



User Guide for Reactor Developer Console

M. Mustafa Yaya

1.2 RELEASE

For other releases please check [GitHub](#).

Contents

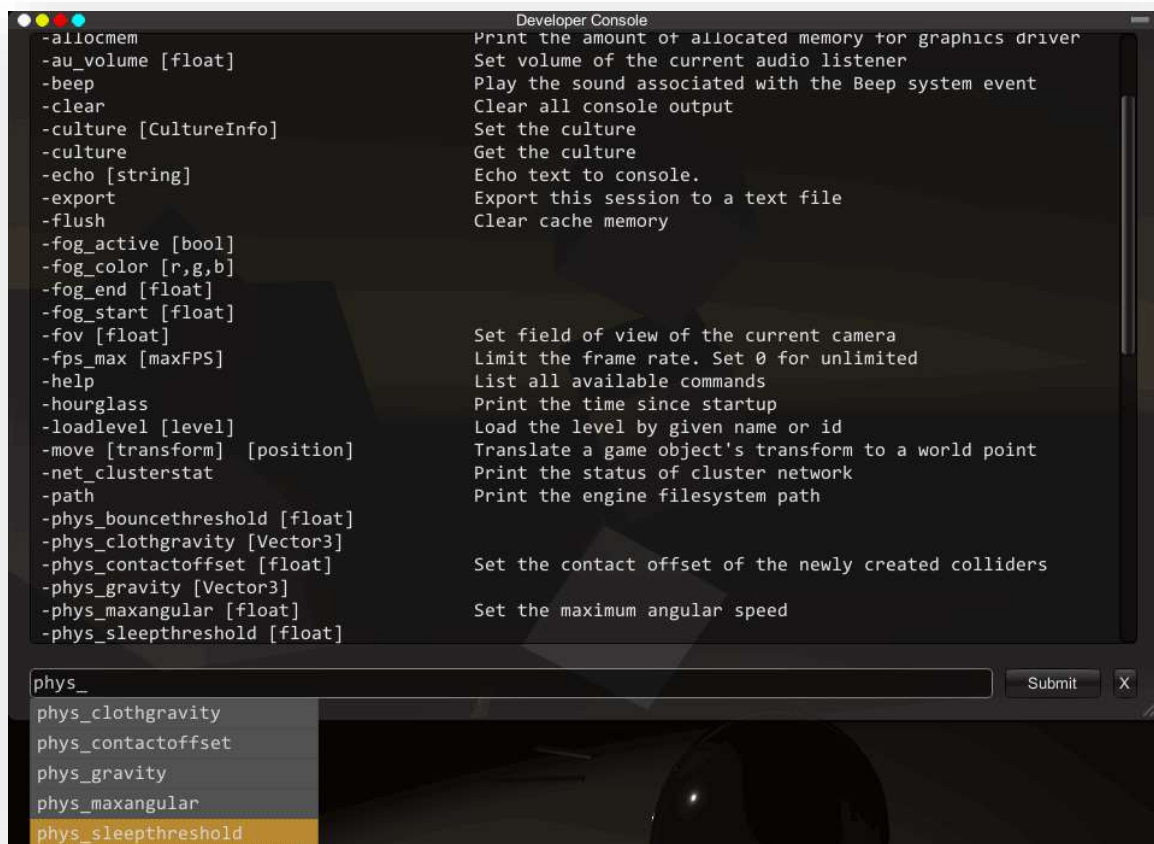
General Information of Software	2
Introduction.....	2
Package.....	3
Installation	4
Anatomy	5
Mobile	7
Syntax	8
Commands	9
Command Parameters.....	9
Invoke Definitions.....	10
Adding new commands	10
Support	12
Licenses	12
Reactor Developer Console	12
Consolas.....	12

User Guide for Reactor Developer Console

General Information of Software

Introduction











































Reactor is a developer console for Unity 3D. Prints logs of the client and built-in Unity console, operates over the input given by the client and executes the related command with/without provided parameters. Filtering the output, input predictions and hints are operable.



