Script ground ;

using UnityEngine;

public class GroundController : MonoBehaviour

{

    // Optional: Use for collision detection

    private void OnCollisionEnter(Collision collision)

    {

        Debug.Log($"Collided with {collision.gameObject.name}");

    }

    // Optional: Use for trigger detection

    private void OnTriggerEnter(Collider other)

    {

        Debug.Log($"Trigger entered by {other.gameObject.name}");

    }

}

Scipt crate ;

using UnityEngine;

public class CrateController : MonoBehaviour

{

    public float gridSize = 1.0f; // Size of each grid cell

    public float pushForce = 5.0f; // Force applied to the crate

    private Rigidbody rb;

    private Vector3 targetPosition;

    private bool isMoving = false;

    private void Start()

    {

        rb = GetComponent<Rigidbody>();

        // Lock rotation on all axes to prevent rotation

        rb.freezeRotation = true;

        // Initialize target position

        targetPosition = transform.position;

    }

    private void FixedUpdate()

    {

        // Smoothly move the crate to the target position

        if (isMoving)

        {

            Vector3 direction = targetPosition - transform.position;

            if (direction.magnitude > 0.1f) // Threshold to stop moving

            {

                rb.MovePosition(Vector3.Lerp(transform.position, targetPosition, Time.fixedDeltaTime \* pushForce));

            }

            else

            {

                transform.position = targetPosition;

                isMoving = false;

                SnapToGrid(); // Ensure crate aligns with the grid after movement

            }

        }

    }

    public void PushCrate(Vector3 direction)

    {

        if (!isMoving)

        {

            // Calculate the target position

            targetPosition = transform.position + direction \* gridSize;

            isMoving = true;

        }

    }

    private void SnapToGrid()

    {

        float x = Mathf.Round(transform.position.x / gridSize) \* gridSize;

        float z = Mathf.Round(transform.position.z / gridSize) \* gridSize;

        transform.position = new Vector3(x, transform.position.y, z);

    }

}

Player control

using UnityEngine;

public class PlayerController : MonoBehaviour

{

    public float moveSpeed = 5.0f; // Speed of the player movement

    public float pushForce = 5.0f; // Force applied to push crates

    private Rigidbody rb;

    private Vector3 startPosition;

    private void Start()

    {

        rb = GetComponent<Rigidbody>();

        startPosition = transform.position;

        // Lock rotation on all axes to prevent rotation

        rb.freezeRotation = true;

    }

    private void Update()

    {

        // Get input for movement

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

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

        // Calculate movement direction in X and Z plane

        Vector3 movement = new Vector3(horizontal, 0, vertical).normalized;

        // Move the player

        rb.velocity = movement \* moveSpeed;

        // Maintain constant Y position

        transform.position = new Vector3(transform.position.x, startPosition.y, transform.position.z);

        // Push crate if applicable

        PushCrate(movement);

    }

    private void PushCrate(Vector3 direction)

    {

        // Perform a raycast to detect if the player is pushing a crate

        RaycastHit hit;

        float rayDistance = 1.0f; // Adjust this based on the crate size and distance from the player

        if (Physics.Raycast(transform.position, direction, out hit, rayDistance))

        {

            if (hit.collider.CompareTag("Crate"))

            {

                CrateController crateController = hit.collider.GetComponent<CrateController>();

                if (crateController != null)

                {

                    crateController.PushCrate(direction);

                }

            }

        }

    }

    private void OnCollisionEnter(Collision collision)

    {

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

        {

            Debug.Log("Player hit the wall!");

            // Handle collision with the wall if needed

        }

    }

}

Scene switch :

using UnityEngine;

using UnityEngine.SceneManagement;

public class SceneSwitcher : MonoBehaviour

{

    // Name of the scene to switch to

    public string targetScene;

    // This method will be called when the player collides with another collider

    private void OnCollisionEnter(Collision collision)

    {

        // Check if the object has a specific tag (optional)

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

        {

            // Load the target scene

            SceneManager.LoadScene(targetScene);

        }

    }

}

Trigger scene change :

using UnityEngine;

using UnityEngine.SceneManagement;

public class TriggerSceneChange : MonoBehaviour

{

    // Name of the scene to switch to

    public string targetScene;

    // This method will be called when another collider enters the trigger

    private void OnCollisionEnter(Collision collision)

    {

        // Check if the object that hit has a Rigidbody (optional)

        if (collision.rigidbody != null)

        {

            // Load the target scene

            SceneManager.LoadScene(targetScene);

        }

    }

    // This method will be called when another collider stays in contact with this collider

    private void OnCollisionStay(Collision collision)

    {

        // Load the scene when the collision persists, if needed

        if (collision.rigidbody != null)

        {

            SceneManager.LoadScene(targetScene);

        }

    }

    // This method will be called when another collider leaves the trigger

    private void OnCollisionExit(Collision collision)

