1. Mmorpg游戏里面的ui界面很多，每次要找都要一个一个点开比较麻烦
2. 之前实习的公司NGUI 有一个 检测工具 ，点击后可以自动在Hierarchy显示点击的 UI界面。自己写了一个UGUI的

整体先想一个思路：

\*先做射线检测点击到的UI

\*再映射到Hierarchy上

1.如何检测点击UI

一开始的想法是通过射线检测Ray去做，

从摄像机作为起点和鼠标点击的地方两个点发射一条Ray，Raycast发射射线，返回bool，里面out RaycastHit，RaycastHit保存射线发射信息。



通过查 out hitInfo，hitInfo遍历物体出来查看他们所有的层级，层级属于UI界面的保留，然后去查询是哪个。但是因为项目里面的Layer是不动的UI界面也会有重叠，有的大有的小，物体的z轴和深度也是有不同。

但是还是试了一下，发现不行，说明UGUI的检测不是用碰撞器去访问的

NGUI需要碰撞器，UGUI不需要，NGUI可以用这个方法，那UGUI是怎么检测碰撞的？

查一下

后面看了跟射线和UI相关的内容，发现UGUI和NGUI确实是有不同的地方，NGUI的事件机制其实就是collider的触发机制，但是UGUI在unity 里面其实已经给我们封装好了他的一个射线类型，射线 通过检测GraphicRaycaster去查看碰撞的是不是UGUI的内容



Unity的GraphicRaycaster也帮我们解决了当有多个UI会有层级阻挡问题：UI组件勾选Raycast Target能够按照层级关系阻挡穿透，但其阻挡的是UI组件之间的射线穿透。GraphicRaycaster的源码中有 var foundGraphics = GraphicRegistry.GetGraphicsForCanvas(canvas);只会判断Graphic对象，这是UI元素的父类。调整一下，可以让点击的时候射线透不过去。

Ok，阻挡的。也就是说，ngui 通过普通射线检测碰撞盒去判断 是否点击UI

Ugui 里用GraphicRaycaster组件来检测射线 UI元素的事件，并且把结果都保存在EventSystem里面！！！！

哦，那岂不是在EventSystem里面就能访问到点击了什么UI吗？？

答案是是的

Unity自带的

using UnityEngine.EventSystems;

头文件里面可以访问EventSystem

并且点击数据都会保存在pointerEventData 类里面

（附加Eventsystem和pointEventData的类）

Ok，那剩下只需要去稍微看一下EventSystem里面的属性和方法即可

1. 使用unityEditor 类把这个文件添加到菜单栏

因为不是每次都要用的，把脚本挂上main\_camera ，然后把它active 为false。

然后用unityeditor把它关联了，点击菜单才点击运行

顺便优化一下方法让他有更多功能

我的初代代码

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.EventSystems;

using UnityEngine.UI;

using UnityEditor;

public class UICheckTool : MonoBehaviour {

    [MenuItem("FindUI/找到UI")]  //保存后就会在unity菜单栏中出现MyMenu的菜单栏

    static void FindUI()          //和二级目录Do Something,点击它就会执行DoSomething()方法

    {

    GameObject go = GameObject.Find("MainCamera");

        go.AddComponent<UICheckTool>();

        go.GetComponent<UICheckTool>().uitype =FindType.UITOP;

    }

    [MenuItem("FindUI/找到底层UI")]

    static void FindUI2(){

        GameObject go = GameObject.Find("MainCamera");

        go.AddComponent<UICheckTool>();

        go.GetComponent<UICheckTool>().uitype =FindType.UIBOTTOM;

    }

   [MenuItem("FindUI/找到UI停止游戏")]

    static void FindCanvas(){

        GameObject go = GameObject.Find("MainCamera");

        go.AddComponent<UICheckTool>();

        go.GetComponent<UICheckTool>().uitype =FindType.StopUI;

    }

    bool isTop ;//是否反映最上层UI

    private EventSystem \_mEventSystem;

    PointerEventData pointerEventData;

    enum FindType{

        UITOP,

        UIBOTTOM,

        StopUI,

    }

    FindType uitype; //查找类型

void Update()

{

if (Input.GetMouseButtonDown(0))

{

            IsRaycastUI(uitype);

}

}

    bool IsRaycastUI(FindType uitype)

    {

        if (EventSystem.current == null)

        {    Debug.Log("nothing ");}

            //鼠标点击事件

            pointerEventData = new PointerEventData(EventSystem.current);

            //设置鼠标位置

            pointerEventData.position = Input.mousePosition;

            //射线检测返回结果

            List<RaycastResult> results = new List<RaycastResult>();

            //检测UI

            //graphicRaycaster.Raycast(pointerEventData, results);

            EventSystem.current.RaycastAll(pointerEventData, results);

            //打印结果

            for (int i = 0; i < results.Count; i++)

            {

                if(uitype == FindType.UITOP)

                {

                    if(results[i].depth != 0)//脚本里面depth == 0 的是背景板 ， UI都是有depth的

                    {

                        //调用显示到Hierarchy中

                        EditorGUIUtility.PingObject(results[i].gameObject);

                        Selection.activeGameObject =results[i].gameObject;

                        break;   //break 的原因是， 只返回最上层的UI

                    }

                }

                if(uitype ==FindType.UIBOTTOM){

                    if(results[i].depth != 0)//脚本里面depth == 0 的是背景板 ， UI都是有depth的

                    {

                        //调用显示到Hierarchy中

                        EditorGUIUtility.PingObject(results[i].gameObject);

                        Selection.activeGameObject =results[i].gameObject;

                    }

                }

                if(uitype==FindType.StopUI)

                {

                    if(results[i].depth != 0)//脚本里面depth == 0 的是背景板 ， UI都是有depth的

                    {

                        //调用显示到Hierarchy中

                        EditorGUIUtility.PingObject(results[i].gameObject);

                        Selection.activeGameObject =results[i].gameObject;

                        if(Time.timeScale == 0){

                            Time.timeScale =1;

                        }

                        else {

                            Time.timeScale =0;

                        }

                        break;

                    }

                }

        }

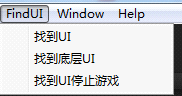
        //返回值>0，即点击到UI

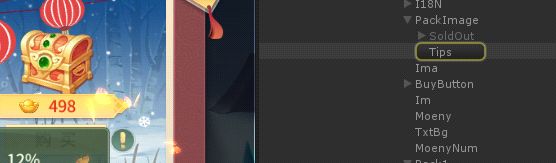
        return results.Count > 0;

    }

}

Ok现在就可以实现在游戏运行的选中想要的ui对应





思考：能否写成一个线程？

#if #endif 是预定义中使用的

其他一些的UGUI和NGUI的区别

1. NGUI本质是unity 的一个插件 ， 所以用的语言是C#， UGUI内置在unity中，unity用c++写的
2. NGUI注重图集概念，UGUI同一个prefab下面的的图会自己打成一个图集。但是对于512\*512或者以上的图片，尽量都分开打图集。
3. 一些其他插件的，比如NGUI里面内置的动画itween， UGUI里面的Mask。（Mask比较特殊，因为他是会另外计算DrawCall）
4. UGUI引入锚点的概念。NGUI宽高比
5. NGUI的渲染前后顺序是通过Widget的Depth，而UGUI渲染顺序根据Hierarchy的顺序，越下面渲染在顶层.

UGUI的同一类型的组件放在一起，假如他们是同一材质和同一贴图， 就会自动合并在一起调用drawcall

6..Canval 的z轴不是0的话也没有办法合并drawcall