Color

Changing the color of an object while pressing the C key

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


public class color : MonoBehaviour
{
    private Renderer ballRenderer;
    // Start is called before the first frame update

    void Start()
    {
        ballRenderer = GetComponent<Renderer>();
    }

    // Update is called once per frame

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.C))
        {
            ChangeColor();
        }
    }


    void ChangeColor()
    {
        Color newColor = new Color(Random.value, Random.value, Random.value);
        ballRenderer.material.color = newColor;
    }
}
```

This code makes use of Unity's Renderer component to control the rendering and color change of objects. The Start() function is used to initialize the Renderer, Update() is used to monitor keystrokes in real time, and ChangeColor() randomly generates and applies a new color. The Changecolor function generates three random floating point numbers (ranging from 0 to 1) as RGB color values through Random.value, and assigns values to change the target color after creating a new color object, so the final color created will be random

CameraFollow

This code allows the camera to follow the player body, in my demo it is moving as a ball for the main character. The locked subject will be fixed in the center of the screen

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

Unity 脚本(1 个资产引用)|0 个引用
public class CameraFollow : MonoBehaviour
{
    public Transform player; // transform
    public Vector3 offest; //dis between cube and c


    Unity 消息|0 个引用
    private void Start()
    {
        offest = transform.position - player.position;
    }

    // Update is called once per frame
    Unity 消息|0 个引用
    void Update()
    {
        transform.position = player.position + offest;
    }
}
```

This script ensures that the camera always follows the player in the game with the same displacement by calculating the initial position difference between the player and the camera. The script calculates the initial relative position difference offest in the Start() function, and then updates the camera's position in the Update() function at each frame so that it follows the player's position while maintaining a fixed relative distance. So no matter how the player moves, the camera will follow and keep the original relative position and distance.

Jump

This code contains physics related features such as using the Rigidbody component for object movement, jumping, and collision detection. It also implements the jumping functionality of the object, which allows the object to jump-fall in response to a given amount of force. (see code next page)

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

Unity 脚本(1 个资产引用)|0 个引用
public class Jump : MonoBehaviour
{
    public float moveSpeed = 1f;
    public float jumpSpeed = 2f;
    public Vector3 initialPosition;
    private bool isMoving = false;
    private Rigidbody rb;

    Unity 消息|0 个引用
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        initialPosition = transform.position;
        rb.freezeRotation = true;
        rb.velocity = Vector3.zero;
    }

    Unity 消息|0 个引用
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.J))
        {
            rb.AddForce(Vector3.up * jumpSpeed, ForceMode.Impulse);
        }
    }

    Unity 消息|0 个引用
    void FixedUpdate()
    {
        if (isMoving)
        {
            rb.velocity = new Vector3(moveSpeed, rb.velocity.y, 0);
        }
    }

    Unity 消息|0 个引用
    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.tag == "Obstacle")
        {
            ResetBall();
        }
    }

    1 个引用
    private void ResetBall()
    {
        rb.velocity = Vector3.zero;
        rb.angularVelocity = Vector3.zero;
        transform.position = initialPosition;
        isMoving = false;
    }

    0 个引用
    public void StartMoving()
    {
        isMoving = true;
    }
}
```

This script is the focus of my project. First the script gets the object's Rigidbody component in the Start() function and records the object's initial position, freezes its rotation and initializes its speed to zero to make sure the object is stationary at the start of the game (otherwise it will always reflect somehow). In the Update() function, which is updated every frame, the script listens for the player to press the J button, and if so, gives the object an upward force via the AddForce function to make it jump. In the FixedUpdate() function for physics update, the script checks for the isMoving boolean value, if true, the object is moved horizontally along the x-axis by modifying its speed while keeping its y-axis (vertical) speed constant.The OnCollisionEnter() function is used to detect collisions between the object and other objects when the object touches an object with a When the object encounters an object with the label Obstacle, the script calls the ResetBall() function to set the object's velocity and angular velocity to zero, reset the object's position to the initial position, and stop the object's movement.The StartMoving() function is a publicly available function that allows for an external call to start the object's movement (it's linked to my movement script). With these functions this code implements jumping, horizontal movement of the object and resetting it when it collides with an obstacle.

Movement

The original purpose of this code was to control the movement of the object, but because of a bug that I didn't figure out, I realized that I had to rewrite the jump function in this code in order for the jump function to work correctly therefore I wrote a duplicate jump function. The good news is that it finally ran successfully.

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

Unity 脚本(1 个资产引用)|0 个引用
public class movement : MonoBehaviour
{
    public float moveSpeed = 3f;
    private bool isMoving = false;
    private Rigidbody rb;

    Unity 消息|0 个引用
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    Unity 消息|0 个引用
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            isMoving = true;
        }

        if (isMoving)
        {
            rb.velocity = new Vector3(moveSpeed, rb.velocity.y, rb.velocity.z);
        }

        if (Input.GetKeyDown(KeyCode.J))
        {
            rb.AddForce(Vector3.up * 10, ForceMode.Impulse);
        }
    }
}
```

The script begins by defining two variables: moveSpeed (the speed at which the object is moving) and isMoving (used to determine if the object is moving). A private variable of type Rigidbody, rb, is also defined for subsequent control of the physical behavior of the object. In the Start() function, the object's Rigidbody component is obtained via GetComponent<Rigidbody>() and assigned to rb for subsequent use. In the Update() function, each frame detects if the space bar (KeyCode.Space) is pressed, and if it is pressed, sets isMoving to true, indicating that the object is starting to move. If isMoving is true, the speed of the object is updated so that it moves according to the moveSpeed value on the x-axis, keeping the y- and z-axis speeds constant. The script also listens for the J key (KeyCode.J) to be pressed, and if it is pressed, an upward

impulse force is applied to the object via the AddForce() function to realize the jumping effect of the object. (This repeated function is to make sure the Jump runs smoothly)

End

It controls an object to play a sound effect when it reaches a specific x-axis position. The code determines whether to trigger the playback of a successful sound effect by monitoring the current position of the object. If the object successfully reaches its destination then the sound effect will be played.

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

Unity 脚本(1 个资产引用)|0 个引用
public class End : MonoBehaviour
{
    public float stopXPosition = 32f;
    public AudioClip successSound;
    private AudioSource audioSource;
    private Rigidbody rb;
    private bool hasStopped = false;

    Unity 消息|0 个引用
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        audioSource = GetComponent<AudioSource>();
    }

    Unity 消息|0 个引用
    void Update()
    {
        if (!hasStopped && transform.position.x >= stopXPosition)
        {
            PlaySuccessSound();
        }
    }

    1 个引用
    void PlaySuccessSound()
    {
        if (successSound != null && audioSource != null)
        {
            audioSource.PlayOneShot(successSound);
        }
    }
}
```

In the Start() function, rb gets the Rigidbody component attached to the object via GetComponent<Rigidbody>(), and audioSource gets the AudioSource component of the object via GetComponent<AudioSource>(). As with the rest of the code, the Update() function is called every frame and first checks to see if the object has reached the stopXPosition position and if the sound effect has not been played yet, using a conditional judgment. If the condition is true, the PlaySuccessSound()

function is called, and it first checks to see if both successSound and audioSource are not empty, and if they are not, it uses audioSource.PlayOneShot(successSound) to play the success sound. This method is designed to play a one-time sound without repeating it or conflicting with other bgm's that may be present (a plan I eventually scrapped).