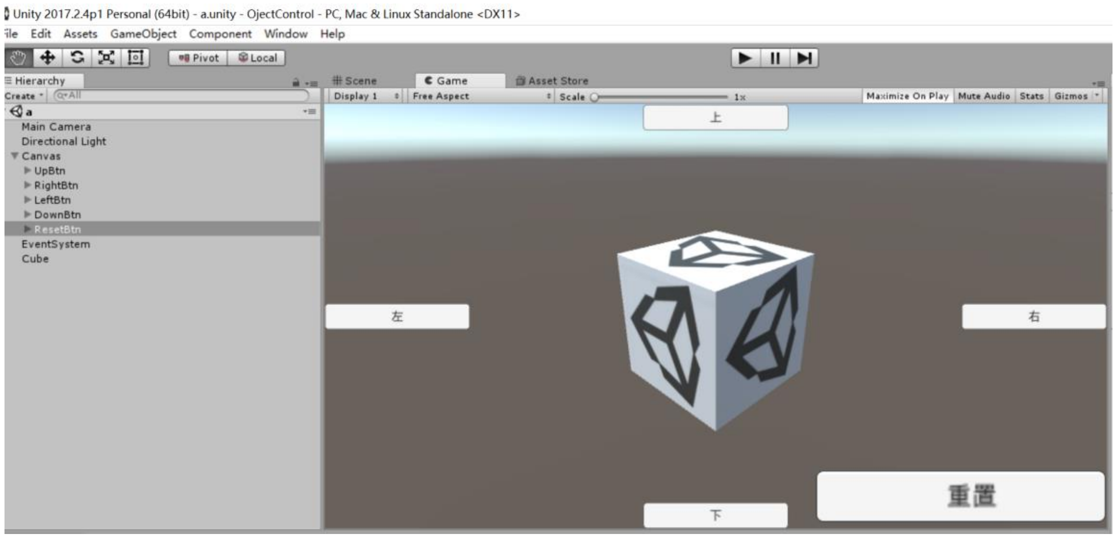
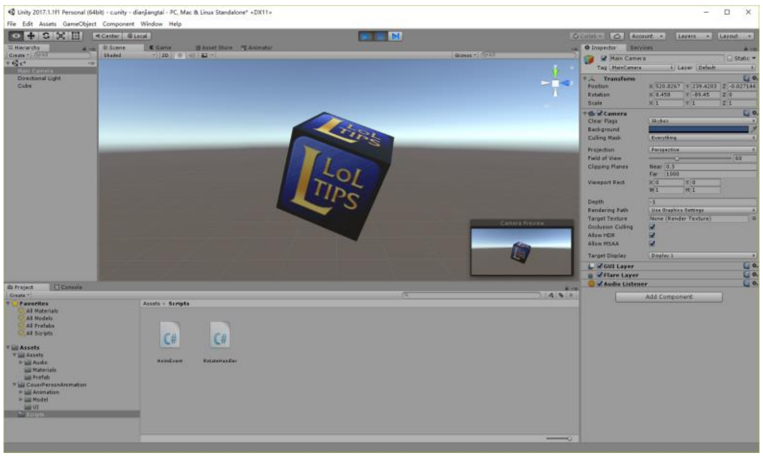
模型对象的操作：缩放、旋转、移动、复位


Unity 2017.2.4p1 Personal (64bit) - a.unity - OjectControl - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Pivot
Local
Hierarchy
Main Camera
Directional Light
Canvas
UpBtn
RightBtn
LeftBtn
DownBtn
ResetBtn
EventSystem
Cube
Scene
Game
Asset Store
上
左
右
下
重置

