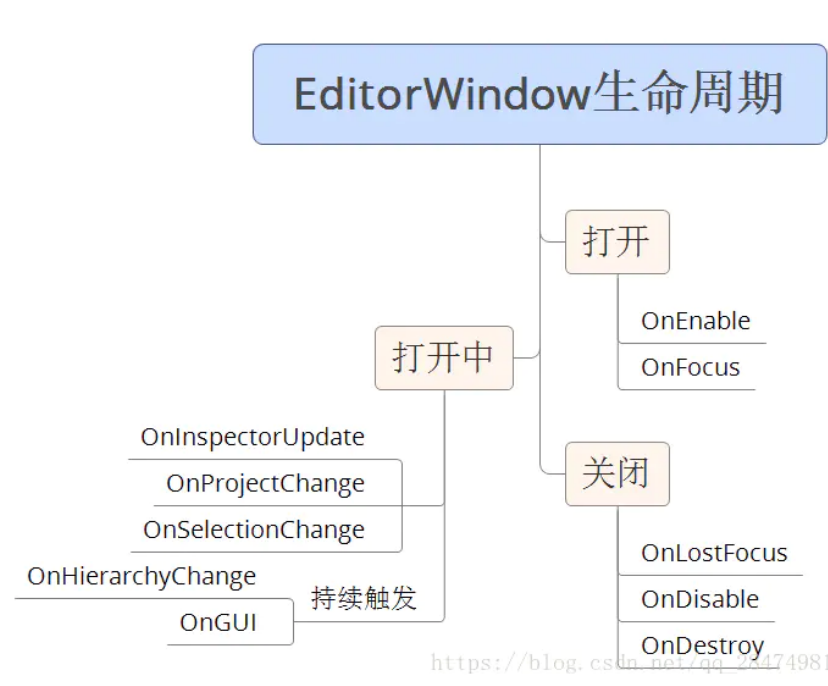
# EditorWindow



一.主要方法

OnEnable()：当打开界面的时候调用  
OnFocus()：当被聚焦的时候调用  
OnGUI()：当渲染UI的时候调用  
OnSelectionChange()：当选择发生更改时调用，选中的可选项（在Project和Hierarchy视图中）

OnLostFocus()：当失去焦点时调用

OnInspectorUpdate()：当属性界面更新时，几乎一直在更新

OnHierarchyChange()：当场景层次界面发生改变时调用");//在Hierarchy界面改变（增加、减少物体）

OnProjectChange()：当项目发生更改时调用");//在Project视图删除、增加文件

OnDisable()：当隐藏的时候调用

OnDestroy()：当销毁的时候调用

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEditor;

public class DemoEditorWindow : EditorWindow {

[MenuItem("CustomTools/Demo")]

public static void OnOpen()

{

EditorWindow.GetWindow<DemoEditorWindow>();

}

private void OnEnable()

{

Debug.Log("当打开界面的时候调用");

}

private void OnInspectorUpdate()

{

Debug.Log("当属性界面更新时");//几乎一直在更新

}

private void OnHierarchyChange()

{

Debug.Log("当场景层次界面发生改变时调用");//在Hierarchy界面改变（增加、减少物体）

}

private void OnProjectChange()

{

Debug.Log("当项目发生更改时调用");//在Project视图删除、增加文件

}

private void OnGUI()

{

Debug.Log("当渲染UI的时候调用");

}

private void OnFocus()

{

Debug.Log("当被聚焦的时候调用");

}

private void OnSelectionChange()

{

Debug.Log("当选择发生更改时调用");//选中的可选项（在Project和Hierarchy视图中）

}

private void OnLostFocus()

{

Debug.Log("当失去焦点时调用");

}

private void OnDisable()

{

Debug.Log("当隐藏的时候调用");

}

private void OnDestroy()

{

Debug.Log("当销毁的时候调用");

}

}

二.应用

1.进度条

//进度条展示

string s = progress \* 100 + "%";

EditorUtility.DisplayProgressBar("进度", s, progress);

//加载完毕，记得关闭进度条（重要）

EditorUtility.ClearProgressBar();

2.EditorWindow 窗口大小设置

EditorWindow ew = EditorWindow.GetWindow(typeof(FightCompareEditor));

ew.minSize = new Vector2(700, 900);

ew.maxSize = new Vector2(800, 1000);

3.label字体颜色、字体大小设置

var \_labelStyle = new GUIStyle(EditorStyles.label);

\_labelStyle.normal.textColor = Color.red;

\_labelStyle.fontSize = 20;

GUILayout.Label("第一步", \_labelStyle);

**4.画一条水平的虚线**

void drawline(Color c)

{

\_colorStyle.normal.textColor = c;

int len = Screen.width / 6;

EditorGUILayout.BeginHorizontal();

for (int i = 0; i < len; i++)

GUILayout.Label("┅", \_colorStyle);

EditorGUILayout.EndHorizontal();

}

**5.EditorWindow 刷新**

因为 OnGUI() 方法是事件驱动的，而非刷帧，见[EditorWindow.OnGUI()](https://links.jianshu.com/go?to=https%3A%2F%2Fdocs.unity3d.com%2FScriptReference%2FEditorWindow.OnGUI.html)。因此若需要定时刷新，可以如下：

private void Update()

{

Repaint();

}

**6.创建EditorWindow**

using UnityEditor;

using UnityEngine;

public class ZZEditorWindow : EditorWindow

{

[MenuItem("ZZTest/CreateWindows")]

static void ZZTestCreateWindow()

