**实验一 场景简单搭建和简单控制**

### 课程名称： 游戏程序设计 实验日期：2024/3/23

### 班级： 人智2103 姓名： 余杰 学号：1193210316

**一、实验目的**

1、简单了解Unity3D中的场景搭建和简单控制

**二、实验要求**

要求将自己编写的**代码复制粘贴**到实验报告中，注意**保留代码格式（尤其是缩进）**，并将**运行结果的界面以截图**的方式放入实验报告。

三、实验任务

1. 简单使用unity3d软件，新建场景，导入物体；（10分）

2. 通过脚本编程，完成物体的移动，分为：自动移动，人为控制移动；（10分）

3. 了解unity3d中物体的旋转，利用c#编程简单控制其位置变化；（10分）

4. 完成物体的自传和公转 （10分）

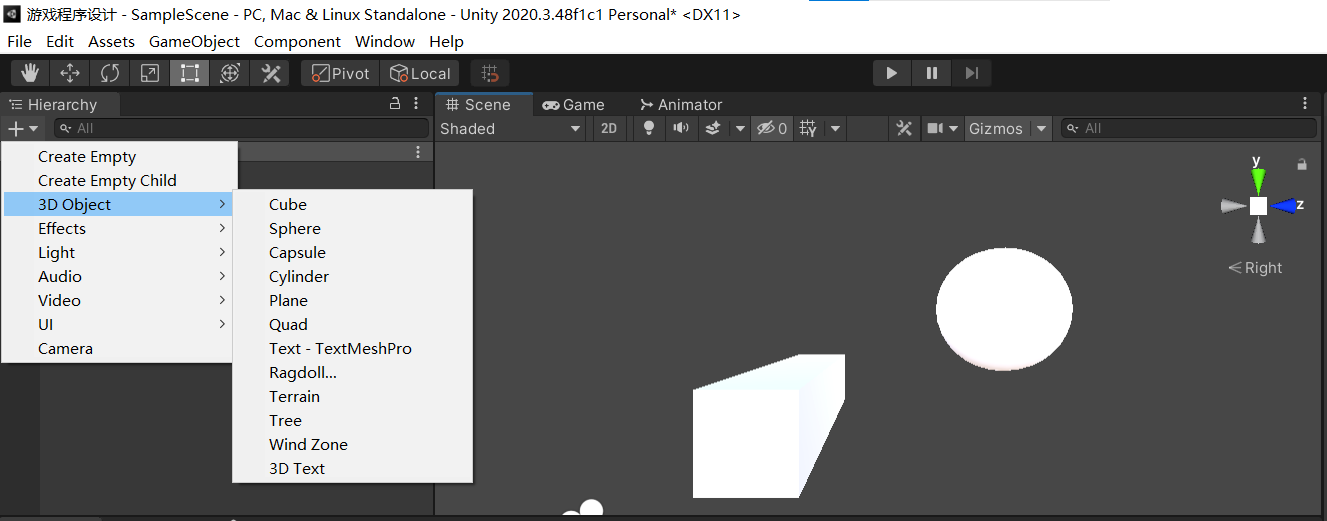
5. 设置public值，通过检查器实时控制物体旋转或者移动的速度，或者面对对象（10分）

6. 了解脚本中函数的调用顺序 awake（10分）

7. 简单了解播放音乐，通过鼠标控制（40分）

**四、实验代码及结果**（先主要代码，后结果）

1. **新建了一个空的3d场景，只有一个主摄像头和光源。选择工具栏Create-->3D object,导入一个立方体和一个球体**



1. **在Asset根目录下新建一个Scripts文件夹，用于存放脚本，打开文件夹新建一个C#脚本CubeController.cs**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CubeContraler : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

}

}

1. 自动移动

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CubeController : MonoBehaviour

{

public float speed = 5.0f; // 移动速度

private Vector3 moveDirection = Vector3.zero; // 移动方向

// Start is called before the first frame update

void Start()

{

// 初始随机方向

ChangeDirection();

}

// Update is called once per frame

void Update()

{

// 随机改变方向

if (Random.Range(0, 100) < 5) // 有5%的概率改变方向

{

ChangeDirection();

}

// 根据方向和速度移动物体

transform.Translate(moveDirection \* speed \* Time.deltaTime);

}

// 随机改变移动方向

void ChangeDirection()

