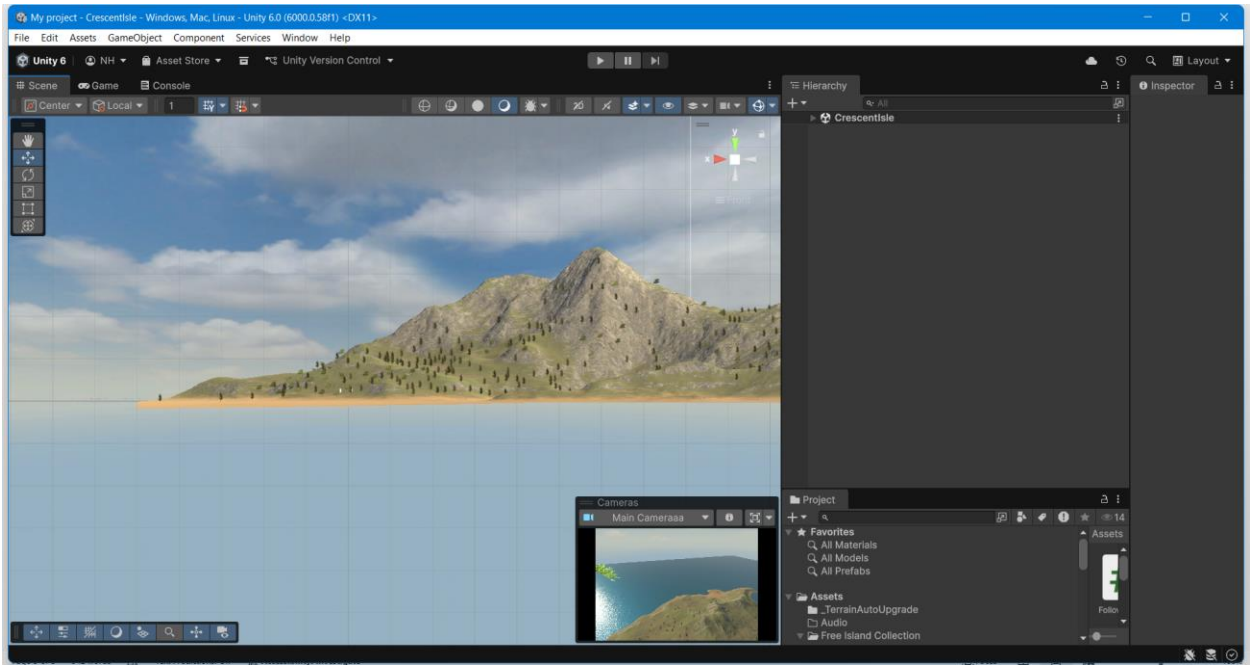
4.1 Terrain Creation

- A Unity **Terrain** was created and shaped to resemble an island.

- The edges of the terrain were raised while the center was kept lower, giving an island-like appearance.

