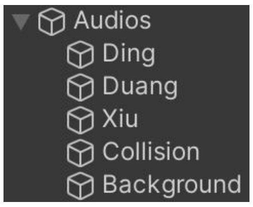
Code and Design Explaination

As the basic ball movement and trigger events have been explained in the official tutorial, I will not repeat them here. I will just explain the code about the feature enhancements that I added.

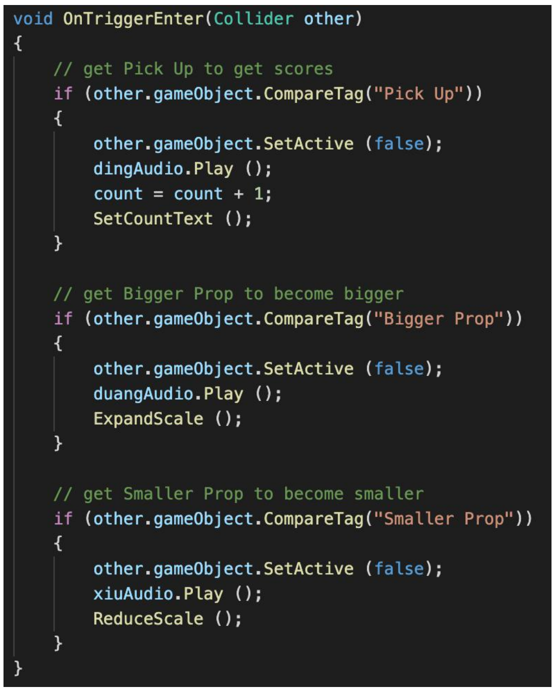
1. Add background music to make the game more vivid.

I created five audio objects, which are the audio when I get yellow, blue and red props, the audio when I hit obstacles, and the background music.



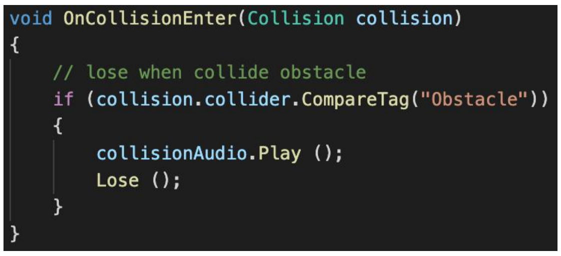
```
▼ Audios
    Ding
    Duang
    Xiu
    Collision
    Background
```

Except the background music, which will be played in a loop on awake, the other audios will be played in different conditions.



```
void OnTriggerEnter(Collider other)
{
    // get Pick Up to get scores
    if (other.gameObject.CompareTag("Pick Up"))
    {
        other.gameObject.SetActive (false);
        dingAudio.Play ();
        count = count + 1;
        SetCountText ();
    }

    // get Bigger Prop to become bigger
    if (other.gameObject.CompareTag("Bigger Prop"))
    {
        other.gameObject.SetActive (false);
        duangAudio.Play ();
        ExpandScale ();
    }

    // get Smaller Prop to become smaller
    if (other.gameObject.CompareTag("Smaller Prop"))
    {
        other.gameObject.SetActive (false);
        xiuAudio.Play ();
        ReduceScale ();
    }
}
```

We can see the OnTriggerEnter function. To get different props, different audios will be played.



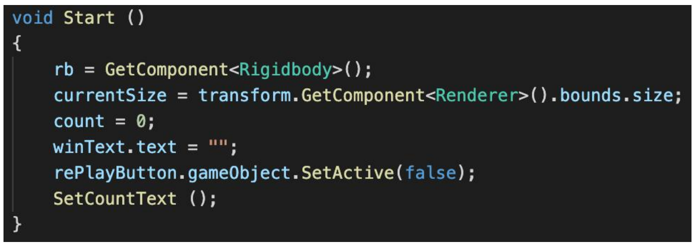
```
void OnCollisionEnter(Collision collision)
{
    // lose when collide obstacle
    if (collision.collider.CompareTag("Obstacle"))
    {
        collisionAudio.Play ();
        Lose ();
    }
}
```

And when the ball collide the moving obstacles (which are tagged by "Obstacle") the collision audio will be played.

This enhancement is so easy to test, you can see the performance in my Gameplay Video.

