

# 实验十二 Unity 交互设计二

## 一、实验目的

- 1、进一步了解 Unity3D 引擎；
- 2、掌握 Unity 引擎小地图设计；
- 3、掌握 Unity 引擎导航列表；
- 4、碰撞检测
- 5、物体运动

## 二、实验环境

硬件要求：

PC 机，主流配置，最好为独立显卡，显存 512M 以上。

软件环境：

操作系统：Windows XP、Windows 7。

应用软件：Unity 3D

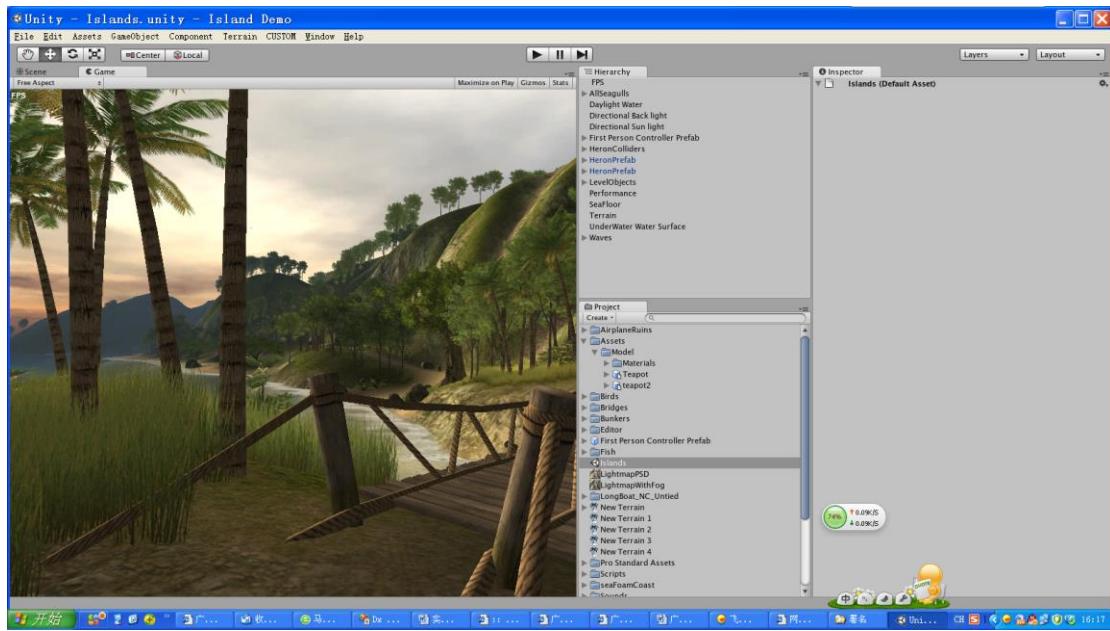
## 三、实验内容与要求

**要求：**

- 1) 将实验的源文件、执行程序打包上传。 打包格式：学号姓名-实验序号-实验名称.rar 。
- 2) 将实验小结（包括收获、自我评价、建议与问题）填到 内容空白处。

