## 第二次作业

叶杰鑫 18342119

1、简答题

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

GameObjects是一个具体的实例，Assets是包括诸多游戏素材的资源。GameObjects是Assets中的一部分，Assets中不仅仅包括GameObject，还有一些C#文件以及音频文件等。

* 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）

资源的目录组织结构主要包含文件、材质、模型、预制件、场景、脚本、标准资源这几个部分。资源里面又包含了图片，游戏需要用到的音乐等等。游戏对象树就如同Windows的文件资源管理器一样，树目录结构：一个游戏对象（文件夹）包含多个子对象（子文件夹），子对象（子文件夹）又可以继续包含多个子对象（子文件夹）。

* 编写一个代码，使用 debug 语句来验证 [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html) 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;

public class Test : MonoBehaviour

{

void Awake() {

Debug.Log ("Awake");

}

void Start () {

Debug.Log ("Start");

}

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");

}

}

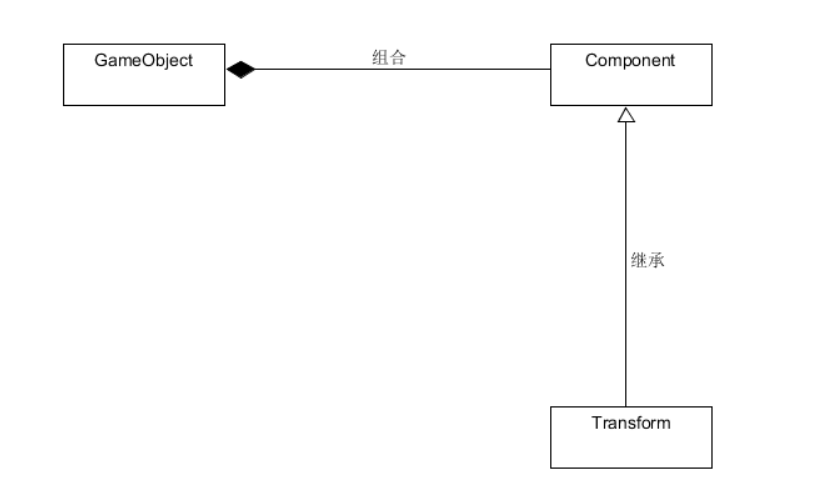
* 查找脚本手册，了解 [GameObject](https://docs.unity3d.com/ScriptReference/GameObject.html)，Transform，Component 对象
  + 分别翻译官方对三个对象的描述（Description）

GameObject：是Unity场景里面所有实体的基类.  
Transform：物体的位置、旋转和缩放。  
Component：一切附加到游戏物体的基类。

* + 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 面板上每一个内容，应该知道对应 API。
    - 例如：table 的对象是 GameObject，第一个选择框是 activeSelf 属性。

table对象的属性： static、layer、tag、prefab 。  
table的transform属性： position：（0，0，0），rotation（0，0，0），scale（1，1，1）。  
table的组件： transform、mesh render、box colider、cube（mesh filter）等。

* + 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？

预设提供了模板，有利于资源的复用，节约时间；通过更改预设的资源，能够将所有调用该资源的对象进行修改。

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

克隆其实就相当于又调用了一次预设的资源。假设A克隆了B，B对应的预设资源是C，那么如果C发生改变，A和B都会改变。而B并不会随着A的改变而改变。

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

table预设放在Resources文件夹下：

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class insert : MonoBehaviour

{

void Start()

{

GameObject instance = (GameObject)Instantiate(Resources.Load("table"));

instance.transform.position = new Vector3(3, 3, 3);

}

void Update()

{

}

}

2、 编程实践，小游戏

* 游戏内容： 井字棋 或 贷款计算器 或 简单计算器 等等
* 技术限制： 仅允许使用 [**IMGUI**](https://docs.unity3d.com/Manual/GUIScriptingGuide.html) 构建 UI
* 作业目的：
  + 了解 OnGUI() 事件，提升 debug 能力
  + 提升阅读 API 文档能力

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Chess : MonoBehaviour {

private int turn = 1;

private int[,] grid = new int[3, 3];

// 重新开始游戏

void Start () {

Restart();

}

void Restart()

{

turn = 1;

for (int i = 0; i < 3; ++i)

{

for (int j = 0; j < 3; ++j)

{

grid[i, j] = 0;

}

}

}

void OnGUI()

{

//设置button中字体的大小

GUI.skin.button.fontSize = 20;

//设置label中字体大小和颜色

GUIStyle style = new GUIStyle();

style.fontSize = 40;

style.normal.textColor = new Color(255, 255, 255);

//判断是否点击Restart按钮

if(GUI.Button(new Rect(360, 500, 80, 80), "Restart")){

Restart();

}

//判断游戏是否结束

int result = judge();

if (result == 1)

{

GUI.Label(new Rect(340, 170, 100, 50), "O wins!", style);

}

else if (result == 2)

{

GUI.Label(new Rect(340, 170, 100, 50), "X wins!", style);

}else if (result == 3)

{

GUI.Label(new Rect(300, 170, 100, 50), "GameOver!", style);

}

//生成棋盘并判断格子是否被点击

for (int i=0; i<3; i++)

{

for(int j=0; j<3; j++) {

if (grid[i, j] == 1)

{

GUI.Button(new Rect(280 + i \* 80, 220 + j \* 80, 80, 80), "O");

}else if (grid[i, j] == 2)

{

GUI.Button(new Rect(280 + i \* 80, 220 + j \* 80, 80, 80), "X");

}else if (GUI.Button(new Rect(280 + i \* 80, 220 + j \* 80, 80, 80), "")&&result==0)

{

if (turn == 1)

{

grid[i, j] = 1;

}else

{

grid[i, j] = 2;

}

turn = -turn;

}

}

}

}

int judge()

{

// 横向连线

for (int i = 0; i < 3; ++i)

{

if (grid[i, 0] != 0 && grid[i, 0] == grid[i, 1] && grid[i, 1] == grid[i, 2])

{

return grid[i, 0];

}

}

//纵向连线

for (int j = 0; j < 3; ++j)

{

if (grid[0, j] != 0 && grid[0, j] == grid[1, j] && grid[1, j] == grid[2, j])

{

return grid[0, j];

}

}

//斜向连线

if (grid[1, 1] != 0 &&

grid[0, 0] == grid[1, 1] && grid[1, 1] == grid[2, 2] ||

grid[0, 2] == grid[1, 1] && grid[1, 1] == grid[2, 0])

{

return grid[1, 1];

}

//全部格子都被点击

bool allClick = true;

for (int i=0; i<3; i++)

{

for (int j=0; j<3; j++)

{

if (grid[i, j] == 0)

{

allClick = false;

}

}

}

if (allClick)

return 3;

return 0;

}

}

