**Unity3D教程：强制屏幕纵宽比**

Posted on 2013年06月21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 54 次

强制屏幕到所需要的长宽比，可以根据需要自由选择是否应用强制比例。其中包括返回校正后的屏幕宽度，高度(Screen.width/height)和鼠标位置(Input.mousePosition)。

将这个脚本放在编译顺序较优先的文件夹中，比如 Standard Assets， (Plugins是最先编译的)，因此你可以用Js，C#等语言调用AspectUtility来使用它。首先要将它附加到Camera上，它同样可以附加给其他物体，如果你赋给其他物体，它将主动尝试找到标签为“Main Camera”的相机。对于WantedAspectRatio这个值，常见的是4:3为1.333333，16:10为1.6，16:9为1.777778，如果屏幕的纵宽比与设定的相同，不会有任何变化。如果与设定的纵宽比不同，将用黑色填充缺少的部分。

这样会导致有些方法返回不正确，比如Screen.width和Screen.height会返回实际屏幕尺寸，而不是主相机的尺寸，为了纠正这个问题，可以使用AspectUtility.screenWidth和AspectUtility.screenHeight。

同样的，Input。mousePosition也会有些问题，这种情况可以使用AspectUtility。mousePosition获取需要的值。

有一点要注意，这些变量在脚本刚唤醒的时候可能不能正确获取到，这就需要等待脚本开始后再访问它们。

当屏幕尺寸变化，例如Webplayer，采用4:3的比例，用户切换全屏后变成了16:9，这时候就需要用AspectUtility.SetCamera()重设脚本相机。

OnGUI代码在用AspectRatioEnforcer时需要做一些额外的工作。OnGUI的代码独立于Camera画在屏幕上的。所以相机改变了矩阵尺寸，它不会自动变化。这时候我们要特别处理一下。

如果你是用GUILayout来排列GUI，可以用GUILayout。BeginArea和EndArea来定义一个范围，像下面这样：

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

|  |  |
| --- | --- |
| 1 | function OnGUI () { |
| 2 | GUILayout.Label("Hello"); |
| 3 | GUILayout.Label("there"); |
| 4 | } |

修改为：

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

|  |  |
| --- | --- |
| 1 | function OnGUI () { |
| 2 | GUILayout.BeginArea(AspectUtility.screenRect); |
| 3 |  |
| 4 | GUILayout.Label("Hello"); |
| 5 | GUILayout.Label("there"); |
| 6 |  |
| 7 | GUILayout.EndArea(); |
| 8 | } |

如果已经用过BeginArea/EndArea，你可以直接在外面再嵌套AspectUtility.screenRect，它仍旧会处理好的。

如果你没有使用GUILayout，那么可以添加x和y的偏移来修正所有GUI的Rect，可以用AspectUtility.xOffset 和 AspectUtility.yOffset来得到偏移量，比如你平时这样写:

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

|  |  |
| --- | --- |
| 1 | function OnGUI () { |
| 2 | GUI.Label(Rect(50, 50, 100, 30), "Hello"); |
| 3 | GUI.Label(Rect(75, 75, 100, 30), "there"); |
| 4 | } |

现在改成：

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

|  |  |
| --- | --- |
| 1 | function OnGUI () { |
| 2 | **var** x = AspectUtility.xOffset; |
| 3 | **var** y = AspectUtility.yOffset; |
| 4 |  |
| 5 | GUI.Label(Rect(x + 50, y + 50, 100, 30), "Hello"); |
| 6 | GUI.Label(Rect(x + 75, y + 75, 100, 30), "there"); |
| 7 | } |

通常在OnGUI里获得鼠标位置用的是Event.current.mousePosition,在这里我们应该改使用AspectUtility.guiMousePosition.

