<http://blog.csdn.net/husheng0/article/details/52568027>

<http://blog.csdn.net/qq_33337811/article/details/62227146>

**架构：Editor与EditorWindow**

对编辑器的相关操作都是放在Unity文件夹下的Editor目录下的

Editor类和EditorWindow类都继承自同一个基类ScriptableObject，因此它们都可以针对某种脚本类来进行操作

Editor只能定制针对脚本的扩展，从脚本内容在Inspector里的显示布局，到变量在Scene视图的可视化编辑

EditorWindow主要是扩展编辑器的功能，不必针对某种脚本（虽然可以做到），而且它有独立的窗口，使用OnGUI函数来回执2D的UI

**2D的显示：GUIUtility，EditorGUIUtility**

**Editor**

using UnityEngine;

using System.Collections;

using UnityEditor;

[CustomEditor(typeof(Test1))]

public class Test1Editor : Editor

{

public Test1 test;

public void OnEnable()

{

test = (Test1)target;

}

public override void OnInspectorGUI()

{

test.myBool = EditorGUILayout.Toggle("是否选中", test.myBool);

test.myString = EditorGUILayout.TextField("输入文本", test.myString);

test.myWheel = EditorGUILayout.IntSlider("输入值", test.myWheel, 0, 100);

}

}

-------------------------------------------------

public class Test1 : MonoBehaviour

{

public bool myBool;

public string myString;

public int myWheel;

void Awake()

{

Debug.Log("myBool = " + myBool + ", myString = " + myString + ", myWheel = " + myWheel);

}

}



**EditorWindow**

public class Test2Editor : EditorWindow

{

[MenuItem("MyEditor/CreateWindow")]

static void AddWindow()

{

Rect wr = new Rect(0, 0, 500, 500);

Test2Editor window = (Test2Editor)EditorWindow.GetWindowWithRect(typeof(Test2Editor), wr);

window.Show();

}

private string text;

public void OnGUI()

{

text = EditorGUILayout.TextField("输入文字", text);

if (GUILayout.Button("打开通知", GUILayout.Width(200)))

{

this.ShowNotification(new GUIContent("This is a Notification"));

}

if (GUILayout.Button("关闭通知", GUILayout.Width(200)))

{

this.RemoveNotification();

}

if (GUILayout.Button("关闭窗口", GUILayout.Width(200)))

{

this.Close();

}

}

void OnFocus()

{

Debug.Log("获得焦点时调用一次");

}

void OnLostFocus()

{

Debug.Log("失去焦点时调用一次");

}

void OnHierarchyChange()

{

Debug.Log("Hierarchy视图中的任何对象发生改变时调用一次");

}

void OnProjectChange()

{

Debug.Log("Project视图中资源发生改变时调用一次");

}

//void OnInspectorUpdate()

//{

// Debug.Log("面板更新");

// this.Repaint();

//}

void OnSelectionChange()

{

// 处于开启状态，并且在Hierarchy视图中选择某个游戏对象时调用

foreach (Transform t in Selection.transforms)

{

Debug.Log("OnSelectionChange name = " + t.name);

}

}

void OnDestroy()

{

Debug.Log("关闭时调用");

}

}



