**DW Console**

DW Console is a powerful tool to help enhance development. While the integrated unity console is useful for getting quick debug information, there is no way to input commands into this console however and this console is only accessible inside of unity.

The DW Console runs in-game like past FPS games, you simply press the bound key (the default is '~') and the console expands from the top of the screen, giving you the ability to view logged messages, warnings, errors or even traces from Debug.Log (optional setting).

All of the console functionally it accessible with static method calls. The two methods that you are most likely to use are, Console.Log(msg, type), which simply prints to the console. The other method is Console.AddCommand(command) allows you to create a new command.

You dont need to put the console prefab in your scene, it will be created for you whenever a console command is called. It is however suggested that you do. The console object wont be destroyed on scene change, and any new instances of the console trying to be created will be destroyed making sure only one is in the scene.

**Printing to the console**

Printing a message to the console is very simple, the easiest way to do it, is to do Console.Log(msg) this will send a message of the default type 'Log'. If you wish to use a different ConsoleMessageType, just call, Console.Log(msg, type), there are 5 different message types, Log, Warning, Error, Trace and Special. These all use the different icons listed in the consoleIcons array. If the Debug.Log sends a message to the console, it will include a button to expand the trace for the log entry.

**Console Commands**

Commands in the console function by just linking to the SendMessage method. To create a command, create a new ConsoleCommand object.

**ConsoleCommand command = new ConsoleCommand("commandYouType", "The help text for the command", CommandParameterType, targetGameObject, "MethodCalledOnTarget");**

Then just add it to the console via the Console.AddCommand(command). When you type 'commandYouType' into the console, it will call the 'MethodCalledOnTarget' on the targetGameObject.

The CommandParameterType is what determines what kind of parameter is passed into the 'MethodCalledOnTarget'. The valid parameter types are None, Int, Float, String, GameObject, Vector3 and Vector2.

**Parameters**

**None**, will simply not use a parameter, if a parameter is passed, it will cause an error, and prevent the method from being called.

**Int Float**, will parse the text (or atleast try) into the correct type and pass it to the message method.

**String**, will not change the passed in parameter at all, just send it over to the method as its pure string.

**GameObject**, is actually a string parameter that uses the GameObject.Find method to select a gameobject and pass that into the message method.

**Vector3**, will convert your string into a vector3 (Formatted like so, '0,0,0').

**Vector2**, will convert your string into a vector2 (Formatted like so, '0,0').

If you want to pass more parameters then one, because of the limitation of sendMessage, you will need to pass it as a string type, then parse it correctly.