AspectUtility.cs脚本，程序代码csharp代码：

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

|  |  |
| --- | --- |
| 001 | **using** UnityEngine; |
| 002 |  |
| 003 | **public** **class** AspectUtility : MonoBehaviour { |
| 004 |  |
| 005 | **public** **float** \_wantedAspectRatio = 1.3333333f; |
| 006 | **static** **float** wantedAspectRatio; |
| 007 | **static** Camera cam; |
| 008 | **static** Camera backgroundCam; |
| 009 |  |
| 010 | **void** Awake () { |
| 011 | cam = camera; |
| 012 | **if** (!cam) { |
| 013 | cam = Camera.main; |
| 014 | } |
| 015 | **if** (!cam) { |
| 016 | Debug.LogError ("No camera available"); |
| 017 | **return**; |
| 018 | } |
| 019 | wantedAspectRatio = \_wantedAspectRatio; |
| 020 | SetCamera(); |
| 021 | } |
| 022 |  |
| 023 | **public** **static** **void** SetCamera () { |
| 024 | **float** currentAspectRatio = (**float**)Screen.width / Screen.height; |
| 025 | *// If the current aspect ratio is already approximately equal to the desired aspect ratio,* |
| 026 | *// use a full-screen Rect (in case it was set to something else previously)* |
| 027 | **if** ((**int**)(currentAspectRatio \* 100) / 100.0f == (**int**)(wantedAspectRatio \* 100) / 100.0f) { |
| 028 | cam.rect = new Rect(0.0f, 0.0f, 1.0f, 1.0f); |
| 029 | **if** (backgroundCam) { |
| 030 | Destroy(backgroundCam.gameObject); |
| 031 | } *//Unity3D教程手册：www.unitymanual.com* |
| 032 | **return**; |
| 033 | } |
| 034 | *// Pillarbox* |
| 035 | **if** (currentAspectRatio > wantedAspectRatio) { |
| 036 | **float** inset = 1.0f - wantedAspectRatio/currentAspectRatio; |
| 037 | cam.rect = new Rect(inset/2, 0.0f, 1.0f-inset, 1.0f); |
| 038 | } |
| 039 | *// Letterbox* |
| 040 | **else** { |
| 041 | **float** inset = 1.0f - currentAspectRatio/wantedAspectRatio; |
| 042 | cam.rect = new Rect(0.0f, inset/2, 1.0f, 1.0f-inset); |
| 043 | } |
| 044 | **if** (!backgroundCam) { |
| 045 | *// Make a new camera behind the normal camera which displays black; otherwise the unused space is undefined* |
| 046 | backgroundCam = new GameObject("BackgroundCam", typeof(Camera)).camera; |
| 047 | backgroundCam.depth = **int**.MinValue; |
| 048 | backgroundCam.clearFlags = CameraClearFlags.SolidColor; |
| 049 | backgroundCam.backgroundColor = Color.black; |
| 050 | backgroundCam.cullingMask = 0; |
| 051 | } |
| 052 | } |
| 053 |  |
| 054 | **public** **static** **int** screenHeight { |
| 055 | **get** { |
| 056 | **return** (**int**)(Screen.height \* cam.rect.height); |
| 057 | } |
| 058 | } |
| 059 |  |
| 060 | **public** **static** **int** screenWidth { |
| 061 | **get** { |
| 062 | **return** (**int**)(Screen.width \* cam.rect.width); |
| 063 | } *//Unity3D教程手册：www.unitymanual.com* |
| 064 | } |
| 065 |  |
| 066 | **public** **static** **int** xOffset { |
| 067 | **get** { |
| 068 | **return** (**int**)(Screen.width \* cam.rect.x); |
| 069 | } |
| 070 | } |
| 071 |  |
| 072 | **public** **static** **int** yOffset { |
| 073 | **get** { |
| 074 | **return** (**int**)(Screen.height \* cam.rect.y); |
| 075 | } |
| 076 | } |
| 077 |  |
| 078 | **public** **static** Rect screenRect { |
| 079 | **get** { |
| 080 | **return** new Rect(cam.rect.x \* Screen.width, cam.rect.y \* Screen.height, cam.rect.width \* Screen.width, cam.rect.height \* Screen.height); |
| 081 | } |
| 082 | } |
| 083 |  |
| 084 | **public** **static** Vector3 mousePosition { |
| 085 | **get** { |
| 086 | Vector3 mousePos = Input.mousePosition; |
| 087 | mousePos.y -= (**int**)(cam.rect.y \* Screen.height); |
| 088 | mousePos.x -= (**int**)(cam.rect.x \* Screen.width); |
| 089 | **return** mousePos; |
| 090 | } |
| 091 | } |
| 092 |  |
| 093 | **public** **static** Vector2 guiMousePosition { |
| 094 | **get** { |
| 095 | Vector2 mousePos = **Event**.current.mousePosition; |
| 096 | mousePos.y = Mathf.Clamp(mousePos.y, cam.rect.y \* Screen.height, cam.rect.y \* Screen.height + cam.rect.height \* Screen.height); |
| 097 | mousePos.x = Mathf.Clamp(mousePos.x, cam.rect.x \* Screen.width, cam.rect.x \* Screen.width + cam.rect.width \* Screen.width); |
| 098 | **return** mousePos; |
| 099 | } |
| 100 | } |
| 101 | } |

