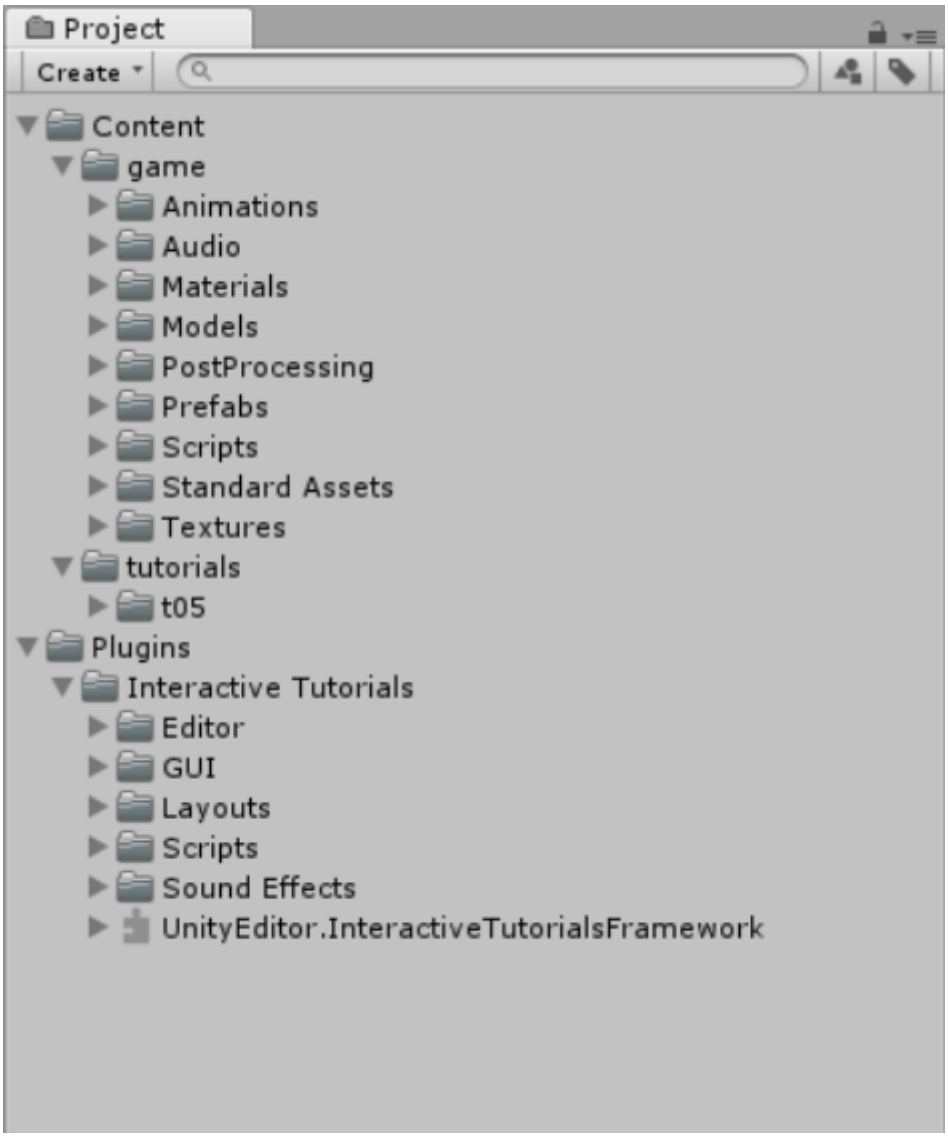
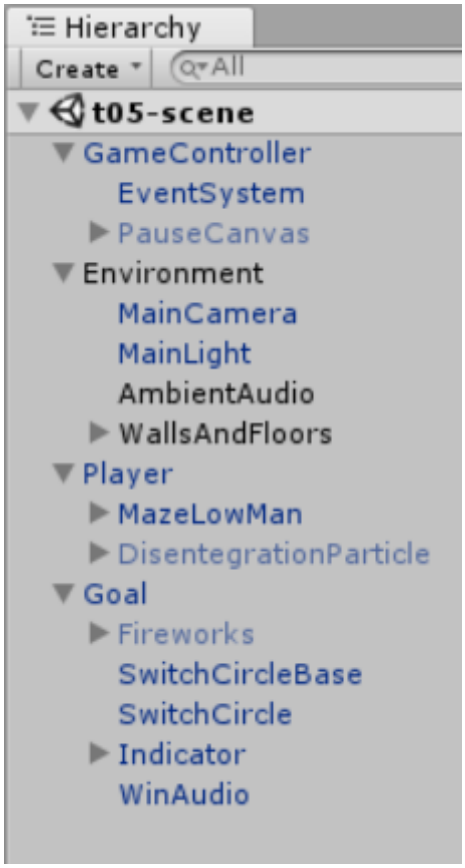
1.解释 游戏对象（GameObjects） 和 资源（Assets）的区别与联系。