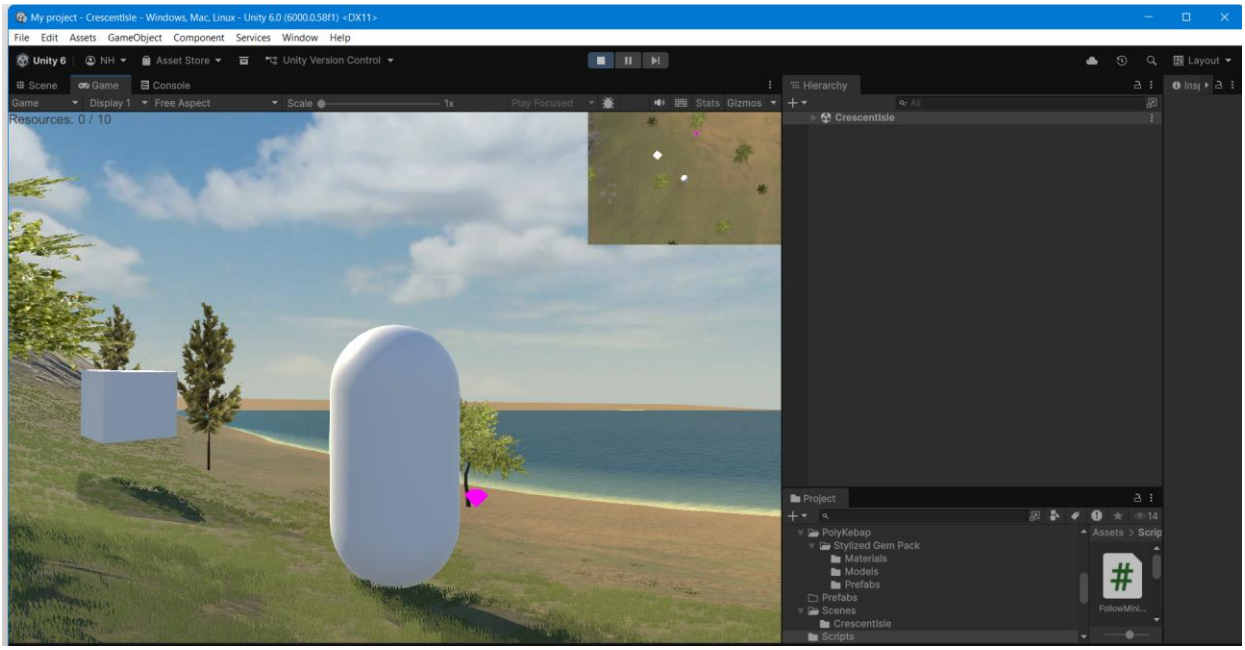


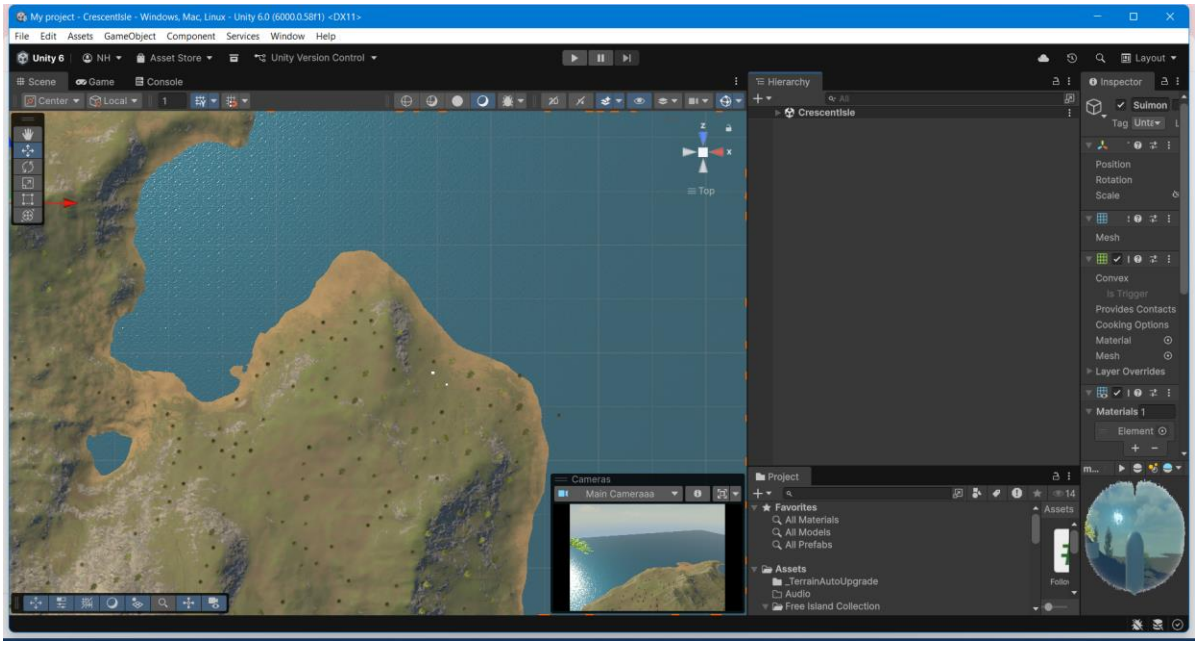




4.2 Environment Details

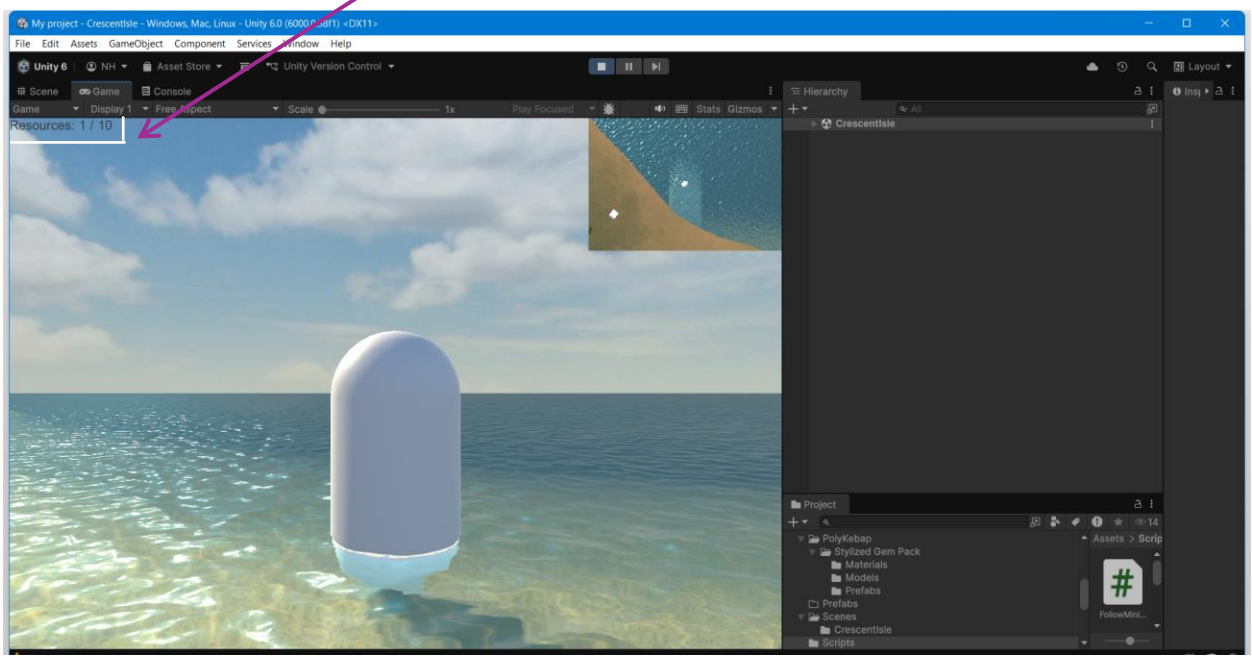
- Trees and grass were added using Unity's Terrain painting tools.
- Wind effects were applied to grass for realism.
- A sand-like material was created and applied to the terrain.
- A suitable skybox (clear sky) was added to enhance visuals.

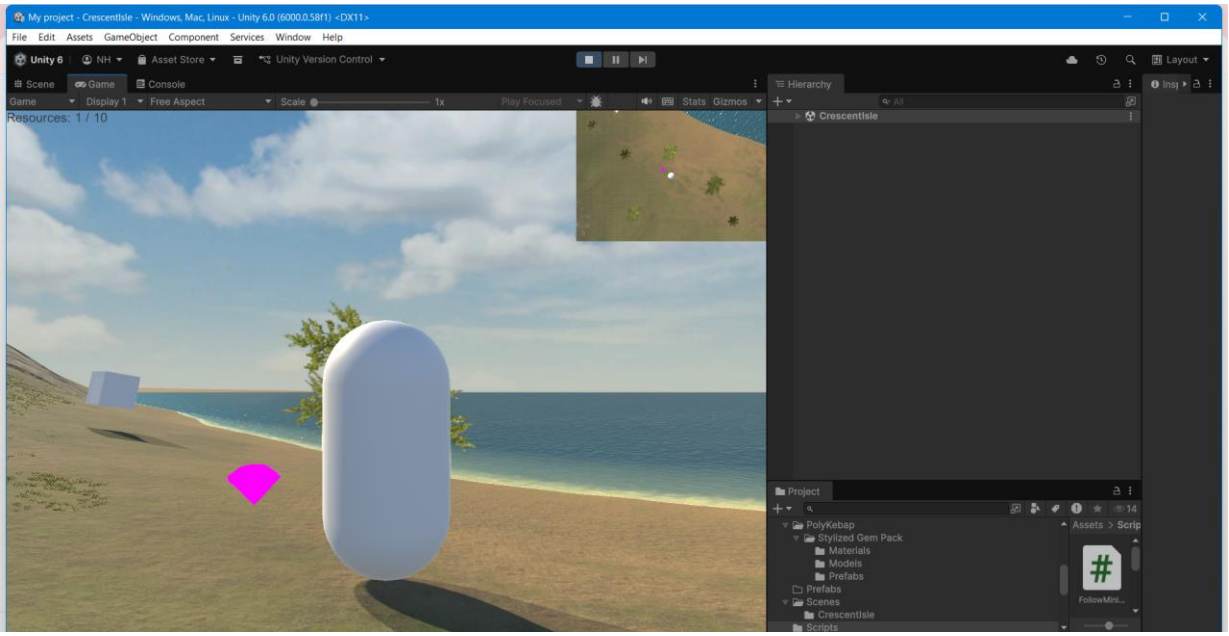




4.3 User Interface

- A **Unity UI Text** element was placed on the screen.
- The text displays the number of collected resources:
- Resources: 0 / 10

