Unity Console

Lee Fan Spring 2013

# Introduction

Included with this Word Document are:

Console.cs The main class for implementing a console.

# Console

The Console code allows for the user to create a developer’s console for Unity. Design of Console was made to be customizable – size, toggle button, and title can all be set for each new console. Support for multiple consoles at one time is also supported. All code resides in Console.cs for convenience. Rather than editing Console.cs, users should extend the Console class.

The Console window can be dragged, and the frame buffer is scrollable. This was done to replicate how a terminal emulator would work on the computer. The console window body is translucent, to emulate consoles in games i.e. Steam games.

In a multi-console environment, focus is given to any console that is just brought to visibility.

# Input and Output

Input and output is managed through a TextWriter using a StringBuilder. This allows for efficiency for the growing frame buffer. Writing to the buffer can be done by using the TextWriter member Out. The contents of the command line can be obtained at any time through the string member In.

# Overriding

There are two main methods that the user can choose to override – Run and Initialize.

## Initialize

The Initializemethod is very unrestrictive – it allows the user to run any code that will happen before the Console is initialized and created. This is most useful for changing settings on the console. There are five main properties that a user may want to set:

int Width, Height

string Header

KeyCode ActivationKey

bool showConsole

Width and Height set the width and height of the console in pixels, Header sets the title of the Console, ActivationKey sets the key that toggles visibility of the Console, and showConsole sets the visibility of the Console. By setting showConsole in Initialize you can choose whether or not the Console is shown on startup.

For instance, a sample Initialize would look like this:

override protected void Initialize(){

Width = 600;

Height = 400;

Header = "Console";

ActivationKey = UnityEngine.KeyCode.F2;

showConsole = false;

}

This also happens to be the default values set for new Consoles. **One thing to be careful of here**: Width and Height only initiate the console size, later changes to them will not be reflected. Instead, one should use consoleRect.width and consoleRect.height for adjusting width and height later on.

## Run

Every time the user enters a new line while contents exist in the input box, the Runmethod is called, passing in the contents of the input box as a string. The default Run method simply writes back what is passed into it. This is where the main functionality of the console is implemented. An obvious example would be to use Object.GetType().InvokeMember to call methods from the command line. If these methods returned string types, then a simple Run method could look something like:

override protected void Run (string command){

Out.WriteLine(this.GetType().InvokeMember(*args, …*));

}

You can also do some other cool things here:

override protected void Run (string command){

Out.WriteLine (">> "+command);

string[] cmd = command.Split (' ');

if(cmd[0].Equals ("width"))

consoleRect.width = Int32.Parse (cmd[1]);

if(cmd[0].Equals ("height"))

consoleRect.height = Int32.Parse (cmd[1]);

}

This would let the user change the height of the console window from the command line. There is currently no simple way to take input from the user while running a method – a workaround would involve saving the state/progress of a process, returning, then resuming on next entry from the user.