CLIU - Command Line Interface for Unity

Version 1.0

Lucas Sampaio Dias
lucassampaiodias@gmail.com

Table of Contents

Introduction

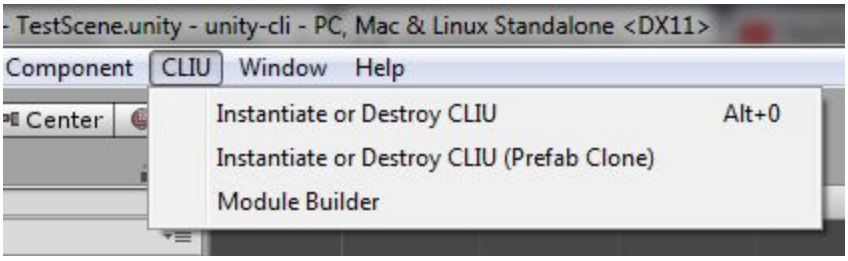
Welcome to the CLIU documentation! In this document you should be able to find all the information you need to use CLIU on your project, as well as many useful tips on how to make the most of this plugin.

If you need to customize anything, all UI elements are made using Unity's default UI solutions (NGUI), so feel free to use what you already are familiar with to adapt the CLIU window. Also, the source code for CLIU is available if you want to change or add anything.

Finally, be sure to send me an email (lucassampaiodias@gmail.com) if you have any suggestions of useful features or bugs to report.

