Console Enhanced

A replacement console for Unity3D

Support

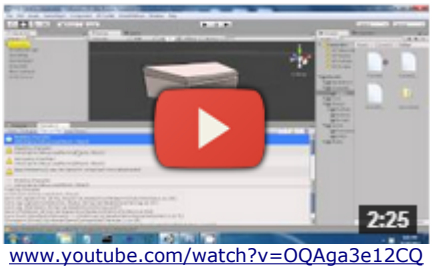
If you have any questions, ask us in the Official Forum Thread or email us directly at support@inhumangames.com

Showing the Console Window

In the Unity Editor, select Window->Console Enhanced. Alternatively, you can press the hotkey Command+Shift+C on OSX or Control+Shift+C on Windows.

Usage

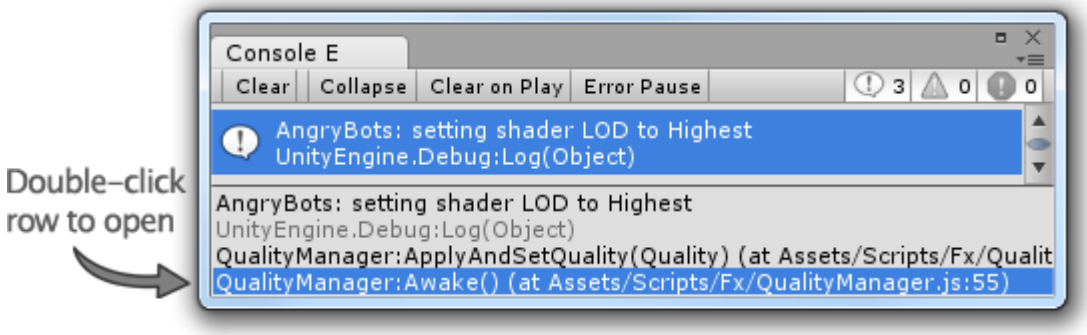
For the most part, using Console Enhanced is done in the same way as using Unity's built-in console. The exact same console entries shown in Unity's console are shown in Console Enhanced. This includes build warnings and errors. This video shows the basics on how to use the console.



www.youtube.com/watch?v=OQAga3e12CQ

Selecting Callstack Rows

A feature of Console Enhanced is the ability to easily open any row of the callstack. Double-click any row to open. Files are opened in the default external editor which can be configured in the Unity Editor under Edit->Preferences->External Tools->External Script Editor.