{

moveDirection = new Vector3(Random.Range(-1f, 1f), Random.Range(-1f, 1f), Random.Range(-1f, 1f)).normalized;

}

}

公共类CubeController，它继承自MonoBehaviour。在Unity中，几乎所有的脚本都会继承自MonoBehaviour，这样才能让这个脚本挂载到游戏对象上，并让Unity的生命周期方法（如Start和Update）生效。

公共变量speed，定义了物体的移动速度。将其设置为公共变量，可以在Unity编辑器中直接修改这个值，而不需要重新编译脚本。

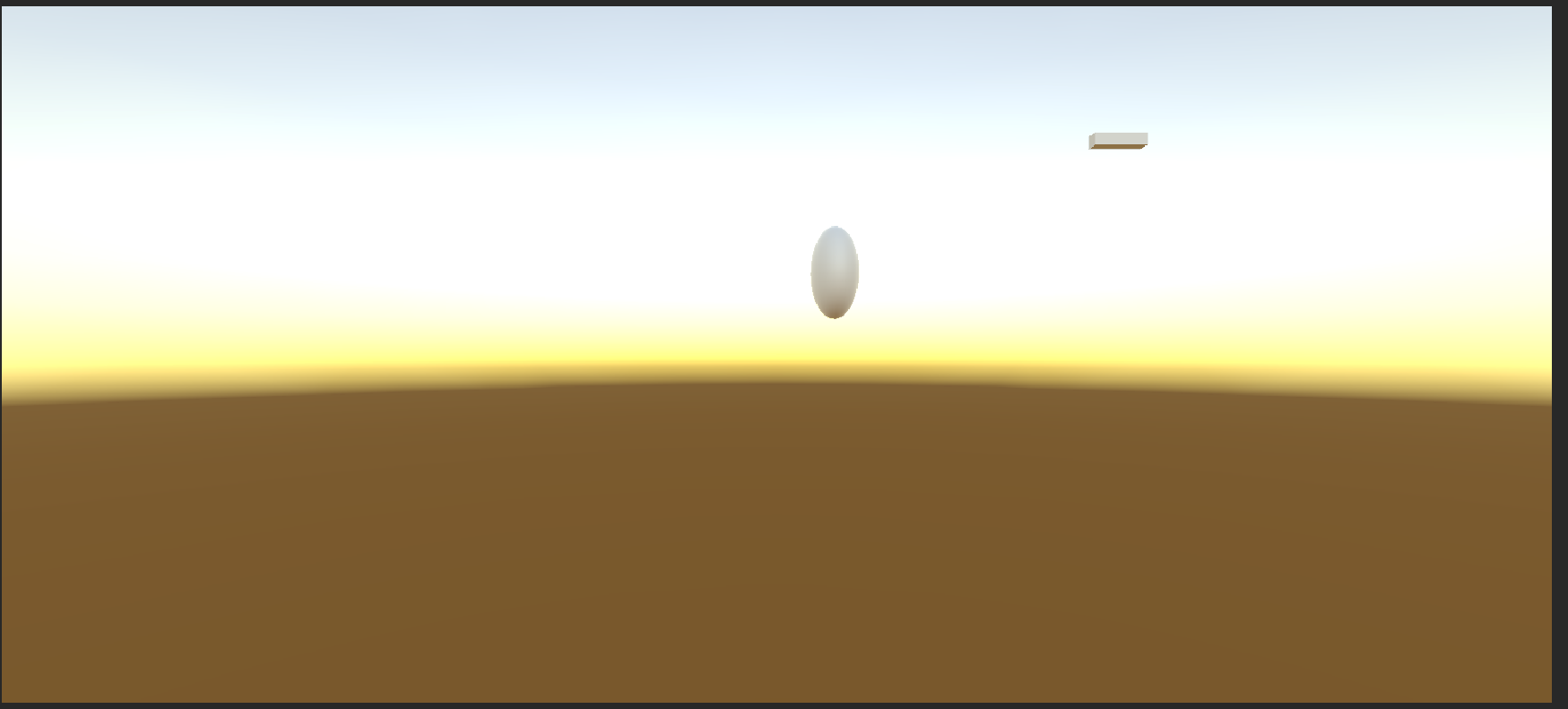
私有变量moveDirection，它存储了物体的移动方向。Vector3.zero是一个方便的方式来初始化这个向量为(0, 0, 0)，表示一开始物体没有移动方向。

