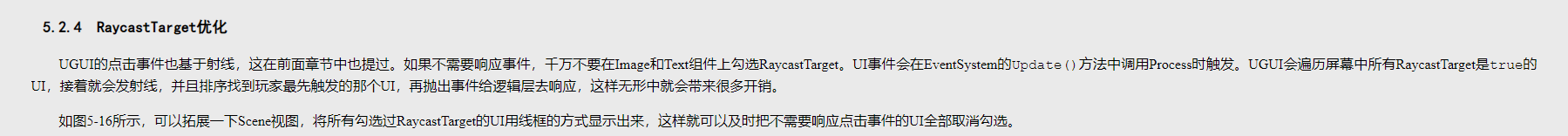
1.

RaycastTarget



UI射线一定要做好管理，不用触发的一定要关掉，由于现在是默认开启的，可以通过编辑器扩展来关掉

同时也可以扩展一下scene场景把勾选了的物体用ongizmodraw显示出来，计算4个顶点，然后drawline

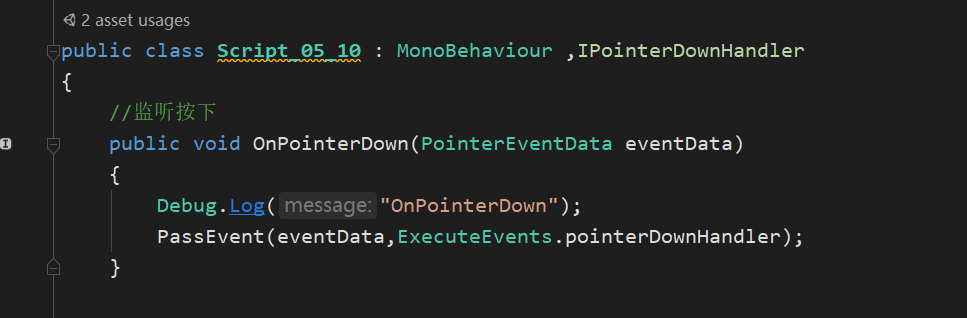


可以随时关

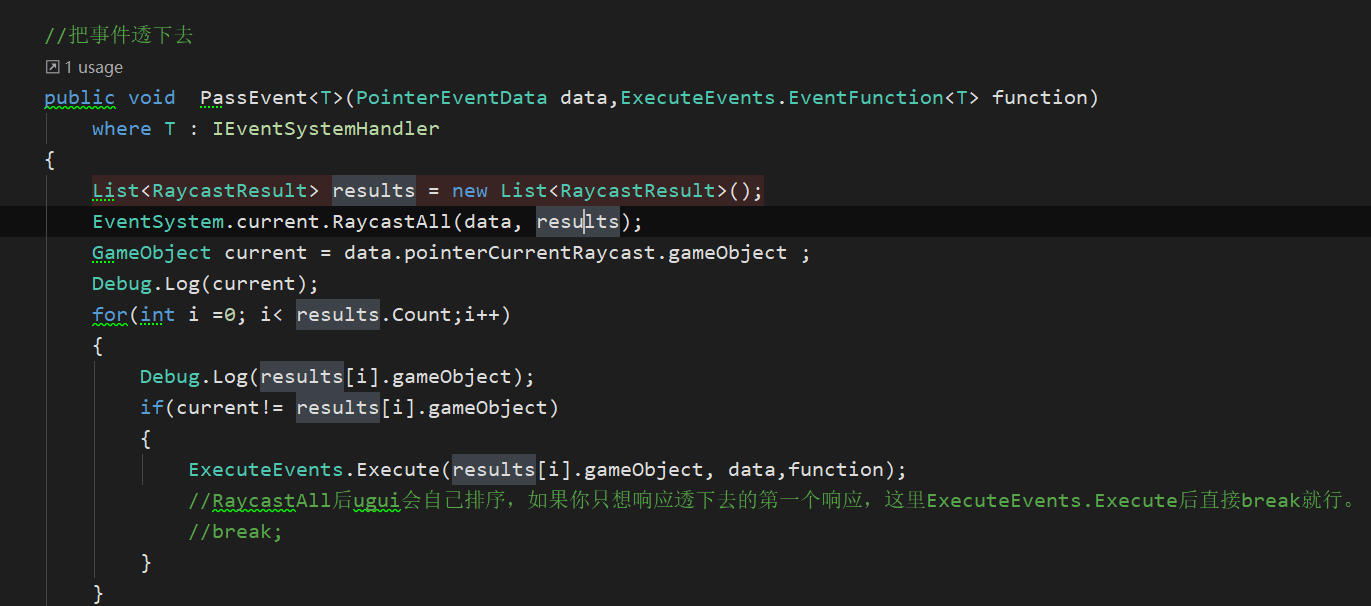
2. 

