**Player Movement**

**Run, Walk, Jump is enough.**

using UnityEngine;

public class PlayerMovement : MonoBehaviour

{

public float walkSpeed = 5f;

public float runSpeed = 10f;

public float jumpForce = 10f;

public Transform groundCheck;

public LayerMask groundMask;

private CharacterController characterController;

private Vector3 moveDirection;

private bool isGrounded;

private void Start()

{

characterController = GetComponent<CharacterController>();

}

private void Update()

{

// Check if the player is grounded

isGrounded = Physics.CheckSphere(groundCheck.position, 0.1f, groundMask);

// Player movement

float moveSpeed = Input.GetKey(KeyCode.LeftShift) ? runSpeed : walkSpeed;

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

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

Vector3 move = transform.forward \* verticalInput + transform.right \* horizontalInput;

moveDirection = move.normalized \* moveSpeed;

// Jumping

if (isGrounded && Input.GetButtonDown("Jump"))

{

moveDirection.y = jumpForce;

}

// Apply gravity

if (!isGrounded)

{

moveDirection.y -= 9.81f \* Time.deltaTime; // Gravity

}

// Move the player

characterController.Move(moveDirection \* Time.deltaTime);

}

}

1 Level with any placeholder assets with few platforms

using UnityEngine;

public class LevelController : MonoBehaviour

{

public GameObject player; // The player character

public GameObject platformPrefab;

public int numberOfPlatforms = 5;

public float platformSpacing = 2.0f;

public float levelWidth = 10.0f;

void Start()

{

// Create the platforms

CreatePlatforms();

}

void CreatePlatforms()

{

float currentX = -levelWidth / 2;

for (int i = 0; i < numberOfPlatforms; i++)

{

Vector3 platformPosition = new Vector3(currentX, Random.Range(1.0f, 3.0f), 0);

Instantiate(platformPrefab, platformPosition, Quaternion.identity);

currentX += platformSpacing;

}

}

void Update()

{

// Check if the player falls below the level

if (player.transform.position.y < -5.0f)

{

// Respawn the player at a safe location (you can define your respawn logic)

player.transform.position = new Vector3(0, 2, 0);

}

}

}

1 Enemy with simplistic AI, keep enemy patrolling on the platform and the player should take damage when it touches the enemy

using UnityEngine;

public class EnemyAI : MonoBehaviour

{

public float moveSpeed = 2.0f;

public Transform[] patrolPoints;

private int currentPatrolIndex = 0;

private Rigidbody2D rb;

private void Start()

{

rb = GetComponent<Rigidbody2D>();

if (patrolPoints.Length > 0)

{

MoveToNextPatrolPoint();

}

}

private void Update()

{

if (Vector2.Distance(transform.position, patrolPoints[currentPatrolIndex].position) < 0.1f)

{

MoveToNextPatrolPoint();

}

}

private void MoveToNextPatrolPoint()

{

currentPatrolIndex = (currentPatrolIndex + 1) % patrolPoints.Length;

Vector2 targetPosition = patrolPoints[currentPatrolIndex].position;

if (transform.position.x < targetPosition.x)

{

transform.localScale = new Vector3(-1, 1, 1);

}

else

{

transform.localScale = new Vector3(1, 1, 1);

}

Vector2 moveDirection = (targetPosition - (Vector2)transform.position).normalized;

rb.velocity = moveDirection \* moveSpeed;

}

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Player"))

{

PlayerHealth playerHealth = collision.GetComponent<PlayerHealth>();

if (playerHealth != null)

{

playerHealth.TakeDamage(1); // You can adjust the damage amount

}

}

}

}

Game Start Basic UI with Start and Quit Buttons

using System;

using System.Windows.Forms;

namespace BasicGameUI

{

public class GameStartUI : Form

{

public GameStartUI()

{

this.Text = "Game Start UI";

this.Size = new System.Drawing.Size(300, 150);

this.FormBorderStyle = FormBorderStyle.FixedDialog;

this.StartPosition = FormStartPosition.CenterScreen;

// Create the Start button

Button startButton = new Button();

startButton.Text = "Start Game";

startButton.Location = new System.Drawing.Point(50, 50);

startButton.Click += new EventHandler(StartButton\_Click);

// Create the Quit button

Button quitButton = new Button();

quitButton.Text = "Quit";

quitButton.Location = new System.Drawing.Point(150, 50);

quitButton.Click += new EventHandler(QuitButton\_Click);

// Add the buttons to the form

this.Controls.Add(startButton);

this.Controls.Add(quitButton);

}

private void StartButton\_Click(object sender, EventArgs e)

{

MessageBox.Show("Game started!");

}

private void QuitButton\_Click(object sender, EventArgs e)

{

Application.Exit();

}

public static void Main(string[] args)

{

Application.Run(new GameStartUI());

}

}

}

Player Health for death condition

using System;

public class Player

{

private int health;

public int Health

{

get { return health; }

private set { health = value; }

}

public Player(int startingHealth)

{

Health = startingHealth;

}

public void TakeDamage(int damageAmount)

{

if (Health > 0)

{

Health -= damageAmount;

if (Health <= 0)

{

Die();

}

}

}

private void Die()

{

Console.WriteLine("Player has died!");

}

}

class Program

{

static void Main()

{

Player player = new Player(100);

player.TakeDamage(25); // Player's health: 75

player.TakeDamage(50); // Player's health: 25

player.TakeDamage(30); // Player has died!

Console.ReadLine();

}

}

Game Over UI Screen on Death Condition.

using UnityEngine;

using UnityEngine.UI;

public class GameOverUI : MonoBehaviour

{

public Text gameOverText;

private bool isGameOver = false;

private void Start()

{

// Initially, hide the Game Over UI

gameObject.SetActive(false);

}

public void ShowGameOverScreen()

{

isGameOver = true;

gameOverText.text = "Game Over";

gameObject.SetActive(true);

}

public void RestartGame()

{

UnityEngine.SceneManagement.SceneManager.LoadScene(UnityEngine.SceneManagement.SceneManager.GetActiveScene().buildIndex);

}

}