1、 鼠标缩放。

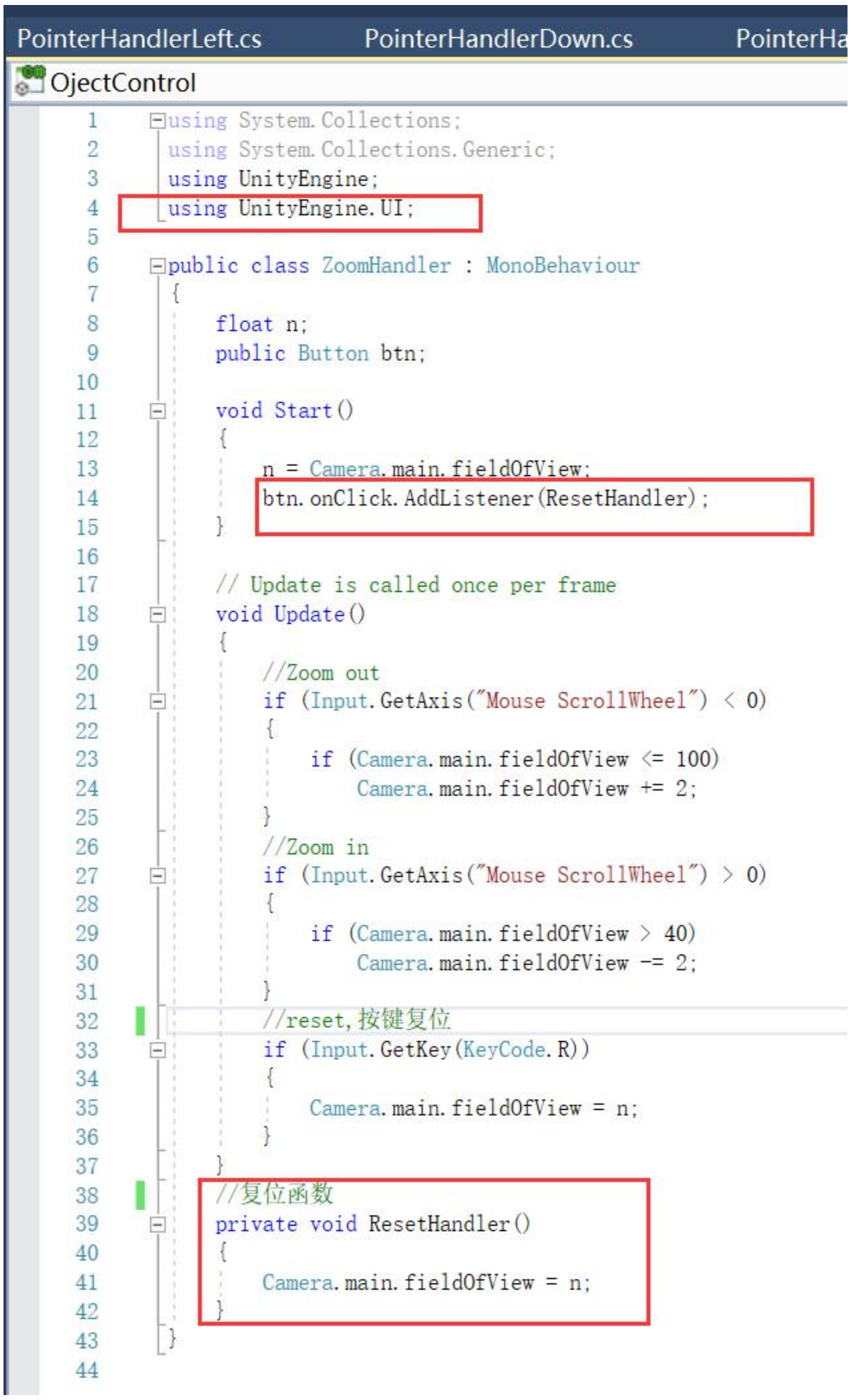
创建脚本 ZoomHandler，并绑定到 Cube 模型对象。



方法一、利用摄像机视角拉伸来调节视野大小，将脚本绑定至 Cube

```
//ZoomHandler.CS
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ZoomHandler : MonoBehaviour {
    float n;

    void Start()
    {
        n = Camera.main.fieldOfView;
    }

     // Update is called once per frame
    void Update () {
        //Zoom out
        if (Input.GetAxis("Mouse ScrollWheel") < 0)
        {
            if (Camera.main.fieldOfView <= 100)
                Camera.main.fieldOfView += 2;
        }
        //Zoom in
        if (Input.GetAxis("Mouse ScrollWheel") > 0)
        {
            if (Camera.main.fieldOfView > 40)
                Camera.main.fieldOfView -= 2;
        }
        //reset
        if (Input.GetKey(KeyCode.R))
        {
            Camera.main.fieldOfView = n;
        }
    }
}
```