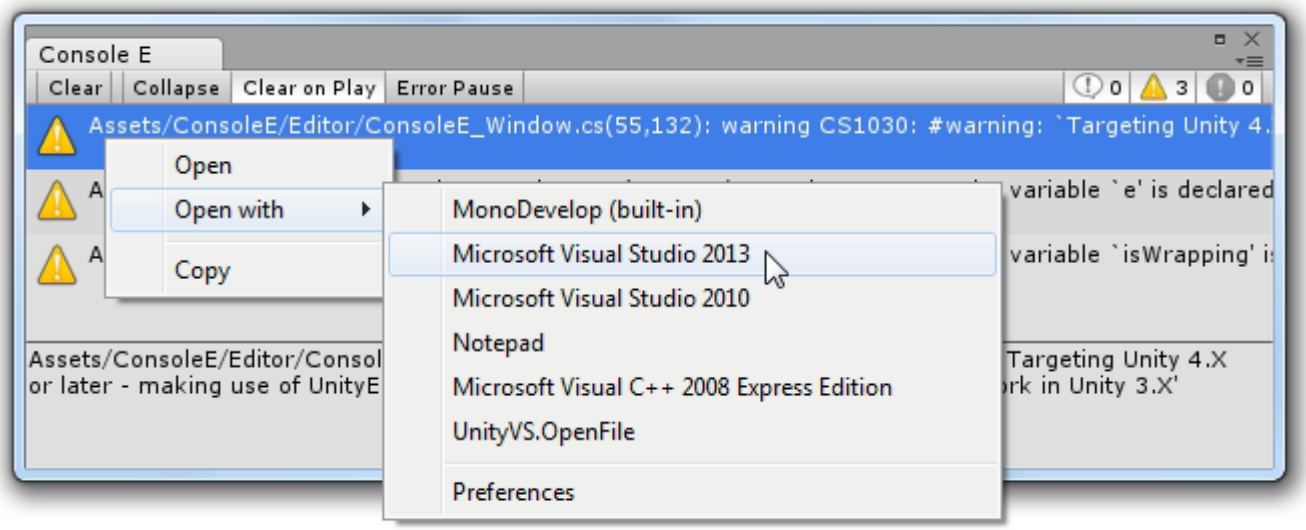


Double-clicking this row will open QualityManager.js at line 55

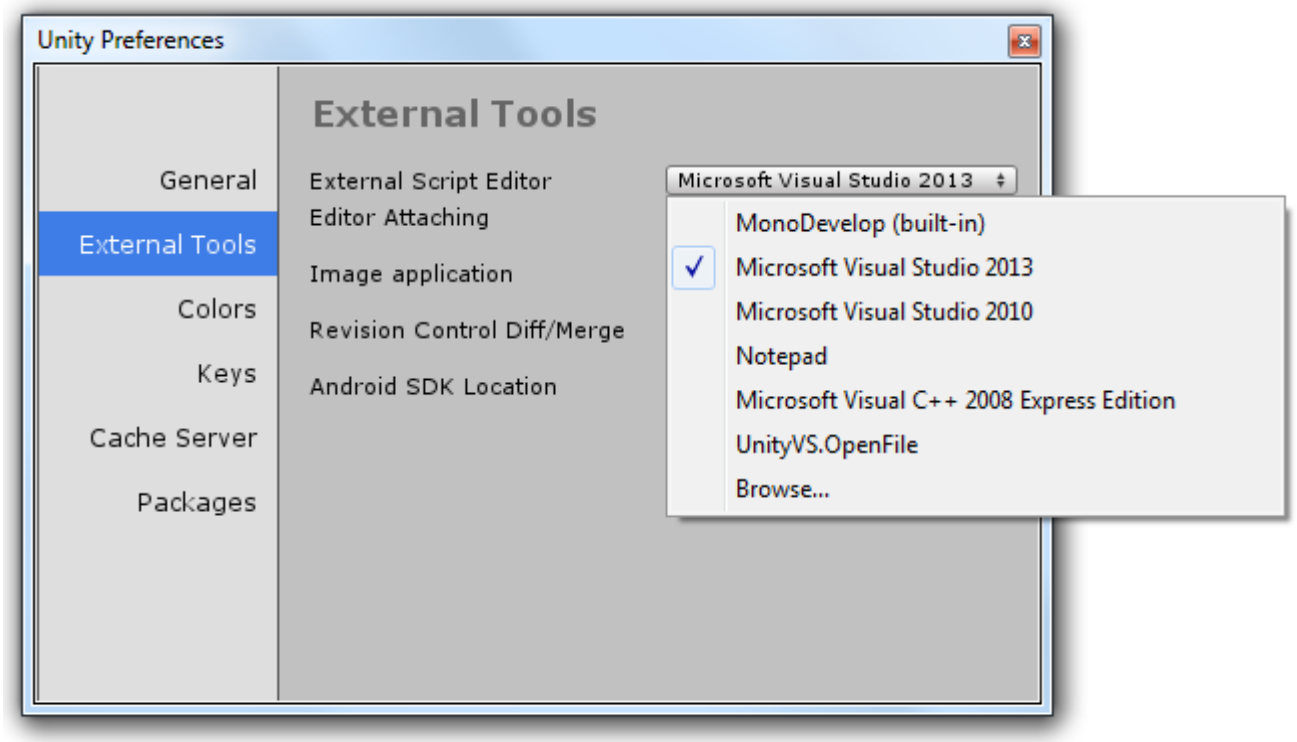
Right-clicking a row reveals a context menu. The context menu includes an option to add or remove the row from the ignore list (see Ignore List below). Also included is an option to show where the row's associated source file is located in the Unity Project Window.

