Part 2: Creating and Optimizing a Simple FPS Game

This section guides you through creating and optimizing a simple first-person shooter (FPS) game in Unity, using .NET and C#. The demo includes player movement, shooting mechanics, enemy AI, and a scoring system. Key optimizations are included at each step to ensure efficient performance.

Step 1: Setting Up the Project

1. Create a New 3D Unity Project.

2. Set up a basic 3D environment with floors, walls, and obstacles to form a simple arena.

3. Add lighting and a camera to follow the player.

Step 2: Player Movement and Mouse Look

Implement player movement and mouse look mechanics using the following script.

PlayerMovement.cs:

```
using UnityEngine;


public class PlayerMovement : MonoBehaviour
{
    public float moveSpeed = 5f;
    public float mouseSensitivity = 2f;
    private Rigidbody rb;
    private float rotationY = 0f;


    void Start()
    {
        rb = GetComponent<Rigidbody>();
        Cursor.lockState = CursorLockMode.Locked;
    }
```

```
    void Update()
    {
        HandleMovement();
        HandleMouseLook();
    }

    void HandleMovement()
    {
        float moveX = Input.GetAxis("Horizontal") * moveSpeed;
        float moveZ = Input.GetAxis("Vertical") * moveSpeed;
        Vector3 move = transform.right * moveX + transform.forward * moveZ;
        rb.MovePosition(rb.position + move * Time.deltaTime);
    }

    void HandleMouseLook()
    {
        float mouseX = Input.GetAxis("Mouse X") * mouseSensitivity;
        float mouseY = Input.GetAxis("Mouse Y") * mouseSensitivity;

        rotationY -= mouseY;
        rotationY = Mathf.Clamp(rotationY, -90f, 90f);
          transform.localRotation = Quaternion.Euler(rotationY, transform.eulerAngles.y + mouseX,
0f);
    }
}
```

Step 3: Implementing Shooting Mechanics with Object Pooling

Implement shooting mechanics using object pooling to optimize bullet handling.

PlayerShooting.cs:

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

```
public class PlayerShooting : MonoBehaviour
{
    public GameObject bulletPrefab;
    public Transform firePoint;
    public int poolSize = 20;
    private List<GameObject> bulletPool;
    public float bulletSpeed = 20f;
    private float shootCooldown = 0.2f;
    private float lastShotTime;

    void Start()
    {
        bulletPool = new List<GameObject>();
        for (int i = 0; i < poolSize; i++)
        {
            GameObject bullet = Instantiate(bulletPrefab);
            bullet.SetActive(false);
            bulletPool.Add(bullet);
        }
    }

    void Update()
    {
        if (Input.GetButtonDown("Fire1") && Time.time > lastShotTime + shootCooldown)
        {
            lastShotTime = Time.time;
            Shoot();
        }
    }

    void Shoot()
    {
```

```
        GameObject bullet = GetPooledBullet();
        if (bullet != null)
        {
            bullet.transform.position = firePoint.position;
            bullet.transform.rotation = firePoint.rotation;
            bullet.SetActive(true);
            bullet.GetComponent<Rigidbody>().velocity = firePoint.forward * bulletSpeed;
        }
    }

    GameObject GetPooledBullet()
    {
        foreach (var bullet in bulletPool)
        {
            if (!bullet.activeInHierarchy)
            {
                return bullet;
            }
        }
        return null;
    }
}
```

Step 4: Enemy AI and Health System

Create simple enemy AI and health systems.

EnemyAI.cs:

```
using UnityEngine;

public class EnemyAI : MonoBehaviour
{
    public Transform player;
```

```
    public float speed = 3f;
    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void Update()
    {
        Vector3 direction = (player.position - transform.position).normalized;
        rb.MovePosition(rb.position + direction * speed * Time.deltaTime);
    }
}
```

EnemyHealth.cs:

```
using UnityEngine;

public class EnemyHealth : MonoBehaviour
{
    public int maxHealth = 100;
    private int currentHealth;

    void Start()
    {
        currentHealth = maxHealth;
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Bullet"))
        {
            TakeDamage(20);
```

```
            collision.gameObject.SetActive(false);
        }
    }

    void TakeDamage(int amount)
    {
        currentHealth -= amount;
        if (currentHealth <= 0)
        {
            Destroy(gameObject);
        }
    }
}
```

Step 5: Implementing a Simple Score System

Add a score system to track player progress.

GameManager.cs:

```
using UnityEngine;
using UnityEngine.UI;

public class GameManager : MonoBehaviour
{
    public Text scoreText;
    private int score = 0;

    public void IncreaseScore(int amount)
    {
        score += amount;
        scoreText.text = "Score: " + score;
    }
}
```

Integration in EnemyHealth.cs:

```
public GameManager gameManager; // Reference to GameManager


void Start()
{
    currentHealth = maxHealth;
    gameManager = FindObjectOfType<GameManager>(); // Find GameManager in the scene
}


void TakeDamage(int amount)
{
    currentHealth -= amount;
    if (currentHealth <= 0)
    {
        gameManager.IncreaseScore(10); // Add score when enemy is defeated
        Destroy(gameObject);
    }
}
```