```
using TMPro;
using UnityEngine;
using System.Collections.Generic;

public class TiltControls : MonoBehaviour
{
   public GameObject maze;
   public TextMeshProUGUI scoreText;
   public GameObject winPanel;
   public GameObject losePanel;
   public Transform startPoint;
   public GameObject collectiblePrefab;
   public GameObject cam;
   public float sensitivity = 9.8f;
   public float fallThreshold = -50f;
   public float tiltSmoothing = 0.1f;

   private Rigidbody playerRb;
   private int score = 0;
   private List<GameObject> collectibles = new List<GameObject>();
   private Quaternion initialMazeRotation;
   private Vector3 initialTilt;

   void Start()
   {
      playerRb = GetComponent<Rigidbody>();
      StoreCollectibles();
      initialMazeRotation = maze.transform.rotation;

      // Set camera position
      if (SystemInfo.deviceType == DeviceType.Handheld)
      {
         cam.transform.SetPositionAndRotation(new Vector3(-3.1f, 120f, -90f),
Quaternion.Euler(55, 0, 0));
         playerRb.useGravity = true;
         CalibrateTilt();  // Calibrate on mobile
      }
      else
      {
         cam.transform.SetPositionAndRotation(new Vector3(-3.1f,63.7f,-48.9f),
Quaternion.Euler(55, 0, 0));
      }

      Reset();
   }

   private void FixedUpdate()
   {
    Vector3 tilt = Vector3.zero;

    // Get tilt input
    if (SystemInfo.deviceType == DeviceType.Handheld)
    {
        tilt = Input.acceleration - initialTilt;  // Apply calibration
    }
    else
    {
        tilt = new Vector3(Input.GetAxis("Vertical"), 0, Input.GetAxis("Horizontal"));
    }

    // Apply tilt threshold to reduce jitter
    if (Mathf.Abs(tilt.x) < tiltSmoothing) tilt.x = 0;
    if (Mathf.Abs(tilt.z) < tiltSmoothing) tilt.z = 0;

    // Rotate maze with adjusted sensitivity
    maze.transform.Rotate(tilt * sensitivity * 0.5f * Time.fixedDeltaTime);

    // Check if player falls off the maze
    if (transform.position.y < fallThreshold)
    {
        Lose();
    }
}

private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Collectible"))
    {
        collision.gameObject.SetActive(false);
        score++;
        scoreText.text = "Score: " + score;
    }
    else if (collision.gameObject.CompareTag("EndWall"))
    {
        Win();
    }
}

private void Win()
{
    Debug.Log("You Won!");
    winPanel.SetActive(true);
    ResetCollectibles();
}

private void Lose()
```

```
    {
        Debug.Log("You fell off the maze!");
        playerRb.linearVelocity = Vector3.zero;
        playerRb.angularVelocity = Vector3.zero;
        gameObject.SetActive(false);
        losePanel.SetActive(true);
    }

    public void Reset()
    {
        // Reset maze rotation
        maze.transform.rotation = initialMazeRotation;

        // Reset player position and physics
        gameObject.SetActive(true);
        transform.position = startPoint.position;
        playerRb.linearVelocity = Vector3.zero;
        playerRb.angularVelocity = Vector3.zero;

        // Reset UI and score
        score = 0;
        scoreText.text = "Score: 0";

        winPanel.SetActive(false);
        losePanel.SetActive(false);

        // Reset collectibles
        ResetCollectibles();
    }

    private void StoreCollectibles()
    {
        GameObject[] collectibleObjects =
GameObject.FindGameObjectsWithTag("Collectible");
        collectibles.Clear();
        collectibles.AddRange(collectibleObjects);
    }

    private void ResetCollectibles()
    {
        foreach (GameObject collectible in collectibles)
        {
            collectible.SetActive(true);
        }
    }

    private void CalibrateTilt()
    {
        initialTilt = Input.acceleration;
        Debug.Log("Tilt calibrated: " + initialTilt);
    }
}
```