Open With

Right-click on any entry or callstack row to show the Open With menu. This feature is useful if you edit your source code in an editor like Visual Studio, and debug your code using MonoDevelop.

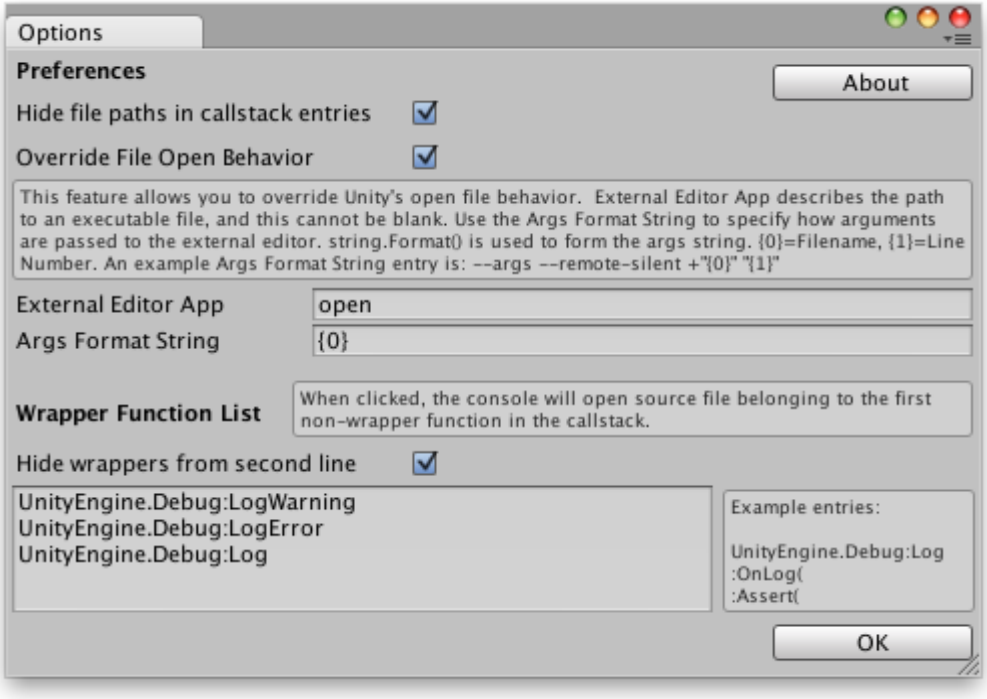


The list of options found in the Open With menu is automatically generated based on your Unity preferences. To edit this list, click the Preferences entry. You will then be taken to the External Tools section of your Unity preferences dialog where you can make any changes to your Open With list.



Options Dialog

To show the Options Dialog, right-click anywhere on the console window, when nothing is selected.



Wrapper Function List

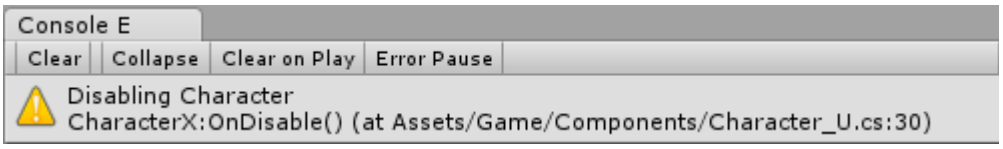
The console checks the Wrapper Function List when considering the best source file to open. The list is used to tell the console which functions are not important. A good example is a wrapper function that calls UnityEngine.Debug.Log().

For example, let's suppose your project implements a custom assert. Let's suppose UnityEngine.Debug.Log() is used to notify the developer when the assertion fails. Normally when you double-click a console entry generated from UnityEngine.Debug.Log(), you are taken to UnityEngine.Debug.Log(). In the case of the custom assert, you really want to be taken to the code that calls Assert(). This is achieved in Console Enhanced by placing your custom assert function in the ignore list.

