**GUI常用的脚本类**

Posted on 2013年05月27日 by U3d / [Unity3D脚本/插件](http://www.unitymanual.com/category/script)/被围观 180 次

1 按钮生成

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

|  |  |
| --- | --- |
| 1 | view plaincopy |
| 2 | function OnGUI () { |
| 3 | **if** (GUI.Button (Rect (10,10,150,100), “I am a button”)) { |
| 4 | print (“You clicked the button!”); |
| 5 | } |
| 6 | } |

2 按钮场景载入

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Example level loader \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | *// Make a background box* |
| 08 |  |
| 09 | GUI.Box (Rect (10,10,100,90), “Loader Menu”); |
| 10 |  |
| 11 | *// Make the first button. If it is pressed, Application.Loadlevel (1) will be executed* |
| 12 |  |
| 13 | **if** (GUI.Button (Rect (20,40,80,20), “Level 1”)) { |
| 14 |  |
| 15 | Application.LoadLevel (1); |
| 16 |  |
| 17 | } |
| 18 |  |
| 19 | *// Make the second button.* |
| 20 |  |
| 21 | **if** (GUI.Button (Rect (20,70,80,20), “Level 2”)) { |
| 22 |  |
| 23 | Application.LoadLevel (2); |
| 24 |  |
| 25 | } |
| 26 |  |
| 27 | } |

3 按钮点击生效时间

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Flashing button example \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | **if** (Time.time % 2 < 1) { |
| 08 |  |
| 09 | **if** (GUI.Button (Rect (10,10,200,20), “Meet the flashing button”)) { |
| 10 |  |
| 11 | print (“You clicked me!”); |
| 12 |  |
| 13 | } |
| 14 |  |
| 15 | } |
| 16 |  |
| 17 | } |

4 创建按钮背景BOX

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Screen.width & Screen.height example \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | GUI.Box (Rect (0,0,100,50), “Top&minus;left”); |
| 08 |  |
| 09 | *//Rect 生成2D矩形的函数，用于摄像机，画面，GUI* |
| 10 |  |
| 11 | GUI.Box (Rect (Screen.width &minus; 100,0,100,50), “Top&minus;right”); |
| 12 |  |
| 13 | GUI.Box (Rect (0,Screen.height &minus; 50,100,50), “Bottom&minus;right”); |
| 14 |  |
| 15 | GUI.Box (Rect (Screen.width &minus; 100,Screen.height &minus; 50,100,50), “Bottom&minus;left”); |
| 16 |  |
| 17 | } |

5 在按钮上显示文字

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* String Content example \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | GUI.Label (Rect (0,0,100,50), “**This** is the text **string** **for** a Label Control”); |
| 08 |  |
| 09 | *//显示文字* |
| 10 |  |
| 11 | } |

6 显示图像，声明一个公共变量的Texture2D，并通过这样的内容作为参数变量的名称

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Texture2D Content example \*/* |
| 04 |  |
| 05 | **var** controlTexture : Texture2D; *//controlTexture为图像的名称* |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | GUI.Label (Rect (0,0,100,50), controlTexture); |
| 10 |  |
| 11 | } |

7 显示图像的例子

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Button Content examples \*/* |
| 04 |  |
| 05 | **var** icon : Texture2D; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | **if** (GUI.Button (Rect (10,10, 100, 50), icon)) { |
| 10 |  |
| 11 | print (“you clicked the icon”); |
| 12 |  |
| 13 | } |
| 14 |  |
| 15 | **if** (GUI.Button (Rect (10,70, 100, 20), “**This** is text”)) { |
| 16 |  |
| 17 | print (“you clicked the text button”); |
| 18 |  |
| 19 | } |
| 20 |  |
| 21 | } |

8 显示在一个图形用户界面控制的图像和文字在一起。可以为内容提供一个参数GUIContent对象，并定义字符串和图像显示的是在GUIContent。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Using GUIContent to display an image and a string \*/* |
| 04 |  |
| 05 | **var** icon : Texture2D; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | GUI.Box (Rect (10,10,100,50), GUIContent(“**This** is text”, icon)); |
| 10 |  |
| 11 | } |