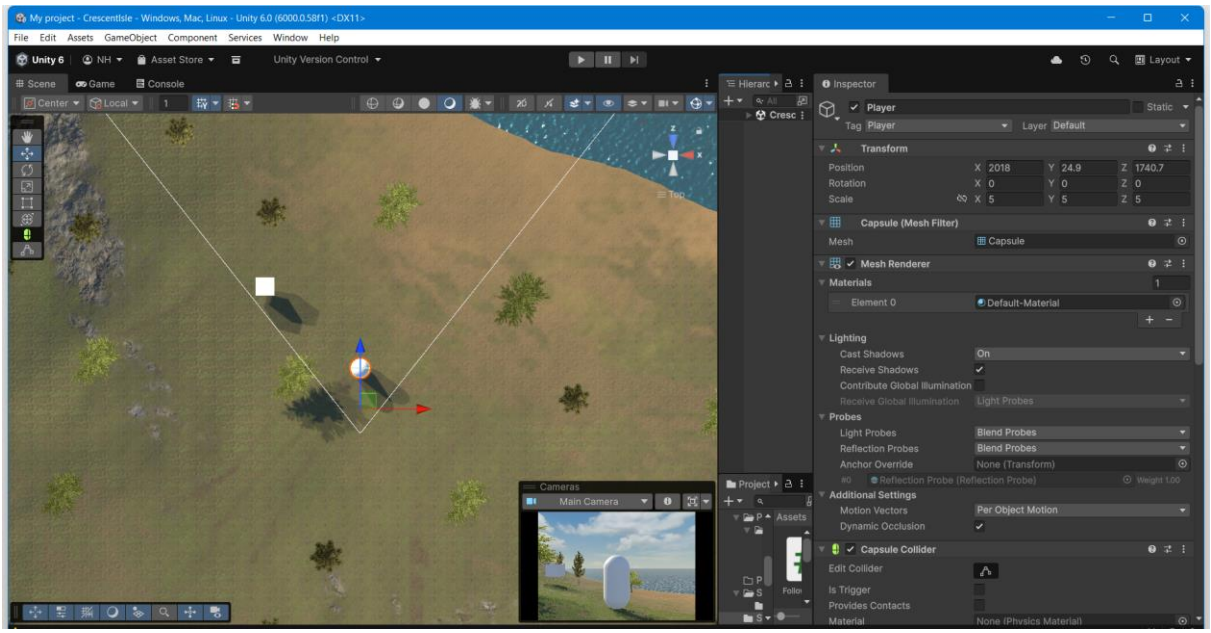




5. Player Implementation

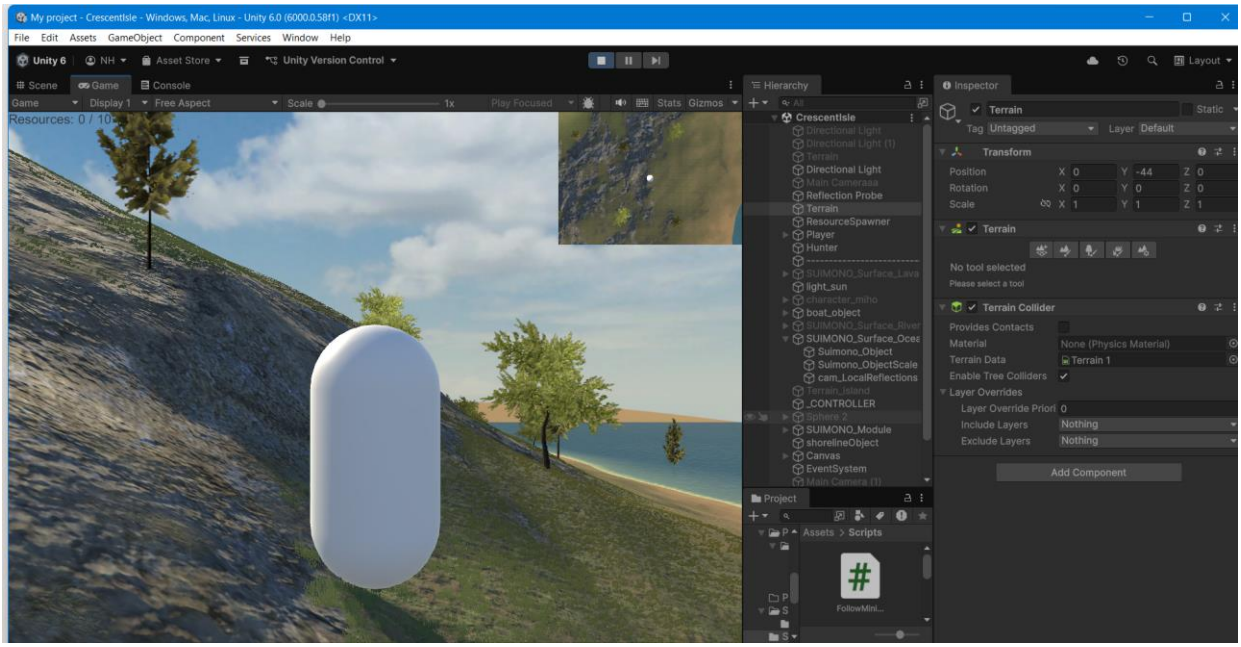
5.1 Player Setup

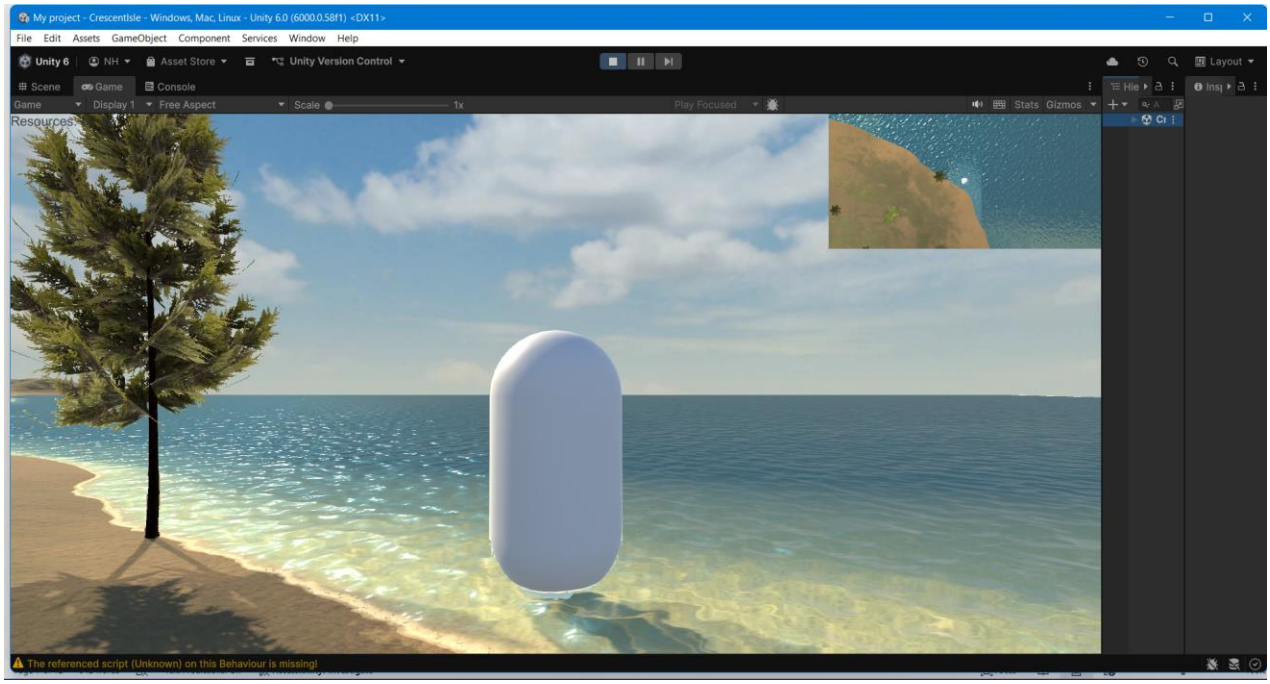
- The player is represented by a **Capsule** object.
- The object is tagged as **"Player"**.
- A **Rigidbody** component is added for physics-based movement.
- Rigidbody rotation is frozen on X and Z axes to prevent tipping.



5.2 Player Movement Logic

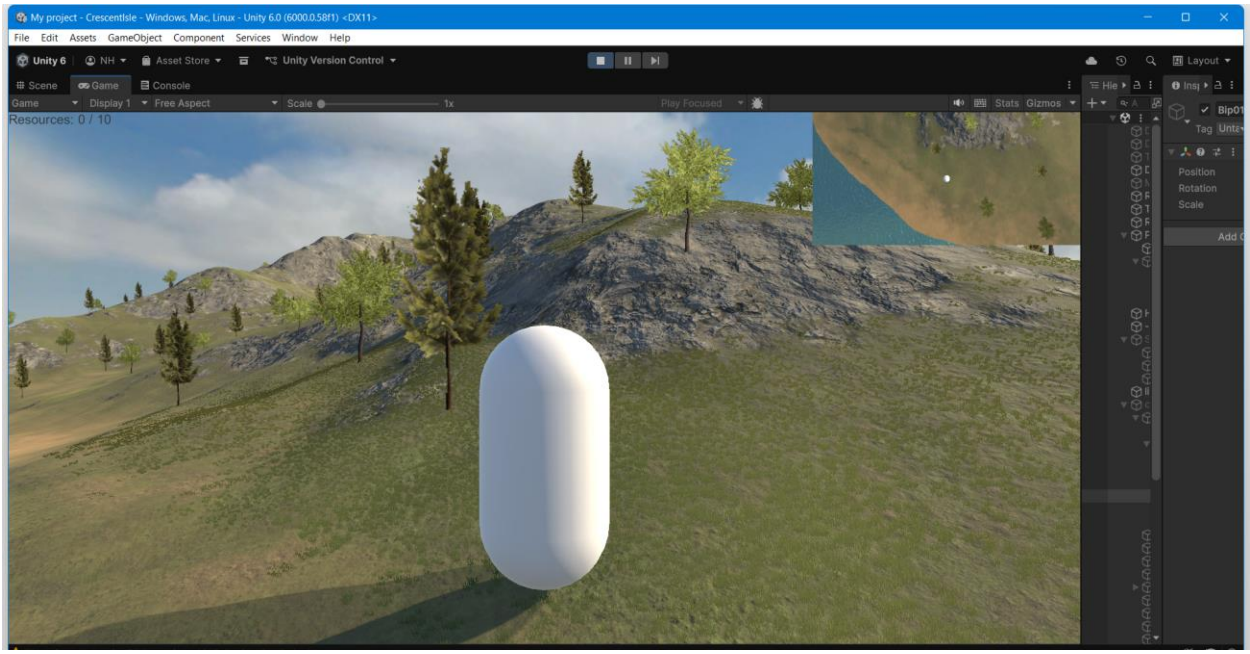
- Player movement is handled in FixedUpdate() using:
    - Input.GetAxis("Horizontal")
    - Input.GetAxis("Vertical")
- Movement is applied using Rigidbody.AddForce() to ensure realistic physics behavior.