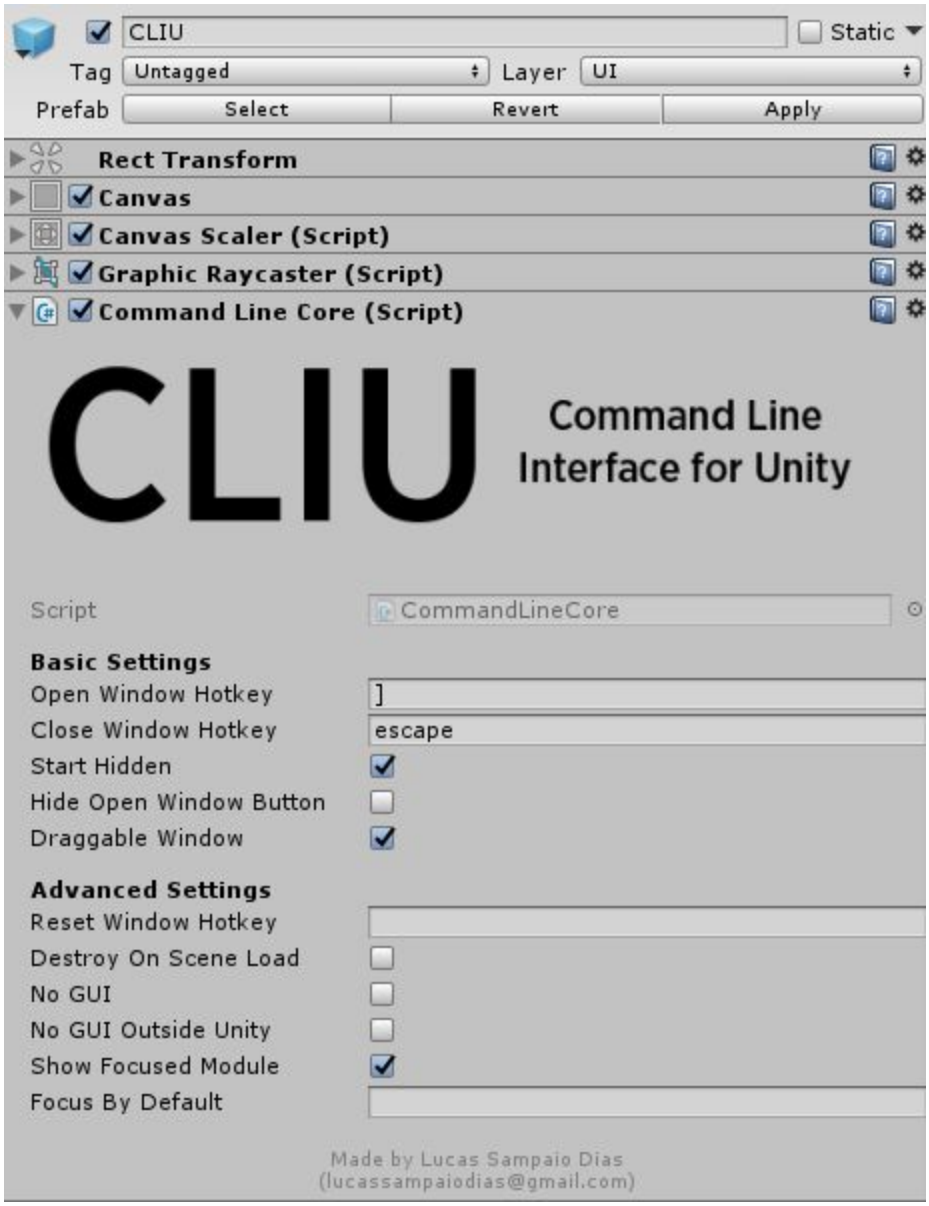
Getting Started

Integrating CLIU to your project is really simple! Go to the ‘CLIU’ submenu and click on ‘Instantiate or Destroy CLIU’, or just press Alt + 0.

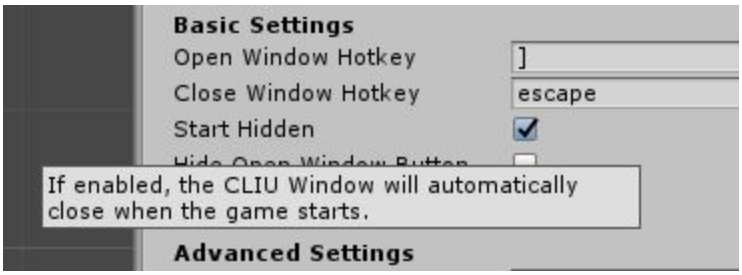


I suggest you to add the CLIU prefab on the first scene that is loaded when your game launches. By doing this, CLIU will always be available by default when you play the game. However, if you want to add CLIU to specific scenes, do the process above for each scene and change the ‘Destroy on Scene Load’ setting of the prefab to true (you can find the prefab at CommandLine/Resources).

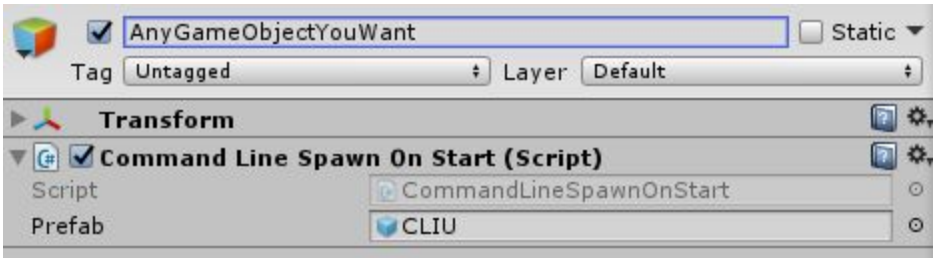
And that’s it! After placing the CLIU prefab into your scene everything should work already. Press Play to test your game and the CLIU window should appear if you press the ‘Open CLIU’ button or press ‘]’ on your keyboard.



Be sure to take a look on all the available settings to make the most of CLIU. There is a tooltip for all settings if you need any help.



Note: you can also use the CommandLineSpawnOnStart script to Instantiate the CLIU prefab when a scene starts. Just attach the script to a game object present on the scene you want CLIU to initialize.



Commands and Modules

Using CLIU should feel quite similar to using pretty much any other CLI. All you have to do is open the window, type your commands (separated by spaces) and press Enter/Return after typing the entire command. To see some basic commands hover over the '?' next to the close button.

Commands are separated into two types: 'Core' commands and 'Module' commands.

Core commands are read first and have priority in execution. Module commands are "custom" commands built on top of CLIU. These are what makes a command line interface useful.

There are many modules that come with CLIU by default (like the 'transform' module, that comes with commands that manipulate the transform attributes of Game Objects), but you can also make your own modules to satisfy the needs of your projects. More on that on the next section of this document.

