


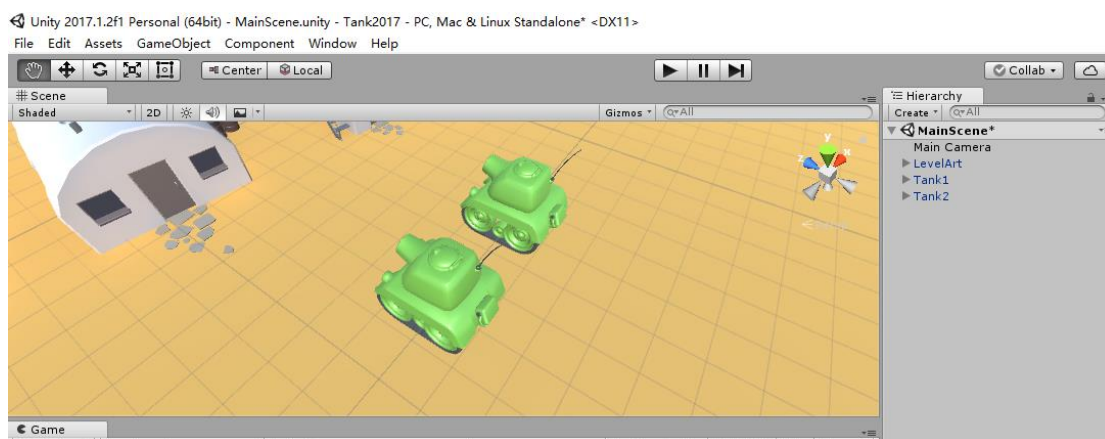
Course3 Two Tank movement and Camera Control

备注：群里已经上传到第三次课为止的全部项目，文件名：
Tank2017_第三次课_完整项目(含代码).unitypackage。大家若是前几次课未跟上，可以下载这个 unitypackage,在项目中导入进去就行（如何导入：Asset->Import Package->customer package，找到下载的文件，导入就行了）。此外可能 Input 输入没有设置会让 tank 无法跑起来，下面详细有说明如何改 Input 输入。

1.Add Player2

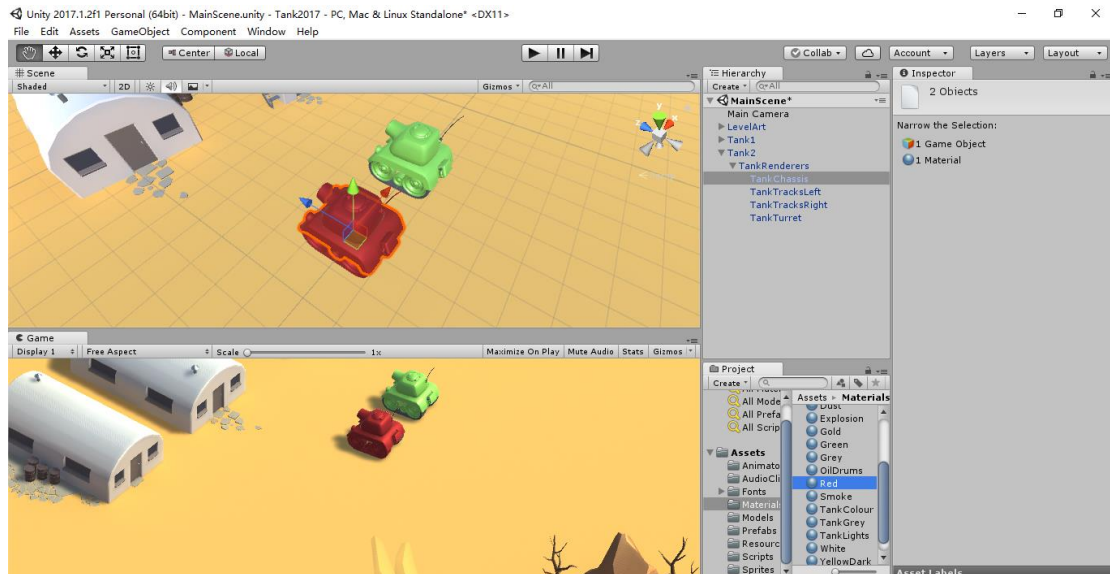
OK，我们在上节课的时候完成了对一个 tank 对象的移动，现在，让我们创建两个 tank 对象，并让他们都跑起来吧。

将原先的 Tank 重命名为 Tank1，选中该物体，右键在弹出的列表中选择 duplicate，此时会复制出一个新 tank，将其命名为 Tank2，由于复制而产生的物体与原物体重合，我们使用位移工具 ，对应快捷键 w（请关闭输入法），将两者分开：

