一个场景中只有一个Canvas就足够显示所有UI，但是也可以有多个Canvas。可以用嵌套Canvas（一个Canvas放在另一个的子对象上）来达到优化的目的。嵌套Canvas与parent的renderMode相同。

一般，UI渲染就像是普通的图形直接绘制在屏幕上。也就是说，他们没有需要摄像机观察的3D场景的概念。Unity提供了屏幕渲染的方式，但是在一个场景中，也需要将ui当作一个object并根据Render Mode去渲染。RenderMode的类型包括：ScreenSpace-Overlay，Screen Space-Camera，World Space。

**Screen Space – Overlay：**

这个模式下，Canva 通过缩放来适应屏幕，然后直接渲染，不需要参照 Camera（就算Scene中完全没有Camera，UI也会渲染）。如果Screen的size或者分辨率发生变化，那么UI也会进行缩放适应变化。UI会始终渲染着其他任何相机视图的上层。

**注意：Screen SpaceOverlay canvas**需要放在Hierarchy所有元素的顶层，否则UI可能会从视图中消失。这是Unity内置的限制。

### Screen Space - Camera

**在这个模式，Canvas渲染时，类似于将对象绘制在距离Camera一定距离的plane上。当UI进行缩放适应相机截面时，在荧幕上的size不受与相机距离影响。如果屏幕的size或者相机的视锥发生变化，UI会自动缩放适应。任何3d对象都比UI距离camera更近，才能在ui前面渲染，否则会被遮盖。**

### World Space

**这个模式，UI渲染被看作一个平面对象，不像Space-Camera模式那样无论何时都需要面向Camera，这个模式下你可以任意调整UI朝向。Canvas的Size可以通过他的RectTransform来设置，但是他的屏幕Szie依赖相机观察的角度以及与相机的距离，其他场景的对象可以在Canvas的前，后显示，甚至穿插，**

# ****项目遇到的问题****

1.我的DropDown的block以及dropdownlist的canvas的rendermode与根Canvas不一样，设置了override。

2.dropdown发现dropdownlist，block事件被阻挡。

问题原因：事件接收的camera不一样，dropdownList的canvas的worldcamera或者说graphicraycaster的eventCamera与rootcamera不一致，导致了，事件阻挡发生问题。

问题结决：

项目中的Camera不能乱用，一来是对性能有影响，二来容易造成渲染混乱，或者事件接收混乱。所以，决定去掉项目中的gameuiCamera，共用gameCamera（mainCamera）。之前之所以使用gameuicamera是犯了一个想当然错误，认为ui应该始终在游戏最上层使用，所以认为只用使用Camer的curllingmask与depth相结合解决。其实Canvas的PlaneDistance属性就是指的UI与canvas的worldCamera间的距离，可以理解为Z轴，这个值可以解决渲染层级问题。