**Written by Wintana**

**Lantern Pickup**

using UnityEngine;

using System.Collections;

public class LanternPickup : MonoBehaviour

{

    private GameObject heldItem;

    public GameObject interactionUI;

    public GameObject lanternPrefab;

    private bool playerIsNearby;

    void Start()

    {

        heldItem = gameObject;

        interactionUI.SetActive(false);

        lanternPrefab.SetActive(false);

    }

    private void OnTriggerEnter(Collider collider)

    {

        if (collider.CompareTag("Reach"))

        {

            playerIsNearby = true;

            interactionUI.SetActive(true);

        }

    }

    private void OnTriggerExit(Collider collider)

    {

        if (collider.CompareTag("Reach"))

        {

            playerIsNearby = false;

            interactionUI.SetActive(false);

        }

    }

    void Update()

    {

        if (playerIsNearby && Input.GetButtonDown("Interact"))

        {

            PickupLantern();

        }

    }

    private void PickupLantern()

    {

        interactionUI.SetActive(false);

        lanternPrefab.SetActive(true);

        StartCoroutine(DestroyItemAfterDelay());

    }

    private IEnumerator DestroyItemAfterDelay()

    {

        yield return new WaitForSeconds(0.01f);

        Destroy(heldItem);

    }

}

**Sway**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Sway : MonoBehaviour

{

    public float swayAmount;          // How much the object sways in response to mouse movement

    public float swayMax;            // The maximum amount the object can sway

    public float swaySmoothSpeed;    // Speed at which swaying returns to normal

    private Vector3 originalPosition;

    void Start()

    {

        // Save the object's initial local position

        originalPosition = transform.localPosition;

    }

    void Update()

    {

        // Capture mouse movement and calculate the sway amount

        float swayInputX = -Input.GetAxis("Mouse X") \* swayAmount;

        float swayInputY = -Input.GetAxis("Mouse Y") \* swayAmount;

        // Clamp the calculated sway values to stay within the defined max range

        swayInputX = Mathf.Clamp(swayInputX, -swayMax, swayMax);

        swayInputY = Mathf.Clamp(swayInputY, -swayMax, swayMax);

        // Set the target position by combining the sway input with the original position

        Vector3 targetSwayPosition = new Vector3(swayInputX, swayInputY, 0) + originalPosition;

        // Smoothly interpolate the object's position towards the desired target position

        transform.localPosition = Vector3.Lerp(transform.localPosition, targetSwayPosition, Time.deltaTime \* swaySmoothSpeed);

    }

}

**Door**

using UnityEngine;

using UnityEngine.SceneManagement;

public class DoorInteraction : MonoBehaviour

{

    public GameObject interactionPrompt; // UI element to indicate interaction possibility

    public GameObject warningMessage; // UI element for feedback when the door can't be opened

    public GameObject playerKey; // Simulates the inventory key

    public GameObject transitionEffect; // Visual effect for scene transition

    public string targetScene; // Name of the scene to load

    private bool isPlayerNearby = false; // Tracks whether the player is near the door

    void Start()

    {

        // Initially deactivate all UI elements and effects

        interactionPrompt.SetActive(false);

        warningMessage.SetActive(false);

        playerKey.SetActive(false);

        transitionEffect.SetActive(false);

    }

    void OnTriggerEnter(Collider other)

    {

        if (other.CompareTag("Reach"))

        {

            isPlayerNearby = true;

            interactionPrompt.SetActive(true); // Show interaction prompt when in range

        }

    }

    void OnTriggerExit(Collider other)

    {

        if (other.CompareTag("Reach"))

        {

            isPlayerNearby = false;

            interactionPrompt.SetActive(false); // Hide interaction prompt when out of range

            warningMessage.SetActive(false); // Clear warning message

        }

    }

    void Update()

    {

        // Handle interaction logic

        if (isPlayerNearby && Input.GetButtonDown("Interact"))

        {

            if (!playerKey.activeInHierarchy)

            {

                // If the key is not collected, show a warning

                warningMessage.SetActive(true);

            }

            else

            {

                // If the key is present, trigger transition and load the next scene

                interactionPrompt.SetActive(false);

                warningMessage.SetActive(false);

                transitionEffect.SetActive(true);

                StartCoroutine(LoadTargetScene());

            }

        }

    }

    IEnumerator LoadTargetScene()

    {

        // Add a slight delay before transitioning to the next scene

        yield return new WaitForSeconds(0.6f);

        SceneManager.LoadScene(targetScene);

    }

}