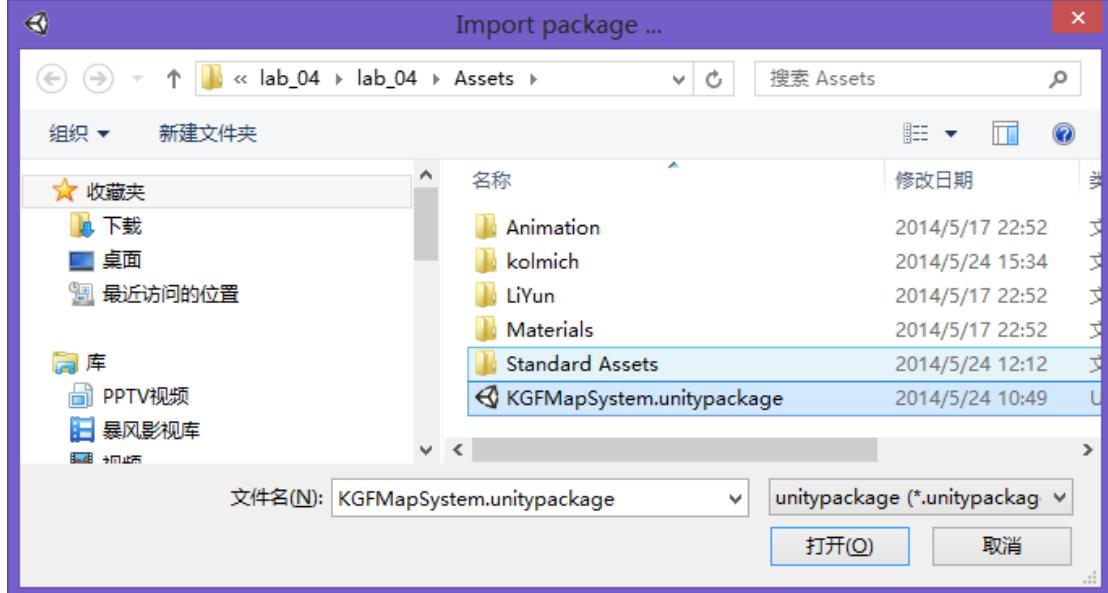
**内容：**

- 1、打开 Unity3D 引擎上次的工程文件

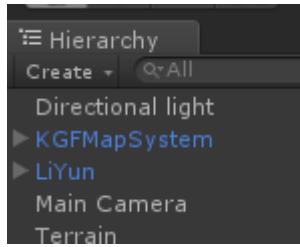


## 2、小地图制作

- 1) 将下载的小地图资源包 KGFMapSystem.unitypackage 拷贝到工程文件夹 Assets 子文件夹中。
- 2) 导入插件资源包, Importing->Custom Package