5.3 Player Rotation

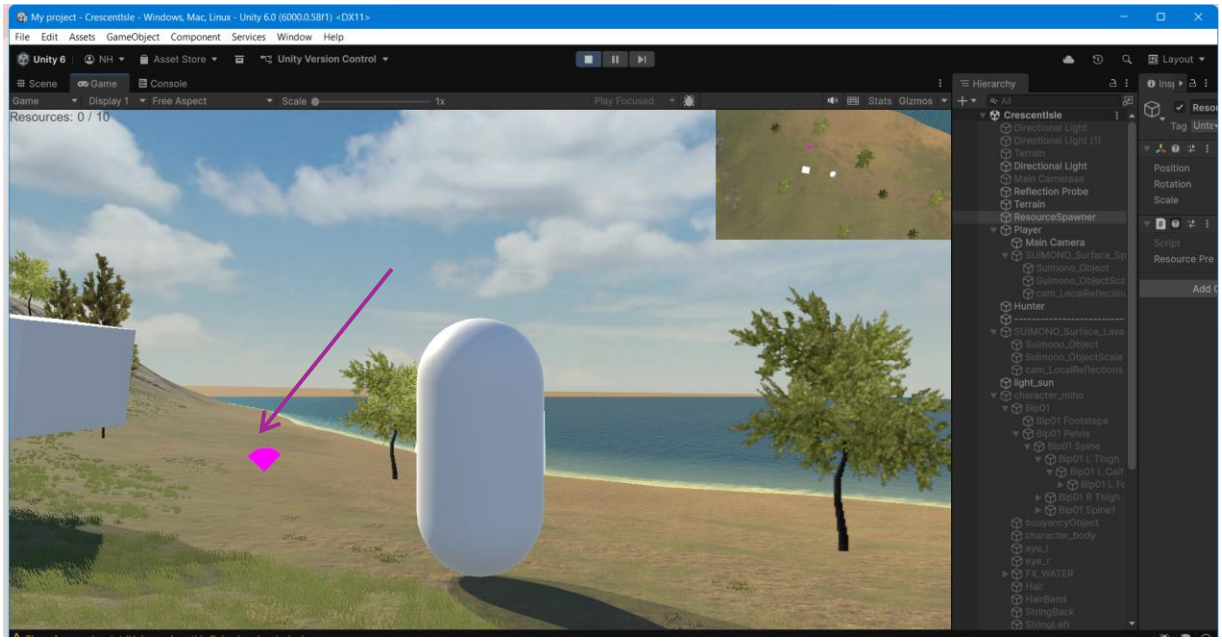
- Player rotation is handled in Update() using:
    - Input.GetKey(KeyCode.A)
    - Input.GetKey(KeyCode.D)
- Rotation uses transform.Rotate() with Time.deltaTime for smooth motion.



6. Collectibles & Resource Spawner

6.1 Resource Prefab

- A crystal object was created as a **Resource**.
- A new material was applied for visual distinction.
- A **Collider** was added and set as **Is Trigger**.
- The object was tagged as **"Resource"**.
- A rotation animation was added using a custom script.

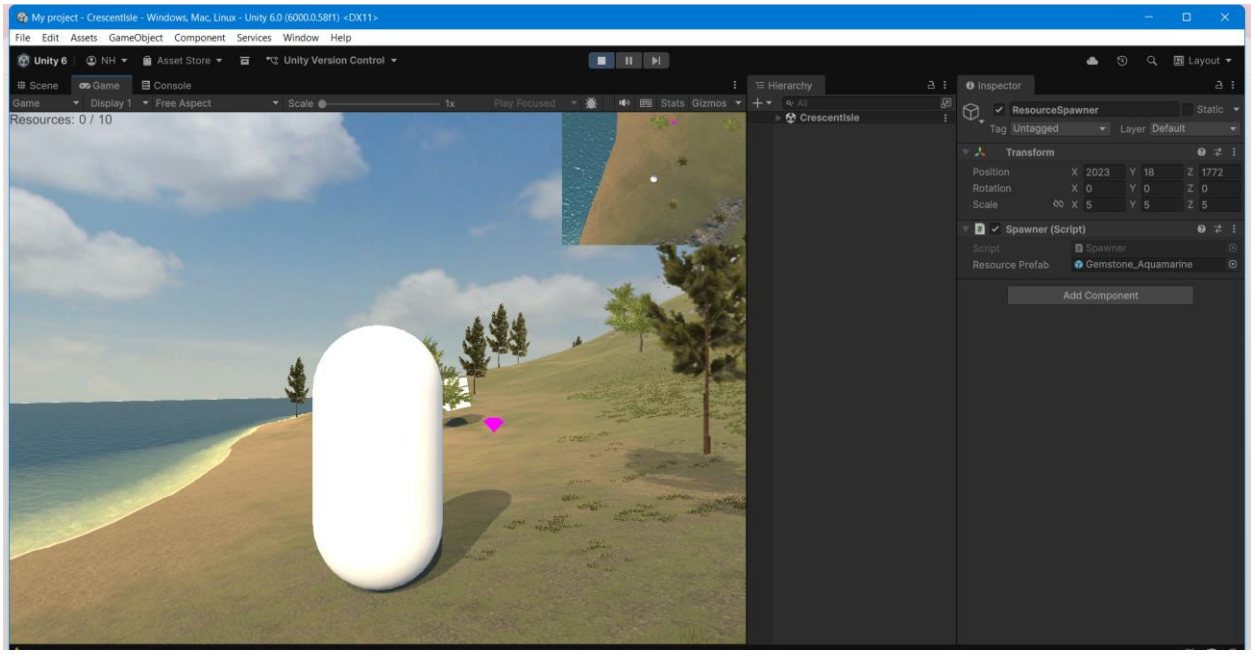


6.2 Resource Collection Logic

- Resource collection is handled using OnTriggerEnter().
- When the player collides with a resource:
    - The resource count is incremented.
    - The UI text is updated.
    - The resource object is destroyed.

6.3 Resource Spawner System

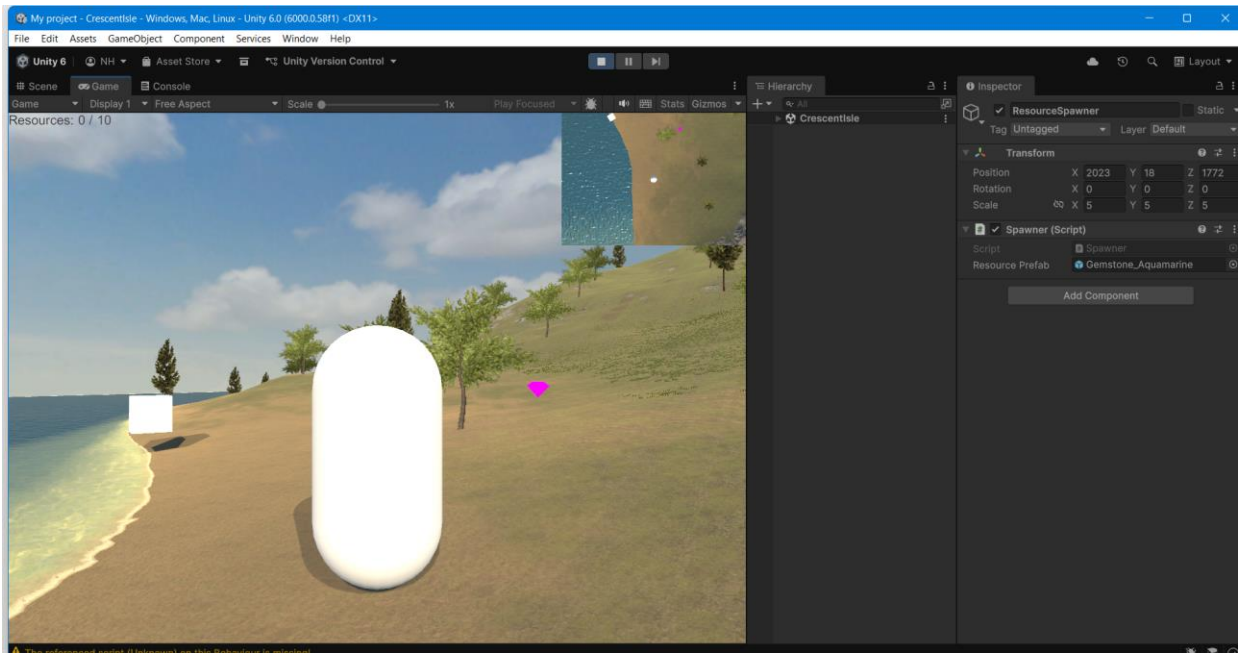
- An empty GameObject named **ResourceSpawner** was created.
- A Spawner script is attached to it.
- The resource prefab is assigned via the Inspector.
- InvokeRepeating() is used to spawn resources every 5 seconds.
- Only **one resource exists at a time**.
- Raycasting ensures the resource spawns exactly on terrain height.