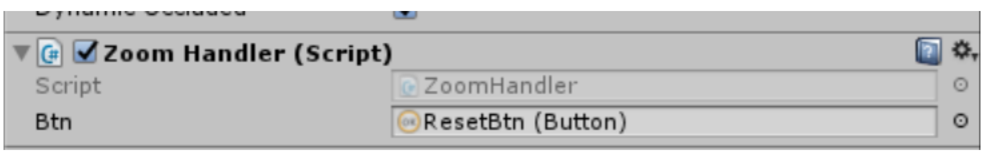
修改脚本，增加按钮控制复位。
注意标红的区域，其为利用按钮 UI 进行控制的关键代码。



PointerHandlerLeft.cs　PointerHandlerDown.cs　PointerHa

OjectControl

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

public class ZoomHandler : MonoBehaviour
{
    float n;
    public Button btn;

    void Start()
    {
        n = Camera.main.fieldOfView;
        btn.onClick.AddListener(ResetHandler);
    }

    // Update is called once per frame
    void Update()
    {
        //Zoom out
        if (Input.GetAxis("Mouse ScrollWheel") < 0)
        {
            if (Camera.main.fieldOfView <= 100)
                Camera.main.fieldOfView += 2;
        }
        //Zoom in
        if (Input.GetAxis("Mouse ScrollWheel") > 0)
        {
            if (Camera.main.fieldOfView > 40)
                Camera.main.fieldOfView -= 2;
        }
        //reset,按键复位
        if (Input.GetKey(KeyCode.R))
        {
            Camera.main.fieldOfView = n;
        }
    }
    //复位函数
    private void ResetHandler()
    {
        Camera.main.fieldOfView = n;
    }
}
```



Zoom Handler (Script)

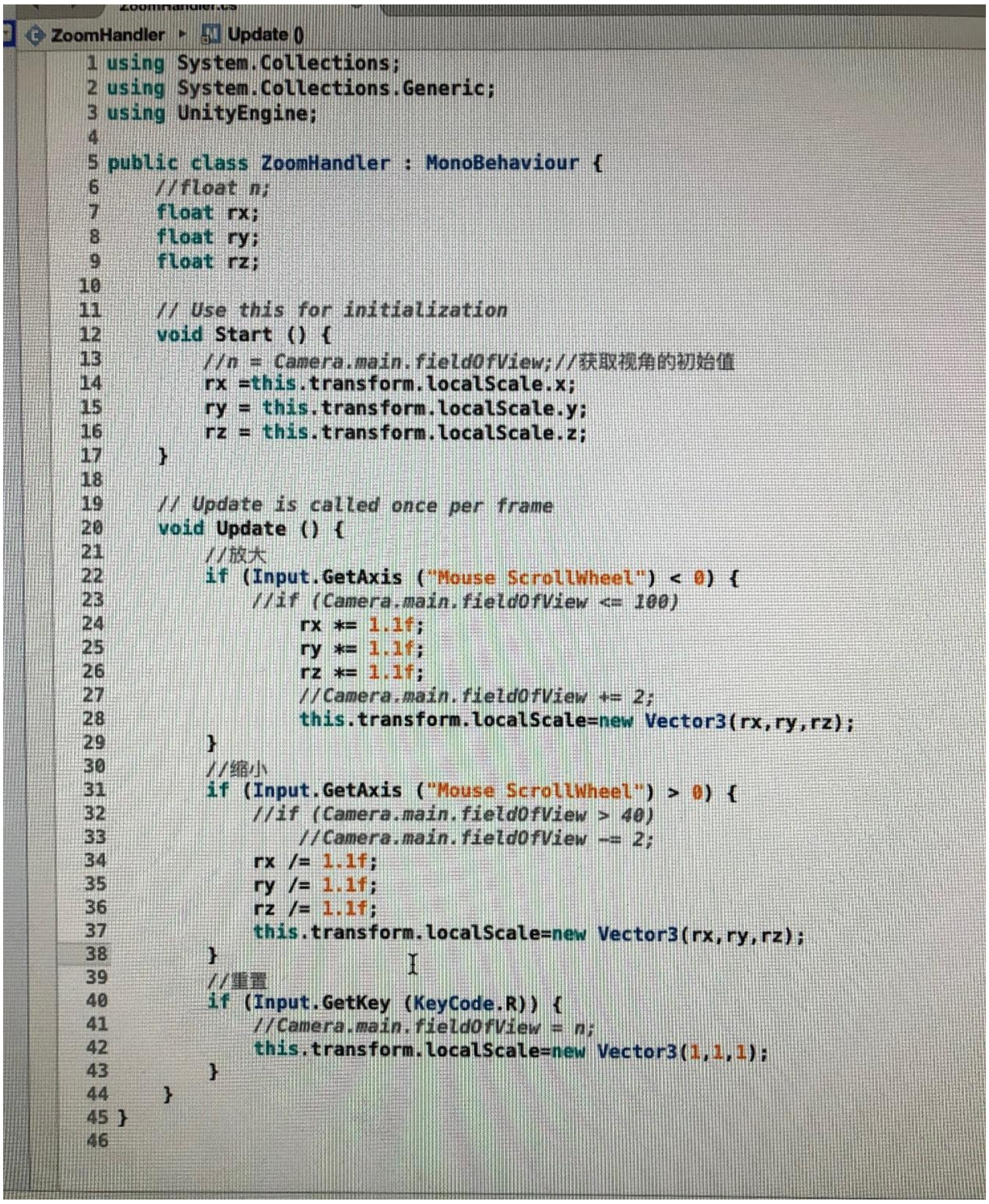
Script　ZoomHandler

Btn　ResetBtn (Button)

方法二、利用模型比例缩放来调节对象大小

//ZoomHandler.CS

//对代码稍作修改即可实现