Start是一个Unity生命周期方法，它在物体首次激活时调用一次。这里调用ChangeDirection方法来给物体设置一个初始的随机方向。

Update是另一个Unity生命周期方法，它在每一帧中都会被调用。首先，使用Random.Range(0, 100) < 5判断是否需要改变方向，这实现了大约5%的概率。如果条件满足，就调用ChangeDirection来改变移动方向。然后，使用transform.Translate方法根据当前方向和速度移动物体，Time.deltaTime确保移动速度不会受到帧率的影响。

最后，ChangeDirection方法通过生成三个随机数作为向量的x、y、z分量来改变移动方向，normalized确保这个向量的长度为1，这样不会因为方向改变而影响移动速度。

这段代码组合起来就实现了一个基本的随机移动效果。物体会在不同方向上以设定的速度移动，并且在每一帧都有小概率改变其移动方向。



1. 人为控制移动

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SphereController : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

public float moveSpeed = 5.0f; // 基础移动速度

public float sprintMultiplier = 2.0f; // 加速倍数

public float verticalSpeed = 3.0f; // 垂直移动速度

// Update is called once per frame

void Update()

{

float speed = moveSpeed;

// 按住Shift键加速

if (Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))

{

speed \*= sprintMultiplier;

}

// 获取WASD键的输入

float moveHorizontal = Input.GetAxis("Horizontal"); // AD键对应的水平移动

float moveVertical = Input.GetAxis("Vertical"); // WS键对应的前后移动

// 获取空格键和Ctrl键的输入进行上下移动

float moveUp = Input.GetKey(KeyCode.Space) ? 1.0f : 0.0f;

float moveDown = Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl) ? -1.0f : 0.0f;

// 根据输入计算移动方向

Vector3 movement = new Vector3(moveHorizontal, moveUp + moveDown, moveVertical);

// 更新物体的位置

transform.Translate(movement \* (movement.y == 0 ? speed : verticalSpeed) \* Time.deltaTime, Space.World);