如何渗透UI？

书上就是这样

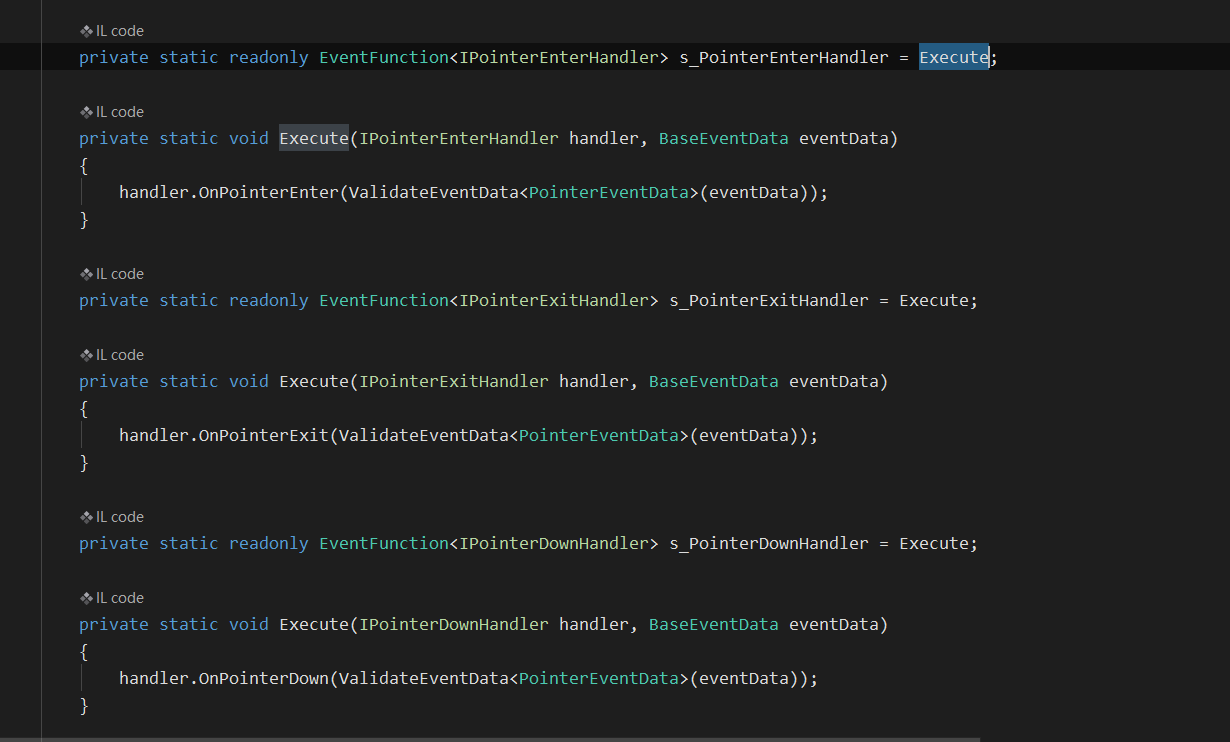


这里首先就是实现原本UI事件接口来写

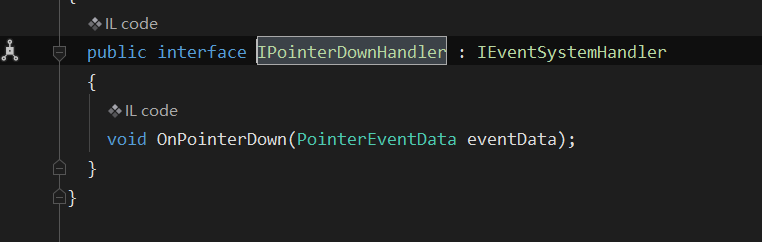


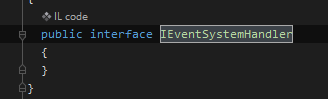
然后传进来这里eventfunction<T>UI事件，这个是UGUI源码里实现的一个UI事件的泛型委托，所以只能是传这个委托的对象方法进来，这里首先是获取到所有射线结果，然后循环遍历触发，这里一定要规避掉当前点击物体，否则会死循环递归，因为当前物体的点击触发就是上面的pointerdown了，如果这里再来就会递归，这里PassEvent只是为了穿透下去事件

