uConsole Command Documentation

Unity In-Game Rich Text Console Interface.

This document helps you learn how to use the uConsole in Unity.

CommandManager

This class manages all the commands and their operation inside and outside the console.

CommandRegistration

Register the commands that will be used in the console.

PROPERTIES

Property	Function
command	Command name to invoke.
handler	Command function to invoke.
help	Help information text to show in Help command.

AppendLog

Add a String to in-game console log.

PROPERTIES

Property	Function
line	String to print in the log.

RunCommandString

Register and Run command.

PROPERTIES

Property	Function
commandString	Command string arguments to invoke.

RunCommand

It processes the arguments to be executed, if the syntax is erroneous it prints an error message in the log.

PROPERTIES

Property	Function
command	Command name.
args	Commands arguments.

ParseArguments

Parses the sentence of the command console entry.

PROPERTIES

Property	Function
commandString	Commans String with arguments.

ConsoleCommand

Handles events and data between CommandManager and UI.

PROPERTIES

Property	Function
Instance	Console Instance, don't destroy on load scene.
commandRegister	Command register of Command Manager.
openKey	Keyboard key to open / close the console.
pauseOnOpen	Pause game if the console is open.
isOpen	True if console is open.
consoleLog	Log text component
inputField	Consoles input field
scrollView	Log Scrollview
showFPS	Show current fps in fpsText
fpsText	Frame per seconds text
onShow	this event is invoked when the console change to visible.
onHide	this event is invoked when the console change to invisble.

Command Scripts Reference

To create new commands must be created within the CommandManager class.

Basic Function Format Example

A basic format code command function.

```
void Example(String[] args)
        {
            //Do Stuff
        }
```

Print Log

Basic example of print log

```
void PrintLog(String[] args)
        {
            Debug.Log("Hello World");
        }
```

Create Cube Example

A test command to demonstrate how to instantiate an object.

```
void createcube(string[] args)
    {
        GameObject cube = GameObject.CreatePrimitive(PrimitiveType.Cube);
        cube.transform.position = new Vector3(0, 6, 0);
        cube.AddComponent<Rigidbody>();
    }
```

Echo

A test command to demonstrate argument checking/parsing.

Get argument variable

```
void echo(string[] args)
        {
            StringBuilder sb = new StringBuilder();
            if (args.Length != 0)
            {
                foreach (string arg in args)
                {
                    sb.AppendFormat("{0},", arg);
                }
                sb.Remove(sb.Length - 1, 1);
                Debug.Log(sb.ToString());
            }
            else
            {
                Debug.Log("Expected 1 arguments.");
            }
        }
```

SetQuality Echo Implementation

A test command to demonstrate argument checking/parsing.

Get argument int a set Graphic quality.

```
void setquality(string[] args)
    {
        StringBuilder sb = new StringBuilder();
        if (args.Length == 1)
        {
            foreach (string arg in args)
            {
                sb.AppendFormat("{0},", arg);
            }
            sb.Remove(sb.Length - 1, 1);
            int level;
            Int32.TryParse(args[0], out level);
            if (level >= 0 && level <= QualitySettings.names.Length - 1)
            {
                QualitySettings.SetQualityLevel(level);
                Debug.Log("Quality Level: " + QualitySettings.names[level]);
            }
            else
            {
                Debug.Log("Value out of range. The value must be integer between 0 and " +
(QualitySettings.names.Length - 1) + ".");
            }
        }
        else
        {
            Debug.Log("Expected 1 arguments.");
        }
    }
```

Babble

A test command to demonstrate argument checking/parsing.

Will repeat the given word a specified number of times.

```
void babble(string[] args)
        {
            if (args.Length < 2)
            {
                Debug.Log("Expected 2 arguments.");
                return;
            }
            string text = args[0];
            if (string.IsNullOrEmpty(text))
            {
                Debug.Log("Expected arg1 to be text.");
            }
            else
            {
                int repeat = 0;
                if (!Int32.TryParse(args[1], out repeat))
                {
                    Debug.Log("Expected an integer for arg2.");
                }
                else
                {
                    for (int i = 0; i < repeat; ++i)
                    {
                        Debug.Log(string.Format("{0} {1}", text, i));
                    }
                }
            }
        }
```

Default Command

the commands in this list are in CommandManager class and can be edited.

Property	Function
About	Show Unity project information.
Babble	Example command that demonstrates how to parse arguments. babble [word] [# of times to repeat]
Clear	Clear console log.
Echo	echoes arguments back as array (for testing argument parser)
Help	Print this help.
Reload	Reload current scene.
Resetprefs	Reset & save PlayerPrefs.
Showfps	Toggle FPS display.
Systeminfo	Show device and system information.
quit	Close this game.

F.A.Q.

How to print a message on the console from a script in my game?

To create a message on the console there must be an instance of the console interface, be sure to run the game with an instance. Use Debug Log Method for print a message in both console.

```
Debug.Log("Hello world!");
```

How to run a command from a script in my game?

To register a command there must be an instance of the console interface, be sure to run the game with an instance.

```
ConsoleCommand.Instance.RunCommand("args");
```

How to register a command from a script in my game?

To register a command there must be an instance of the console interface, be sure to run the game with an instance.

```
m_Console.commandRegister.RegisterCommand("hello", Hello, "Hello world!.");
```

How to creating a Command with an Argument?

Use the Echo example as a guide to create your command

Code Example

```
void Echo(string[] args)
        {
            StringBuilder sb = new StringBuilder();
            if (args.Length != 0)
            {
                foreach (string arg in args)
                {
                    sb.AppendFormat("{0},", arg);
                }
                sb.Remove(sb.Length - 1, 1);
                // My code here...
            }
            else
            {
                Debug.Log("Expected 1 arguments.");
            }
        }
```

Another example is the create command, where the word in the argument determines the action.

```
void Create(string[] args)
    {
        StringBuilder sb = new StringBuilder();
        if (args.Length != 0)
        {
            foreach (string arg in args)
            {
                sb.AppendFormat("{0},", arg);
            }
            sb.Remove(sb.Length - 1, 1);
            switch (sb.ToString())
            {
                case "capsule":
                    // Create capsule
                    break;
                case "cube":
                    // Create cube
                    break;
                case "cylinder":
                    // Create cylinder
                    break;
                case "sphere":
                    // Create sphere
                    break;
                default:
                    Debug.Log("primitive no valid. try capsule, cube, cylinder, sphere");
                    break;
            }
        }
        else
        {
            Debug.Log("Expected 1 arguments.");
        }
    }
```