}

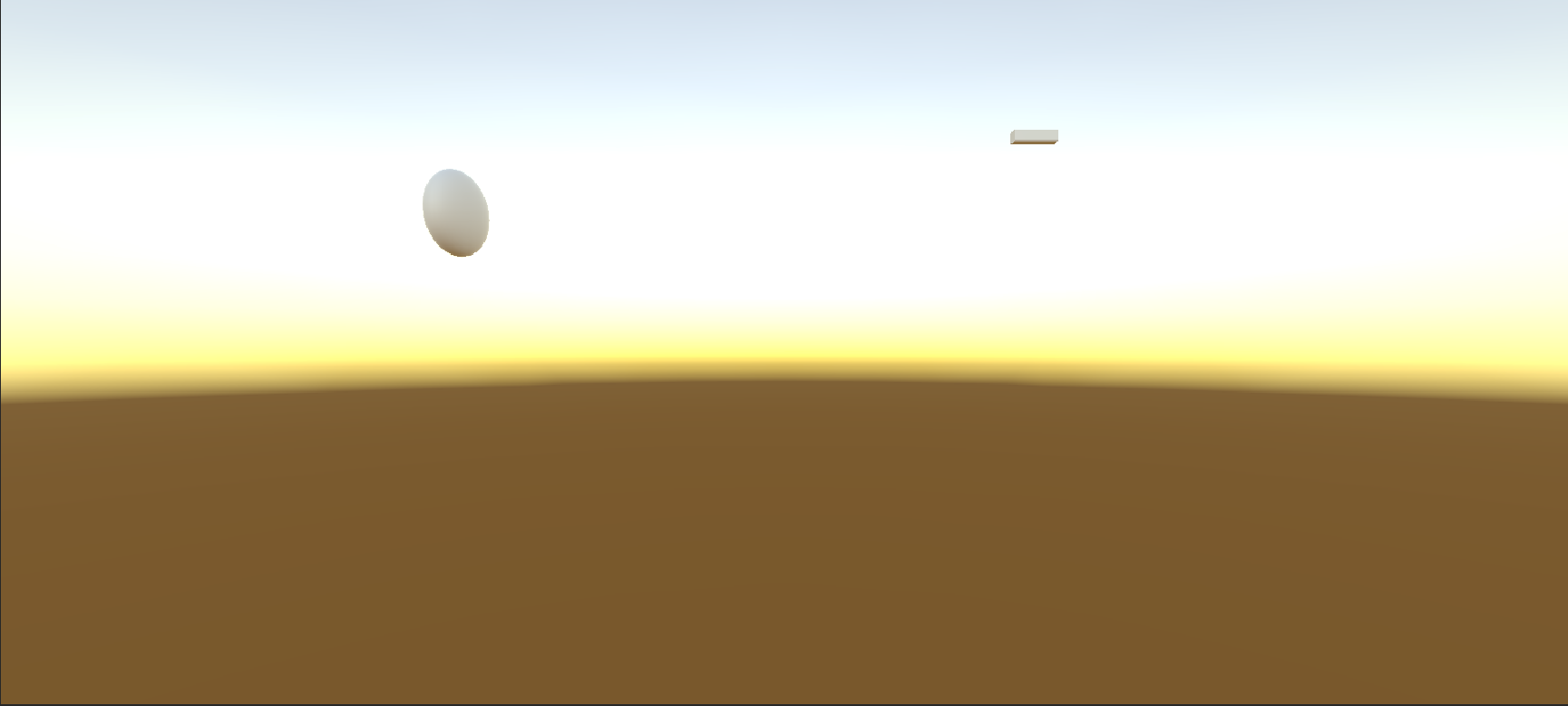
}

通过键盘的WASD键加上空格键和Ctrl键，就可以实现物体在三维空间内自由移动的效果。

moveSpeed：基础移动速度，控制物体在水平方向（前后左右）移动的速度。

sprintMultiplier：加速倍数，用于在按住Shift键时增加移动速度。

verticalSpeed：垂直移动速度，控制物体上升和下降的速度。



1. **物体的旋转**

public class CubeController : MonoBehaviour

{

public float rotationSpeed = 90.0f; // 旋转速度，每秒旋转度数

// Update is called once per frame

void Update()