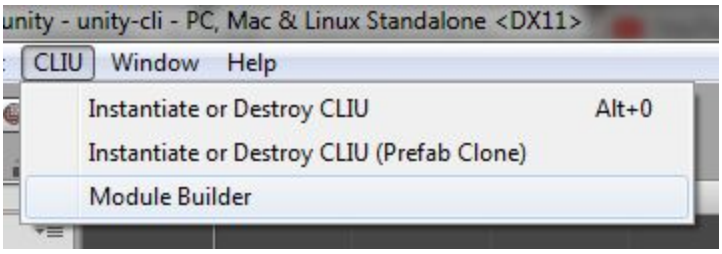
Here is *some* of the awesome commands found on the default modules:

- time timescale 1: sets the timescale to the value specified
- scene loadscene 5: load the scene with build index equals to 5
- obj call Player Die: call the method Die() on the Player game object

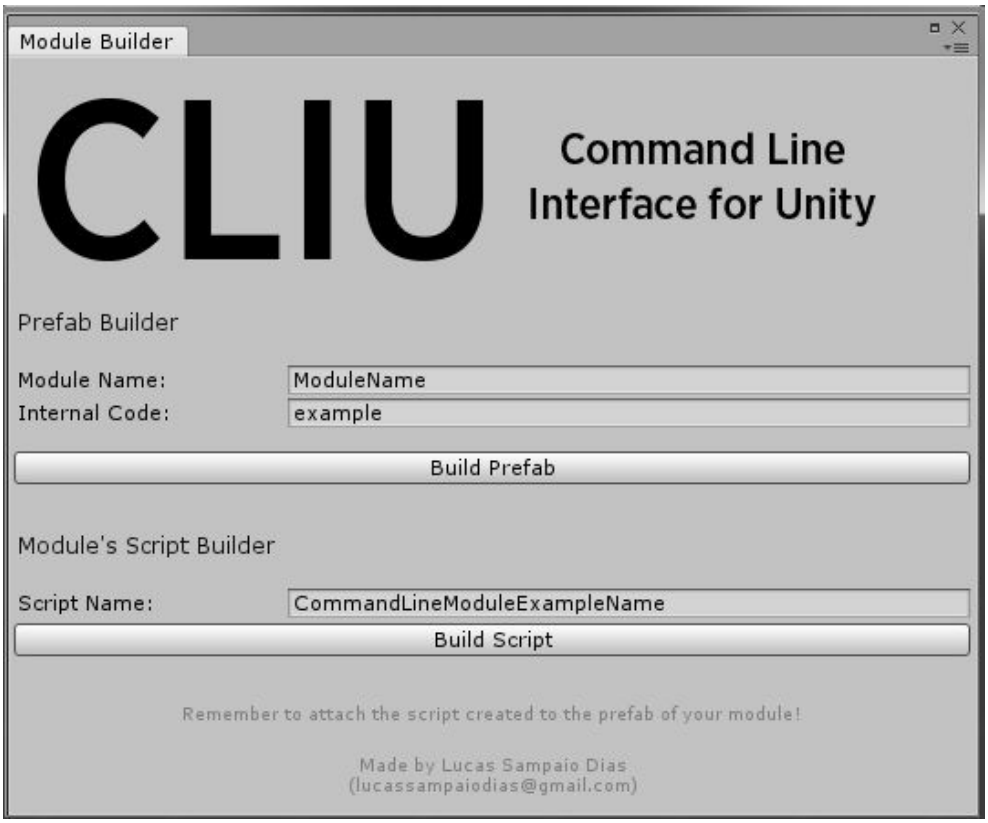
There are two **very** important things you should know about commands. The first thing is that the first word on the commands above are the module codes of the modules where these commands are implemented. The second thing is that **you don’t have to use the module code** every time you want to use a module command! For the first command, just use “timescale 1” instead and it will work just fine! The only “side effect” is that all commands found with this “name” will be executed (even on multiple modules). It is worth noting that all commands available on the default modules are unique, so this problem of two or more commands being executed will never happen for them.