- Gameobject是游戏中的基本组件，是游戏过程中的基本单位。它具有一系列，可扩展的component，并为他们提供封装。
- Assets是游戏中各种可能用到的资源，material，sounds，pictures等等，（不知道预制算不算assets，我记得在做table预制时是保存在assets里的）
- Assets可以作为部件，组成在Gameobject中，若预制算Assets，那么Gameobject也可以组成为预制作为Assets的一部分;

2.下载几个游戏案例，分别总结资源、对象组织的结构（指资源的目录组织结构与游戏对象树的层次结构）



- 上面的样例可见，该游戏资源的内容分为预制、脚本等等，按照文件的类型的不同，添加到不同的文件夹当中，便于后面的资源整理和利用



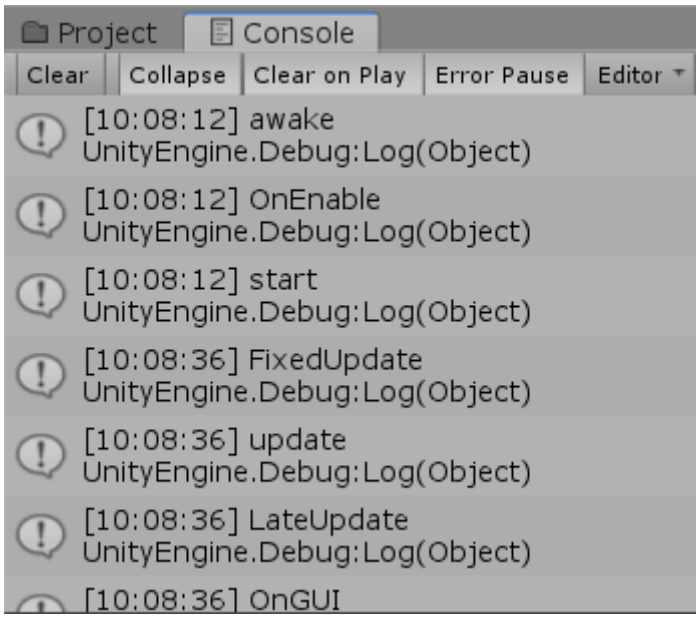
- 游戏对象分为了，game control, environment, players and game goals.可以看出主要从游戏机制，环境，角色，游戏目标来对整个游戏进行分类。
- 对象结构与资源结构，基本符合游戏引擎架构图中的designer部分。

3.编写一个代码，使用 debug 语句来验证MonoBehaviour基本行为或事件触发的条件

- 基本行为包括 Awake() Start() Update() FixedUpdate() LateUpdate()
- 常用事件包括 OnGUI() OnDisable() OnEnable()

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

// 编写一个代码，使用 debug 语句来验证 MonoBehaviour 基本行为或事件触发的条件
// 基本行为包括 Awake() Start() Update() FixedUpdate() LateUpdate()
// 常用事件包括 OnGUI() OnDisable() OnEnable()

public class test : MonoBehaviour
{
    // Awake is called before any Start function
    void Awake()
    {
        Debug.Log("awake");
    }

    // Start is called before the first frame update
    void Start()
    {
        Debug.Log("start");
    }