7. Hunter (AI Enemy)

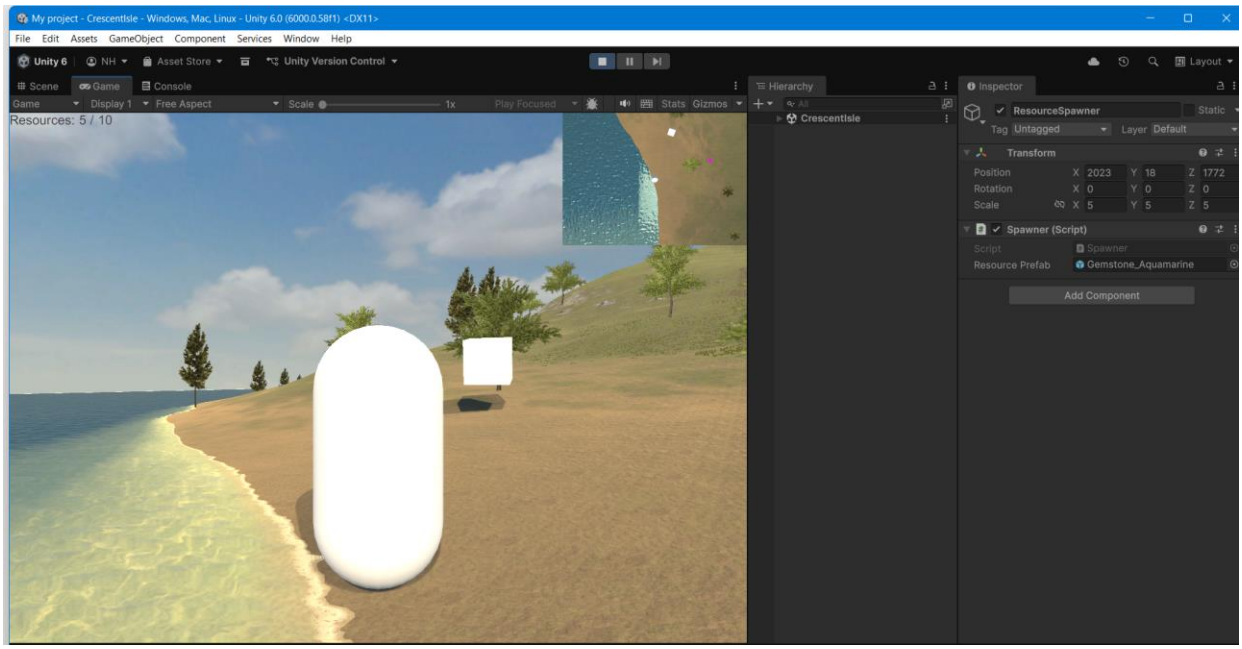
7.1 Hunter Prefab

- The Hunter is created using simple cube shapes.
- It has:
    - A **Rigidbody**
    - A **Collider**
- The object is tagged as **"Enemy"**.
- Converted into a prefab.



7.2 Hunter AI Logic

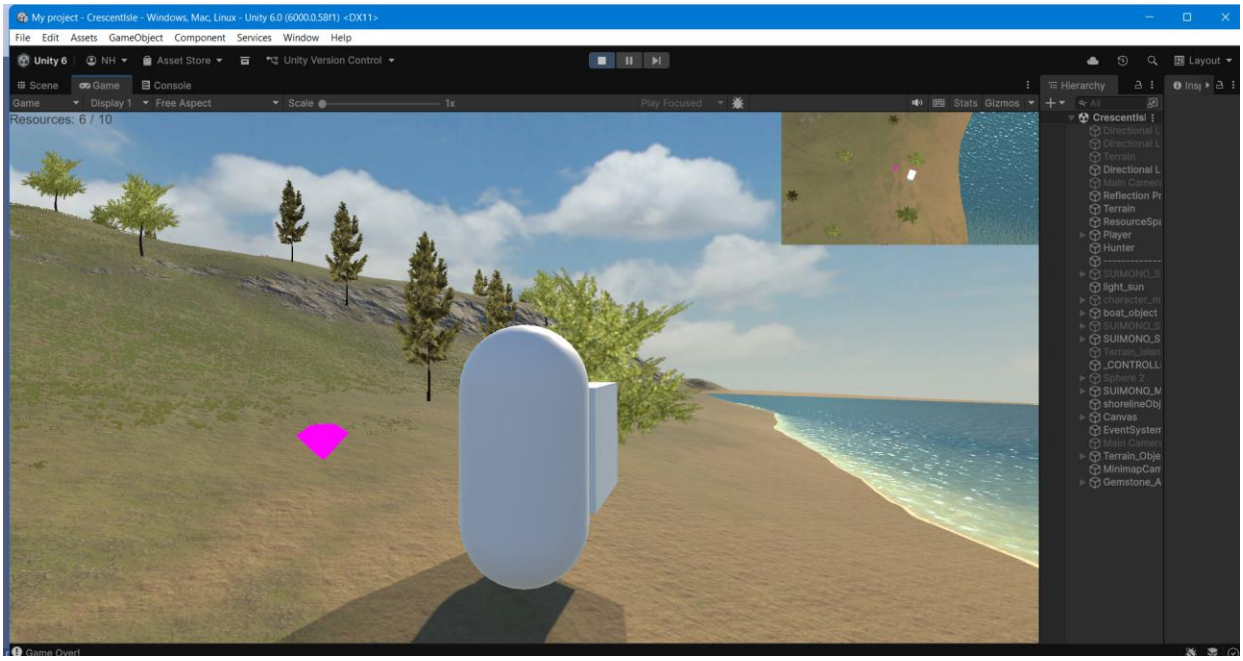
- The player's transform is found using:
- GameObject.Find("Player").transform;
- The distance between the Hunter and the player is calculated using:
- Vector3.Distance()
- If the player is within a defined range, the Hunter:
    - Moves towards the player using Vector3.MoveTowards()
    - Faces the player using transform.LookAt()