- 3) 在 Project 视图 All Prefabs 中选择 KGFMapSystem. perfab, 拖入到场景 Hierarchy 面板。

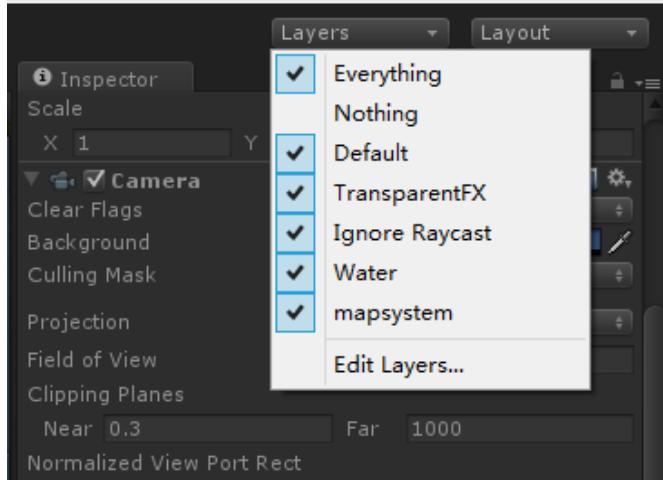


4) 这个时候在 Hierarchy 视图中点击 KGFMapSystem，可以看到其属性面板参数设定栏中提示目标不能为空。需要创建一个新的图层 layer

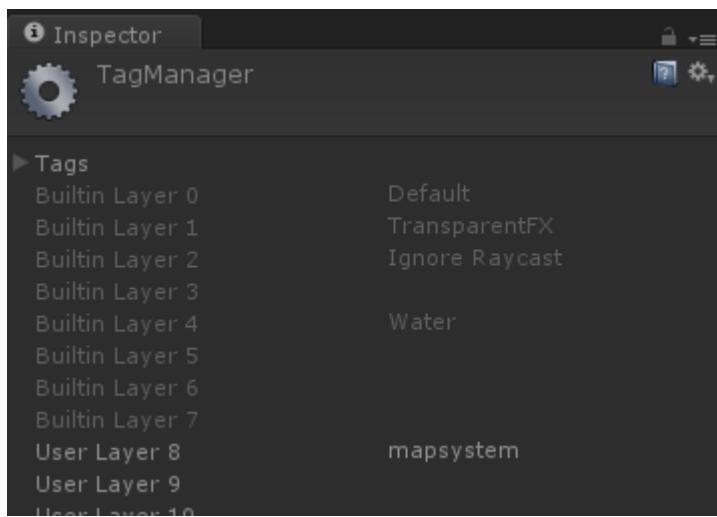


5) 创建新图层 mapsystem

点击右上角 Layers-> Edit Layers

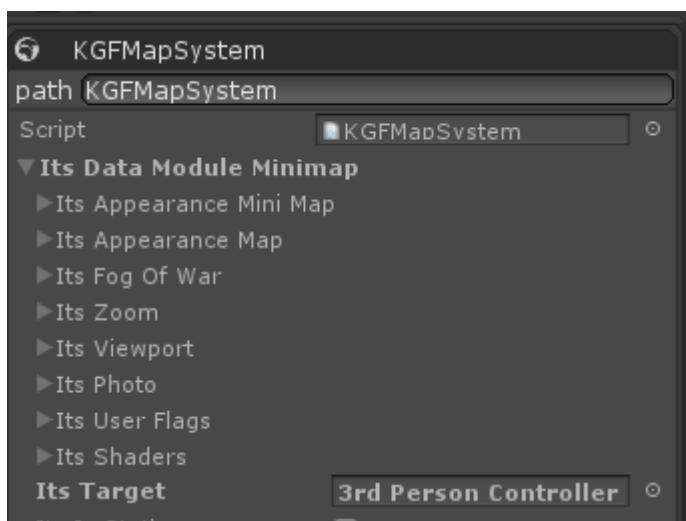


6) 将用户层第 8 层的图层名改为 mapsystem 图层



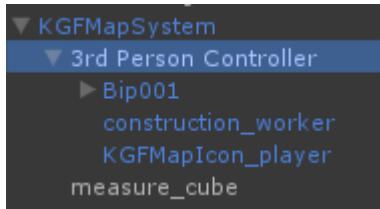
7) 将第三人称角色拖给 KGFMapSystem 属性面板中的 Its Target 栏目中。