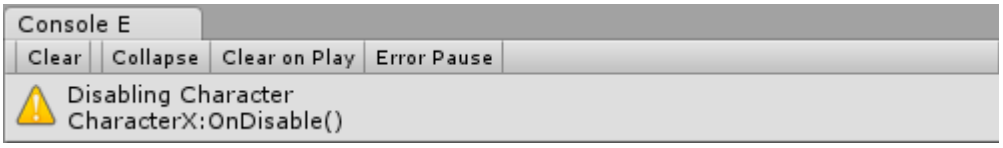
You can manually edit the Wrapper Function List from the options dialog. Alternatively, you can right-click a callstack row, and mark the row as a wrapper.

Hide file paths in second line

The second-line entry (of any main entry) sometimes contains a filename of a source file. This option removes the filename and path from the second line.



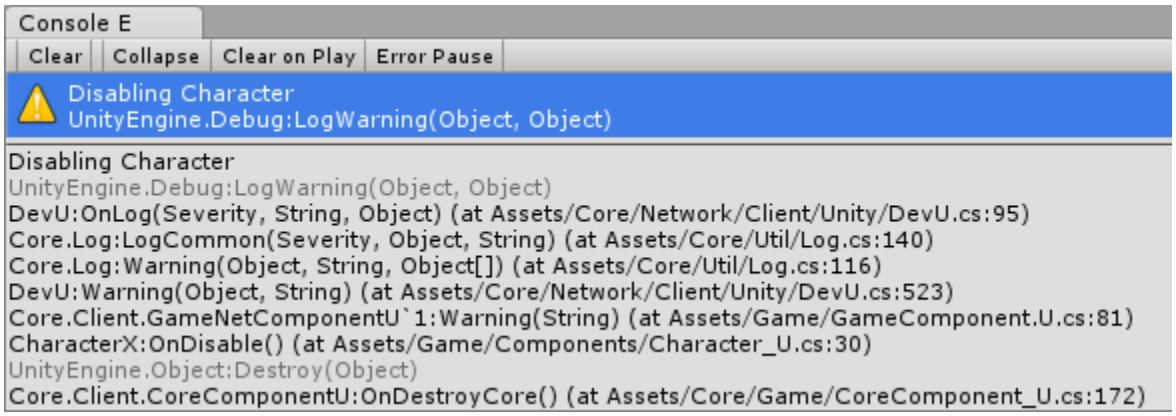
Hide file paths in second line is unchecked



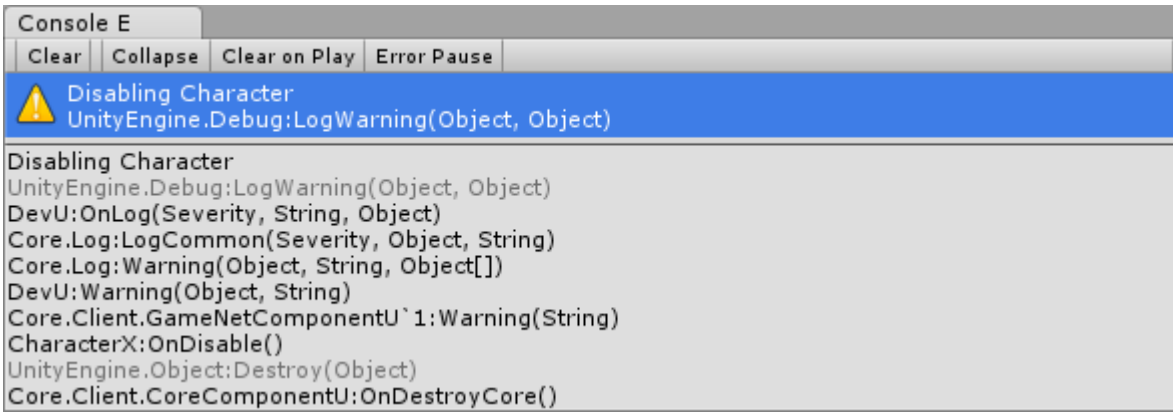
Hide file paths in second line is checked

Hide file paths in callstack entries

Callstack entries usually contain a filename and path of a source file. This option removes the filename, leaving just the function call.