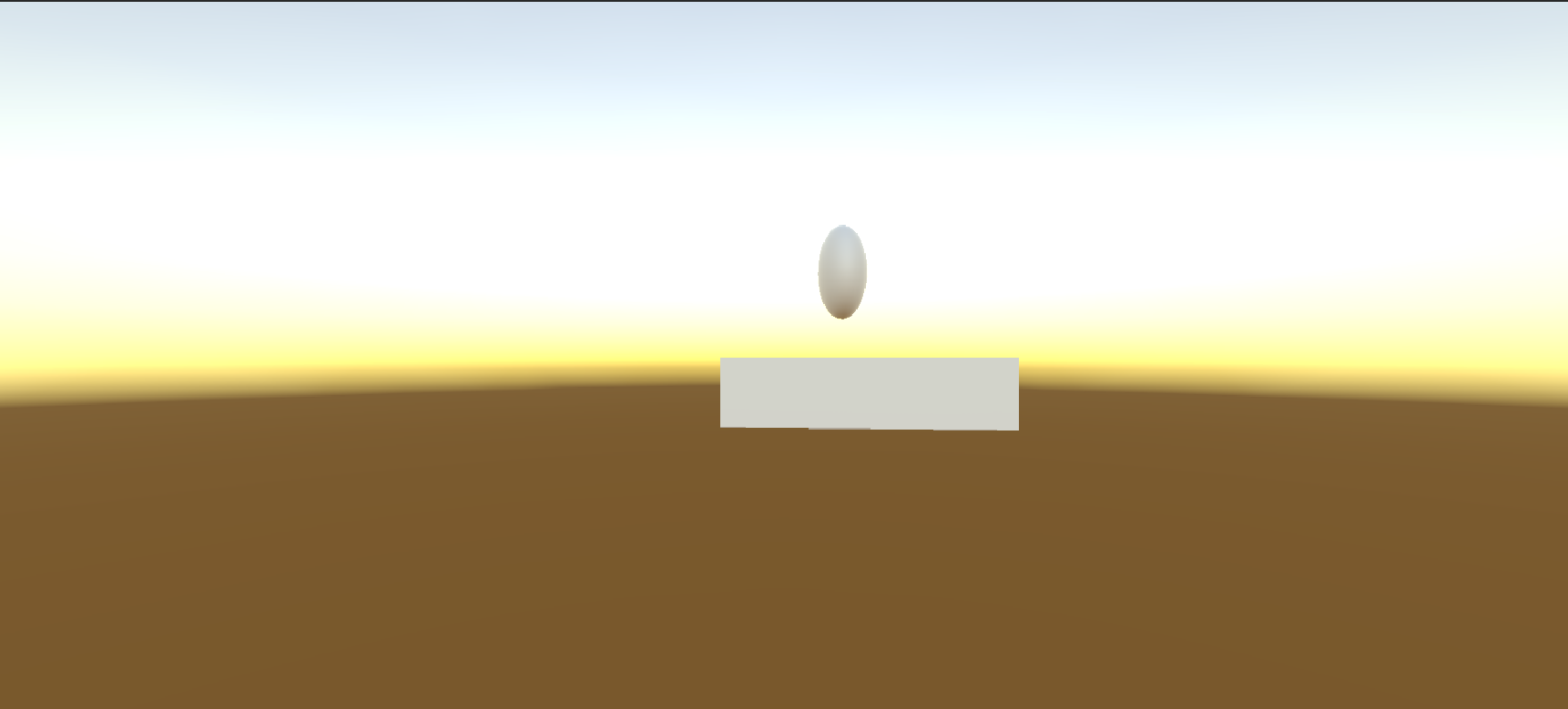
{

// 使物体绕y轴旋转

transform.Rotate(0, rotationSpeed \* Time.deltaTime, 0);

}

}



1. **物体的自转和公转**

public class CubeController : MonoBehaviour

{

public GameObject pivot; // 公转的中心点

public float selfRotationSpeed = 90.0f; // 自转速度，每秒旋转度数

public float orbitalRotationSpeed = 45.0f; // 公转速度，每秒旋转度数

// Start is called before the first frame update

void Start()

{

// 确保 pivot 不为空

if (pivot == null)

{

// 如果未指定，创建一个新的 pivot 点在原点

pivot = new GameObject("Pivot");

pivot.transform.position = new Vector3(0, 0, 10); ; // 设置公转中心点位置

}

// 设置当前对象为 pivot 的子对象

transform.parent = pivot.transform;

}

// Update is called once per frame

void Update()

{

// 自转

transform.Rotate(Vector3.up, selfRotationSpeed \* Time.deltaTime, Space.Self);

// 公转

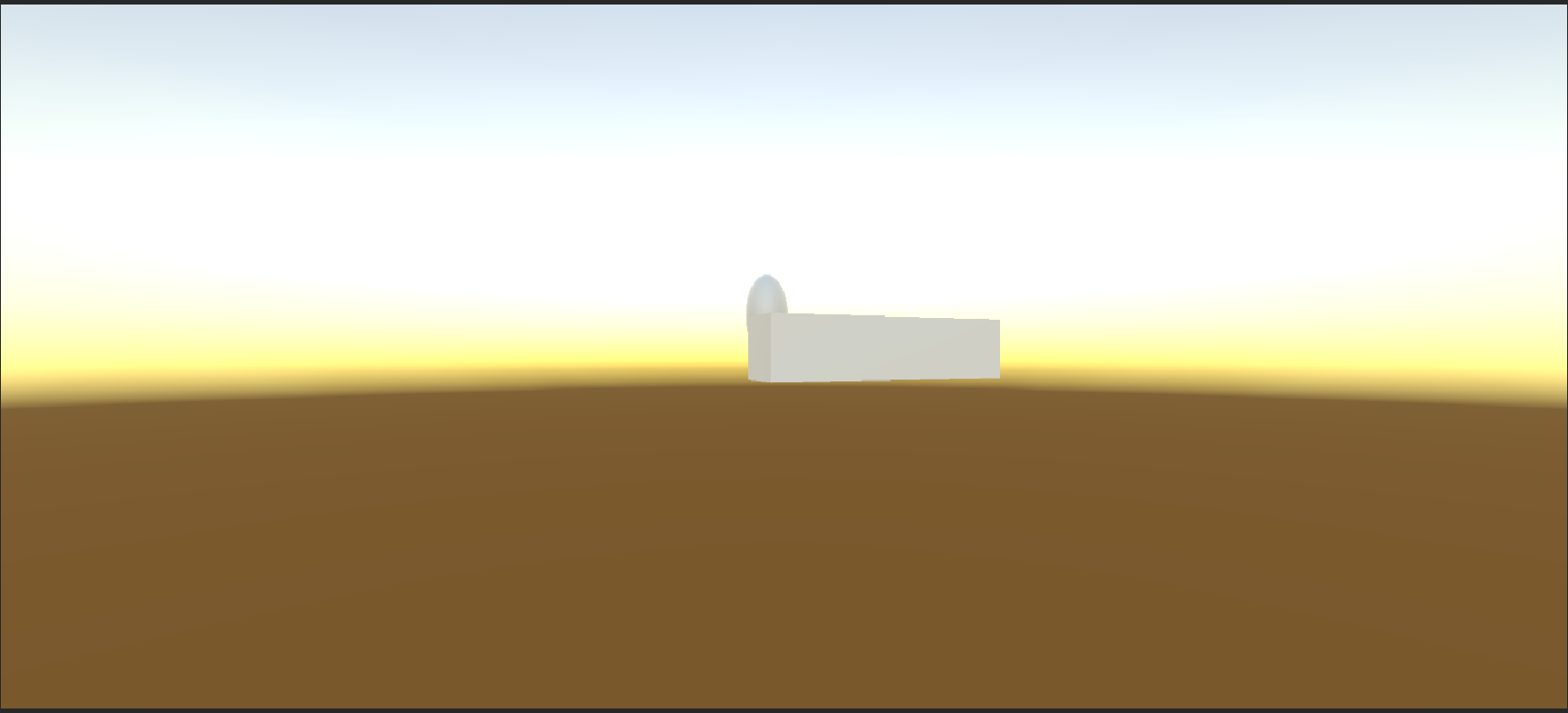
pivot.transform.Rotate(Vector3.up, orbitalRotationSpeed \* Time.deltaTime, Space.World);