展开 **Its Data Module Minimap**，就可以看得到 Its Target 栏目

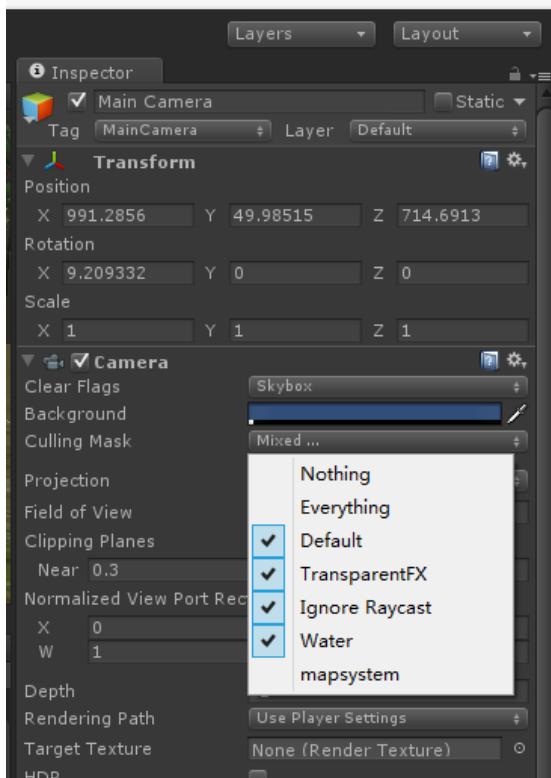


8) 在 Project 视图 All Prefabs 中找到 KGFMapIcon\_player ,将其拖给 Hierarchy 视图

中的第三人称角色。



9) 在 Hierarchy 视图中选中主相机，在它的属性面板中选中 Culling mask 属性，去掉勾选 mapsystem 图层



10) 运行观看小地图效果



### 11) 调整小地图的焦距

点击并展开 Hierarchy 视图中的 KGFMapSystem 属性面板中的 Its Zoom, 调整最小焦距和最大焦距 以便小地图显示更为清晰的地图



调整之前小地图和调整之后小地图对比



清晰小地图



12) 小地图的功能分为放大视野范围、缩小视野范围、锁定相机方向和大小窗口切换四个功能，下面对四个功能进行简单操作说明。

在场景右上角存在如上图的小地图区域。可以点击“+”“—”进行地图的缩放；

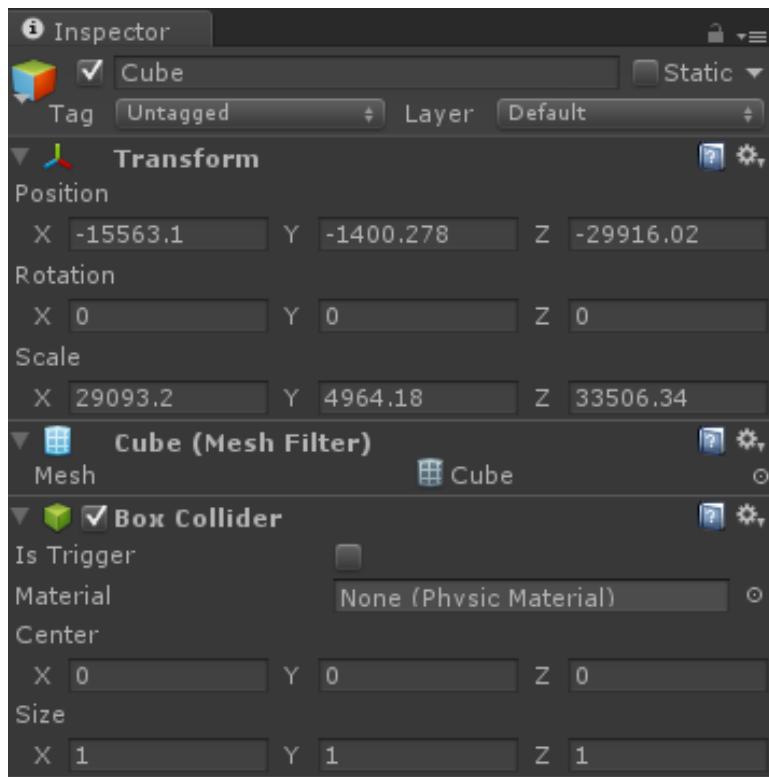
点击锁型图标锁定视角，使箭头始终朝向上方；

点击右下地球形状按钮可以切换到大地图窗口，其按钮的功能和小地图一样，不同的是地图按钮是切换回小地图。

### 3、碰撞检测

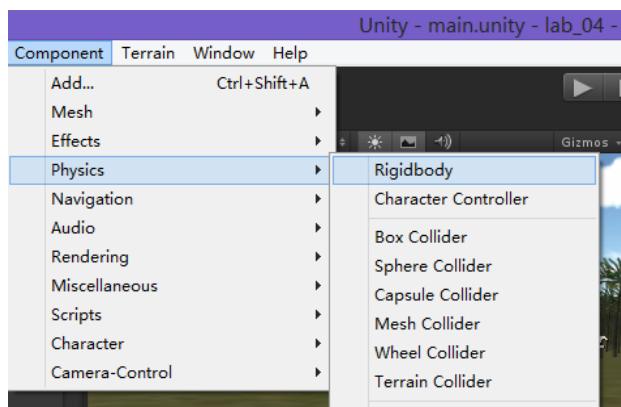
需要对每栋建筑或某个游戏物体进行碰撞封装。具体方法是创建一个 Cube，使之依附在一栋建筑或游戏物体上面，然后调整 Cube 的大小，使得 Cube 边界包含建筑或某个游戏物体，再去掉 Cube 的 Mesh Renderer 属性（即透明化）。Box Collider 属性勾选上。