AspectUtilityEnhanced.cs脚本：

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

|  |  |
| --- | --- |
| 001 | **using** UnityEngine; |
| 002 |  |
| 003 | **public** **class** AspectUtility : MonoBehaviour { |
| 004 |  |
| 005 | **public** **float** \_wantedAspectRatio = 1.5f; |
| 006 | **public** **bool** landscapeModeOnly = **true**; |
| 007 | **static** **public** **bool** \_landscapeModeOnly = **true**; |
| 008 | **static** **float** wantedAspectRatio; |
| 009 | **static** Camera cam; |
| 010 | **static** Camera backgroundCam; |
| 011 |  |
| 012 | **void** Awake () { |
| 013 | \_landscapeModeOnly = landscapeModeOnly; |
| 014 | cam = camera; |
| 015 | **if** (!cam) { |
| 016 | cam = Camera.main; |
| 017 | Debug.Log ("Setting the main camera " + cam.name); |
| 018 | } |
| 019 | **else** { |
| 020 | Debug.Log ("Setting the main camera " + cam.name); |
| 021 | } |
| 022 |  |
| 023 | **if** (!cam) { |
| 024 | Debug.LogError ("No camera available"); |
| 025 | **return**; |
| 026 | } |
| 027 | wantedAspectRatio = \_wantedAspectRatio; |
| 028 | SetCamera(); |
| 029 | } |
| 030 |  |
| 031 | **public** **static** **void** SetCamera () { |
| 032 | **float** currentAspectRatio = 0.0f; |
| 033 | **if**(Screen.orientation == ScreenOrientation.LandscapeRight || |
| 034 | Screen.orientation == ScreenOrientation.LandscapeLeft) { |
| 035 | Debug.Log ("Landscape detected..."); |
| 036 | currentAspectRatio = (**float**)Screen.width / Screen.height; |
| 037 | } |
| 038 | **else** { |
| 039 | Debug.Log ("Portrait detected...?"); |
| 040 | **if**(Screen.height > Screen.width && \_landscapeModeOnly) { |
| 041 | currentAspectRatio = (**float**)Screen.height / Screen.width; |
| 042 | } |
| 043 | **else** { |
| 044 | currentAspectRatio = (**float**)Screen.width / Screen.height; |
| 045 | } |
| 046 | } |
| 047 | *// If the current aspect ratio is already approximately equal to the desired aspect ratio,* |
| 048 | *// use a full-screen Rect (in case it was set to something else previously)* |
| 049 |  |
| 050 | Debug.Log ("currentAspectRatio = " + currentAspectRatio + ", wantedAspectRatio = " + wantedAspectRatio); |
| 051 |  |
| 052 | **if** ((**int**)(currentAspectRatio \* 100) / 100.0f == (**int**)(wantedAspectRatio \* 100) / 100.0f) { |
| 053 | cam.rect = new Rect(0.0f, 0.0f, 1.0f, 1.0f); |
| 054 | **if** (backgroundCam) { |
| 055 | Destroy(backgroundCam.gameObject); |
| 056 | } *//Unity3D教程手册：www.unitymanual.com* |
| 057 | **return**; |
| 058 | } |
| 059 |  |
| 060 | *// Pillarbox* |
| 061 | **if** (currentAspectRatio > wantedAspectRatio) { |
