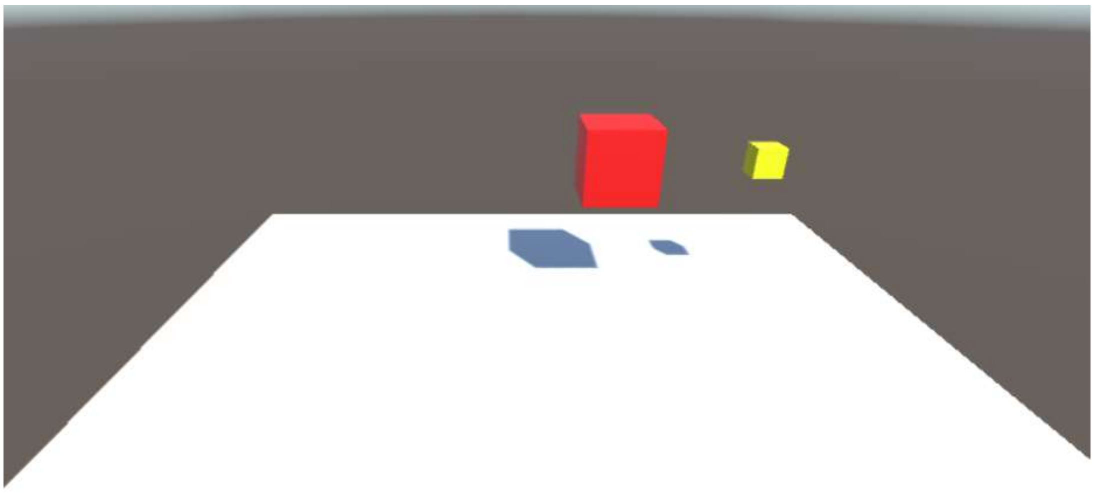
1.物体放置与上色：

如图所示放置一个**平面**和两个**立方体**，小立方体是大立方体大小的 **0.4** 倍，小立方体**相对于**大立方体坐标为 **Vector3(2,0,0)**。并为大方块涂上红色，小方块涂上黄色。