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

public class ZoomHandler : MonoBehaviour {
    //float n;
    float rx;
    float ry;
    float rz;

    // Use this for initialization
    void Start () {
        //n = Camera.main.fieldOfView;//获取视角的初始值
        rx =this.transform.localScale.x;
        ry = this.transform.localScale.y;
        rz = this.transform.localScale.z;
    }

    // Update is called once per frame
    void Update () {
        //放大
        if (Input.GetAxis ("Mouse ScrollWheel") < 0) {
            //if (Camera.main.fieldOfView <= 100)
                rx *= 1.1f;
                ry *= 1.1f;
                rz *= 1.1f;
                //Camera.main.fieldOfView += 2;
                this.transform.localScale=new Vector3(rx,ry,rz);
        }
        //缩小
        if (Input.GetAxis ("Mouse ScrollWheel") > 0) {
            //if (Camera.main.fieldOfView > 40)
                //Camera.main.fieldOfView -= 2;
            rx /= 1.1f;
            ry /= 1.1f;
            rz /= 1.1f;
            this.transform.localScale=new Vector3(rx,ry,rz);
        }
        //重置
        if (Input.GetKey (KeyCode.R)) {
            //Camera.main.fieldOfView = n;
            this.transform.localScale=new Vector3(1,1,1);
        }
    }
}
```