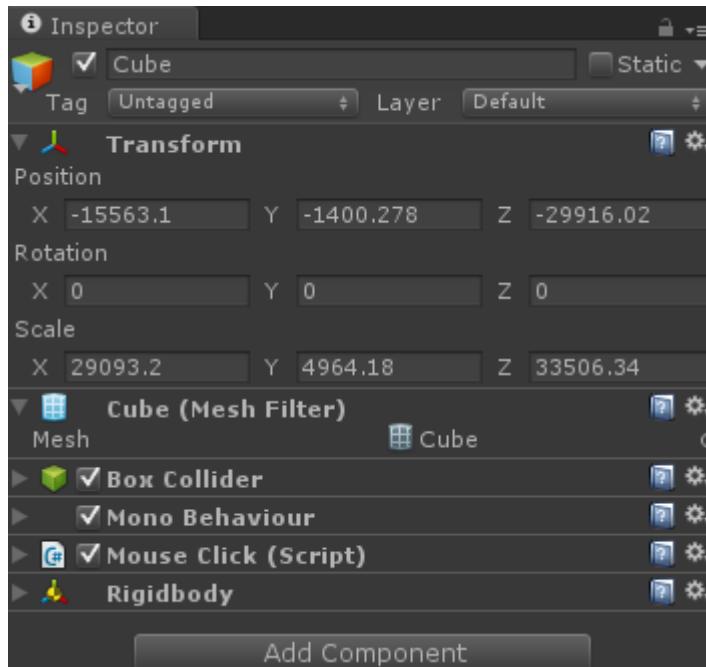
当人物在场景中漫游时就有了对这些加上 Cube 的建筑或游戏物体碰撞检测效果。



再给 Cube 加上 Rigidbody 属性，可以使得物体与场景中其他物体碰撞时又有了碰撞检测效应：

在 Hierarchy 视图中选择 Cube，在 Component 菜单->Physics->Rigidbody 选择添加即可。





## 4、物体的运动

选择场景中某个物体，例如立方体。点击立方体弹出窗口，窗口中有按钮显示可控制立方体的左右旋转，前进后退和左右移动。有标签显示立方体在场景中的位置和方向。创建脚本Movement.cs：

```
using UnityEngine;
using System.Collections;
public class Movement : MonoBehaviour
{
    public Rect windowRect = new Rect(20, 20, 320, 300);
    public bool Window;
    public int TranslateSpeed = 20;
    public int RotateSpeed = 1000;
    // Use this for initialization
    void Start()
    {
        Window = false;
    }

    // Update is called once per frame
    void Update()
    {
    }
    void OnMouseDown()
    {
        Debug.Log("Click");
        Window = true;
    }
    void OnGUI()
    {
        if (Window)
            windowRect = GUI.Window(0, windowRect, DoMyWindow, "Cube Movement");
    }
    void DoMyWindow(int windowID)
    {
```

```

        GUI.Label(new Rect(10, 40, 200, 30), "Cube Position" + transform.position.ToString());
        GUI.Label(new Rect(10, 70, 200, 30), "Cube Rotation" + transform.rotation.ToString());

        if (GUI.Button(new Rect(10, 100, 100, 20), "LeftRotated"))
            transform.Rotate(Vector3.up * Time.deltaTime * (-RotateSpeed));
        if (GUI.Button(new Rect(150, 100, 100, 20), "RightRoatated"))
            transform.Rotate(Vector3.up * Time.deltaTime * RotateSpeed);
        if (GUI.Button(new Rect(10, 140, 100, 20), "Forward"))
            transform.Translate(Vector3.forward * Time.deltaTime * TranslateSpeed);
        if (GUI.Button(new Rect(150, 140, 100, 20), "Back"))
            transform.Translate(Vector3.forward * Time.deltaTime * (-TranslateSpeed));
        if (GUI.Button(new Rect(10, 180, 100, 20), "LeftMove"))
            transform.Translate(Vector3.right * Time.deltaTime * (-TranslateSpeed));
        if (GUI.Button(new Rect(150, 180, 100, 20), "RightMove"))
            transform.Translate(Vector3.right * Time.deltaTime * TranslateSpeed);
        if (GUI.Button(new Rect(10, 220, 100, 20), "Close"))
            Window = false;

    }
}