2.简单编程 **1**：*(编程题目可以使用末尾附录的 API，也可以使用其他方式实现)*

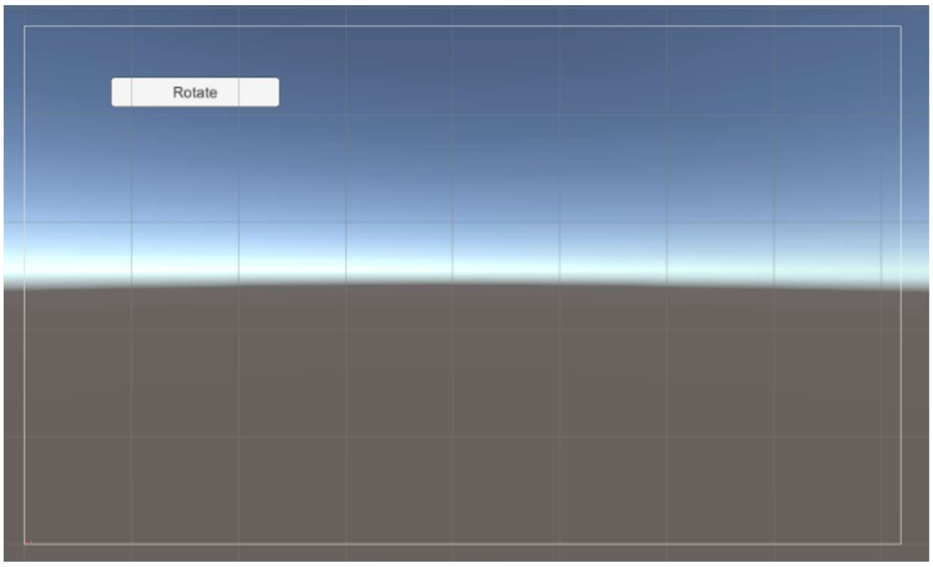
使大立方体以 **20.0** 的速度围绕**自身 y** 轴进行自转

3.简单编程 **2**：

使小立方体以 **25.5** 的速度围绕**大立方体**进行公转

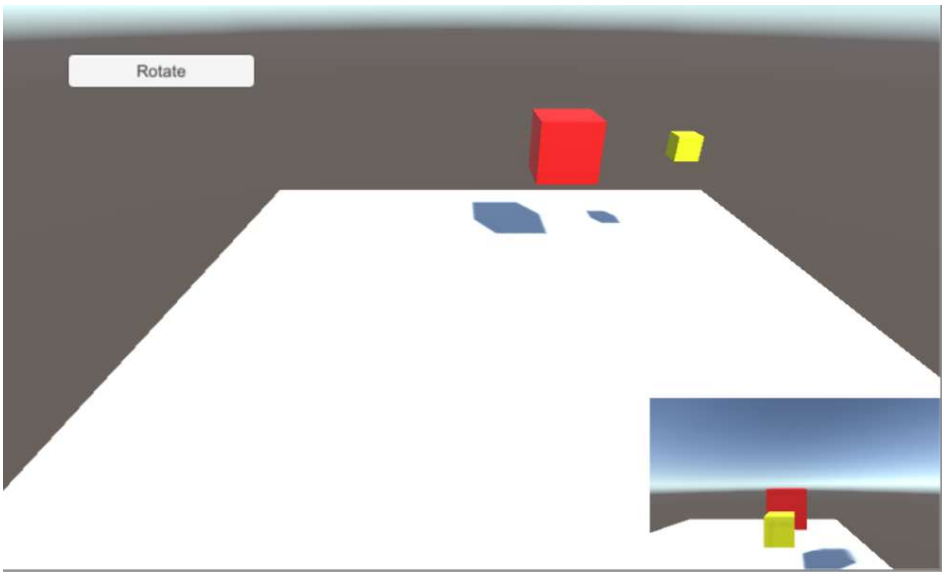
4.简单 **UI**：

创建一个 UI Button 并命名为 **Rotate**，放置在 UI 界面**左上角**，当**点击按钮**时候，小立方体开始围绕大立方体进行**公转**，在旋转时，再次点击按钮会**停止**公转。



5.添加摄像机：

创建第二个摄像机，拍摄立方体的侧面，并且显示在界面的右下角



附录：可能会用到的 API

Transform.Rotate

public void **Rotate**(Vector3 **eulers**, Space **relativeTo** = Space.Self);

Parameters

eulers	The rotation to apply.
relativeTo	Determines whether to rotate the GameObject either locally to the GameObject or relative to the Scene in world space.

Description

Applies a rotation of eulerAngles.z degrees around the z-axis, eulerAngles.x degrees around the x-axis, and eulerAngles.y degrees around the y-axis (in that order).

Rotate takes a Vector3 argument as an Euler angle. The second argument is the rotation axes, which can be set to local axis (Space.Self) or global axis (Space.World). The rotation is by the Euler amount.

public void **Rotate**(float **xAngle**, float **yAngle**, float **zAngle**, Space **relativeTo** = Space.Self);

Parameters

relativeTo	Determines whether to rotate the GameObject either locally to the GameObject or relative to the `Scene` in world space.
xAngle	Degrees to rotate the GameObject around the X axis.
yAngle	Degrees to rotate the GameObject around the Y axis.
zAngle	Degrees to rotate the GameObject around the Z axis.

Description

The implementation of this method applies a rotation of `zAngle` degrees around the z axis, `xAngle` degrees around the x axis, and `yAngle` degrees around the y axis (in that order).

Rotate can have the euler angle specified in 3 floats for x, y, and z.

The example shows two cubes: one cube uses Space.Self (the local space and axes of the GameObject) and the other uses Space.World (the space and axes in relation to the /Scene/). They are both first rotated 90 in the X axis so they are not aligned with the world axis by default. Use the xAngle, yAngle and zAngle values exposed in the inspector to see how different rotation values apply to both cubes. You might notice the way the cubes visually rotate is dependant on the current orientation and Space option used. Play around with the values while selecting the cubes in the scene view to try to understand how the values interact.

public void **Rotate**(Vector3 **axis**, float **angle**, Space **relativeTo** = Space.Self);

Parameters

angle	The degrees of rotation to apply.
axis	The axis to apply rotation to.
relativeTo	Determines whether to rotate the GameObject either locally to the GameObject or relative to the Scene in world space.

Description

Rotates the object around the given axis by the number of degrees defined by the given angle.

Rotate has an axis, angle and the local or global parameters. The rotation axis can be in any direction.

Transform.RotateAround

SWITCH TO MANUAL

public void **RotateAround**(Vector3 **point**, Vector3 **axis**, float **angle**);

Description

Rotates the transform about `axis` passing through `point` in world coordinates by `angle` degrees.

This modifies both the position and the rotation of the transform.

```
using UnityEngine;

//Attach this script to a GameObject to rotate around the target position.
public class Example : MonoBehaviour
{
    private Vector3 target = new Vector3(5.0f, 0.0f, 0.0f);

    void Update()
    {
        // Spin the object around the world origin at 20 degrees/second.
        transform.RotateAround(target, Vector3.up, 30 * Time.deltaTime);
    }
}
```