8. Win & Loss Conditions

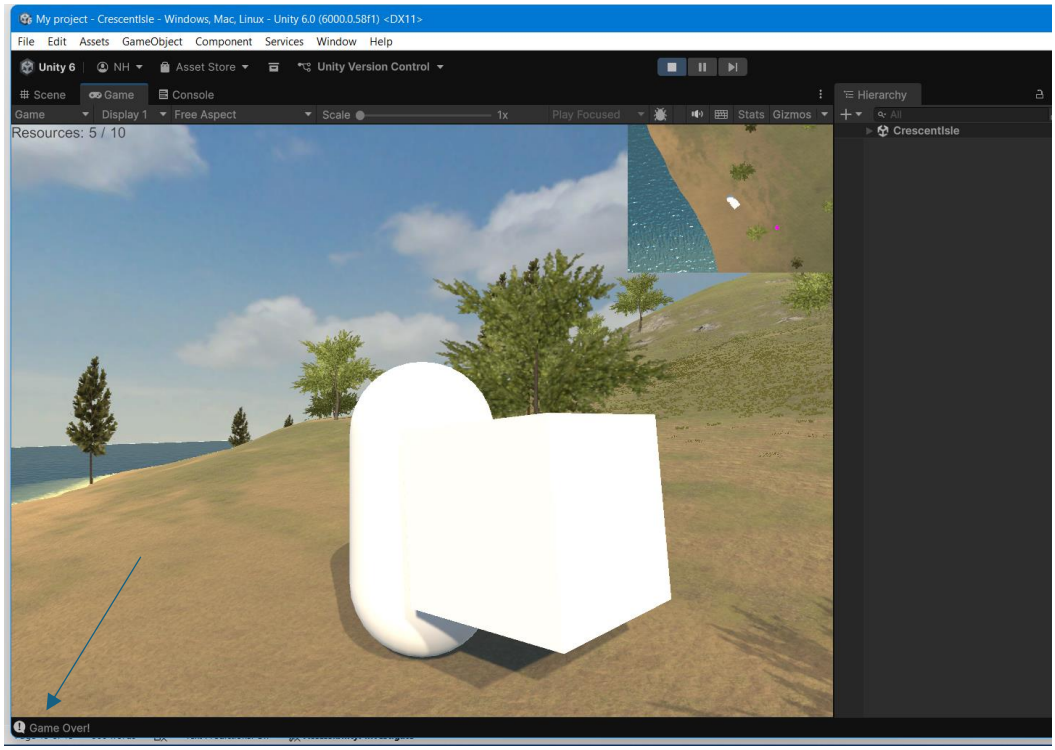
8.1 Win Condition

- When the player collects 10 resources:
    - "You Win!" is printed to the console.
    - Time.timeScale = 0 freezes the game.



8.2 Loss Condition

- When the Hunter collides with the player:
    - "Game Over!" is printed to the console.
    - The game is frozen using Time.timeScale = 0.

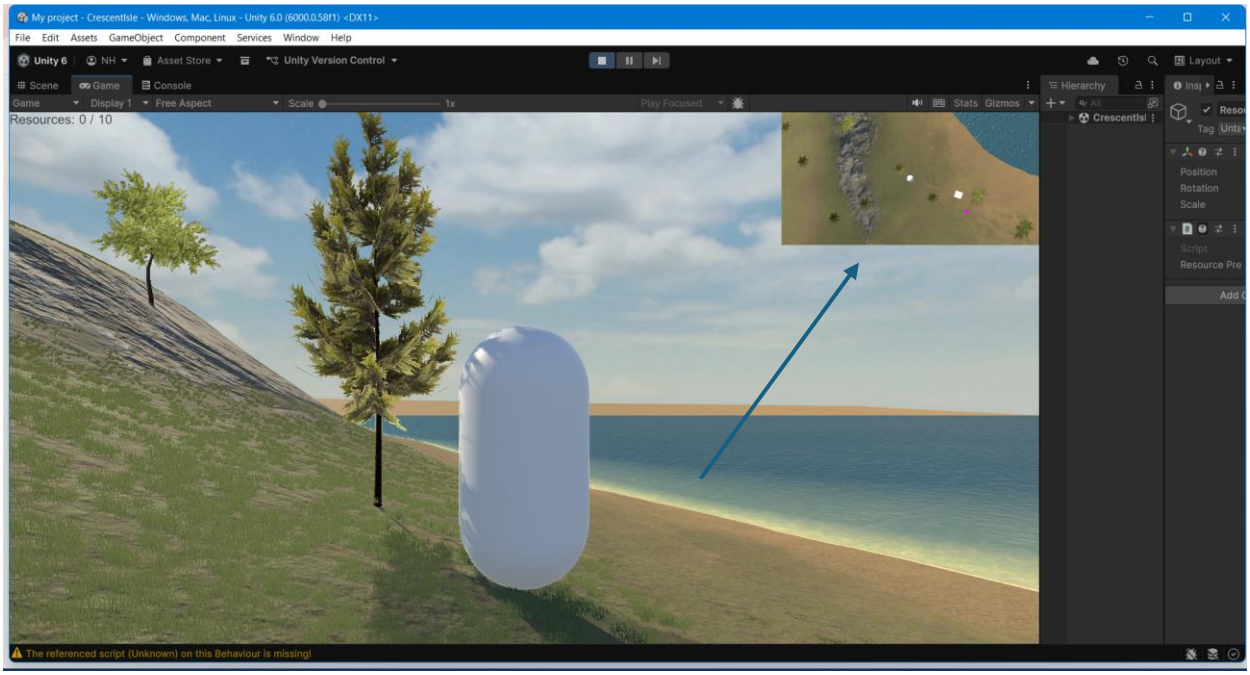


9. Mini Map System (Additional Feature)

- A top-down camera was added for the minimap.
- The camera follows the player's position and rotation.
- The minimap behaves similarly to GTA-style minimaps.

FollowMinimap Script

- Uses LateUpdate() to ensure smooth camera following.
- Keeps camera height fixed.
- Matches player's Y-axis rotation.



10. Audio & Sound Effects (Additional Feature)

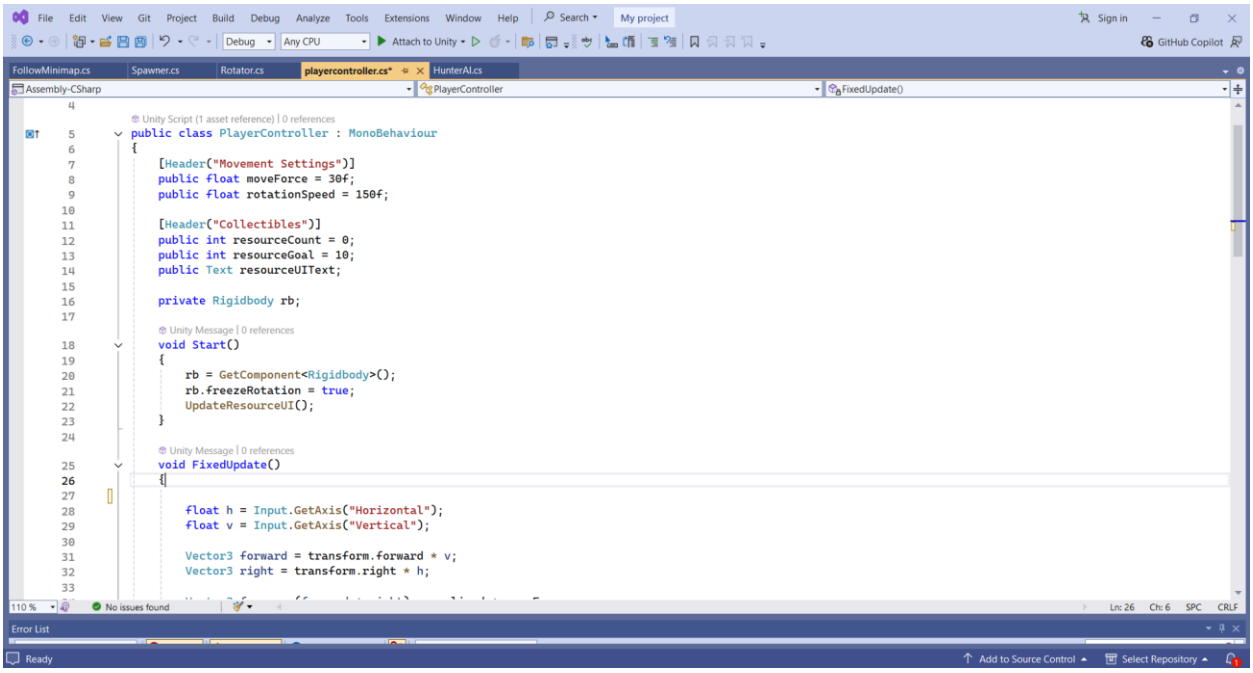
To enhance realism and immersion, audio effects were added to the game environment.