所以这里有点问题就是，这个current是必须只能让一个物体有，如果穿透的物体也是有这个PassEvent方法就完犊子了，这里会循环到上一个射线打击物体触发方法，然后又会穿透调用又来死循环递归

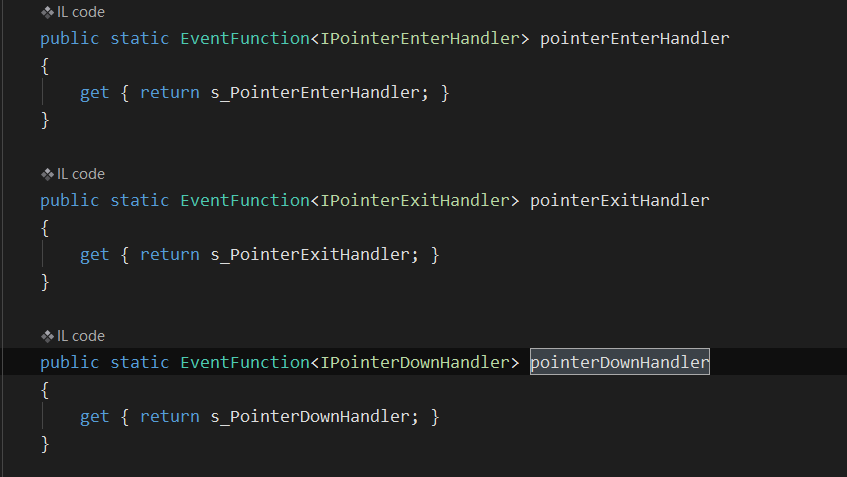


源码这里很巧妙，三个同名函数，只是参数不同，就靠定义的泛型接口来赋值了





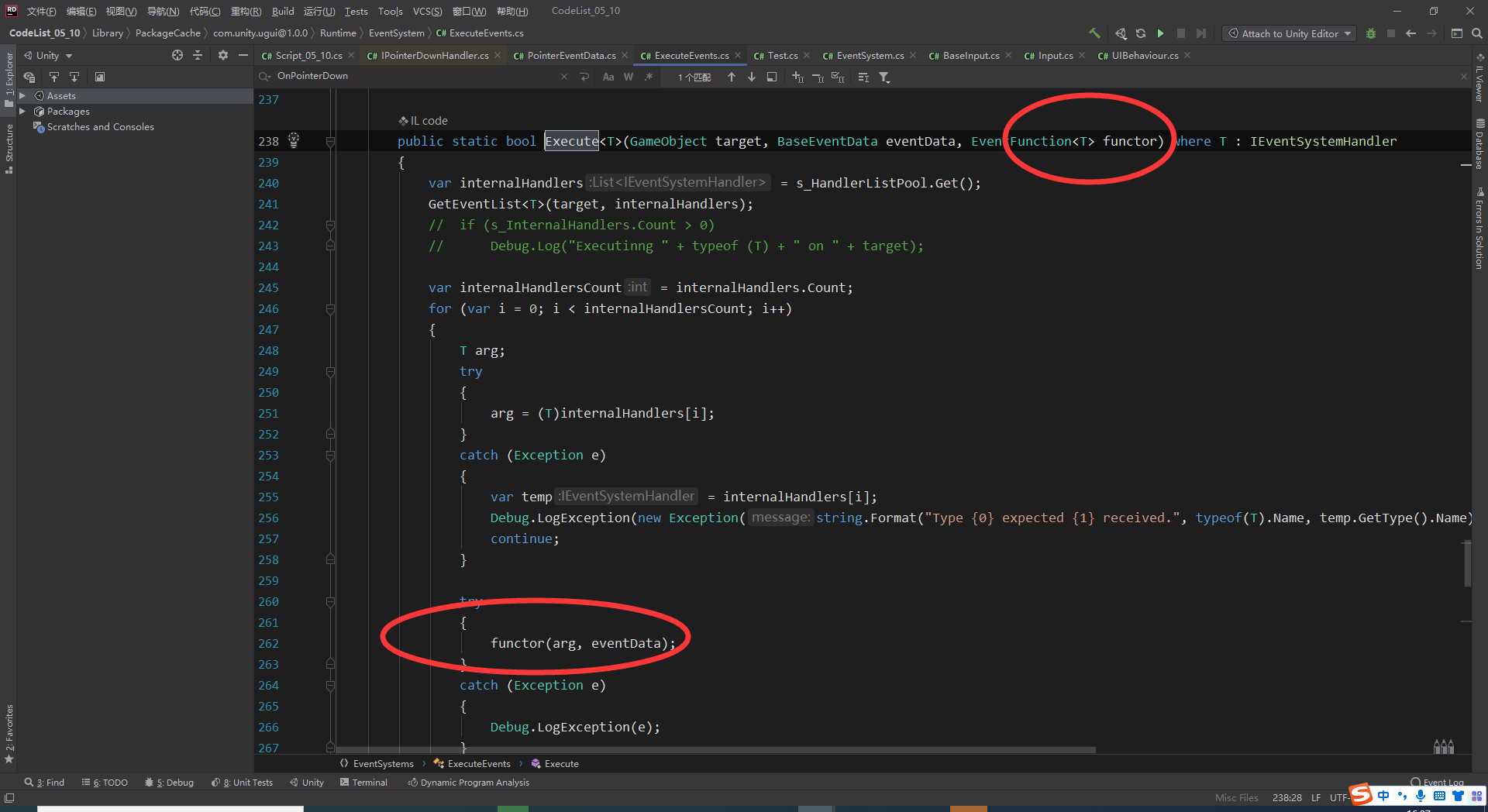
用一个总事件接口来方便这里泛型约束where T:然后这些继承下来的接口分别调用各自状态的回调，比如点击OnPointerDown，那么如果想像现在这样直接在别的地方进行调用检测怎么办呢，没错就是上面那样获取到