9 还可以定义在GUIContent工具提示，当他鼠标停留在按钮上时显示提示

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Using GUIContent to display a tooltip \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | *// This line feeds “This is the tooltip” into GUI.tooltip* |
| 08 |  |
| 09 | GUI.Button (Rect (10,10,100,20), GUIContent (“Click me”, “**This** is the tooltip”)); |
| 10 |  |
| 11 | *// This line reads and displays the contents of GUI.tooltip* |
| 12 |  |
| 13 | GUI.Label (Rect (10,40,100,20), GUI.tooltip); |
| 14 |  |
| 15 | } |

10 也可以使用GUIContent来显示字符串，图标，工具提示

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Using GUIContent to display an image, a string, and a tooltip \*/* |
| 04 |  |
| 05 | **var** icon : Texture2D; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | GUI.Button (Rect (10,10,100,20), GUIContent (“Click me”, icon “**This** is the tooltip”)); |
| 10 |  |
| 11 | GUI.Label (Rect (10,40,100,20), GUI.tooltip); |
| 12 |  |
| 13 | } |

11 当鼠标停留在按钮上时显示提示【调用函数为GUI.tooltip】

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | function OnGUI () { |
| 04 |  |
| 05 | *// Make a button using a custom GUIContent parameter to pass in the tooltip.* |
| 06 |  |
| 07 | GUI.Button (Rect (10,10,100,20), GUIContent (“Click me”, “**This** is the tooltip”)); |
| 08 |  |
| 09 | *// Display the tooltip from the element that has mouseover or keyboard focus* |
| 10 |  |
| 11 | GUI.Label (Rect (10,40,100,40), GUI.tooltip); |
| 12 |  |
| 13 | } |

12 GUI 显示样式

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

|  |  |
| --- | --- |
| 1 | view plaincopy |
| 2 |  |
| 3 | **static** function Button (position : Rect, content : GUIContent, style : GUIStyle) : **bool** |
| 4 |  |
| 5 | *//bool 布尔类型 【1 or 0】* |

13 也可以使用GUIContent来显示字符串，图标，工具提示

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Using GUIContent to display an image, a string, and a tooltip \*/* |
| 04 |  |
| 05 | **var** icon : Texture2D; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | GUI.Button (Rect (10,10,100,20), GUIContent (“Click me”, icon “**This** is the tooltip”)); |
| 10 |  |
| 11 | GUI.Label (Rect (10,40,100,20), GUI.tooltip); *// GUI.tooltip用来显示提示的函数* |
| 12 |  |
| 13 | } |

14 GUI设置样式

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

|  |  |
| --- | --- |
| 1 | view plaincopy |
| 2 |  |
| 3 | **var** mystyle:GUIStyle; |
| 4 |  |
| 5 | function OnGUI(){ |
| 6 |  |
| 7 | GUI.Button(Rect(10,10,100,20),“0”,mystyle); |
| 8 |  |
| 9 | } |

15 Lable标签非交互式。它是仅用于显示。它不能被点击或以其他方式移动。最好是只显示信息

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

|  |  |
| --- | --- |
| 1 | view plaincopy |
| 2 |  |
| 3 | */\* GUI.Label example \*/* |
| 4 |  |
| 5 | function OnGUI () { |
| 6 |  |
| 7 | GUI.Label (Rect (25, 25, 100, 30), “Label”); |
| 8 |  |
| 9 | } |

16 Button是一个典型的互动式按钮。按钮包裹在一个if语句的GUI.Button功能。在if语句中是将要执行的按钮被点击时的代码

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.Button example \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | **if** (GUI.Button (Rect (25, 25, 100, 30), “Button”)) { |
| 08 |  |
| 09 | *// This code is executed when the Button is clicked* |
| 10 |  |
| 11 | } |
| 12 |  |
| 13 | } |

17 RepeatButton点击和松开Button分别触发，普通的Button只有一次触发

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.RepeatButton example \*/* |
| 04 |  |
| 05 | function OnGUI () { |
| 06 |  |
| 07 | **if** (GUI.RepeatButton (Rect (25, 25, 100, 30), “RepeatButton”)) { |
| 08 |  |
| 09 | *// This code is executed every frame that the RepeatButton remains clicked* |
| 10 |  |
| 11 | } |
| 12 |  |
| 13 | } |