    {

        // Optional: You could add a condition to load the scene on exit

    }

}

Wall collider :

using UnityEngine;

public class WallController : MonoBehaviour

{

    // Optional: Define behavior when the player hits the wall

    private void OnCollisionEnter(Collision collision)

    {

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

        {

            Debug.Log("Player hit the wall!");

            // Implement any additional logic for when the player hits the wall

        }

    }

}

Script for enemy to follow waypoint

using UnityEngine;

public class EnemyGridMover : MonoBehaviour

{

    public float moveSpeed = 2f; // Speed of the enemy movement

    private Vector3 nextWaypoint; // The position of the next waypoint

    private bool hasNextWaypoint = false; // Check if the enemy has a valid waypoint

    private void Update()

    {

        if (hasNextWaypoint)

        {

            MoveTowardsWaypoint();

        }

    }

    public void SetNextWaypoint(Vector3 waypointPosition)

    {

        nextWaypoint = waypointPosition;

        hasNextWaypoint = true;

    }

    private void MoveTowardsWaypoint()

    {

        transform.position = Vector3.MoveTowards(transform.position, nextWaypoint, moveSpeed \* Time.deltaTime);

        if (Vector3.Distance(transform.position, nextWaypoint) < 0.01f)

        {

            hasNextWaypoint = false; // Stop moving when the waypoint is reached

        }

    }

}

Script for waypoint ( waypoint 0 Cuma pajangan )

using UnityEngine;

public class WaypointManager : MonoBehaviour

{

    public Transform[] waypoints; // Array of waypoints to move through

    public float moveSpeed = 2f;  // Speed of the enemy movement

    public float pullRange = 0.1f; // Distance at which the enemy is considered to have reached the waypoint

    public EnemyGridMover targetEnemy; // The enemy to move

    private int currentWaypointIndex = 0; // Index of the current waypoint in the list

    private bool isMoving = false; // Track if the enemy is currently moving

    private void Start()

    {

        // Initialize the enemy's target to the first waypoint

        if (waypoints.Length > 0 && targetEnemy != null)

        {

            targetEnemy.SetNextWaypoint(waypoints[currentWaypointIndex].position);

        }

    }

    private void Update()

    {

        if (targetEnemy != null && waypoints.Length > 0)

        {

            // Check if the enemy is moving and has reached the current waypoint

            if (isMoving)

            {

                if (Vector3.Distance(targetEnemy.transform.position, waypoints[currentWaypointIndex].position) < pullRange)

                {

                    // Move to the next waypoint

                    currentWaypointIndex = (currentWaypointIndex + 1) % waypoints.Length; // Loop back to the start

                    targetEnemy.SetNextWaypoint(waypoints[currentWaypointIndex].position);

                    // Check if we have completed the loop

                    if (currentWaypointIndex == 0)

                    {

                        isMoving = false; // Stop moving if back to the first waypoint

                    }

                }

            }

            else

            {

                // Start moving if the enemy is not currently moving

                isMoving = true;

                targetEnemy.SetNextWaypoint(waypoints[currentWaypointIndex].position);

            }

        }

    }

}

Updated script untuk player script ( biar bisa lompat )

using UnityEngine;

public class PlayerController : MonoBehaviour

{

    public float moveSpeed = 5.0f;  // Speed of the player movement

    public float pushForce = 5.0f;  // Force applied to push crates

    public float jumpForce = 7.0f;  // Force applied for jumping

    private Rigidbody rb;

    private bool isGrounded;

    private void Start()

    {

        rb = GetComponent<Rigidbody>();

        // Lock rotation on all axes to prevent rotation

        rb.freezeRotation = true;

    }

    private void Update()

    {

        // Get input for movement

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

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

        // Calculate movement direction in X and Z plane

        Vector3 movement = new Vector3(horizontal, 0, vertical).normalized;

        // Move the player, only changing X and Z velocity, leaving Y for jump

        rb.velocity = new Vector3(movement.x \* moveSpeed, rb.velocity.y, movement.z \* moveSpeed);

        // Jumping logic (using spacebar)

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

        {

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

        }

        // Push crate if applicable

        PushCrate(movement);

    }

    private void PushCrate(Vector3 direction)

    {

        // Perform a raycast to detect if the player is pushing a crate

        RaycastHit hit;

        float rayDistance = 1.0f; // Adjust this based on the crate size and distance from the player

        if (Physics.Raycast(transform.position, direction, out hit, rayDistance))

        {

            if (hit.collider.CompareTag("Crate"))

            {

                CrateController crateController = hit.collider.GetComponent<CrateController>();

                if (crateController != null)

                {

                    crateController.PushCrate(direction);

                }

            }

        }

    }

    private void OnCollisionEnter(Collision collision)

    {

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

        {

            isGrounded = true;  // Player is on the ground

        }

    }

    private void OnCollisionExit(Collision collision)

    {

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

        {

            isGrounded = false; // Player left the ground

        }

    }

}