针对按钮复位功能，参考方法一

2、 键盘控制旋转

```
//rotateHandler.CS
//脚本绑定至 Cube
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
//using UnityEngine.UI;

public class RotateHandler : MonoBehaviour
{
    /*
    public Button upBtn;
     public Button downBtn;
     public Button leftBtn;
     public Button rightBtn;
    */
     int flag=0;

    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        //修改键盘指令
        if (Input.GetKeyDown(KeyCode.A))
        {
            flag = 1;
        }
        else if (Input.GetKeyUp(KeyCode.A))
        {
            flag = 0;
        }
        if (Input.GetKeyDown(KeyCode.D))
        {
            flag = 2;
        }
        else if (Input.GetKeyUp(KeyCode.D))
        {
            flag = 0;
        }
        if (Input.GetKeyDown(KeyCode.W))
```

注意：该函数只能作为键盘控制旋转以及复位，若要采用按钮控制还要单独制作，参考后续步骤。

```
        {
            flag = 3;
        }
        else if (Input.GetKeyUp(KeyCode.W))
        {
            flag = 0;
        }
        if (Input.GetKeyDown(KeyCode.S))
        {
            flag = 4;
        }
        else if (Input.GetKeyUp(KeyCode.S))
        {
            flag = 0;
        }
        //旋转
        if (flag == 1)
        {
            transform.Rotate(Vector3.up * 100 * Time.deltaTime);//left
        }
        else if (flag == 2)
        {
            transform.Rotate(Vector3.down * 100 * Time.deltaTime);//right
        }
        else if (flag == 3)
        {
            transform.Rotate(Vector3.left * 100 * Time.deltaTime);//up
        }
        else if (flag == 4)
        {
            transform.Rotate(Vector3.right * 100 * Time.deltaTime);//right
        }
        //复原
        if (Input.GetKeyDown(KeyCode.E))
        {
            transform.localEulerAngles = new Vector3(0, 0, 0);
        }
    }
}
```

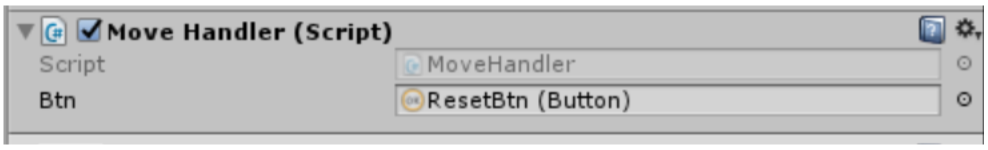
注意：该函数只能作为键盘控制旋转以及复位，若要采用按钮控制还要单独制作，参考后续步骤。

3、 鼠标控制移动

```
//MoveHandler.CS
//脚本绑定至 Cube
using UnityEngine;
using System.Collections;

public class MoveHandler : MonoBehaviour
{

    // Use this for initialization
    void Start()
    {
        StartCoroutine(OnMouseDown());
    }

    IEnumerator OnMouseDown()
    {
        //将物体由世界坐标系转换为屏幕坐标系
        Vector3 screenSpace = 
Camera.main.WorldToScreenPoint(transform.position);//三维物体坐标转屏幕坐标
        //完成两个步骤 1.由于鼠标的坐标系是2维，需要转换成3维的世界坐标系
        //    //           2.只有3维坐标情况下才能来计算鼠标位置与物理的距离，offset
即是距离
        //将鼠标屏幕坐标转为三维坐标，再算出物体位置与鼠标之间的距离
        Vector3 offset = transform.position - Camera.main.ScreenToWorldPoint(new 
Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z));
        while (Input.GetMouseButton(0))
        {
            //得到现在鼠标的2维坐标系位置
            Vector3 curScreenSpace = new Vector3(Input.mousePosition.x, 
Input.mousePosition.y, screenSpace.z);
            //将当前鼠标的2维位置转换成3维位置，再加上鼠标的移动量
            Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenSpace) + 
offset;
            //curPosition就是物体应该的移动向量赋给transform的position属性
            transform.position = curPosition;
            yield return new WaitForFixedUpdate(); //这个很重要，循环执行
        }
    }
}
```

接下来，我们添加 1 个 Button，制作重置坐标功能的按钮。

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class MoveHandler : MonoBehaviour
{
    public Button btn;
    float dx;
    float dy;
    float dz;

    // Use this for initialization
    void Start()
    {
        StartCoroutine(OnMouseDown());
        //按钮监听，记录原始坐标
        btn.onClick.AddListener(reset);
        dx = transform.position.x;
        dy = transform.position.y;
        dz = transform.position.z;
    }

    IEnumerator OnMouseDown()
    {
        //将物体由世界坐标系转换为屏幕坐标系
        Vector3 screenSpace =
Camera.main.WorldToScreenPoint(transform.position);//三维物体坐标转屏幕坐标
        //完成两个步骤 1.由于鼠标的坐标系是2维，需要转换成3维的世界坐标系
        //    //             2.只有3维坐标情况下才能来计算鼠标位置与物理的距离，offset
即是距离
        //将鼠标屏幕坐标转为三维坐标，再算出物体位置与鼠标之间的距离
        Vector3 offset = transform.position - Camera.main.ScreenToWorldPoint(new
Vector3(Input.mousePosition.x, Input.mousePosition.y, screenSpace.z));
        while (Input.GetMouseButton(0))
        {
            //得到现在鼠标的2维坐标系位置
            Vector3 curScreenSpace = new Vector3(Input.mousePosition.x,
Input.mousePosition.y, screenSpace.z);
            //将当前鼠标的2维位置转换成3维位置，再加上鼠标的移动量
            Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenSpace) +
offset;
            //curPosition就是物体应该的移动向量赋给transform的position属性
            transform.position = curPosition;
            yield return new WaitForFixedUpdate(); //这个很重要，循环执行
        }
    }