18 TextField为显示函数，可以编辑显示中的字符串 【单行显示】

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.TextField example \*/* |
| 04 |  |
| 05 | **var** textFieldString = “text field”; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | textFieldString = GUI.TextField (Rect (25, 25, 100, 30), textFieldString); |
| 10 |  |
| 11 | } |

19 TextArea 可编辑一个包含多行文本的字符串并显示 【多行显示】

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.TextArea example \*/* |
| 04 |  |
| 05 | **var** textAreaString = “text area”; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | textAreaString = GUI.TextArea (Rect (25, 25, 100, 30), textAreaString); |
| 10 |  |
| 11 | } |

20 Toggle函数，建立一个复选框，用户通过点击复选框，来切换开关状态，同时根据点击返回布尔真假值

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.Toggle example \*/* |
| 04 |  |
| 05 | **var** toggleBool = **true**; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | toggleBool = GUI.Toggle (Rect (25, 25, 100, 30), toggleBool, “Toggle”); |
| 10 |  |
| 11 | } |

21 Toolbar函数，在工具栏是通过整数积极跟踪按钮。因此必须提供给函数参数一个整数。为了使工具栏的互动，必须指定整数函数的返回值。数组中的内容将决定显示在工具栏上所提供的按钮数量。点击按钮时，按钮一直处于被激活状态，直到点击另一个按钮。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.Toolbar example \*/* |
| 04 |  |
| 05 | **var** toolbarInt = 0; |
| 06 |  |
| 07 | **var** toolbarStrings : **String**[] = [“Toolbar1”, “Toolbar2”, “Toolbar3”]; |
| 08 |  |
| 09 | function OnGUI () { |
| 10 |  |
| 11 | toolbarInt = GUI.Toolbar (Rect (25, 25, 250, 30), toolbarInt, toolbarStrings); |
| 12 |  |
| 13 | } |

22 SelectionGrid函数，控制一个多行的工具栏，可以改变按钮的列数和行数，只有一个按钮可以处于激活状态。SelectionGrid按钮是通过跟踪一个整数。因此必须提供的函数参数中的一个整数。为了使SelectionGrid互动性，必须指定整数函数的返回值。在数组中的内容是在SelectionGrid显示的按钮的数量。还可以通过函数的参数决定的列数。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* GUI.SelectionGrid example \*/* |
| 04 |  |
| 05 | **var** selectionGridInt : **int** = 0; |
| 06 |  |
| 07 | **var** selectionStrings : **String**[] = [“Grid 1”, “Grid 2”, “Grid 3”, “Grid 4”]; |
| 08 |  |
| 09 | function OnGUI () { |
| 10 |  |
| 11 | selectionGridInt = GUI.SelectionGrid (Rect (25, 25, 100, 30), selectionGridInt, selectionStrings, 2); |
| 12 |  |
| 13 | } |

23 HorizontalSlider 横向滑动函数，可以拖动到预定的变化之间的最大值和最小值。滑块的旋钮位置存储为一个浮动。要显示旋钮的位置，由于在函数的参数之一浮动。需要提供有两种确定最低和最高的值。如果想滑块旋钮可调，分配给滑块函数返回值。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Horizontal Slider example \*/* |
| 04 |  |
| 05 | **var** hSliderValue : **float** = 0.0; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | hSliderValue = GUI.HorizontalSlider (Rect (25, 25, 100, 30), hSliderValue, 0.0, 10.0); |
| 10 |  |
| 11 | } |

24 VerticalSlider 垂直滑动函数， 可以拖动到预定的变化之间的最小值和最大值。滑块的旋钮位置存储为一个浮动。要显示旋钮的位置，由于在函数的参数之一浮动。需要提供有两种确定最低和最高值。如果想滑块旋钮可调，分配给滑块函数返回值。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Vertical Slider example \*/* |
| 04 |  |
| 05 | **var** vSliderValue : **float** = 0.0; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | vSliderValue = GUI.VerticalSlider (Rect (25, 25, 100, 30), vSliderValue, 10.0, 0.0); |
| 10 |  |
| 11 | } |