| 062 | **float** inset = 1.0f - wantedAspectRatio/currentAspectRatio; |
| 063 | cam.rect = new Rect(inset/2, 0.0f, 1.0f-inset, 1.0f); |
| 064 | } |
| 065 | *// Letterbox* |
| 066 | **else** { |
| 067 | **float** inset = 1.0f - currentAspectRatio/wantedAspectRatio; |
| 068 | cam.rect = new Rect(0.0f, inset/2, 1.0f, 1.0f-inset); |
| 069 | } |
| 070 | **if** (!backgroundCam) { |
| 071 | *// Make a new camera behind the normal camera which displays black; otherwise the unused space is undefined* |
| 072 | backgroundCam = new GameObject("BackgroundCam", typeof(Camera)).camera; |
| 073 | backgroundCam.depth = **int**.MinValue; |
| 074 | backgroundCam.clearFlags = CameraClearFlags.SolidColor; |
| 075 | backgroundCam.backgroundColor = Color.black; |
| 076 | backgroundCam.cullingMask = 0; |
| 077 | } |
| 078 | } |
| 079 |  |
| 080 | **public** **static** **int** screenHeight { |
| 081 | **get** { |
| 082 | **return** (**int**)(Screen.height \* cam.rect.height); |
| 083 | } |
| 084 | } |
| 085 |  |
| 086 | **public** **static** **int** screenWidth { |
| 087 | **get** { |
| 088 | **return** (**int**)(Screen.width \* cam.rect.width); |
| 089 | }*//Unity3D教程手册：www.unitymanual.com* |
| 090 | } |
| 091 |  |
| 092 | **public** **static** **int** xOffset { |
| 093 | **get** { |
| 094 | **return** (**int**)(Screen.width \* cam.rect.x); |
| 095 | } |
| 096 | } |
| 097 |  |
| 098 | **public** **static** **int** yOffset { |
| 099 | **get** { |
| 100 | **return** (**int**)(Screen.height \* cam.rect.y); |
| 101 | } |
| 102 | } |
| 103 |  |
| 104 | **public** **static** Rect screenRect { |
| 105 | **get** { |
| 106 | **return** new Rect(cam.rect.x \* Screen.width, cam.rect.y \* Screen.height, cam.rect.width \* Screen.width, cam.rect.height \* Screen.height); |
| 107 | } |
| 108 | } |
| 109 |  |
| 110 | **public** **static** Vector3 mousePosition { |
| 111 | **get** { |
| 112 | Vector3 mousePos = Input.mousePosition; |
| 113 | mousePos.y -= (**int**)(cam.rect.y \* Screen.height); |
| 114 | mousePos.x -= (**int**)(cam.rect.x \* Screen.width); |
| 115 | **return** mousePos; |
| 116 | } |
| 117 | } |
| 118 |  |
| 119 | **public** **static** Vector2 guiMousePosition { |
| 120 | **get** { |
| 121 | Vector2 mousePos = **Event**.current.mousePosition; |
| 122 | mousePos.y = Mathf.Clamp(mousePos.y, cam.rect.y \* Screen.height, cam.rect.y \* Screen.height + cam.rect.height \* Screen.height); |
| 123 | mousePos.x = Mathf.Clamp(mousePos.x, cam.rect.x \* Screen.width, cam.rect.x \* Screen.width + cam.rect.width \* Screen.width); |
| 124 | **return** mousePos; |
| 125 | } |
| 126 | } |