using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Sphcontrol : MonoBehaviour

{

    public float speed = 5f;               // Movement speed

    public float jumpForce = 5f;          // Force applied for jumping

    public GameObject plane;              // Reference to the plane GameObject

    public GameObject projectilePrefab;   // Prefab for the sphere to shoot

    public float projectileSpeed = 10f;   // Speed of the projectile

    public Transform shootPoint;          // Point from which the projectile will be shot

    private Vector3 planeBoundsMin;       // Calculated minimum bounds

    private Vector3 planeBoundsMax;       // Calculated maximum bounds

    private Rigidbody rb;                 // Rigidbody component

    private bool isGrounded = true;       // Check if the sphere is on the ground

    private Renderer sphereRenderer;      // Renderer component for the sphere

    void Start()

    {

        // Get the Rigidbody component

        rb = GetComponent<Rigidbody>();

        // Get the Renderer component

        sphereRenderer = GetComponent<Renderer>();

        // Calculate the bounds based on the plane's size and position

        if (plane != null)

        {

            Vector3 planeSize = plane.GetComponent<Renderer>().bounds.size;

            Vector3 planeCenter = plane.transform.position;

            planeBoundsMin = planeCenter - (planeSize / 2);

            planeBoundsMax = planeCenter + (planeSize / 2);

        }

        else

        {

            Debug.LogError("Plane GameObject is not assigned!");

        }

    }

    void Update()

    {

        // Get keyboard input

        float moveHorizontal = Input.GetAxis("Horizontal");

        float moveVertical = Input.GetAxis("Vertical");

        // Calculate movement direction

        Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

        // Apply movement using Rigidbody

        Vector3 newPosition = rb.position + movement \* speed \* Time.deltaTime;

        // Clamp the position within the plane's bounds

        newPosition.x = Mathf.Clamp(newPosition.x, planeBoundsMin.x, planeBoundsMax.x);

        newPosition.z = Mathf.Clamp(newPosition.z, planeBoundsMin.z, planeBoundsMax.z);

        // Move the sphere

        rb.MovePosition(newPosition);

        // Handle jumping

        if (Input.GetKeyDown(KeyCode.Space) && isGrounded)

        {

            rb.AddForce(Vector3.up \* jumpForce, ForceMode.Impulse);

            isGrounded = false;

            // Change color on jump

            ChangeColor();

        }

        // Handle shooting

        if (Input.GetMouseButtonDown(0)) // Left mouse button

        {

            ShootProjectile();

        }

    }

    void OnCollisionEnter(Collision collision)

    {

        if (collision.gameObject.CompareTag("Plane"))

        {

            isGrounded = true;

        }

    }

    void ChangeColor()

    {

        // Generate a random color

        Color randomColor = new Color(Random.value, Random.value, Random.value);

        // Apply the random color to the sphere's material

        if (sphereRenderer != null)

        {

            sphereRenderer.material.color = randomColor;

        }

    }

    void ShootProjectile()

    {

        if (projectilePrefab != null)

        {

            // Check if shootPoint is set; if not, use the sphere's position

            Vector3 shootOrigin = shootPoint != null ? shootPoint.position : transform.position + transform.forward \* 1.5f;

            // Create a new projectile

            GameObject projectile = Instantiate(projectilePrefab, shootOrigin, Quaternion.identity);

            // Add a Rigidbody to the projectile

            Rigidbody projectileRb = projectile.GetComponent<Rigidbody>();

            if (projectileRb != null)

            {

                // Apply force to shoot the projectile forward

                projectileRb.velocity = transform.forward \* projectileSpeed;

            }

        }

        else

        {

            Debug.LogError("Projectile Prefab is not assigned!");

        }

    }

}

<https://docs.google.com/document/d/1Gw1h80oJK8pm-Bhfr8ULPMhWJpUUXazIfzLtK4zUczc/edit?usp=sharing>