25 HorizontalScrollbar横向滚动函数，控制是类似于滑块控制，但视觉类似于Web浏览器或字处理器滚动元素。这种控制用于导航滚动控制。实现相同的水平滚动条和水平滑块一个例外：有一个额外的参数，它控制旋钮本身的滚动条宽度。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Horizontal Scrollbar example \*/* |
| 04 |  |
| 05 | **var** hScrollbarValue : **float**; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | hScrollbarValue = GUI.HorizontalScrollbar (Rect (25, 25, 100, 30), hScrollbarValue, 1.0, 0.0, 10.0); |
| 10 |  |
| 11 | } |

26 VerticalScrollbar纵向滚动函数，控制是类似于滑块控制，但视觉类似于Web浏览器或字处理器滚动元素。这种控制用于导航滚动控制。垂直滚动实施垂直滑块相同，但有一个例外：有一个额外的参数，它控制旋钮的滚动条本身的高度。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* Vertical Scrollbar example \*/* |
| 04 |  |
| 05 | **var** vScrollbarValue : **float**; |
| 06 |  |
| 07 | function OnGUI () { |
| 08 |  |
| 09 | vScrollbarValue = GUI. VerticalScrollbar (Rect (25, 25, 100, 30), vScrollbarValue, 1.0, 10.0, 0.0); |
| 10 |  |
| 11 | *// vScrollbarValue =中的vScrollbarValue为返回值，括号内的vScrollbarValue为设置滑块的滑动速度* |
| 12 |  |
| 13 | } |

27 ScrollViews函数，显示一个更大的可视区域且可滚动。ScrollViews需要两个Rect作为参数。第一个Rect定义滚动屏幕上的可视区域的位置和大小。第二个Rect定义的可视区域内所载的空间的大小。如果可视区域内的空间是更大的，滚动条会显示为适当。还必须指定并提供二维向量，它存储可视区域显示的位置。

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | */\* ScrollView example \*/* |
| 04 |  |
| 05 | **var** scrollViewVector : Vector2 = Vector2.zero; *// Vector2是一个二维向量，Vector2.zero是二维向量的X轴Y轴的值都为0* |
| 06 |  |
| 07 | **var** innerText : **String** = “I am inside the ScrollView”; *//String类，初始化字符串* |
| 08 |  |
| 09 | function OnGUI () { |
| 10 |  |
| 11 | *// Begin the ScrollView* |
| 12 |  |
| 13 | scrollViewVector = GUI.BeginScrollView (Rect (25, 25, 100, 100), scrollViewVector, Rect (0, 0, 400, 400)); |
| 14 |  |
| 15 | *// Put something inside the ScrollView* |
| 16 |  |
| 17 | innerText = GUI.TextArea (Rect (0, 0, 400, 400), innerText); |
| 18 |  |
| 19 | *// End the ScrollView* |
| 20 |  |
| 21 | GUI.EndScrollView(); |
| 22 |  |
| 23 | } |

28 Window函数，建立窗口。它们的实现稍微不同于其他控件。每个窗口都有一个ID号。Window是唯一需要一个额外的功能才能正常工作的控件。你必须提供一个ID号码和一个函数的名称必须在窗口中执行。在窗口的函数中，可以建立实际的行为或包含的控件。

GUI.Window调用格式

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

|  |  |
| --- | --- |
| 01 | view plaincopy |
| 02 |  |
| 03 | **static** function Window (id : **int**, clientRect : Rect, func : WindowFunction, title : GUIContent, style : GUIStyle) : Rect |
| 04 |  |
| 05 | */\* Window example \*/* |
| 06 |  |
| 07 | **var** windowRect : Rect = Rect (20, 20, 120, 50); |
| 08 |  |
| 09 | function OnGUI () { |
| 10 |  |
| 11 | windowRect = GUI.Window (0, windowRect, WindowFunction, “My Window”); |
| 12 |  |
| 13 | } |
| 14 |  |
| 15 | function WindowFunction (windowID : **int**) { |
| 16 |  |
| 17 | *// Draw any Controls inside the window here* |
| 18 |  |
| 19 | } |