    //重置
    void reset()
    {
        transform.localPosition = new Vector3(dx, dy, dz);
    }
}
```

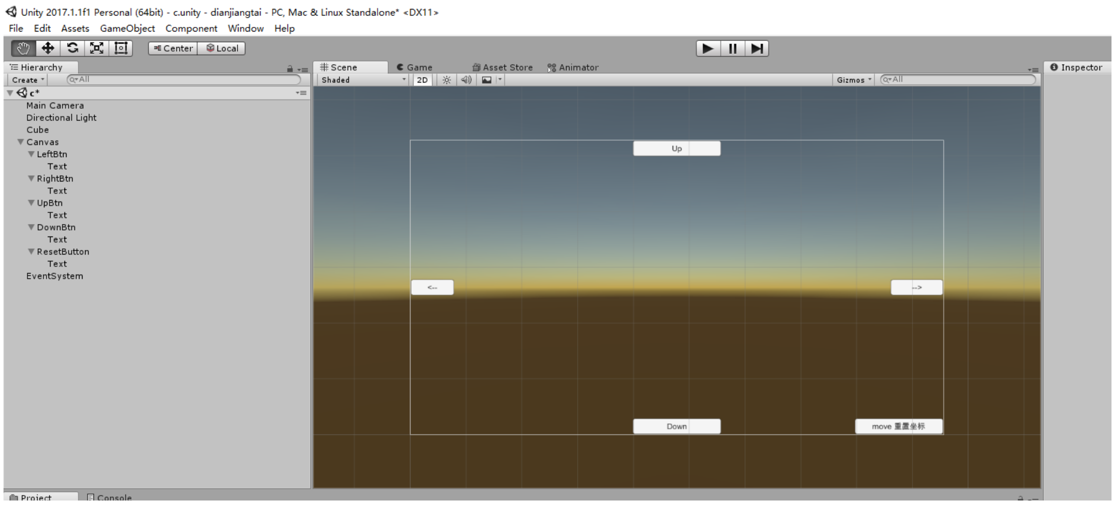


Move Handler (Script)

Script	MoveHandler
Btn	ResetBtn (Button)

4、【测试】关于按钮的长按与弹起

//PointHandler.CS

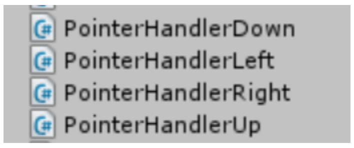


```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;

public class PointHander : MonoBehaviour,IPointerDownHandler,IPointerUpHandler
{

//将脚本绑定到需要监听鼠标长按与弹起的对象上即可

    void IPointerDownHandler.OnPointerDown(PointerEventData eventData)
    {
        //throw new NotImplementedException();
        Debug.Log("Btn Down OK");
    }

    void IPointerUpHandler.OnPointerUp(PointerEventData eventData)
    {
       // throw new NotImplementedException();
        Debug.Log("Button UP OK");
    }
}
```

5、鼠标单击按钮控制旋转

取消步骤 4 的 PointHandler.CS

创 建 4 个 脚 本 ： PointerHandlerLeft 、 PointerHandlerRight 、 PointerHandlerUp 、 PointerHandlerDown，绑定至 Cube


PointerHandlerDown
PointerHandlerLeft
PointerHandlerRight
PointerHandlerUp

// 编写脚本 PointerHandlerLeft.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
//将本脚本绑定到需要监听鼠标长按与弹起的对象上即可

public class PointerHandlerLeft : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    bool flag = false;
    public GameObject obj;
    public Button btn;


    void Update()
    {
        if (flag == true)
        {
            obj.transform.Rotate(Vector3.up * 100 * Time.deltaTime);//left
        }
        btn.onClick.AddListener(resetHandler);
    }
    void IPointerDownHandler.OnPointerDown(PointerEventData eventData)
    {
        //throw new NotImplementedException();
        Debug.Log("Button PressDown OK");
        flag = true;
    }

    void IPointerUpHandler.OnPointerUp(PointerEventData eventData)
    {
        // throw new NotImplementedException();
        Debug.Log("Button PressUp OK");
        flag = false;
    }


    void resetHandler() {
        obj.transform.localEulerAngles = new Vector3(0, 0, 0);
    }

}
```