{

ZZEditorWindow zzWindow = EditorWindow.GetWindow(typeof(ZZEditorWindow), false, "ZZTestWindow", false) as ZZEditorWindow;

zzWindow.Show();

}

void OnEnable()

{

Debug.Log("ZZEditorWindow->OnEnable");

}

}

public static EditorWindow GetWindow(System.Type t, [DefaultValue("false")] bool utility, [DefaultValue("null")] string title, [DefaultValue("true")] bool focus)

{

return EditorWindow.GetWindowPrivate(t, utility, title, focus);

}

第一个参数是窗口类型，注意是一定要继承自EditorWindow。

第二个参数是窗口是否浮动，如果是就不能内嵌到unity其他窗口中去，如果不是就能嵌入其他窗口。默认是内嵌

第三个参数是窗口的标题，如果为空的话就采用类的名称来当标题。默认是空

第四个参数是窗口是否是焦点 默认是焦点

**三.具体介绍**

## 1.静态属性

focusedWindow 目前有键盘焦点的EditorWindow。（只读）

mouseOverWindow EditorWindow目前在鼠标光标下。（只读）

## 2.属性

autoRepaintOnSceneChange 　　　每当场景改变时，窗口是否自动重绘？  
maximized 　　　　　　　　　　　 这个窗口最大化了吗？  
maxSize 　　　　　　　　　　　　 这个窗口的最大尺寸。  
minSize 　　　　　　　　　　　　　这个窗口的最小尺寸。  
position 　　　　　　　　　　　　 窗口在屏幕空间中的所需位置。  
titleContent 　　　　　　　　　　　用于绘制EditorWindows标题的GUIContent。  
wantsMouseEnterLeaveWindow 　 检查在此编辑器窗口的GUI中是否接收到MouseEnterWindow和MouseLeaveWindow事件。  
wantsMouseMove 　　　　　　　　检查在此编辑器窗口的GUI中是否收到MouseMove事件。

## 3.公共方法

BeginWindows 　　　　 标记所有弹出窗口的开始区域。  
Close 　　　　　　　　 关闭编辑器窗口。  
EndWindows 　　　　 关闭一个以EditorWindow.BeginWindows开始的窗口组。  
Focus 　　　　　　　　 将键盘焦点移到另一个EditorWindow。  
RemoveNotification 　　 停止显示通知消息。  
Repaint 　　　　　　　　使窗口重绘。  
SendEvent 　　　　　　 发送一个事件到一个窗口。  
Show 　　　　　　　　 显示EditorWindow。  
ShowAsDropDown 　　 显示具有下拉行为和样式的窗口。  
ShowAuxWindow 　　　 在辅助窗口中显示编辑器窗口。  
ShowNotification 　　　 显示通知消息。  
ShowPopup 　　　　 使用弹出式框架显示编辑器窗口。  
ShowUtility 　　　　　　 将EditorWindow显示为浮动实用程序窗口。

## 4.静态方法

FocusWindowIfItsOpen 如果它是打开的，重点指定类型的第一个找到的EditorWindow。  
GetWindow 　　　　　　返回当前在屏幕上的类型t的第一个EditorWindow。  
GetWindowWithRect 返回当前在屏幕上的类型t的第一个EditorWindow。

## 5.消息

Awake 　　　　 　　在新窗口打开时调用。  
OnDestroy 　　　　　　调用OnDestroy关闭EditorWindow窗口。  
OnFocus 　　当窗口获得键盘焦点时调用。  
OnGUI 　　在这里实现你自己的编辑器GUI。  
OnHierarchyChange 　　当层次结构中的一个对象或一组对象发生变化时发送的消息的处理程序。  
OnInspectorUpdate　　 OnInspectorUpdate以每秒10帧的速度调用，以便检查员有机会进行更新。  
OnLostFocus 　　当窗口失去键盘焦点时调用。  
OnProjectChange 每当项目状态发生变化时发送消息的处理程序。  
OnSelectionChange 　 每当选择发生变化时调用。  
Update 　　　　　　　 在所有可见的窗口上每秒调用多次。

## 6.继承的成员

7.1 属性

hideFlags　　该物体应该隐藏，保存在场景中还是用户可以修改？  
name 　　　 对象的名称

7.2 公共方法

GetInstanceID 　　返回对象的实例ID。  
ToString 　　　　返回游戏对象的名称。

7.3 静态方法

Destroy 　　　　　　 删除一个游戏对象，组件或资产。  
DestroyImmediate 　　 立即销毁obj对象。强烈建议您改用Destroy。  
DontDestroyOnLoad 　　 加载新场景时，使对象目标不会自动销毁。  
FindObjectOfType 　　 返回Type类型的第一个活动加载的对象。  
FindObjectsOfType 　　 返回Type类型的所有活动加载对象的列表。  
Instantiate 　　　　　　　克隆原始对象并返回克隆。  
CreateInstance 　　　　　创建可脚本化对象的实例。

7.4 操作符

bool 　　　　　　对象是否存在？  
operator!= 　　 比较两个对象是否引用不同的对象。  
operator== 比较两个对象引用，看它们是否引用同一个对象。

7.5 消息

Awake 　　　　这个函数在ScriptableObject脚本启动时被调用。  
OnDisable 　　 这个函数在脚本对象超出范围时被调用。  
OnEnable 　　 这个函数在对象加载时被调用。