

Rolling Ball on iPhone

CIM 640 Creative Coding
Jinqi Li

Introduction

It's a mobile game with a ball rolling on a track while the player needs to tilt the phone to collect stars and avoid barriers in order to get more scores.

Demonstration



https://youtu.be/lsya_hZqyWM

Features

- UI button
 - Start button
 - Restart button

```

public class StartGame : MonoBehaviour
{
    public GameObject PanelBefore;
    //public GameObject PlayerBall;
    public float BallSpeed = 10;
    public bool OnUpdate = false;

    public GameObject ButtonLeft;
    public GameObject ButtonRight;

    // Start is called before the first frame update
    void Start()
    {
        //GetComponent<Rigidbody>().isKinematic = true;
        ButtonLeft.gameObject.SetActive(false);
        ButtonRight.gameObject.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        if(OnUpdate == true)
        {
            transform.position = transform.position + new Vector3(BallSpeed * Time.deltaTime * -1, 0, 0);
        }
    }

    public void BeginGame()
    {
        PanelBefore.gameObject.SetActive(false);
        ButtonLeft.gameObject.SetActive(true);
        ButtonRight.gameObject.SetActive(true);
        //GetComponent<Rigidbody>().isKinematic = false;
        OnUpdate = true;
    }
}

```

```

public class Restart : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
    }

    public void RestartGame()
    {
        SceneManager.LoadScene("MainScene");
    }
}

```

Features (Cont'd)

- Gyroscope
 - Tilting the phone to move the object left and right

```

public class GyroManager : MonoBehaviour
{
    #region Instance
    private static GyroManager instance;
    public static GyroManager Instance
    {
        get
        {
            if(instance == null)
            {
                instance = FindObjectOfType<GyroManager>();
                if(instance == null)
                {
                    instance = new GameObject("Spawned GyroManager", typeof(GyroManager)).GetComponent<GyroManager>();
                }
            }

            return instance;
        }

        set
        {
            instance = value;
        }
    }
    #endregion

    [[Header("Logic")]]
    private Gyroscope gyro;
    private Quaternion rotation;
    private bool gyroActive;

    public void EnableGyro()
    {
        if (gyroActive)
            return;

        if(SystemInfo.supportsGyroscope)
        {
            gyro = Input.gyro;
            gyro.enabled = true;
            gyroActive = gyro.enabled;
        }
    }

    private void Update()
    {
        if(gyroActive)
        {
            rotation = gyro.attitude;
            Debug.Log(rotation);
        }
    }

    public Quaternion GetGyroRotation()
    {
        return rotation;
    }
}

```

```

public class FollowGyro : MonoBehaviour
{
    [Header("Tweaks")]
    [SerializeField] private Quaternion baseRotation = new Quaternion(0, -1, 0, 0);

    private void Start()
    {
        GyroManager.Instance.EnableGyro();
    }

    private void Update()
    {
        transform.localRotation = GyroManager.Instance.GetGyroRotation() * baseRotation;
    }
}

```

Features (Cont'd)

- Collision
 - Barrier collision: game over
 - Star collision: add score - star disappear
 - Collide on the last star: add score - star disappear - game over
 - Collide on the transparent barrier in the end: game over


```

public class BarrierCollision : MonoBehaviour
{
    public Text FinalScore;
    public Text GameOver;
    public Text CurrentScore;

    public GameObject PanelAfter;

    private StartGame script;
    // Start is called before the first frame update
    void Start()
    {
        script = GetComponent<StartGame>();
    }

    // Update is called once per frame
    void Update()
    {

    }

    void OnCollisionEnter(Collision collision)
    {
        if(collision.gameObject.tag == "barrier" || collision.gameObject.tag == "finalStar")
        {
            GetComponent<Rigidbody>().isKinematic = true;
            script.OnUpdate = false;

            GameOver.text = "Game Over";

            int currentScore = int.Parse(CurrentScore.text);
            FinalScore.text = "Your score is " + currentScore.ToString();

            PanelAfter.gameObject.SetActive(true);
        }
    }
}

```

```

public class TotalScore : MonoBehaviour
{
    public Text CurrentScore;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "star" || collision.gameObject.tag == "finalStar")
        {
            GetComponent<Rigidbody>().isKinematic = true;
            int curScore = int.Parse(CurrentScore.text);
            curScore++;
            CurrentScore.text = curScore.ToString();
            Destroy(collision.gameObject);
            GetComponent<Rigidbody>().isKinematic = false;
        }
    }
}

```

Features (Cont'd)

- Camera following player without rotating
- Object moving automatically
- Scene management

```

public class Restart : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
    }

    public void RestartGame()
    {
        SceneManager.LoadScene("MainScene");
    }
}

```

```

public class CameraControl : MonoBehaviour
{
    public GameObject target;
    Vector3 offset;
    // Use this for initialization
    void Start()
    {
        offset = transform.position - target.transform.position;
    }

    // Update is called once per frame
    void LateUpdate()
    {
        Vector3 desiredPosition = target.transform.position + offset;
        transform.position = desiredPosition;
    }
}

```

```

public class PlayerController : MonoBehaviour
{
    public float speed;

    private Rigidbody rb;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        //float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        Vector3 movement = new Vector3(0.0f, 0.0f, moveVertical);

        rb.AddForce(movement * speed);
    }
}

```