User Guide for Reactor Developer Console

Package

Reactor asset package has 36 files.

-  DeveloperConsole
-  DeveloperConsole.prefab
-  LICENSE
 -  Core
 -  Command.cs
 -  Commands.cs
 -  ConsoleUtility.cs
 -  DeveloperConsole.cs
 -  Widgets.cs
 -  Fonts
 -  [CONSOLA.TTF](#)
 -  Misc
 -  background-active.png
 -  background-hover.png
 -  background.png
 -  button active.png
 -  button hover.png
 -  button.png
 -  Reactor-Default.guiskin
 -  corner.png
 -  error-hover.png
 -  error.png
 -  log-hover.png
 -  log.png
 -  minus-active.png
 -  minus-hover.png
 -  minusbutton.png
 -  network-hover.png
 -  network.png
 -  toggle-hover.png
 -  toggle-normal.png
 -  warning-hover.png
 -  warning.png
 -  SampleScene
 -  ground.mat
 -  transform.mat
 -  transform.physicMaterial
 -  SampleScene.unity
 -  Widgets
 -  ConsoleActivator.cs
 -  StatsWidget.cs
 -  WireframeWidget.cs

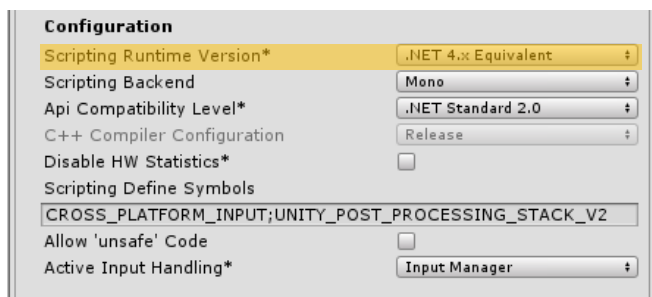
User Guide for Reactor Developer Console

Installation

Before installation please check that you have the [all files](#) in your project.

This software supports Unity 2018.1 or later releases. If you are using an older version, problems may occur.

Make sure your project's scripting runtime version is **.NET 4.x Equivalent**.



- 1) **Open a Unity 3D 2018.1 or later project. (If you have got this asset from the Asset Store, skip to the third step)**
- 2) **Open a scene.**
- 3) **Open `DeveloperConsole` folder in Project view.**
- 4) **Find `DeveloperConsole.prefab` and drag it to your Hierarchy view.**