Water Sound Effect

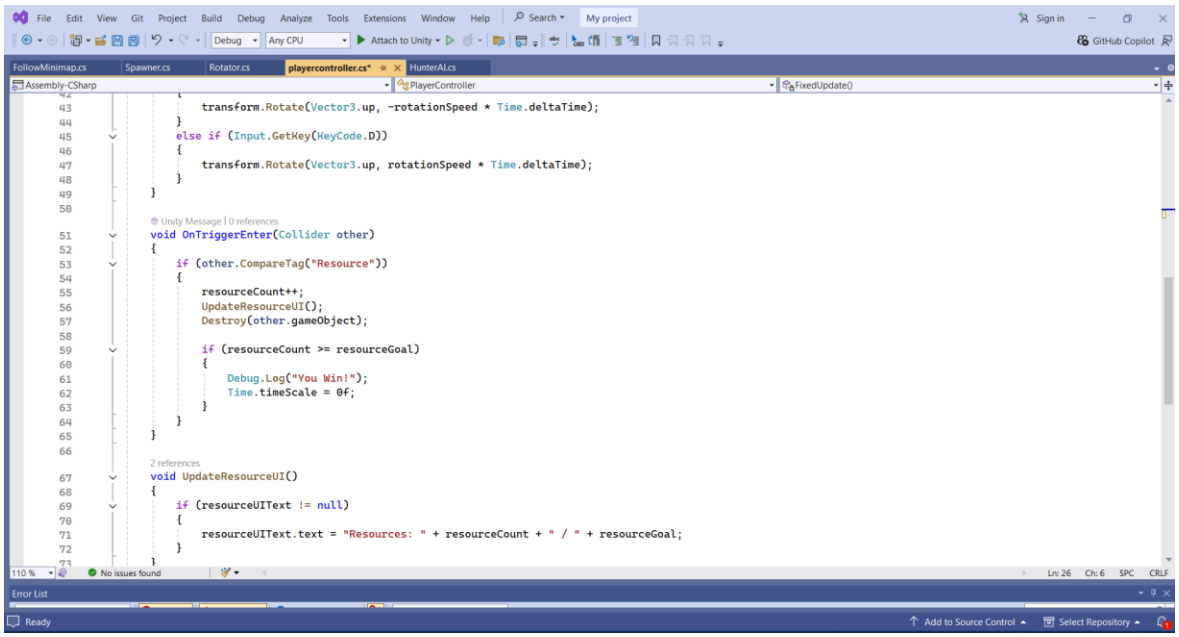
- A water ambient sound was added near the island's surrounding water.
- An Audio Source component was attached to the water GameObject.
- The sound was set to:
    - Loop (for continuous playback)
    - Spatial Blend = 3D (so sound changes based on player position)
- This creates a realistic ocean-like environment while the player explores the island.

11. Scripts

Playercontroller.cs

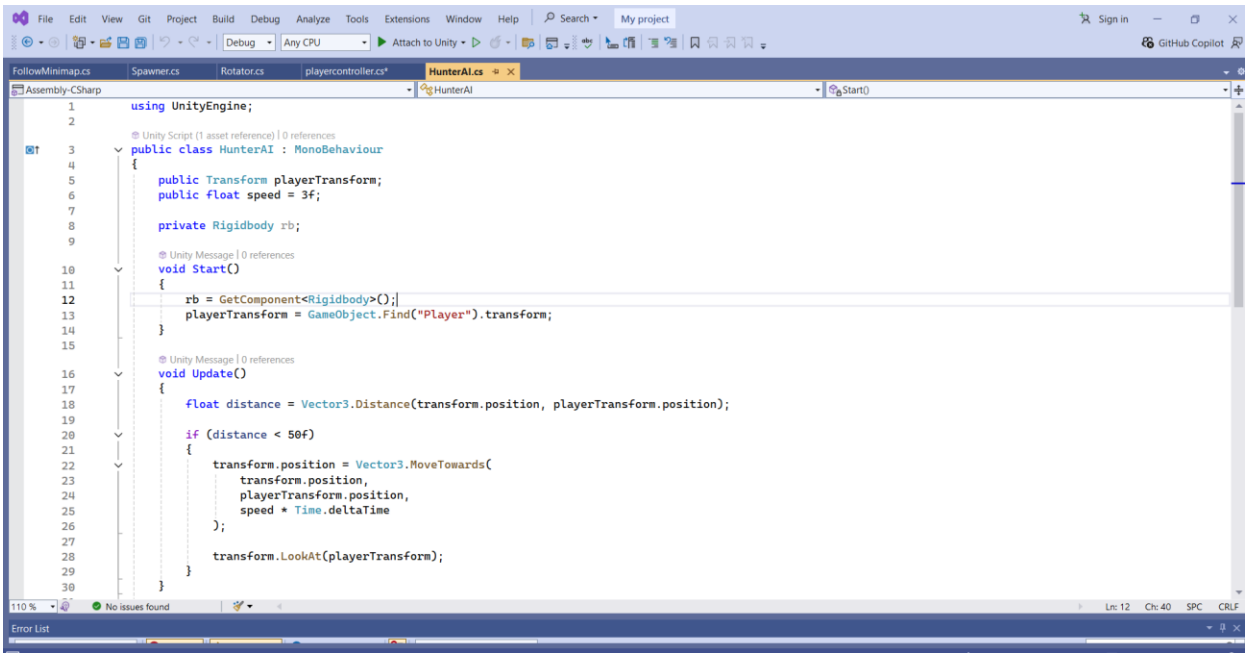


```
Unity Script (1 asset reference) | 0 references
public class PlayerController : MonoBehaviour
{
    [Header("Movement Settings")]
    public float moveForce = 30f;
    public float rotationSpeed = 150f;

    [Header("Collectibles")]
    public int resourceCount = 0;
    public int resourceGoal = 10;
    public Text resourceUIText;

    private Rigidbody rb;

    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        rb.freezeRotation = true;
        UpdateResourceUI();
    }

    Unity Message | 0 references
    void FixedUpdate()
    {

        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");

        Vector3 forward = transform.forward * v;
        Vector3 right = transform.right * h;
```



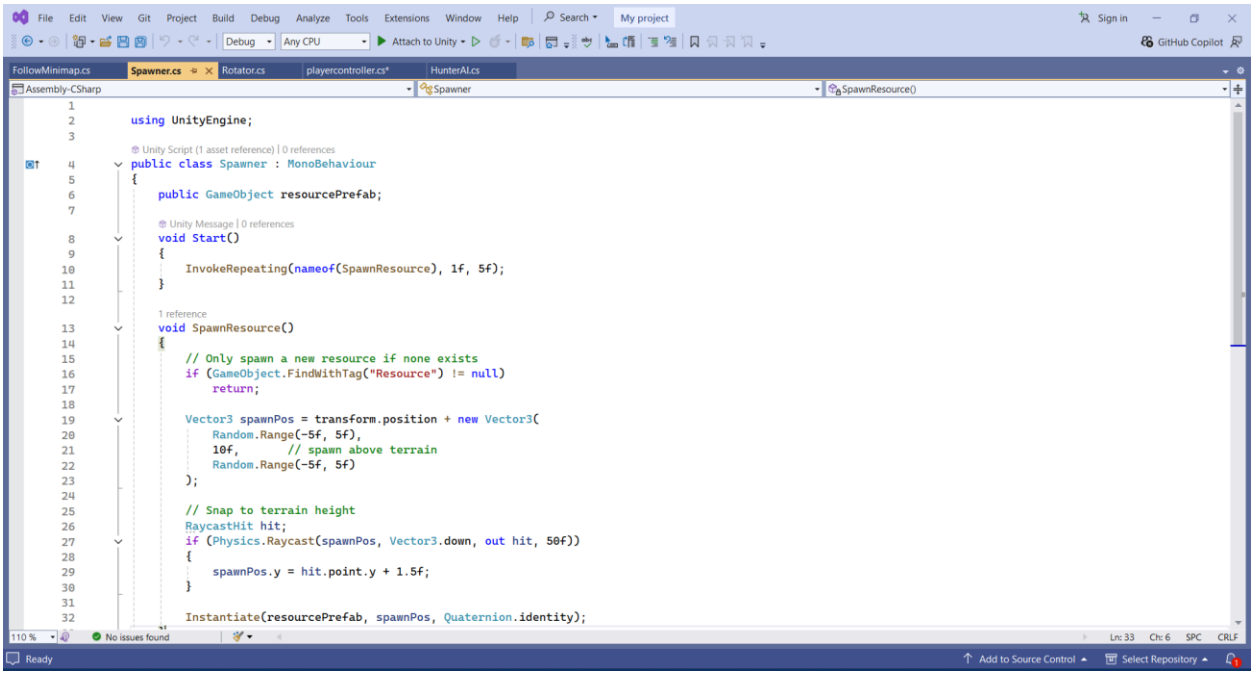
```
            transform.Rotate(Vector3.up, -rotationSpeed * Time.deltaTime);
        }
        else if (Input.GetKey(KeyCode.D))
        {
            transform.Rotate(Vector3.up, rotationSpeed * Time.deltaTime);
        }
    }

    Unity Message | 0 references
    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Resource"))
        {
            resourceCount++;
            UpdateResourceUI();
            Destroy(other.gameObject);

            if (resourceCount >= resourceGoal)
            {
                Debug.Log("You Win!");
                Time.timeScale = 0f;
            }
        }
    }

    2 references
    void UpdateResourceUI()
    {
        if (resourceUIText != null)
        {
            resourceUIText.text = "Resources: " + resourceCount + " / " + resourceGoal;
        }
    }
```