以下三个控制脚本原理一致，只要修改个别参数即可。

//编写脚本 PointerHandlerRight.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
//将本脚本绑定到需要监听鼠标长按与弹起的对象上即可

public class PointerHandlerRight : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    bool flag = false;
    public GameObject obj;
    public Button btn;


    void Update()
    {
        if (flag == true)
        {
            obj.transform.Rotate(Vector3.down * 100 * Time.deltaTime);//left
        }
        btn.onClick.AddListener(resetHandler);
    }

    void IPointerDownHandler.OnPointerDown(PointerEventData eventData)
    {
        //throw new NotImplementedException();
        Debug.Log("Button PressDown OK");
        flag = true;
    }

    void IPointerUpHandler.OnPointerUp(PointerEventData eventData)
    {
        // throw new NotImplementedException();
        Debug.Log("Button PressUp OK");
        flag = false;
    }


    void resetHandler()
    {
        obj.transform.localEulerAngles = new Vector3(0, 0, 0);
    }

}
```

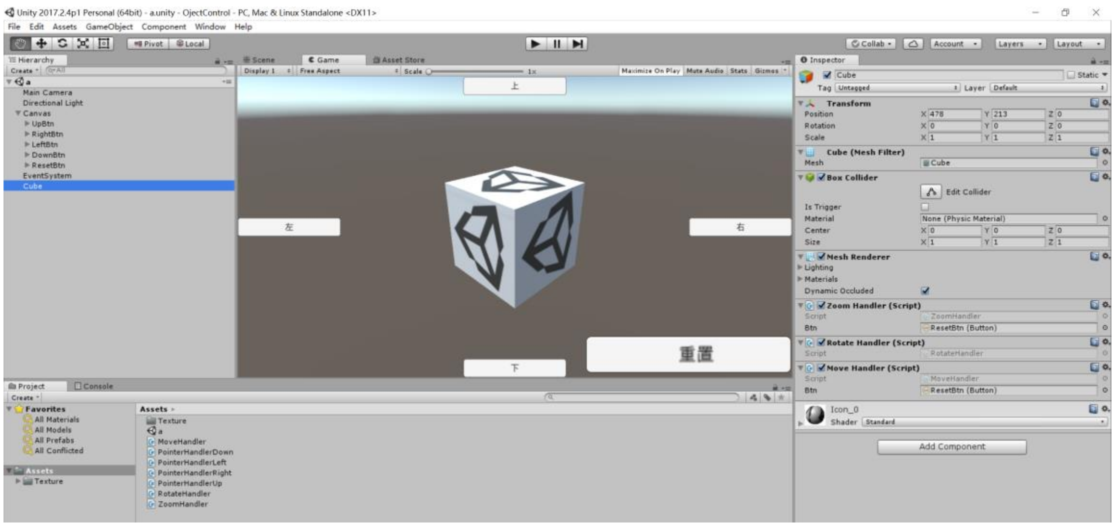
//编写脚本 PointerHandlerUp.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
//将本脚本绑定到需要监听鼠标长按与弹起的对象上即可

public class PointerHandlerUp : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    bool flag = false;
    public GameObject obj;
    public Button btn;
    void Update()
    {
        if (flag == true)
        {
            obj.transform.Rotate(Vector3.left * 100 * Time.deltaTime);//left
        }
        btn.onClick.AddListener(resetHandler);
    }

    void IPointerDownHandler.OnPointerDown(PointerEventData eventData)
    {
        //throw new NotImplementedException();
        Debug.Log("Button PressDown OK");
        flag = true;
    }

    void IPointerUpHandler.OnPointerUp(PointerEventData eventData)
    {
        // throw new NotImplementedException();
        Debug.Log("Button PressUp OK");
        flag = false;
    }



    void resetHandler()
    {
        obj.transform.localEulerAngles = new Vector3(0, 0, 0);
    }

}
```