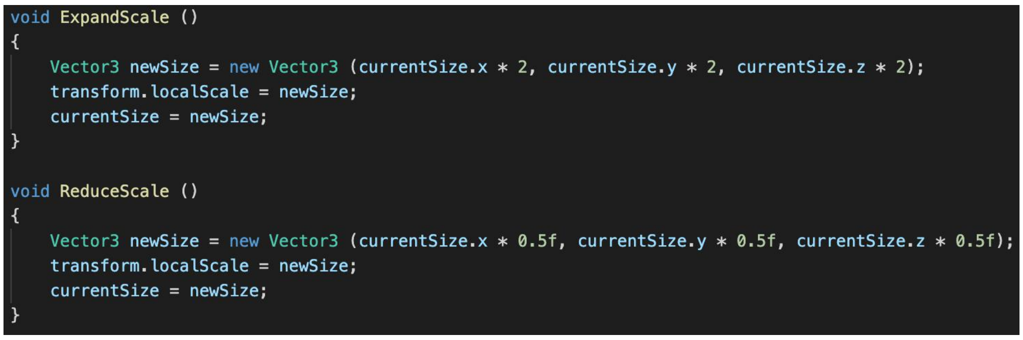
2. By getting different props, change the size of the ball to pass through the following terrain.

In this enhancement, the most important thing is how to change the size of ball. I record the original size of the ball in Start function using bounds.size.



```
void Start ()
{
    rb = GetComponent<Rigidbody>();
    currentSize = transform.GetComponent<Renderer>().bounds.size;
    count = 0;
    winText.text = "";
    rePlayButton.gameObject.SetActive(false);
    SetCountText ();
}
```

In different conditions, I will create new size according to original size to replace the size of the ball.

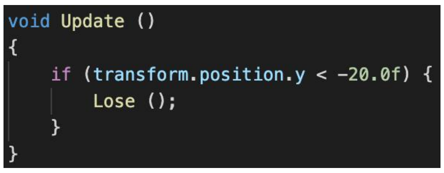


```
void ExpandScale ()
{
    Vector3 newSize = new Vector3 (currentSize.x * 2, currentSize.y * 2, currentSize.z * 2);
    transform.localScale = newSize;
    currentSize = newSize;
}

void ReduceScale ()
{
    Vector3 newSize = new Vector3 (currentSize.x * 0.5f, currentSize.y * 0.5f, currentSize.z * 0.5f);
    transform.localScale = newSize;
    currentSize = newSize;
}
```

For testing, I gave several props near the ball, and tied the related tags to the props. Then, the trigger events worked well. The ball become bigger or smaller when collecting the related props.

3. Add different landforms to improve the difficulty of the ball to reach the destination.

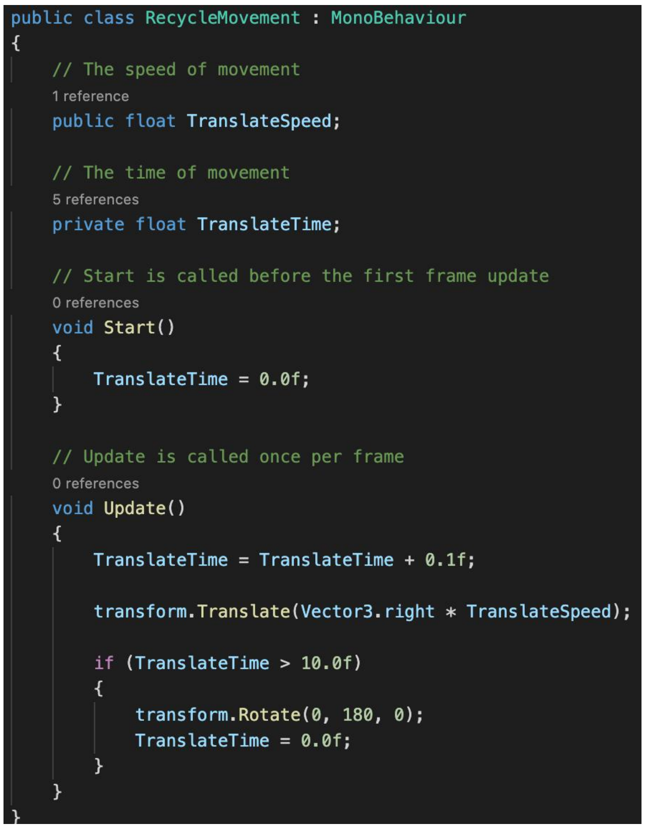
In this one, I set the minimum of y position. When the ball move out of the bounds of the landforms, it will fall forever. I set -20.0f as the reference. When the ball's y position is smaller than -20.0f, that means that this ball is over the bounds and the game is over. The code is as follows.



```
void Update ()
{
    if (transform.position.y < -20.0f) {
        Lose ();
    }
}
```

4. Make the obstacles move to increase the difficulty of the game.

To make the obstacles move, I use the following code.



