Ball spawner

using UnityEngine;

public class BalloonPrefab : MonoBehaviour

{

public GameObject balloonPrefab;

public float spawnInterval = 2f;

public float minXSpawnPosition = 0f; // Minimum x-coordinate for spawning

private float nextSpawnTime;

void Start()

{

// Start spawning balloons immediately

nextSpawnTime = Time.time;

}

void Update()

{

// Spawn balloons at regular intervals

if (Time.time > nextSpawnTime)

{

SpawnBalloon();

nextSpawnTime = Time.time + spawnInterval;

}

}

void SpawnBalloon()

{

// Spawn a balloon at a random position on the positive side of the x-axis

float randomX = Random.Range(minXSpawnPosition, 5f); // Adjust 5f as needed

Vector3 spawnPosition = new Vector3(randomX, -6f, 0f);

Instantiate(balloonPrefab, spawnPosition, Quaternion.identity);

}

}

DART

using UnityEngine;

using System.IO.Ports;

public class Dart : MonoBehaviour

{

public float dartSpeed = 1f;

public float moveDistance = 1f;

public float thresholdPositive = 1f; // Threshold for positive movement

public float thresholdNegative = -1f; // Threshold for negative movement

private SerialPort serialPort;

void Start()

{

// Initialize serial port communication

try

{

serialPort = new SerialPort("COM7", 9600); // Change COM port and baud rate as necessary

serialPort.Open();

}

catch (System.Exception e)

{

Debug.LogError("Error opening serial port: " + e.Message);

}

}

void Update()

{

// Check if serial port is open

if (serialPort != null && serialPort.IsOpen)

{

// Read data from the gyroscope

string gyroData = serialPort.ReadLine();

// Validate gyro data format

string[] gyroValues = gyroData.Split(',');

if (gyroValues.Length >= 1)

{

if (float.TryParse(gyroValues[0], out float gyroX))

{

// Move the dart based on gyro data

float moveInput = 0f;

if (gyroX > thresholdPositive)

{

moveInput = 1f; // Move forward

}

else if (gyroX < thresholdNegative)

{

moveInput = -1f; // Move backward

}

// Calculate the target position

float targetX = transform.position.x + moveInput \* dartSpeed \* Time.deltaTime \* moveDistance;

// Clamp the target position within the specified range

targetX = Mathf.Clamp(targetX, -6.5f, 6.5f);

// Set the new position

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

}

else

{

Debug.LogError("Invalid gyro data format: " + gyroData);

}

}

else

{

Debug.LogError("Invalid gyro data format: " + gyroData);

}

}

}

void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Balloon"))

{

// Burst the balloon if dart touches it

Destroy(other.gameObject);

GameManager.Instance.IncrementScore(); // Increment score using GameManager instance

}

}

}

GAME MANAGER

using UnityEngine;

using System.Collections;

using TMPro;

using UnityEngine.UI;

public class GameManager : MonoBehaviour

{

public static GameManager Instance; // Singleton instance

public float gameDuration = 60f; // Duration of the game in seconds

public TextMeshProUGUI scoreText;

public TextMeshProUGUI gameOverText;

public TextMeshProUGUI timeLeftText; // Text element to display time left

public TextMeshProUGUI startGameText; // Text element to display "Start playing..." message

public TextMeshProUGUI keepGoingText; // Text element to display "Keep Going!" message

private float gameTimer;

private int score;

private bool isGameOver;

private bool gameStarted;

public bool IsGameOver // Property to check if the game is over

{

get { return isGameOver; }

}

void Awake()

{

// Ensure only one instance of GameManager exists

if (Instance == null)

{

Instance = this;

}

else

{

Destroy(gameObject);

}

}

void Start()

{

StartGame();

}

public void StartGame()

{

gameTimer = gameDuration;

score = 0;

isGameOver = false;

gameStarted = false;

StartCoroutine(StartGameSequence());

}

IEnumerator StartGameSequence()

{

// Display "Start playing..." message for 2 seconds

startGameText.gameObject.SetActive(true);

yield return new WaitForSeconds(2f);

startGameText.gameObject.SetActive(false);

// Start the game

gameStarted = true;

// Display "Keep Going!" message every 10 seconds

StartCoroutine(KeepGoingSequence());

}

IEnumerator KeepGoingSequence()

{

while (!isGameOver)

{

// Display "Keep Going!" message

keepGoingText.gameObject.SetActive(true);

yield return new WaitForSeconds(2f);

keepGoingText.gameObject.SetActive(false);

yield return new WaitForSeconds(8f); // Wait for the remaining 8 seconds

}

}

void Update()

{

if (!isGameOver && gameStarted)

{

// Update time left

gameTimer -= Time.deltaTime;

timeLeftText.text = "Time Left: " + Mathf.Round(gameTimer); // Round the time to display it as an integer

if (gameTimer <= 0f)

{

// Game over

isGameOver = true;

gameOverText.gameObject.SetActive(true);

}

}

}

public void IncrementScore()

{

if (!isGameOver)

{

score++;

UpdateScoreUI();

}

}

void UpdateScoreUI()

{

scoreText.text = "Score: " + score;

}

}

