**屏幕动态画框脚本**

Posted on 2013年07月03日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script) /被围观 64 次

这个步骤很简单，就是在鼠标按下的时候记下鼠标位置，然后在鼠标移动时在当前鼠标位置和按下的位置之间画一个方形就行了。注意GL绘图都是每帧进行的，所以不需要清除，直接不绘制方框就消失了。

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

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 | **using** System.Collections; |
| 03 | **public** **class** DrawRectangle : MonoBehaviour { |
| 04 | **public** Color rectColor = Color.green; |
| 05 |  |
| 06 | **private** Vector3 start = Vector3.zero;*//记下鼠标按下位置* |
| 07 | **private** Material rectMat = **null**;*//画线的材质 不设定系统会用当前材质画线 结果不可控* |
| 08 | **private** **bool** drawRectangle = **false**;*//是否开始画线标志* |
| 09 |  |
| 10 | *// Use this for initialization* |
| 11 | **void** Start () { |
| 12 | rectMat = new Material( "Shader **\"**Lines/Colored Blended**\"** {" + |
| 13 | "SubShader { Pass { " + |
| 14 | " Blend SrcAlpha OneMinusSrcAlpha " + |
| 15 | " ZWrite Off Cull Off Fog { Mode Off } " + |
| 16 | " BindChannels {" + |
| 17 | " Bind **\"**vertex**\"**, vertex Bind **\"**color**\"**, color }" + |
| 18 | "} } }" );*//生成画线的材质* |
| 19 | rectMat.hideFlags = HideFlags.HideAndDontSave; |
| 20 | rectMat.shader.hideFlags = HideFlags.HideAndDontSave; |
| 21 | } *//Unity3D教程手册：www.unitymanual.com* |
| 22 | **void** Update () { |
| 23 | **if**(Input.GetMouseButtonDown(0)){ |
| 24 | drawRectangle = **true**;*//如果鼠标左键按下 设置开始画线标志* |
| 25 | start = Input.mousePosition;*//记录按下位置* |
| 26 | }**else** **if** (Input.GetMouseButtonUp(0)){ |
| 27 | drawRectangle = **false**;*//如果鼠标左键放开 结束画线* |
| 28 | } |
| 29 | } |
| 30 |  |
| 31 | **void** OnPostRender() {*//画线这种操作推荐在OnPostRender（）里进行 而不是直接放在Update，所以需要标志来开启* |
| 32 | **if**(drawRectangle){ |
| 33 | Vector3 end = Input.mousePosition;*//鼠标当前位置* |
| 34 | GL.PushMatrix();*//保存摄像机变换矩阵* |
| 35 |  |
| 36 | **if** (! rectMat) |
| 37 | **return**; |
| 38 | *//Unity3D教程手册：www.unitymanual.com* |
| 39 | rectMat.SetPass( 0 ); |
| 40 | GL.LoadPixelMatrix();*//设置用屏幕坐标绘图* |
| 41 | GL.Begin(GL.QUADS); |
| 42 | GL.Color( new Color(rectColor.r,rectColor.g,rectColor.b,0.1f) );*//设置颜色和透明度，方框内部透明* |
| 43 | GL.Vertex3( start.x,start.y,0); |
| 44 | GL.Vertex3( end.x,start.y,0); |
| 45 | GL.Vertex3( end.x,end.y,0 ); |
| 46 | GL.Vertex3( start.x,end.y,0 ); |
| 47 | GL.End(); |
| 48 | GL.Begin(GL.LINES); |
| 49 | GL.Color(rectColor);*//设置方框的边框颜色 边框不透明* |
| 50 | GL.Vertex3( start.x,start.y,0); |
| 51 | GL.Vertex3( end.x,start.y,0); |
| 52 | GL.Vertex3( end.x,start.y,0); |
| 53 | GL.Vertex3( end.x,end.y,0 ); |
| 54 | GL.Vertex3( end.x,end.y,0 ); |
| 55 | GL.Vertex3( start.x,end.y,0 ); |
| 56 | GL.Vertex3( start.x,end.y,0 ); |
| 57 | GL.Vertex3(start.x,start.y,0); |
| 58 | GL.End(); |
| 59 | GL.PopMatrix();*//恢复摄像机投影矩阵* |
| 60 | } |
| 61 | } |
| 62 | } |