HunterAI.cs



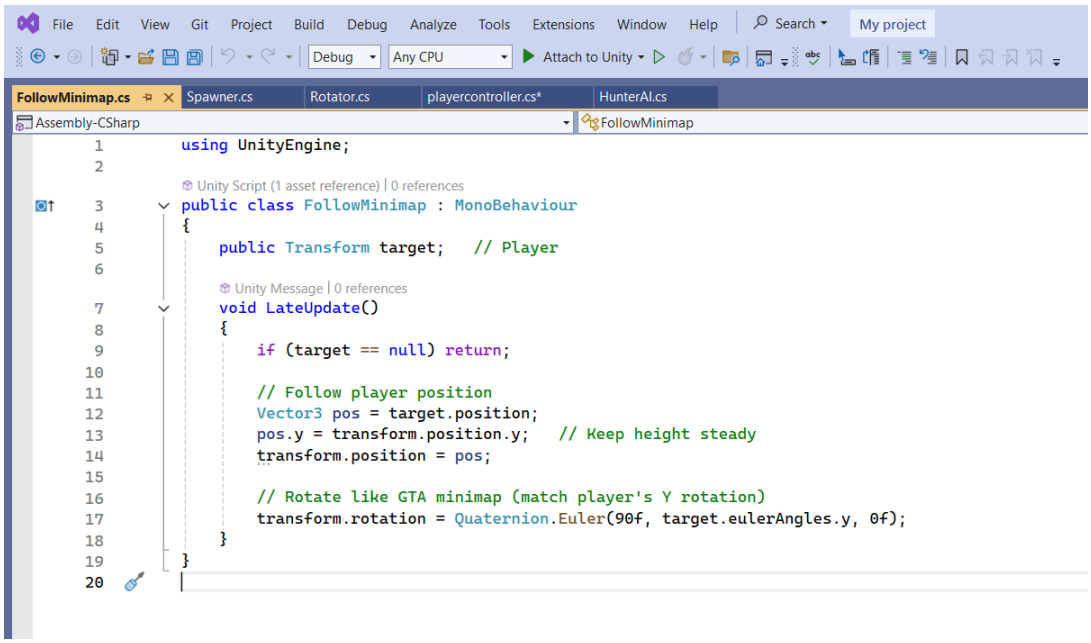
```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class HunterAI : MonoBehaviour
{
    public Transform playerTransform;
    public float speed = 3f;

    private Rigidbody rb;

    Unity Message | 0 references
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        playerTransform = GameObject.Find("Player").transform;
    }

    Unity Message | 0 references
    void Update()
    {
        float distance = Vector3.Distance(transform.position, playerTransform.position);

        if (distance < 50f)
        {
            transform.position = Vector3.MoveTowards(
                transform.position,
                playerTransform.position,
                speed * Time.deltaTime
            );

            transform.LookAt(playerTransform);
        }
    }
```

Spawner.cs

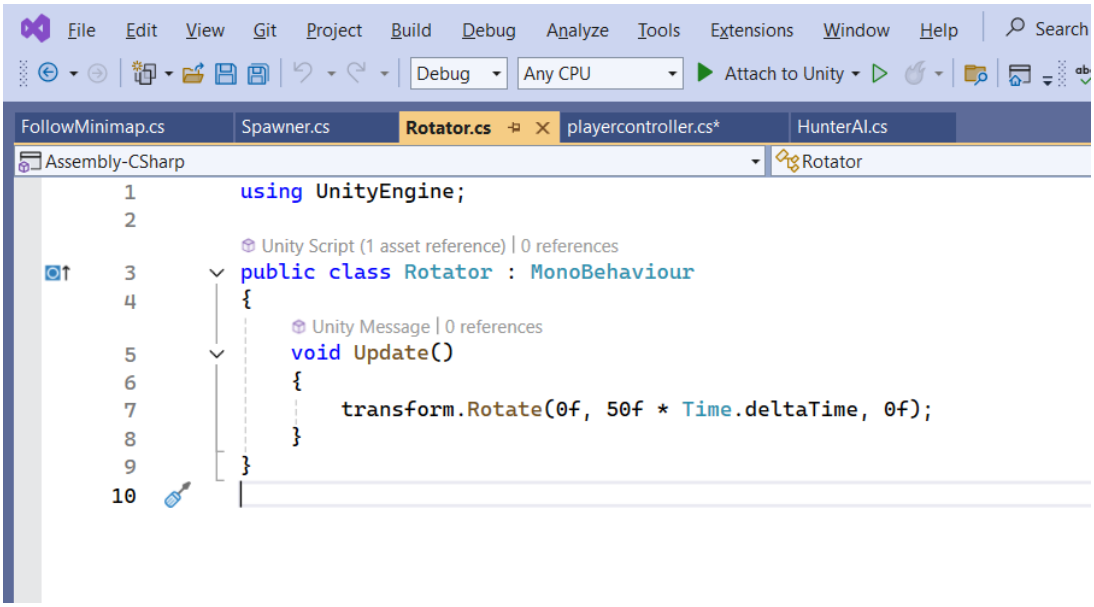


```
using UnityEngine;

Unity Script (1 asset reference) | 0 references
public class Spawner : MonoBehaviour
{
    public GameObject resourcePrefab;

    Unity Message | 0 references
    void Start()
    {
        InvokeRepeating(nameof(SpawnResource), 1f, 5f);
    }

    1 reference
    void SpawnResource()
    {
        // Only spawn a new resource if none exists
        if (GameObject.FindWithTag("Resource") != null)
            return;

        Vector3 spawnPos = transform.position + new Vector3(
            Random.Range(-5f, 5f),
            10f,        // spawn above terrain
            Random.Range(-5f, 5f)
        );

        // Snap to terrain height
        RaycastHit hit;
        if (Physics.Raycast(spawnPos, Vector3.down, out hit, 50f))
        {
            spawnPos.y = hit.point.y + 1.5f;
        }

        Instantiate(resourcePrefab, spawnPos, Quaternion.identity);
```

FollowMinimap.cs


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
My project
Debug
Any CPU
Attach to Unity
FollowMinimap.cs
Spawner.cs
Rotator.cs
playercontroller.cs*
HunterAI.cs
Assembly-CSharp
FollowMinimap
using UnityEngine;
Unity Script (1 asset reference) | 0 references
public class FollowMinimap : MonoBehaviour
{
public Transform target; // Player
Unity Message | 0 references
void LateUpdate()
{
if (target == null) return;
// Follow player position
Vector3 pos = target.position;
pos.y = transform.position.y; // Keep height steady
transform.position = pos;
// Rotate like GTA minimap (match player's Y rotation)
transform.rotation = Quaternion.Euler(90f, target.eulerAngles.y, 0f);
}
}

Rotator.cs


File Edit View Git Project Build Debug Analyze Tools Extensions Window Help
Search
Debug
Any CPU
Attach to Unity
FollowMinimap.cs
Spawner.cs
Rotator.cs
playercontroller.cs*
HunterAI.cs
Assembly-CSharp
Rotator
using UnityEngine;
Unity Script (1 asset reference) | 0 references
public class Rotator : MonoBehaviour
{
Unity Message | 0 references
void Update()
{
transform.Rotate(0f, 50f * Time.deltaTime, 0f);
}
}

12. Conclusion

The **Crescent Isle** project successfully demonstrates practical implementation of Unity game development fundamentals. In addition to fulfilling all required tasks, environmental audio effects were integrated to enhance immersion.

The project covers terrain design, physics-based player movement, collectible systems, AI behavior, UI updates, win/loss logic, and audio integration, making it a complete and functional mini-game aligned with the course learning outcomes.

13. Future Improvements

- Add animations to the player
- Enhance UI with health bar and score screen