    // Update is called once per frame
    void Update()
```

```
    {
        Debug.Log("update");
    }

    void FixedUpdate()
    {
        Debug.Log("FixedUpdate");
    }

    void LateUpdate()
    {
        Debug.Log("LateUpdate");
    }

    void OnGUI()
    {
        Debug.Log("OnGUI");
    }

    void OnDisable()
    {
        Debug.Log("OnDisable");
    }

    void OnEnable()
    {
        Debug.Log("OnEnable");
    }
}
```



事件名称	执行条件或时机
Awake	This function is always called before any Start functions and also just after a prefab is instantiated. (If a GameObject is inactive during start up Awake is not called until it is made active.)
Start	Start is called before the first frame update only if the script instance is enabled.
FixUpdate	**FixedUpdate** is often called more frequently than **Update**. It can be called multiple times per frame.
Update	**Update** is called once per frame. It is the main workhorse function for frame updates.
LateUpdate	**LateUpdate** is called once per frame, after **Update** has finished.
OnGUI	Called multiple times per frame in response to GUI events. The Layout and Repaint events are processed first, followed by a Layout and keyboard/mouse event for each input event.
OnEnable	Only called if object is active!
OnDisable	This function is called when the behaviour becomes disabled or inactive.

Form is not right, OnEnalbe should be in front of Start.

4.查找脚本手册，了解GameObject，Transform，Component 对象

- 分别翻译官方对三个对象的描述（Description）

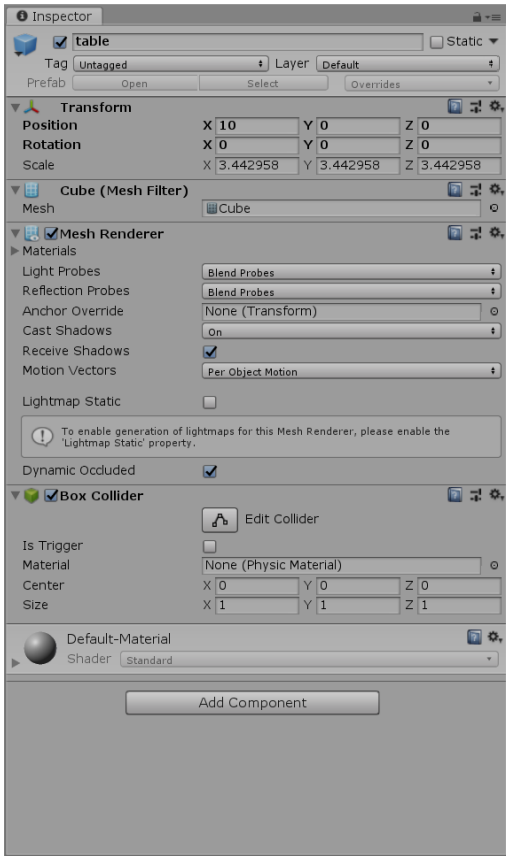
游戏对象是统一体中代表人物、道具和场景的基本对象。它们本身并没有完成多少工作，但是它们充当组件的容器，组件实现真正的功能。

Transform组件决定场景中每个对象的位置、旋转和比例。每个gameobject都有一个transform。

Component是一切附加在游戏物体的基类。

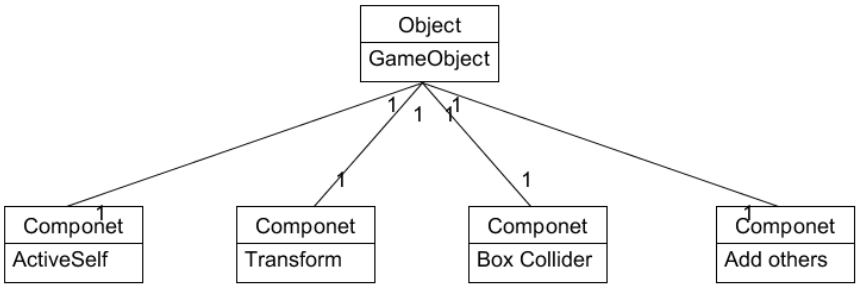
5.描述下图中 table 对象（实体）的属性、table 的 Transform 的属性、 table 的部件

- 本题目要求是把可视化图形编程界面与 Unity API 对应起来，当你在 Inspector 面板上阅读每一个内容，应该知道对应 API。



> 第一个选择框是activeSelf：可以定义对象的名称，动静态等属性，比如上图，对象名称是table
> 第二个选择框是Transform：可以定义对象的位置、面向方向、大小
> 第四个选择框是Box Collider：可以调整坐标系的位置、大小
> 第五个选择框是Add Component：可以给对象增加行为,比如给对象增加C#script
> 原文链接：https://blog.csdn.net/gzx1002/article/details/100623692

- 用 UML 图描述 三者的关系（请使用 UMLet 14.1.1 stand-alone版本出图）



- 资源预设（Prefabs）与 对象克隆 (clone)
  - 预设（Prefabs）有什么好处?

  预设可以提前将设计游戏中所需要的游戏对象进行设计打包，成为一个模板。在设计的过程中，随时可以直接从资源当中加载，成为一个游戏对象。预设的存在，方便了面向对象思想的应用，方便我们在设计游戏中更加便捷。

  - 预设与对象克隆 (clone or copy or Instantiate of Unity Object) 关系?

克隆是将已经存在的游戏对象，或者是资源当中的预设进行复制。克隆可以是对已经添加在视图当中的游戏对象进行复制，也可以是对资源当中，我们提前预设好的模板进行克隆。克隆的进行往往伴随着预设。

- 制作 table 预制，写一段代码将 table 预制资源实例化成游戏对象

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

public class LoadBeh : MonoBehaviour {

    public Transform res;

    // Use this for initialization
    void Start () {
        // Load Resources
        GameObject newobj = Instantiate<Transform> (res,
this.transform).gameObject;
        newobj.transform.position = new Vector3 (0, Random.Range (-5,
5), 0);
    }
}
```