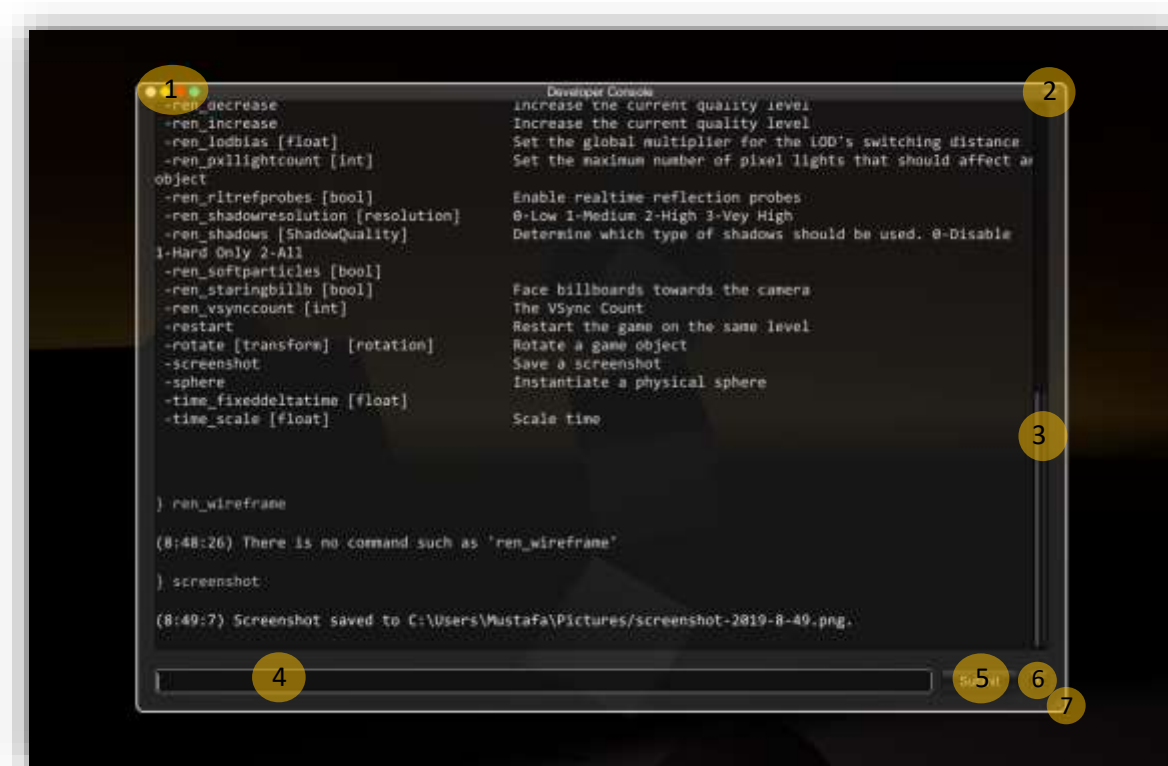
After completing this steps console will be ready. Run the scene, the console will open automatically. If you want to use the default F4 key for activating/disable the console, add `ConsoleActivator.cs` to the console game object. Otherwise, you can use `DeveloperConsole.active` for controlling the console.

If you are using Unity 2018 or an older release, some commands can be disabled.

	Unity 2019	Older
<code>phys_maxangular</code>	Available	Disabled
<code>phys_clothgravity</code>	Available	Disabled

User Guide for Reactor Developer Console

Anatomy



1) Filters

User can filter the output by clicking filter buttons located at the left upper corner. If the button has no color, that type of output is disabled.

Button	Output Type
-	User
-	System
○	Log
●	Warning
●	Error
●	Network

User Guide for Reactor Developer Console

2) Close Button

Opens and closes the console window. On mobile, it doesn't disappear when the console window is closed. This behavior can be handled with `drawCloseButtonOnMobile`.

3) Output Box

The output will be printed in the output box. If the output is too long, the output can be scrolled.

4) Input Field

User can type commands here. Predictions and hints will be drawn in this field.

5) Submit Button

Submits the entry to the console.

6) Clear Button

Clears the output.

7) Resize Drag

Resizes the console window. This is disabled at mobile builds.

User Guide for Reactor Developer Console

Mobile

Fullscreen mode is enabled on mobile builds.



User Guide for Reactor Developer Console

Syntax

Reactor combines characters between brackets. You must use brackets for combining words.

```
move (Main Camera) (0,5,3)
```

Consider there are 2 invoke definitions `culture` & `culture [CultureInfo]`. The first one prints the current culture and the second one changes it.

```
} culture  
Culture is en-US  
}  
} culture tr-TR  
Culture is now tr-TR
```


User Guide for Reactor Developer Console

Commands

Operates the logic and returns the output. A command can be invoked by submitting its query identity to the console. There can be 2 commands that have the same query identity. They called invoke definitions. Commands can have parameters. A command must have `[ConsoleCommand]` attribute. Usage of `[ConsoleCommand]` attribute is `[ConsoleCommand(string queryIdentity, string description)]`. For instance;

```
[ConsoleCommand("help", "List all available commands")]
class Help: Command
{
}
```

If a command is a cheat or only for developers, you can set that command as development build only by setting the third parameter `onlyAllowedOnDeveloperVersion` true.

```
[ConsoleCommand("invincibility", "Make player invincible",true)]
class Invincibility: Command
{
    //cheat here
}
```

