Unity Command Terminal

A simple and highly performant in-game drop down Console.

gif

Command Terminal is based on an implementation by Jonathan Blow done in the Jai programming language.

Usage

Copy the contents from CommandTerminal to your Assets folder. Attach a `Terminal` Component to a game object. The console window can be toggled with a hotkey (default is backtick), and another hotkey can be used to toggle the full size window (default is shift+backtick).

Enter `help` in the console to view all available commands, use the up and down arrow keys to traverse the command history, and the tab key to autocomplete commands.

Registering Commands

There are 3 options to register commands to be used in the Command Terminal.

1. Using the RegisterCommand attribute:

The command method must be static (public or non-public).

```
[RegisterCommand(Help = "Adds 2 numbers", MinArgCount = 2, MaxArgCount = 2)]
static void CommandAdd(CommandArg[] args) {
    int a = args[0].Int;
    int b = args[1].Int;

    if (Terminal.IssuedError) return; // Error will be handled by Terminal

    int result = a + b;
    Terminal.Log("{0} + {1} = {2}", a, b, result);
}
```

`MinArgCount` and `MaxArgCount` allows the Command Interpreter to issue an error if arguments have been passed incorrectly, this way you can index the `CommandArg` array, knowing the array will have the correct size.

In this case the command name (`add`) will be inferred from the method name, you can override this by setting `Name` in `RegisterCommand`.

```
[RegisterCommand(Name = "MyAdd", Help = "Adds 2 numbers", MinArgCount = 2, MaxArgCount = 2)
```

2. Using a FrontCommand method:

Here you still use the `RegisterCommand` attribute, but the arguments are handled in a separate method, prefixed with `FrontCommand`. This way, `MaxArgCount` and `MinArgCount` are automatically inferred.

This also allows you to keep the argument handling `FrontCommand` methods in another file, or even generate them procedurally during a pre-build.

```
[RegisterCommand(Help = "Adds 2 numbers")]
static void CommandAdd(int a, int b) {
    int result = a + b;
    Terminal.Log("{0} + {1} = {2}", a, b, result);
}

static void FrontCommandAdd(CommandArg[] args) {
    int a = args[0].Int;
    int b = args[1].Int;

    if (Terminal.IssuedError) return;

    CommandAdd(a, b);
}
```

3. Manually adding Commands:

`RegisterCommand` only works for static methods. If you want to use a non-static method, you may add the command manually.

```
Terminal.Shell.AddCommand("add", CommandAdd, 2, 2, "Adds 2 numbers");
```