<http://blog.csdn.net/yyq9111/article/details/51249900>

using UnityEngine;

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEditor;

using System.IO;

public class CustomHotKey : Editor

{

[MenuItem("CustomHotKey/ToggleActive &v")]

public static void ToggleActive()

{

foreach (var obj in Selection.gameObjects)

{

Undo.RegisterCompleteObjectUndo(obj, obj.name);

obj.SetActive(!obj.activeSelf);

}

}

[MenuItem("CustomHotKey/TogglePause &p")]

public static void TogglePause()

{

EditorApplication.isPaused = !EditorApplication.isPaused;

}

[MenuItem("CustomHotKey/TogglePause &s")]

public static void SaveAssets()

{

AssetDatabase.SaveAssets();

Debug.Log("Project Saved");

}

[MenuItem("CustomHotKey/Open Containing Folder &o")]

public static void OpenAssetPath()

{

#if UNITY\_EDITOR

string path = Application.dataPath + "/../" + AssetDatabase.GetAssetPath(Selection.activeObject);

path = GetContainFolder(path);

if (path != string.Empty)

{

System.Diagnostics.Process.Start(path);

}

else

{

Debug.Log("the path is invalid, path = " + path);

}

#else

Debug.LogWarning("not support on current platform");

#endif

}

static bool hide = false;

[MenuItem("CustomHotKey/Toggle Selected wireframe &x")]

public static void ToggleWireframe()

{

if (Selection.activeGameObject == null) return;

Renderer[] renders = Selection.activeGameObject.GetComponentsInChildren<Renderer>();

for (int i = 0, size = renders.Length; i < size; i++)

{

EditorUtility.SetSelectedWireframeHidden(renders[i], hide);

}

hide = !hide;

}

[MenuItem("CustomHotKey/Clear Console &c")]

public static void ClearConsole()

{

Type logEntries = System.Type.GetType("UnityEditorInternal.LogEntries, UnityEditor.dll");

var method = logEntries.GetMethod("Clear", System.Reflection.BindingFlags.Static | System.Reflection.BindingFlags.Public);

if (method != null)

{

method.Invoke(null, null);

}

else

{

Debug.LogWarning("Can't find method");

}

}

[MenuItem("CustomHotKey/Export Package &e")]

public static void ExportPackage()

{

string path = GetSaveFolder("Save Package", "ExportPackage.SaveFolder");

if (string.IsNullOrEmpty(path))

{

Debug.LogWarning("path is null or empty");

return;

}

UnityEngine.Object[] selectedAsset = Selection.GetFiltered(typeof(object), SelectionMode.Deep);

List<string> list = new List<string>();

for (int i = 0, size = selectedAsset.Length; i < size; i++)

{

list.Add(AssetDatabase.GetAssetPath(selectedAsset[i]));

}

if (list.Count <= 0)

{

Debug.LogWarning("not resource is selected");

return;

}

ExportPackageOptions options = ExportPackageOptions.IncludeDependencies | ExportPackageOptions.Recurse;

AssetDatabase.ExportPackage(list.ToArray(), path, options);

}

// 保存对话框

public static string GetSaveFolder(string title, string key)

{

string lastFolder = PlayerPrefs.GetString(key);

string folder = EditorUtility.SaveFilePanel(title, lastFolder, "export.unitypackage", "");

if (!string.IsNullOrEmpty(folder))

{

PlayerPrefs.SetString(key, folder);

}

return folder;

}

// 获取当前path的包含目录

public static string GetContainFolder(string path)

{

if (string.IsNullOrEmpty(path)) return string.Empty;

if (Directory.Exists(path) || File.Exists(path)) // 目录或文件

{

int lastIndex = path.LastIndexOf(@"/");

if (lastIndex == -1)

{

Debug.LogWarning("the path is invalid, path = " + path);

return string.Empty;

}

path = path.Substring(0, lastIndex);

return path;

}

return string.Empty;

}

}

其他快捷键参考：

%X： ctrl + X

#X： shift + X

&X： alt + X

\_X: X

%&X: ctrl + alt + X