Command Parameters

Command parameters are variables of command classes. A command parameter must have `[CommandParameter]` attribute and be public.

```
//Define command query identity, description and optionally for only developer
version mode
[ConsoleCommand("culture", "Set the culture", true)]
class CultureSet : Command//Inherits class from Console.Command
{
    [CommandParameter("CultureInfo")] //Add parameter for command
    public System.Globalization.CultureInfo value;

    public override ConsoleOutput Logic() //Virtual logic method for every
command,
    {
        base.Logic();
        var cultureInfo = value;//Command logic

        //Return console output with a message, output type and optionally
time signature
        return new ConsoleOutput("Culture is now "+ cultureInfo,
ConsoleOutput.OutputType.Log, false);
    }
}
```

A command parameter's type must have a global definition. If there are multiple definitions, the optimal definition will be used to define.

User Guide for Reactor Developer Console

Invoke Definitions

Two commands that have the same query identity must be separated. We can use the same query identity and create different usages.

```
help -List all available commands
```

```
help [command] - "Provide help information for a command."
```

Adding new commands

To add a new command you have to create a new class that inherits from `Console.Command` and attribute it with the `ConsoleCommand` attribute. Do not forget to make the parameter field public. Example proper command:

```
[ConsoleCommand("culture", "Get the culture")]
class CultureGet : Command
{
    //No parameters
    public override ConsoleOutput Logic()
    {
        base.Logic();
        var oldCulture =
System.Globalization.CultureInfo.CurrentCulture.Name;

        return new ConsoleOutput("Culture is " + oldCulture,
ConsoleOutput.OutputType.Log, false);
    }
}
```

1) Create a new C# script.

```
using System;
using UnityEngine;

public class MyCustomCommands
{
}
```

2) Add `using Console;` to top of your script.

```
using System;
using UnityEngine;
using Console;

public class MyCustomCommands
{
}
```

User Guide for Reactor Developer Console

3) Create a new class that inherits from `Command` class.

```
public class MyCustomCommands
{
    class MyMultiplyCommand: Command
    {
    }
}
```

4) Insert `ConsoleCommand` attribute to your command.

```
public class MyCustomCommands
{
    [ConsoleCommand("multiply", "Multiply two numbers.")]
    class MyMultiplyCommand: Command
    {
    }
}
```

5) Override the `ConsoleOutput Logic()` method and program your command's behavior.

```
public class MyCustomCommands
{
    [ConsoleCommand("multiply", "Multiply two numbers.")]
    class MyMultiplyCommand: Command
    {
        public override ConsoleOutput Logic()
        {
            var result = 5 * 3;
            return new ConsoleOutput("Result is " + result,
                ConsoleOutput.OutputType.Log);
        }
    }
}
```

6) To add parameters, use `CommandParameter` attribute.

```
public class MyCustomCommands
{
    [ConsoleCommand("multiply", "Multiply two numbers.")]
    class MyMultiplyCommand : Command
    {
        [CommandParameter("firstParameterDescription")]
        public int i;
        [CommandParameter("secondParameterDescription")]
        public int j;
        public override ConsoleOutput Logic()
        {
            var result = i * j;
            return new ConsoleOutput("Result is " + result,
                ConsoleOutput.OutputType.Log);
        }
    }
}
```

Your command is ready to use. You can type `multiply 5 10` to console. It will print 50.

User Guide for Reactor Developer Console

Support

If you are having an issue, please check the [forum](#) and please report to [issue tracking](#).

You can contact me at mustafa.yaya@outlook.com.tr

Licenses

Reactor Developer Console

MIT License

Copyright (c) **2019 Mustafa Yaya**

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Consolas

Consolas is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Microsoft Corporation

Luc(as) de Groot

<http://www.microsoft.com/typography/ctfonts>

<http://fontfabrik.com>

You may use this font as permitted by the EULA for the product in which this font is included to display and print content. You may only (i) embed this font in content as permitted by the embedding restrictions included in this font; and (ii) temporarily download this font to a printer or other output device to help print content.

<http://www.microsoft.com/typography/fonts/default.aspx>