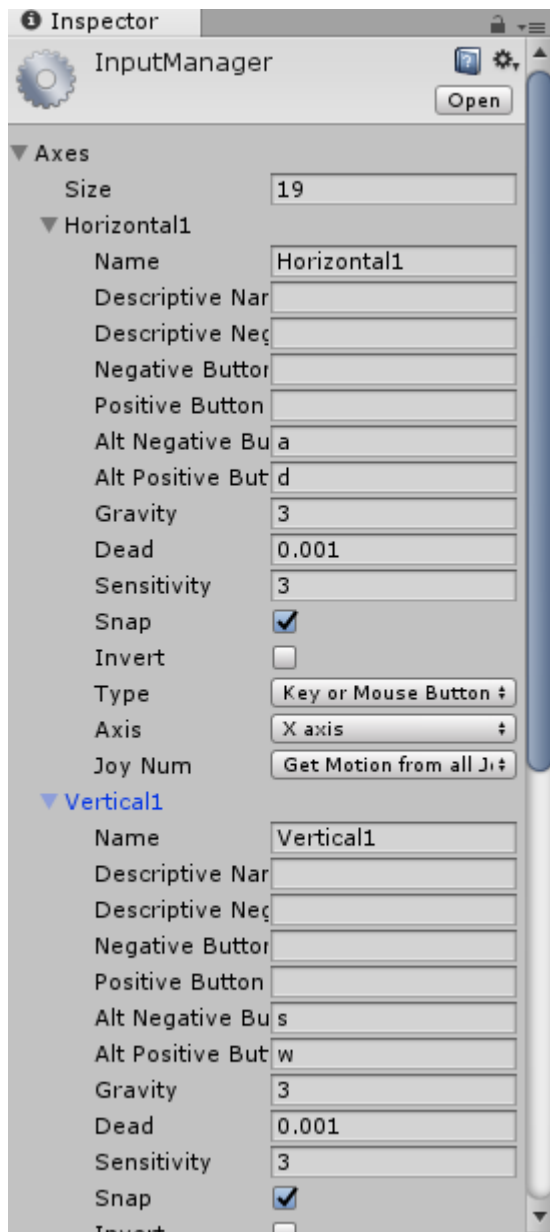


选中 Tank2->TankRenderers,拖动 materials 文件夹中的 red 给其下的四

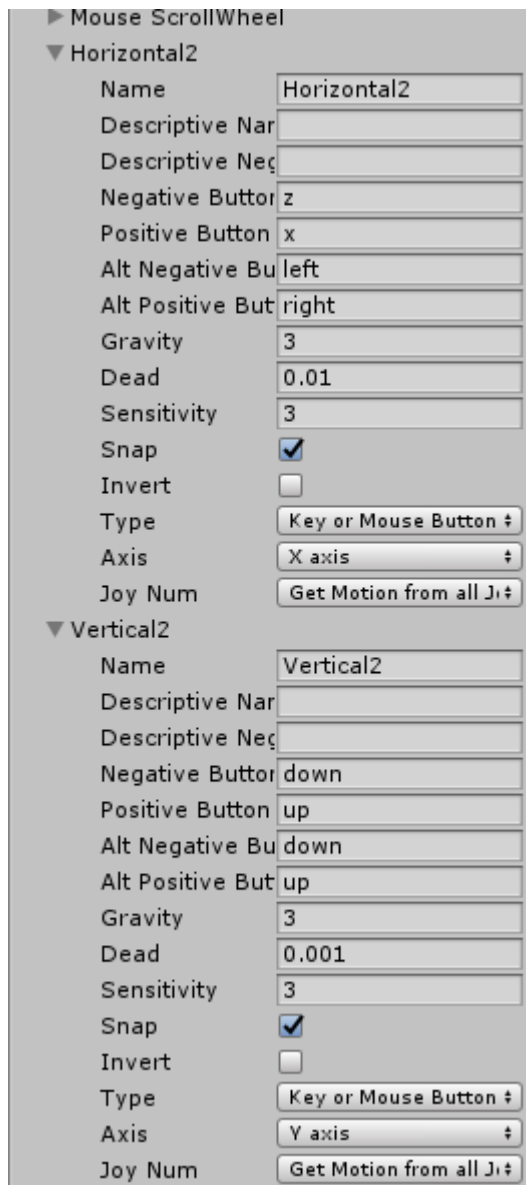
个子物体上：



然后在左上方,选中: Edit->ProjectSettings->Input, 在右边出现的 Axes 中打开我们上节课设置的 Horizontal1 与 vertical1, 将其中 NegativeButton 与 PositiveButton 的内容删掉:



在下面的新 Horizontal 与 vertical 里进行重命名与设置，将上一步骤中删掉的东西写到这里面去，并将其命名与 Horizontal2 与 vertical2:



然后点击运行，看看会出现什么情况？

两个 tank 一起移动，这是因为我们是复制的 tank1，因此在两个 tank 物体上都有我们上节课写到的 TankmMove 脚本，而脚本中有这么一行：

```
void Update () {  
    m_movementValue = Input.GetAxis("Vertical");  
    m_rotateValue = Input.GetAxis("Horizontal");  
}
```

因此,tank1 与 tank2 都是受到 wasd 的控制,因此我们在此修改逻辑:

```
// Update is called once per frame
void Update () {
    //m_movementValue = Input.GetAxis("Vertical");
    // m_rotateValue = Input.GetAxis("Horizontal");
    m_movementValue = Input.GetAxis("Vertical"+(gameObject.name.Replace("Tank","")));
    m_rotateValue = Input.GetAxis("Horizontal"+(gameObject.name.Replace("Tank","")));
}
```

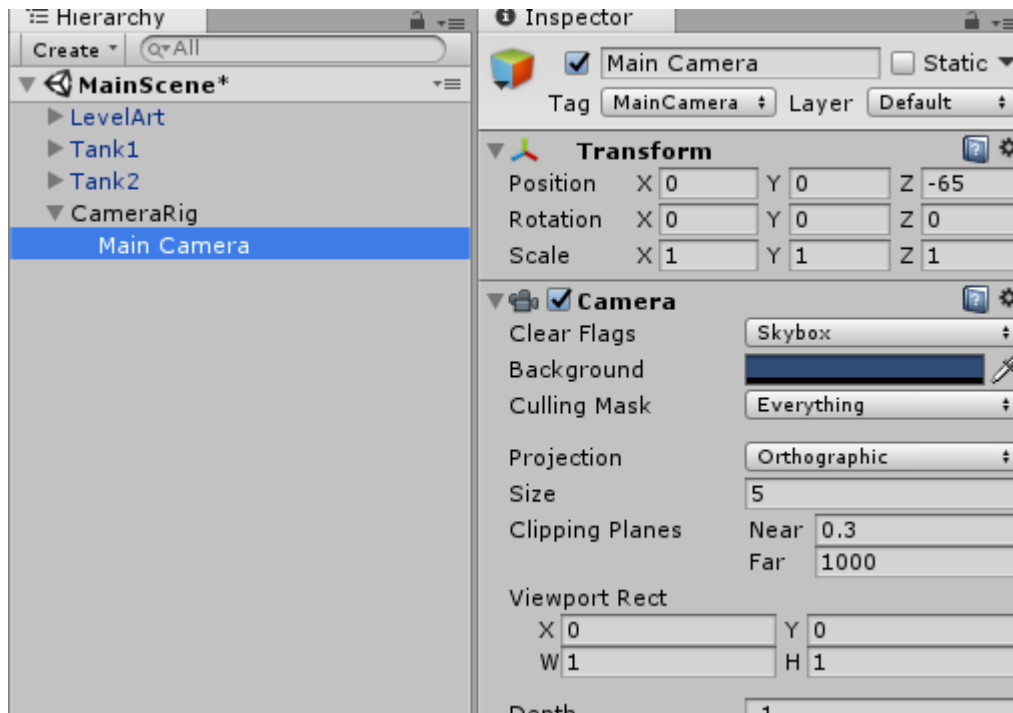
（记得修改代码后保存代码，不然不会有效果）

运行一下，wasd 可以控制 Tank1，上下左右可以控制 Tank2 啦！

2.Camera Control

接下来，我们要开始控制 camera 的位置，让它总是可以看到我们的两个 tank（即摄像机也应该随 tank 位置的变化而开始移动）

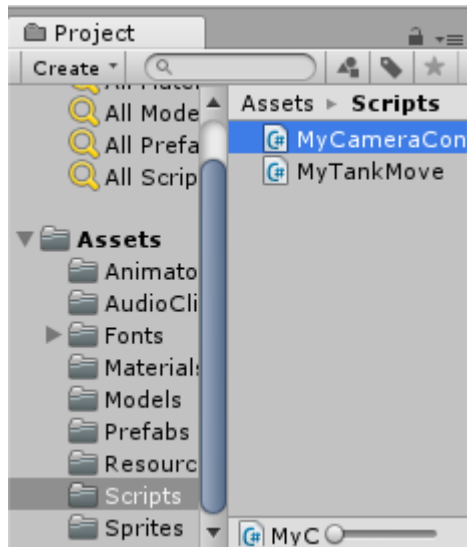
首先，创建一个空物体，命名为 CameraRig，reset 其 position（点击 position 组件右边一个小齿轮，选择 reset，即可），然后将其 rotation 设为（40,60,0）；将 MainCamera 拖到其下作为它的子物体，reset 一下，然后将 position 设置为（0,0, -65），其 Camera 组件成员 Culling Mask 改为 Orthographic