井字棋作业

视频地址:https://www.bilibili.com/video/BV1hA411J74J/

作业比较简单，但是有很多细节只有在自己边做边玩中才能发现，

比如：no winner 情况下的不显示，或者鼠标的还可以继续操作等等

比如出现下图中搞笑的错误：一个赢了，另一个在点一次也赢了，很尴尬

```
if(GUI.Button(new Rect(250+i*50,200+j*50,50,50),"")&&map[i,j] == 0&&test == 0)//
注意只有在，没有下过的位置，且没有玩家胜出的时候可以构建。
```

所以对上面的代码进行修改，解决了问题，代码就和生活一样，一个这么小的游戏我都无法一次喜好，实在是感慨游戏设计程序员的不易，这是第二次提交了，second version，解决掉自己发现的一些bug。

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

public class tictactoe : MonoBehaviour
{
    //游戏每一次按下按钮都会导致棋盘值的变动，再进而影响渲染的结果
    //引擎会对代码进行渲染，初始渲染的有：棋盘，按钮（reset）可以跟据个人喜好进行调整
    //我们希望在每一次点击后，在指定位置，出现相应玩家的棋子，所以需要变量：player（1，2），
map[3,3]
    //map有三种状态：未点击（0），player1（1），player2（2）
    //我们需要检测游戏结果，test（），其中包含结果呈现

    //现在来逐步完成这个游戏

    public int player = 1;//默认玩家1
    public int[,] map = new int[3, 3];//默认未点击
    public Texture2D player1;
    public Texture2D player2;//提供player1，2的素材接口，也可以使用GUIContent
    int counter = 0;
    int testWin()//1代表player1，-1代表player2，0代表和棋
    {
        //胜利有三种，横纵斜：
        //Horizontal
        for(int i = 0;i < 3;i++)
        {
            if(map[i,0]==map[i,1]&&map[i,2]==map[i,1])
            {
                return map[i,1];
            }
        }
        //Vertical
        for(int i = 0;i < 3;i++)
        {
            if(map[1,i]==map[0,i]&&map[2,i]==map[1,i])
            {
                return map[1,i];
            }
        }
        //Diagonal
        if(map[1,1]==map[0,0]&&map[1,1]==map[2,2])
        {
            return map[1,1];
        }
        if(map[1,1]==map[0,2]&&map[1,1]==map[2,0])
        {
            return map[1,1];
        }
```

```
        return 0;
    }
    void OnGUI()
    {
GUIStyle fontStyle = new GUIStyle();
fontStyle.normal.background = null;
fontStyle.normal.textColor= new Color(1, 0, 0);
fontStyle.fontSize = 20;

    //显示玩家和其对应的头像；
    GUI.Button (new Rect (100,10, 75, 75), player1);
    GUI.Button (new Rect (100,85, 75, 50), "player1");
    GUI.Button (new Rect (475,10, 75, 75), player2);
    GUI.Button (new Rect (475,85, 75, 50), "player2");

    if(GUI.Button(new Rect(275,400,100,50),"New game"))
    {
        player = 1;
        for(int i = 0;i < 3;i++)
        for(int j = 0;j < 3;j++)
        {
            map[i,j] = 0;
        }
        counter = 0;
    }
    //检查是否结束
    int test;
    test = testWin();
    if(test == 1){GUI.Button(new Rect(275,50,100,50),"Player1 win!");
        }
    if(test == -1){GUI.Button(new Rect(275,50,100,50),"Player2 win!");
       }

    if(counter==9&&test==0){GUI.Button(new Rect(275,50,100,50),"No winner!");
    }
    //构建棋盘；
    for(int i = 0;i < 3;i++)
        for(int j = 0;j < 3;j++)
        {
            if(GUI.Button(new Rect(250+i*50,200+j*50,50,50),"")&&map[i,j] ==
0&&test == 0)//注意只有在，没有下过的位置，且没有玩家胜出的时候可以构建。
            {
                map[i,j] = player;
                player *= -1;
                counter = counter + 1;
            }//这里可以有很多种实现方法，现在所示的这种属于比较简单的；
            else if(map[i,j] == 1)
            {
                GUI.Button(new Rect(250+i*50,200+j*50,50,50),player1);
            }
            else if(map[i,j] == -1)
            {
                GUI.Button(new Rect(250+i*50,200+j*50,50,50),player2);
            }
        }

    //在构建好期盼后，我们需要检验，当一方胜利后，弹出提示
    //检测函数可以写在ongui外，只在内部调用其结果，开始写
```

```
    }
}
```