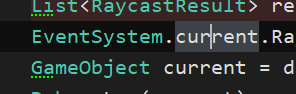


这些handler，然后就可以直接调用了，上面的

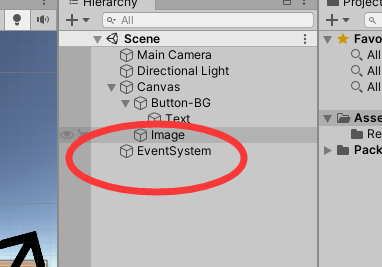
ExecuteEvents.Execute

方法也是，直接调用的传进来的func，用了trycatch



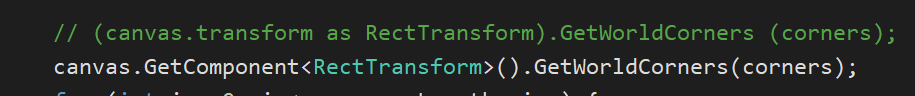


这个current可能就是

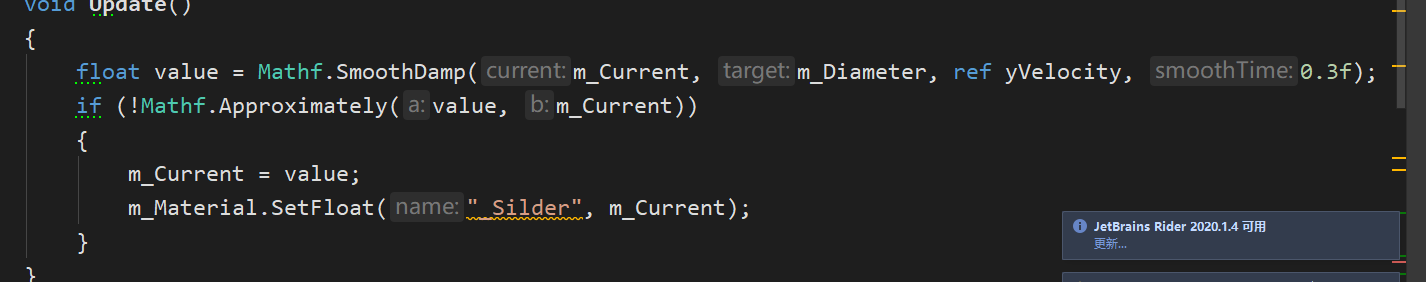


这里我们总结看到，别人写的轮子，我们可以很方便的扩展，比如这里的穿透事件，所以

3.通过上面的渗透原理我们可以有这种新的教程实现方法就是穿透UI点击触发事件，这样我们就可以不用想办法去设置这个区域大小了，并且通过UI自己本身去计算这个白圈，这样假如UI出现错位也能正确显示上去（虽然这样已经很严重的bug）了



我们一般这样写



然后通过smoothdamp

4.