然后在 Game 视图里，你应该看到的是这种，如若不是，看 LevelArtposition 是不是不对（应该是 (0,0,0)），或是其他位置设置错误。



然后在 Scripts 文件夹中新建一个脚本，名为 MyCameraControll。



打开，键入以下代码：

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

public class MyCameraControl : MonoBehaviour {

    public float m_DampTime = 0.2f;           //摄像机移动时间
    public float m_ScreenEdgeBuffer = 4f;     //
    public float m_MinSize = 6.5f;           //摄像机Size最小值
    public Transform[] m_Players;             //存储两个Player的位置信息

    private Camera m_camera;                  //对子物体摄像机的引用
    private float m_ZoomSpeed;                //摄像机size变换速度
    private Vector3 m_MoveVelocity;           //摄像机移动速度
    private Vector3 m_DesiredPosition;        //CameraRig要移动到的目标位置

    // Use this for initialization
    void Start () {
        m_camera = GetComponentInChildren<Camera>();
        m_Players = new Transform[2];
        m_Players[0] = GameObject.Find("Tank1").GetComponent<Transform>();
        m_Players[1] = GameObject.Find("Tank2").GetComponent<Transform>();
    }

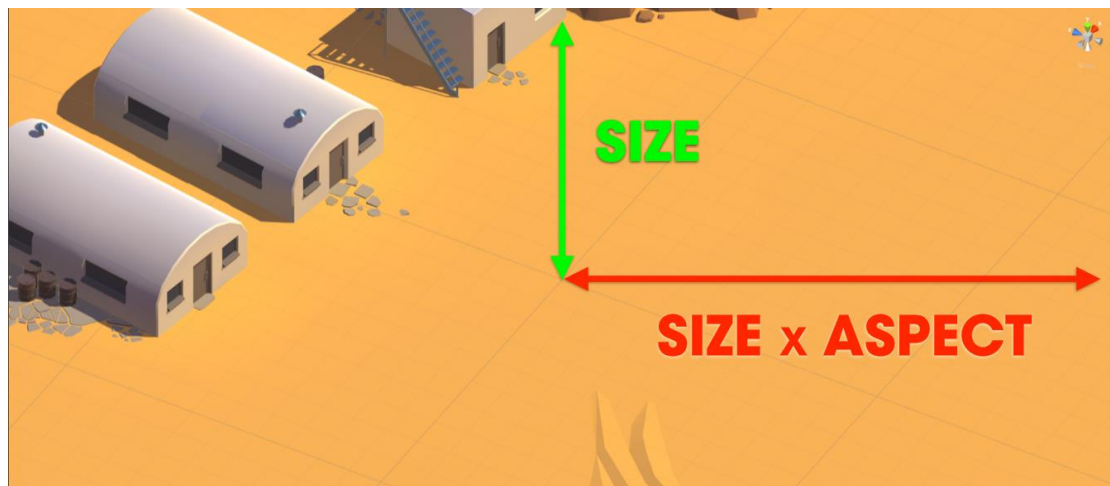
    private void FixedUpdate()
    {
        Move();                               //控制cameraRig移动
        Zoom();                               //控制camera的size大小
    }

    private void Move()
    {
        FindAveragePosition();
        transform.position = Vector3.SmoothDamp(transform.position, m_DesiredPosition, ref m_MoveVelocity, m_DampTime);
    }
}
```

```

34
35 private void FindAveragePosition()
36 {
37     Vector3 averagePos = new Vector3();
38     for(int i = 0; i < m_Players.Length; i++)
39     {
40         averagePos += m_Players[i].position;
41     }
42     averagePos /= 2;
43     averagePos.y = transform.position.y;
44     m_DesiredPosition = averagePos;
45 }
46 private void Zoom()
47 {
48     float requiredSize = FindRequiredSize();
49     m_camera.orthographicSize = Mathf.SmoothDamp(m_camera.orthographicSize, requiredSize, ref m_ZoomSpeed, m_DampTime);
50 }
51 private float FindRequiredSize()
52 {
53     float size = 0;
54     Vector3 desiredLocalPos = transform.InverseTransformPoint(m_DesiredPosition);
55     for(int i = 0; i < m_Players.Length; i++)
56     {
57         Vector3 targetLocalPos = transform.InverseTransformPoint(m_Players[i].position);
58         Vector3 desiredPosToTarget = targetLocalPos - desiredLocalPos;
59         size = Mathf.Max(size, Mathf.Abs(desiredPosToTarget.y));
60         size = Mathf.Max(size, Mathf.Abs(desiredPosToTarget.x) / m_camera.aspect);
61     }
62     size += m_ScreenEdgeBuffer;
63     size = Mathf.Max(size, m_MinSize);
64     return size;
65 }
66
67

```



保存，并将代码附到 CameraRig 上，运行就行啦！