Creating Custom Modules

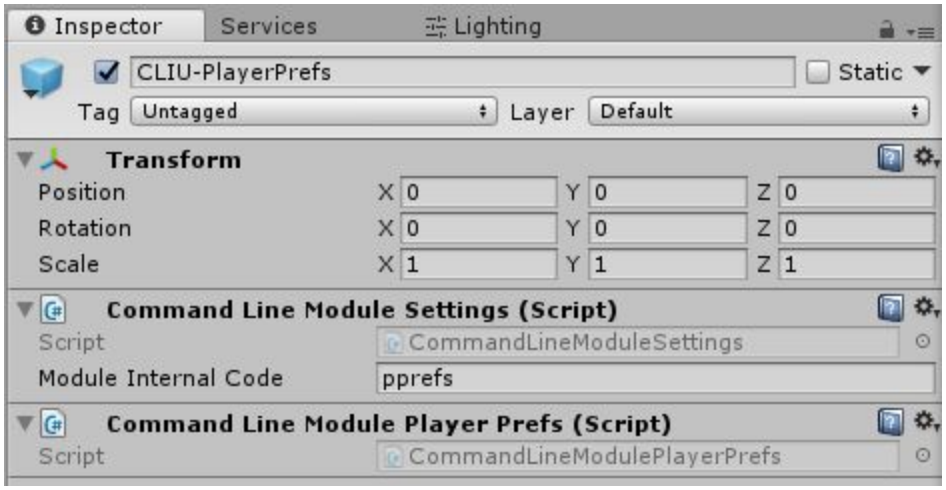
Building new modules is made very easy using the Module Builder that comes with CLIU. To open it, go to the CLIU submenu and click on ‘Module Builder’.



This should open a window just like the one on the image below:



First, insert the name of the module (which will be the name of the prefab) and the module's code (like pprefs is the internal code for the PlayerPrefs module, for example) and press 'Build Prefab'. Then, insert the script name of your module (I suggest CommandLineModulePlayer for a module called Player, for example) and press 'Build Script'. Finally, find the module prefab (CommandLine/Resources/Modules) and attach to it the script you created. The prefab should look like this:



When you press Play again the module will be already working with a few example commands. To add the commands you want open the script the Module Builder created (the one attached to the module prefab) and change the code of the Execute() method. If you need any guidance, follow the patterns used on the scripts of the default modules.

If you find any errors while using the Module Builder see if you can find a solution on the FAQ section at the end of this document. If you can't, send me an email (lucassampaiodias@gmail.com) and I will look into it as soon possible.

List of All Module Commands

Object (code: obj)

- call string:methodName string:gameObjectName
    - Finds the specified game object and call the method on its attached scripts.
    - Example: call Kill Player
- callbytag string:methodName string:gameObjectTag
    - Finds all game objects with this tag and call the method on its attached scripts
    - Example: call Die Enemy
- callall string:methodName
    - Calls the method on every single game object
    - Example: callall Reset
- destroy string:gameObjectName
    - Destroys the specified game object
    - Example: destroy Mighty Orc
- destroytag string:gameObjectsTag
    - Destroys all game objects with this tag
    - Example: destroytag Enemy
- instantiate string:pathInAResourcesFolder
    - Instantiates a prefab located inside a resources folder or one of its subfolders (like Assets/Resources/Prefabs)
    - Example: instantiate Prefabs/MyPrefab

PlayerPrefs (cods: pprefs)

- getint string:key
    - Gets the int value of the specified key on you PlayerPrefs
    - Example: getint CurrentStage
- getfloat string:key
    - Gets the float value of the specified key on you PlayerPrefs

    - Example: getfloat Score
- getstring string:key
    - Gets the string value of the specified key on you PlayerPrefs
    - Example: getstring PlayerName
- setint string:key int:value
    - Sets an int value to the specified key on you PlayerPrefs (for keys with spaces use ‘|’ instead, without the ‘’)
    - Example: setint CurrentStage 8
- setfloat string:key float:value
    - Sets a float value to the specified key on you PlayerPrefs (for keys with spaces use ‘|’ instead, without the ‘’)
    - Example: setfloat ScoreValue 56789.0
- setstring string:key string:value
    - Sets a string value to the specified key on you PlayerPrefs (for keys with spaces use ‘|’ instead, without the ‘’)
    - Example: setstring PlayerName John Doe
- saveprefs
    - Writes all modified preferences to disk (PlayerPrefs.Save)
- deleteallprefs
    - Removes all keys and values from the preferences. Use with caution (PlayerPrefs.DeleteAll)