}

}

Start 方法在脚本实例化并启用时调用一次。它检查 pivot 是否已经被指定，如果没有，则创建一个新的 GameObject 作为公转的中心点，并将其位置设置为 (0,0,10)。然后，它将当前物体（脚本所附加的GameObject）设置为 pivot 的子对象，这是实现公转效果的关键。

这样物体同时进行自转和公转。



1. **设置public变量**

public GameObject pivot; // 公转的中心点

public float selfRotationSpeed = 90.0f; // 自转速度，每秒旋转度数

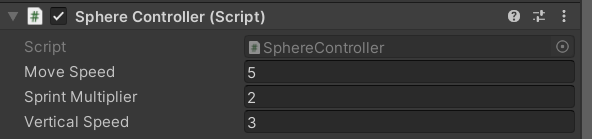
public float orbitalRotationSpeed = 45.0f; // 公转速度，每秒旋转度数

三个公共变量：

pivot：作为公转的中心点。

selfRotationSpeed：表示物体自转的速度（每秒旋转的度数）。

orbitalRotationSpeed: 表示物体围绕公转中心点旋转的速度。



1. **了解函数的调用顺序**

public class CubeController : MonoBehaviour

{

public GameObject pivot; // 公转的中心点

public float selfRotationSpeed = 90.0f; // 自转速度，每秒旋转度数

public float orbitalRotationSpeed = 45.0f; // 公转速度，每秒旋转度数

// Awake is called when the script instance is being loaded

void Awake()

{

// 确保 pivot 不为空

if (pivot == null)

{

// 如果未指定，创建一个新的 pivot 点

pivot = new GameObject("Pivot");

pivot.transform.position = new Vector3(0, 0, 10); // 设置公转中心点位置

}

}

// Start is called before the first frame update

void Start()

{

// 设置当前对象为 pivot 的子对象

transform.parent = pivot.transform;

}

// Update is called once per frame

void Update()

{

// 自转

transform.Rotate(Vector3.up, selfRotationSpeed \* Time.deltaTime, Space.Self);

// 公转

pivot.transform.Rotate(Vector3.up, orbitalRotationSpeed \* Time.deltaTime, Space.World);

}

}

在游戏开始加载时，Unity首先调用所有激活的游戏对象上的 Awake 方法。这确保了所有必需的初始化都在游戏逻辑开始之前完成。对于 CubeController 脚本，这意味着 pivot 的检查和初始化（如果需要）会最先进行。

接下来，Unity在首个 Update 调用之前，对所有已激活的游戏对象调用 Start 方法。这允许进行进一步的初始化，特别是那些依赖于其他对象已经完成 Awake 初始化的操作。对 CubeController 脚本而言，这个阶段会将物体设置为 pivot 的子对象。

在所有的 Start 方法调用之后，游戏进入运行状态，Unity将开始在每一帧调用 Update 方法。对于 CubeController 脚本，这意味着它将在每帧中执行自转和公转的逻辑。

