**Unity3D教程：GUI的布局模式**

Posted on 2013年06月11日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 75 次

**Fixed Layout vs Automatic Layout**

有两种方法让你去布置和组织你的GUI，一个是固定的，一个是活动的，所有GUI打头的例子都是固定的，要想使用活动的Layout，就得用GUILayout代替GUI.它同样也是写在OnGUI()函数里。

固定Layout是用在你预先设计好界面的时候，活动Layout工作在你不知道会有多少组件或者如何定位他们的时候，这里有个例子:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | function OnGUI () { |
| 02 |  |
| 03 | *// 固定Layout* |
| 04 |  |
| 05 | GUI.Button (Rect (25,25,100,30), “I am a **Fixed** Layout Button”); |
| 06 |  |
| 07 | *// 活动Layout* |
| 08 |  |
| 09 | GUILayout.Button (“I am an Automatic Layout Button”); |
| 10 |  |
| 11 | } |

**排列控制**

在活动Layout模式里，你可以控制组件的位置，并组织他们。在固定模式下你可以把多个控件放进Groups中，在活动模式下可以把他们放在Areas， Horizontal Groups和Vertical Groups里。

**固定Layout中的Groups**

Groups 在固定Layout模式中起到组织可用项的，他让你在屏幕的一个区域中包含多个控件。你要把定义的控件放在GUI.BeginGroup()和 GUI.EndGroup()这对函数中间，所有控件的位置坐标都以Groups的0坐标为起点，假如你更改了group坐标,那么内部的控件也会跟随改变。

举个例子:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | *// 在Group中有一个box和一个button.这里,两个控件是属于Group的.* |
| 02 |  |
| 03 | function OnGUI(){ |
| 04 |  |
| 05 | GUI.BeginGroup (Rect (Screen.width / 2 , Screen.height / 2 &minus; 50, 100, 100)); |
| 06 |  |
| 07 | GUI.Box (Rect (0,0,100,100), “**Group** is here”); |
| 08 |  |
| 09 | GUI.Button (Rect (10,40,80,30), “Click me”); |
| 10 |  |
| 11 | GUI.EndGroup (); |
| 12 |  |
| 13 | } |

Group也可以嵌套，例如:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **var** bgImage : Texture2D; *// background image that is 256 x 32* |
| 02 |  |
| 03 | **var** fgImage : Texture2D; *// foreground image that is 256 x 32* |
| 04 |  |
| 05 | **var** playerEnergy = 1.0; *// a float between 0.0 and 1.0* |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | *// Create one Group to contain both images* |
| 10 |  |
| 11 | *// Adjust the first 2 coordinates to place it somewhere else on&minus;screen* |
| 12 |  |
| 13 | GUI.BeginGroup (Rect (0,0,256,32)); |
| 14 |  |
| 15 | *// Draw the background image* |
| 16 |  |
| 17 | GUI.Box (Rect (0,0,256,32), bgImage); |
| 18 |  |
| 19 | *// Create a second Group which will be clipped* |
| 20 |  |
| 21 | *// We want to clip the image and not scale it, which is why we need the second Group* |
| 22 |  |
| 23 | GUI.BeginGroup (Rect (0,0,playerEnergy \* 256, 32)); |
| 24 |  |
| 25 | *// Draw the foreground image* |
| 26 |  |
| 27 | GUI.Box (Rect (0,0,256,32), fgImage); |
| 28 |  |
| 29 | *// End both Groups* |
| 30 |  |
| 31 | GUI.EndGroup (); |
| 32 |  |
| 33 | GUI.EndGroup (); |
| 34 |  |
| 35 | } |

和我们熟知的if-else语句一样,相互最接近的begin和end为一组。

**活动Layout-Areas**

Areas只用于活动Layout模式.作用和固定模式下的Group一样。

在活动模式下，你可以定义Area的范围，也可以不定义。不定义的时候整个屏幕就是它的范围，如果手工定义了它的范围，那么它内部的控件将以Area的坐标为起始坐标。Unity3D教程手册

一个例子:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | function OnGUI () { |
| 02 |  |
| 03 | GUILayout.Button (“I am not inside an Area”); |
| 04 |  |
| 05 | GUILayout.BeginArea (Rect (Screen.width/2, Screen.height/2, 300, 300)); |
| 06 |  |
| 07 | GUILayout.Button (“I am completely inside an Area”); |
| 08 |  |
| 09 | GUILayout.EndArea (); |
| 10 |  |
| 11 | } |

**自动Layout - Horizontal and Vertical Groups**

自动模式中还有还有两对组函数:GUILayout.BeginHoriztontal()，GUILayout.EndHorizontal()，GUILayout.BeginVertical()，和GUILayout.EndVertical().他们用在Area中，同样也是成对使用的。

其特点是Horiztontal中的控件呈水平排列，Vertical中的控件呈垂直排列。

例子一个:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **var** sliderValue = 1.0; |
| 02 |  |
| 03 | **var** maxSliderValue = 10.0; |
| 04 |  |
| 05 | function OnGUI() |
| 06 |  |
| 07 | { |
| 08 |  |
| 09 | *// Wrap everything in the designated GUI Area* |
| 10 |  |
| 11 | GUILayout.BeginArea (Rect (20,0,200,60)); |
| 12 |  |
| 13 | *// Begin the singular Horizontal Group* |
| 14 |  |
| 15 | GUILayout.BeginHorizontal(); |
| 16 |  |
| 17 | *// Place a Button normally* |
| 18 |  |
| 19 | **if** (GUILayout.RepeatButton (“Increase max\nSlider **Value**”)) |
| 20 |  |
| 21 | { |
| 22 |  |
| 23 | maxSliderValue += 3.0 \* Time.deltaTime; |
| 24 |  |
| 25 | } |
| 26 |  |
| 27 | *// Arrange two more Controls vertically beside the Button* |
| 28 |  |
| 29 | *// 这里面有两个控件 是按照垂直的形式排列的* |
| 30 |  |
| 31 | GUILayout.BeginVertical(); |
| 32 |  |
| 33 | GUILayout.Box(“Slider **Value**: ” + Mathf.Round(sliderValue)); |
| 34 |  |
| 35 | sliderValue = GUILayout.HorizontalSlider (sliderValue, 0.0, maxSliderValue); |
| 36 |  |
| 37 | *// End the Groups and Area* |
| 38 |  |
| 39 | GUILayout.EndVertical(); |
| 40 |  |
| 41 | GUILayout.EndHorizontal(); |
| 42 |  |
| 43 | GUILayout.EndArea(); |
| 44 |  |
| 45 | } |

　　用GUILayoutOptions去定义一些控件 　　比如可以控制按钮的长度,如:

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | function OnGUI () { |
| 02 |  |
| 03 | GUILayout.BeginArea (Rect (100, 50, Screen.width&minus;200, Screen.height&minus;100)); |
| 04 |  |
| 05 | GUILayout.Button (“I am a regular Automatic Layout Button”); |
| 06 |  |
| 07 | GUILayout.Button (“My width has been overridden”, GUILayout.Width (95)); |
| 08 |  |
| 09 | GUILayout.EndArea (); |
| 10 |  |
| 11 | } |

这里GUILayout.Width (95)就定义了按钮的长度为95。