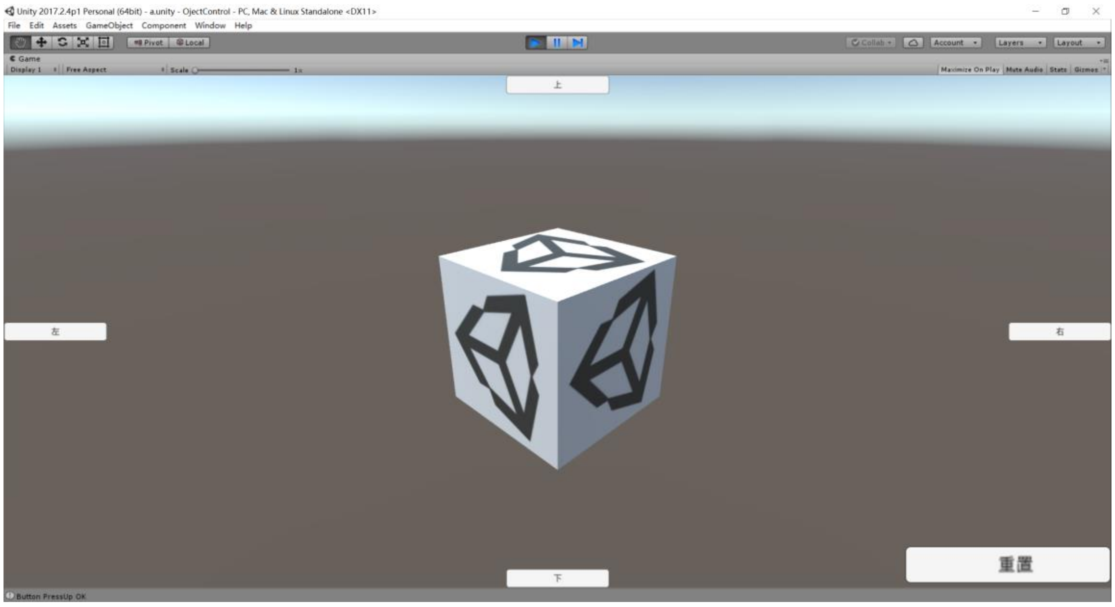
//编写脚本 PointerHandlerDown.CS

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
//将本脚本绑定到需要监听鼠标长按与弹起的对象上即可

public class PointerHandlerDown : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    bool flag = false;
    public GameObject obj;
    public Button btn;


    void Update()
    {
        if (flag == true)
        {
            obj.transform.Rotate(Vector3.right * 100 * Time.deltaTime);//left
        }
        btn.onClick.AddListener(resetHandler);
    }

    void IPointerDownHandler.OnPointerDown(PointerEventData eventData)
    {
        //throw new NotImplementedException();
        Debug.Log("Button PressDown OK");
        flag = true;
    }

    void IPointerUpHandler.OnPointerUp(PointerEventData eventData)
    {
        // throw new NotImplementedException();
        Debug.Log("Button PressUp OK");
        flag = false;
    }



    void resetHandler()
    {
        obj.transform.localEulerAngles = new Vector3(0, 0, 0);
    }

}
```

调试，运行



Unity 2017.2.4p1 Personal (64bit) - a.unity - OjectControl - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
Main Camera
Directional Light
Canvas
UpBtn
RightBtn
LeftBtn
DownBtn
ResetBtn
EventSystem
Cube
上
左
右
下
重置
Inspector
Transform
Box Collider
Mesh Renderer
Zoom Handler (Script)
Rotate Handler (Script)
Move Handler (Script)
Add Component
Assets
MoveHandler
PointerHandlerDown
PointerHandlerLeft
PointerHandlerRight
PointerHandlerUp
RotateHandler
ZoomHandler



Unity 2017.2.4p1 Personal (64bit) - a.unity - OjectControl - PC, Mac & Linux Standalone <DX11>
上
左
右
下
重置
Button PressUp OK