UConsole - Unity Plugin from Cardboard Keep

A Valve-style console for Unity to allow real-time calling of developer commands, in-engine and in builds.

How To Use

Setting up UConsole involves two steps. Firstly drop the UConsole prefab onto the uGUI canvas in any scene where you wish to use it.. Secondly define a list of C# functions as commands to call via the console.

The best way to get started is to watch the video tutorial available here:
https://www.youtube.com/watch?v=MbvvuOMh8b8

Adding the UConsole prefab

uGUI Canvas

UConsole works with the new Unity GUI system included from Unity 4.6 and up. If you're already using uGUI your scene will already have a canvas in it. If not, you can right click on your hierarchy and select UI > Canvas to add one.

UConsole prefab

Once you have a canvas simply locate the prefab called UConsole in the CK_UConsole plugin directory and drag it underneath the Canvas. Make sure it's parented to it, otherwise, like all uGUI elements, it won't be visible on screen.

Defining Commands

There are two ways to define commands for UConsole. The simple way doesn't require use of C# Inheritance, but you may lose your commands if you update the UConsole plugin.

In the Commands script (Simple)

Open up the UConsoleCommands.cs script in your preferred script editor. This script is heavily commented and should be pretty self-explanatory, make especially sure you read the "RULES" comment block, as you will need to do what it says for your functions to be visible in the console.

This script also contains a bunch of example commands, including testfunction, argumentfunction, and mynewfunction, which was created in the above tutorial video.
You can add your own functions in this area, following the rules, and have them influence any other script in your game. (If you're unfamiliar with communicating between scripts in Unity, you could start with SendMessage, static variables or static instances. Here's the officially Unity tutorial on the subject:
https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/communicating-between-components-gameobjects)
This script also has two more functions up the top, GameSpecificActivate and Deactivate. As their comment says, these will be called when the console is opened and closed, so you can put anything you want your game specifically to do on these events here, such as the example of pausing time.

Inherit a new child Commands script (Advanced)

What I recommend you do rather than the above, is inherit a new child script from UConsoleCommands and enter your games functions here. The main benefit of this is if you update to a newer version of UConsole in the future, it won't override your modified UConsoleCommands.cs with the base one. To do this just requires three new steps before the above:

1. Create a new script, I recommend <YourGameName>Commands.cs. For example ours is WardenCommands.cs.
2. Copy the contents of UConsoleCommands into it, so you have the comments, and replace MonoBehaviour on the public class line with UConsoleCommands, so it looks like this:
   `public class <YourGameName>Commands : UConsoleCommands {`
3. Find your UConsole instance in your scene, remove UConsoleCommands from it and add your new script.

That's it! You can now remove the default console functions from your new script (if you copied them over) and start writing your own.

Using UConsole in-game

Activating the console while your game is running, inside the editor or in a build, is as simple as hitting the ~ key (usually in the top left corner of your keyboard).

Something broke/I need help/I have a suggestion!

You can contact us at contact@cardboardkeep.com =)