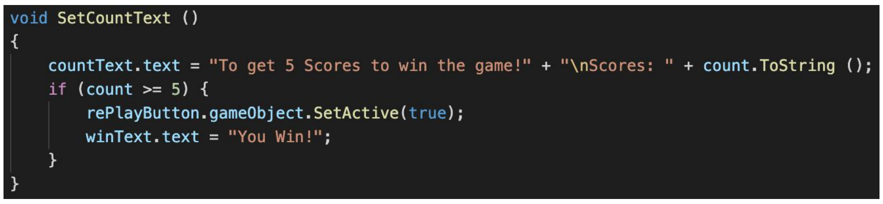
```
public class RecycleMovement : MonoBehaviour
{
    // The speed of movement
    1 reference
    public float TranslateSpeed;

    // The time of movement
    5 references
    private float TranslateTime;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        TranslateTime = 0.0f;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        TranslateTime = TranslateTime + 0.1f;

        transform.Translate(Vector3.right * TranslateSpeed);

        if (TranslateTime > 10.0f)
        {
            transform.Rotate(0, 180, 0);
            TranslateTime = 0.0f;
        }
    }
}
```

The obstacles, which are tied with this script, will move in a designated direction in a pre-set speed for a fixed time. Than, the obstacles will be rotated on the y-axis. And the time will be reset. The obstacles will move in the opposite direction for same time. Than, repeat. It will be a loop.

For testing, you just need to create an game object tied with this script.

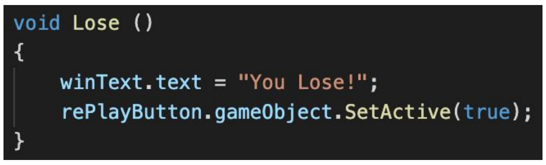
5. Set the end conditions of the game, for example, going out of bounds or hitting a special object, to enhance the playability of the game.

For win, you need to get the goal of scores. Collecting one yellow prop, you will get one score.



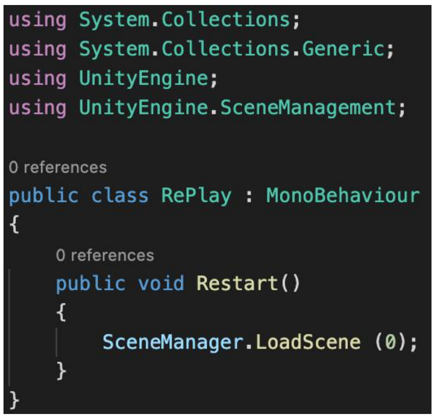
```
void SetCountText ()
{
    countText.text = "To get 5 Scores to win the game!" + "\nScores: " + count.ToString ();
    if (count >= 5) {
        rePlayButton.gameObject.SetActive(true);
        winText.text = "You Win!";
    }
}
```

And if you are out of the bound of terrain that I set or if you collide the moving obstacles, you will lose the game. (The code has already showed before.)



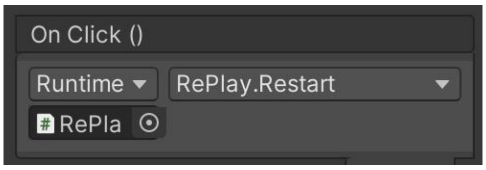
```
void Lose ()
{
    winText.text = "You Lose!";
    rePlayButton.gameObject.SetActive(true);
}
```

If you win or lose, the replay button will be active. For realize the restart function, I use LoadScene function. The code in Replay script is as follows.



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

0 references
public class RePlay : MonoBehaviour
{
    0 references
    public void Restart()
    {
        SceneManager.LoadScene (0);
    }
}
```

This public function also need to be added to replay button.



On Click ()

Runtime | RePlay.Restart

RePla

You can lose or win the game, and test this button.