Scene (code: scene)

- loadscene int:sceneindex
    - Loads the specified scene (based on build index)
    - Example: loadscene 1
- reloadscene
    - Reloads the current scene
- loadnextscene
    - Loads the next scene on the build index
- loadpreviousscene
    - Loads the previous scene on the build index
- sceneindex

    - Prints the build index of the current scene

Time (code: time)

- timescale float:amount
    - Sets the current timescale to the specified value
    - Example: timescale 0.5
- slowmo float:amount float:duration
    - Sets the current timescale to the specified value for a specified duration (in seconds)
    - Example: slowmo 0.5 2.5
    - Can also be called as: slomo, slowmotion
- devicetime
    - Prints the time and date used on the current device
- currenttimescale
    - Prints the current timescale value
    - Can also be called as: currentts

Transform (code: transform)

- position float:x float:y float:z string:gameObjectName
    - Sets the position of a specified game object to the specified position
    - Example: position Player 1 0.5 2
    - Can also be called as: setposition
- move float:x float:y float:z string:gameObjectName
    - Adds the specified values to the X, Y and Z of a game object position
    - Example: move Player 1 0 0
- rotation float:x float:y float:z string:gameObjectName
    - Sets the rotation of a specified game object to the specified rotation (transform.eulerAngles)
    - Example: rotation Cube 10 30 0
    - Can also be called as: setrotation
- rotate float:x float:y float:z string:gameObjectName

    - Calls transform.Rotate() with the specified values to the game object
    - Example: rotate Cube 20 20 20
- scale float:x float:y float:z string:gameObjectName
    - Sets the scale of a specified game object to the specified values
    - Example: scale Sphere 2 2 2
    - Can also be called as: setscale
- addscale float:x float:y float:z string:gameObjectName
    - Adds the specified values to the X, Y and Z of a game object scale
    - Example: addscale Sphere 1.5 3 1.5
- getposition string:gameObjectName
    - Prints the position of a specified game object
- getrotation string:gameObjectName
    - Prints the rotation of a specified game object
- getscale string:gameObjectName
    - Prints the scale of a specified game object

FAQ

Q: I built a custom module but it doesn't recognize any commands. What should I do?
A: First, check if the prefab of the module you created is at CommandLine/Resources/Modules. If it is, see if it has the CommandLineModuleSettings script with the correct code on the inspector. Also, check if the script you created (for example, CommandLineCustomModule) is attached to the prefab. If this doesn't work, be sure to send me an email to tell me what happened.

Q: The ModuleBuilder created the prefab but couldn't create the script. What should I do?
A: There is a script called CommandLineModuleExample at CommandLine/Scripts that contains everything the Module Builder was going to create for you. Copy its contents, create a new script for your module and paste the contents of CommandLineModuleExample there. Don't forget to add the script you just created to the module prefab.

Q: I already have a chat system for my game. Can I use it to run the commands of this plugin?
A: Yes! Enable the 'No GUI' setting and send your commands using the RunCommand() method on the CommandLineCore script (you still need to have the prefab on your scene and call this method from it)

Q: How can I configure CLIU to only take methods from a specific module I created?
A: Use the 'focus' command or the 'Focus by Default' setting if you want this behavior when CLIU initializes.

Q: Is there a way to print messages on the CLIU window without using a module?
A: Yes, and you don't even need a reference to the CLIU object! Use any of the following static methods:

- CommandLineCore.Print
- CommandLineCore.PrintError (red text)
- CommandLineCore.PrintWarning (yellow text)
- CommandLineCore.PrintSucess (green text)

Q: Does CLIU work for mobile?
A: Yes, although you might have to adapt the window interface because of the small screen size and/or screen orientation.

Q: Why does CLIU have an EventSystem?
A: In case your scene have no EventSystem this game object will be activated so the NGUI components of CLIU work normally. If for some reason you don't want it to active, disable the components on the EventSystem game object inside the CLIU prefab.