1. **播放音乐**

public class SphereController : MonoBehaviour

{

public float moveSpeed = 5.0f; // 基础移动速度

public float sprintMultiplier = 2.0f; // 加速倍数

public float verticalSpeed = 3.0f; // 垂直移动速度

private AudioSource audioSource; // 用于播放音频的组件

// Start is called before the first frame update

void Start()

{

// 获取AudioSource组件

audioSource = GetComponent<AudioSource>();

}

// Update is called once per frame

void Update()

{

float speed = moveSpeed;

// 按住Shift键加速

if (Input.GetKey(KeyCode.LeftShift) || Input.GetKey(KeyCode.RightShift))

{

speed \*= sprintMultiplier;

}

// 获取WASD键的输入

float moveHorizontal = Input.GetAxis("Horizontal"); // AD键对应的水平移动

float moveVertical = Input.GetAxis("Vertical"); // WS键对应的前后移动

// 获取空格键和Ctrl键的输入进行上下移动

float moveUp = Input.GetKey(KeyCode.Space) ? 1.0f : 0.0f;

float moveDown = Input.GetKey(KeyCode.LeftControl) || Input.GetKey(KeyCode.RightControl) ? -1.0f : 0.0f;

// 根据输入计算移动方向

Vector3 movement = new Vector3(moveHorizontal, moveUp + moveDown, moveVertical);

// 更新物体的位置

transform.Translate(movement \* (movement.y == 0 ? speed : verticalSpeed) \* Time.deltaTime, Space.World);

// 检测鼠标左键点击

if (Input.GetMouseButtonDown(0)) // 0代表鼠标左键

{

// 播放音效

if (audioSource != null && !audioSource.isPlaying)

{

audioSource.Play();

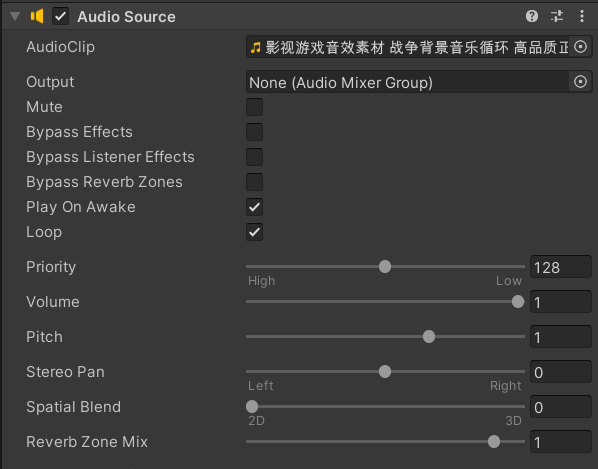
}

}

}

}

首先我从网络上下载了一段“biu~”的发射音效，并挂载在sphere上的音频，将音频文件设置为该组件的AudioClip。



在 Update 方法中，使用 Input.GetMouseButtonDown(0) 检查是否点击了鼠标左键（其中 0 表示鼠标左键）。如果检测到鼠标左键点击，并且 AudioSource 组件不为空且当前没有播放音频，则调用 audioSource.Play() 来播放音频。

1. **实验心得（如出现的具体错误类型，自己如何解决的）**
2. 创建脚本时默认只有start函数没有awake函数，不了解这两的区别，搜素资料学习了

Awake 与 Start 的不同

调用时机:

Awake: 在加载脚本实例时立即调用。这意味着当场景开始加载时，所有的 Awake 函数几乎是首先被调用的，而且是在所有对象的 Awake 方法完成之后，才会开始调用任何对象的 Start 方法。

Start: 在脚本组件首次激活之前的第一帧调用，但是在所有对象的 Awake 方法之后。如果脚本组件未激活（未勾选），则 Start 不会被调用，直到该组件被激活。

用途:

Awake 通常用于设置引用和状态，这些设置不依赖于其他对象的初始化代码。由于 Awake 保证在游戏开始之前调用，因此是设置单例或跨脚本引用的理想选择。

Start 可用于执行需要依赖于其他对象先行初始化的代码。

2.实验中许多效果时动态的或者需要音频形式的，难以通过截图的方式表现出来，我尝试了转成gif文件等方法任然不能嵌入到文件里，索性将所有完成后的效果录制一个demo视频，以供老师检阅