

Developer Console

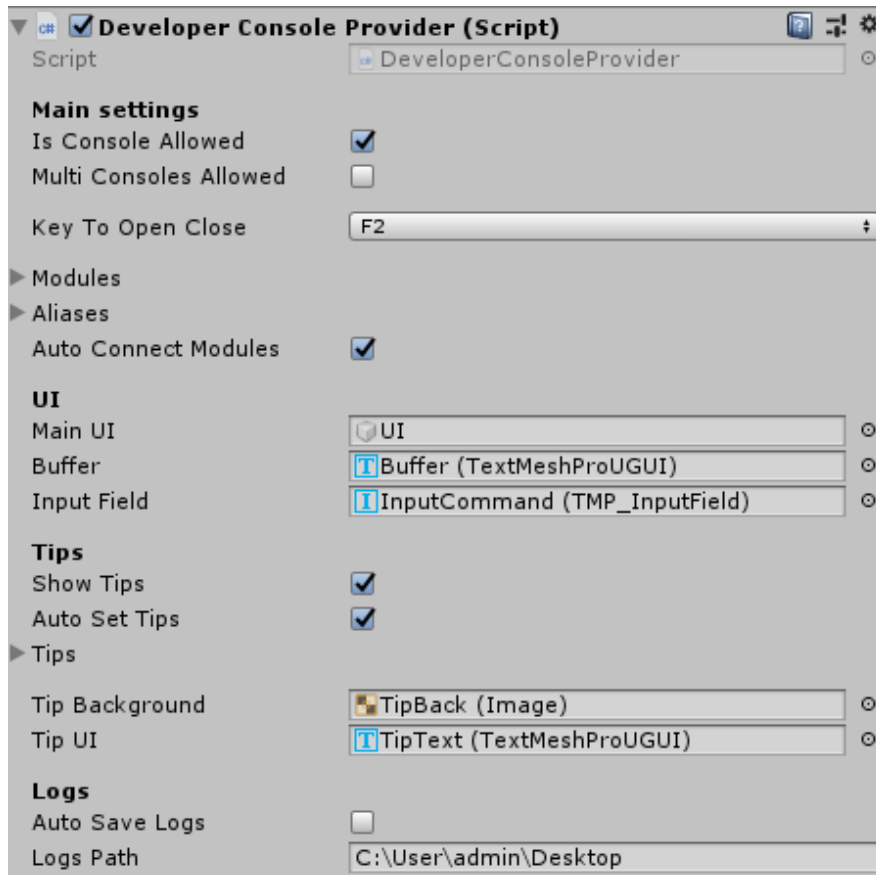
for
Unity games

Add this developer console to your game which you are creating in Unity, to improve game development and test some stuff in your project.

Features:

- Clean code
- Easy to implement (just drag a prefab to scene)
- The ability to quickly add your own commands
- Commands suggestion
- Possibility to save logs from console to file
- Aliases
- By down and up arrow on keyboard, you can review recently executed commands

Main script:



Is Console Allowed - if equals false, it won't be possible to turn on console

Multi Consoles Allowed - if equals true, there can be more than one console on the scene in the same time

Key To Open Close - key which is using to open/close console

Modules - current modules connected to console

Aliases - all aliases available in console

Auto Connect Modules - if equals true, script will automatically find and add all console modules

Main UI - main console UI

Buffer - buffer text field

Input Field - input field where you can type command to execute

Show Tips - if equals true, console will display tips about commands

Auto Set Tips - if equals true, script will automatically add all tips appealing to current connected console modules

Tips - list of all tips in console

Tip Background - tips background image

Tip UI - tips text field

Auto Save Logs - if equals true, console will automatically save logs to file, after every executed command