```

保存脚本到Asserts文件夹，将其拖给Cube立方体即可。

## 5、导航列表

导航列表实现的功能是漫游者可以通过下拉菜单直接到达想要到达的区域。下拉菜单中包含了重要景点的位置，可以在后续优化中增加。

**功能：** 运行主场景时，导航列表弹出。点击导航列表中的 LiYun，第三人称角色去了励耘楼；点击导航列表中的 Cube，第三人称角色去了立方体；点击导航列表中的关闭按钮，导航窗口关闭；按鼠标右键，导航窗口重新打开。

以下是 Navigation.cs 代码示例：

```

using UnityEngine;
using System.Collections;

public class Navigation : MonoBehaviour {

    public Rect windowRect = new Rect(20, 20, 320, 200);
    public bool Window;
    public GameObject LiYun, Cube;

    // Use this for initialization
    void Start () {
        Window = true;
    }

    void OnGUI()
    {
        if (Window)

```

```

        windowRect = GUILayout.Window(0, windowRect, DoMyWindow, "Cube Movement");
    }

    // Update is called once per frame
    void Update () {
        if (Input.GetMouseButtonDown(1))
            Window = true;
    }

    void DoMyWindow(int windowID)
    {

        GUILayout.BeginVertical();
        if (GUILayout.Button("1,Liyunlou"))
        {
            float step = 99999 * Time.deltaTime;
            transform.position = Vector3.MoveTowards(transform.position,
LiYun.transform.position, step);
        }

        if (GUILayout.Button("2,Cube"))
        {
            float step = 99999 * Time.deltaTime;
            transform.position = Vector3.MoveTowards(transform.position,
Cube.transform.position, step);
        }

        if (GUILayout.Button("3,Close"))
            Window = false;

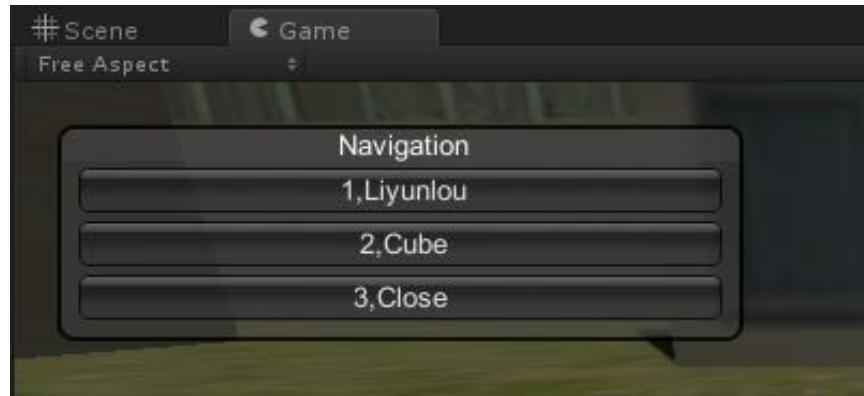
        GUILayout.EndVertical();
    }
}

```

通过 MoveTowards 函数进行场景之间的切换，实现了从当下位置移动到目标地点的功能。其函数本身可以设置移动速度，为了设计需要，把移动速度调到了一个极大量，所以实现的效果是直接移动到目标位置。

- 1) 保存脚本为 Navigation.cs
- 2) 将脚本拖给第三人称角色
- 3) 创建一个 Cube，改名为 Cube2，（定位立方体 Cube 的位置），去掉 MeshRender 属性，取消 Box Collider 属性，位置靠近场景中的立方体 Cube，将 Cube2 拖给脚本中的游戏对象 Cube。
- 4) 创建一个 Cube，改名为 liyun2，（定位励耘楼 LiYun 的位置），去掉 MeshRender 属性，取消 Box Collider 属性，位置靠近场景中的励耘楼 LiYun，将 liyun2 拖给脚本中的游戏对象 LiYun。

导航列表效果如图所示。



导航列表效果图

## 四、实验素材

- 1、上周的打包工程文件
- 2、小地图插件 KGFMapSystem.unitypackage

## 五、参考

Unity3D 游戏开发

宣雨松编著 人民邮电出版社

- ? <http://www.unitymanual.com/>
- ? <http://game.ceeger.com/>
- ? <http://unity3d.9ria.com/>
- ? <http://www.u3dchina.com/forum.php>
- ? <http://www.cgjoy.com/unity3d-1>
- ? <http://www.unitygame.cn/>