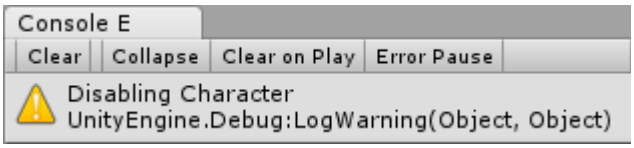
Hide file paths in callstack entries is unchecked



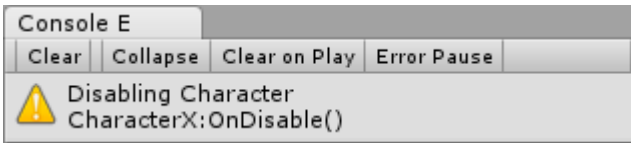
Hide file paths in callstack entries is checked

Hide wrappers from second line

When this option is checked, items in the Wrapper Function List are not shown in the second-line display (of any main entry). Instead, the line shown is the one that is opened if the entry is clicked.



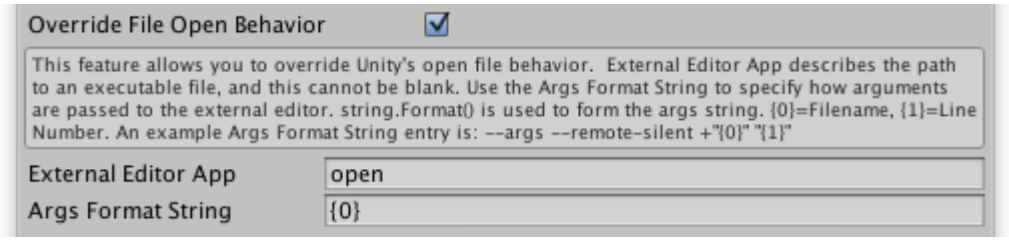
Hide ignored from second line is unchecked



Hide ignored from second line is checked

Override File Open Behavior

Unity 4.3 for Windows allows you to specify editor arguments including file and line info. The Override File Open Behavior feature in Console Enhanced is useful for users who want this ability in their console but are using older versions of Unity or using Unity for OSX.



Note that on OSX, you may need to prefix your Arg Format String with --args. The Arg Format String is formed using the C# string.Format function. {0} is replaced with the filename and {1} is replaced with the line number.

If this feature is enabled, only the primary external editor is overridden. Other editors opened with Open With will use the default open behavior.

Building from Sources

It's possible to build a new console DLL from files included in SourceCode.zip. Note that some low-level rendering source code is not included in the standard package, but if you would like to make high-level changes, these sources should be sufficient. It you need **all** source code, that is, the source code to ConsoleE_Renderer.dll, please contact support@inhumangames.com and include the invoice number of your order.

The types of changes possible by editing the provided sources include

- Editing context menus
- Changing what happens when an entry is opened
- Changing the hotkey used to open the Console Window

Debugging Console Source Changes

One method of building the console is to put the included source files directly in your project, replacing ConsoleE.dll. This method is useful for debugging, but your final console should be built as a DLL. If you keep some of the console in the form of C# source code, you may notice issues. The biggest known issue with this setup is that it's possible your editor window layout will be reset to the default window layout. This can happen if you open the Unity Editor while your project has a compile error. If there's an error, Unity may decide to not compile your changes to the console source. If this happens, the Unity Editor will not know how to render the console window, and then Unity will reset all your editor windows to the default layout. This is not a major problem if you previously saved your editor window layout using Unity's *Save Layout* feature. It's recommended you use your console in DLL form to avoid issues like this.

DLL Build Instructions

Unzip the files in SourceCode.zip to a folder and copy ConsoleE_Renderer.dll there too.

Open ConsoleE.sln in either MonoDevelop or Visual C#.

In your C# IDE, ensure your references to UnityEditor and UnityEngine are correctly pointing to UnityEditor.dll and UnityEngine.dll, respectively. These DLLs can be found where you installed Unity, under Editor/Data/Managed.

After building a new ConsoleE.dll, copy this file to your Unity project, replacing the official one there.