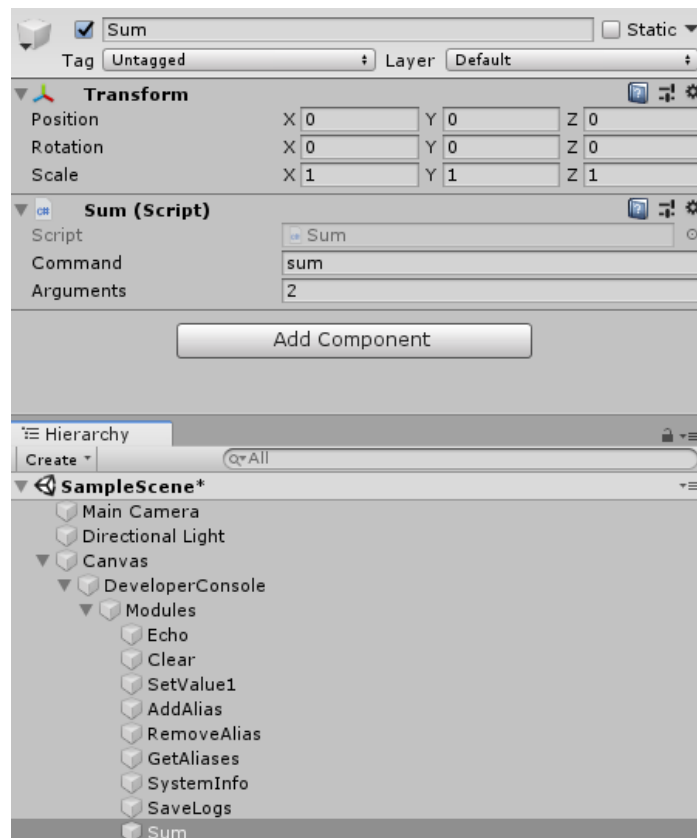
Logs Path - directory where logs will be saved, if null, logs will be saved in Application.dataPath

Create your own command (console module):

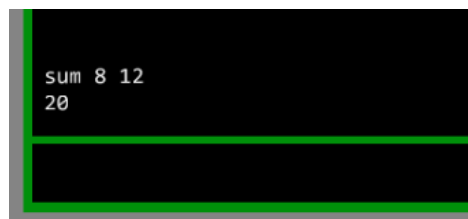
1. Create new class, which inherits from Developer.ConsoleModule class.
2. In private function Awake, set command and count of required arguments, if arguments == -1, there can be an infinite count of arguments.
3. Overwrite public function ExecuteCommand and in body of this function, declare what this command will be doing.
4. There is an example of command that prints sum of two ints:

```
1 namespace Developer
2 {
3     public class Sum : ConsoleModule
4     {
5         private void Awake()
6         {
7             command = "sum";
8             arguments = 2;
9         }
10
11         public override bool ExecuteCommand(DeveloperConsoleProvider host, ref string result, string[] arguments)
12         {
13             int a, b;
14
15             try
16             {
17                 a = int.Parse(arguments[0]);
18                 b = int.Parse(arguments[1]);
19             }
20             catch
21             {
22                 result = CommandResults.WRONG_ARGS_MESS;
23                 return false;
24             }
25
26             host.Print((a + b).ToString());
27             return true;
28         }
29     }
30 }
31
```

5. Host - console which will host this command
result - result of command which will be shown on console
arguments - all arguments passed to command
6. If everything in code is right, then add new empty game object to your scene and add this newly created module to that empty object. That how it should look:



7. If you unchecked the *Auto Connect Modules* in main script, you have to manually add new module to *Modules* list in main script.
8. If you did everything correctly, now you can use your new command:



DeveloperConsoleProvider.cs

```
public void Print(string message) - prints given message in console

public void PrintError(string message) - prints given message in console as error
(red color)

public void PrintWarning(string message) - prints given message in console as
warning (yellow color)

public void ExecuteCommand(string c) - executes given command

public void SaveLogs() - saves logs to file

public bool HasAlias(string aliasToCheck) - returns true if alias with given name
exists in aliases list.

public bool HasCommand(string commandToCheck) - returns true if given command
exists.

public void AddAlias(Alias aliasToAdd) - adds given alias to aliases list.

public void RemoveAlias(string aliasToRemove) - removes alias with given name.

public string GetAliasesList() - return formatted all aliases list.

public void AddModule(ConsoleModule moduleToAdd) - adds given ConsoleModule to
console.

public void ClearBuffer() - clears console buffer.
```