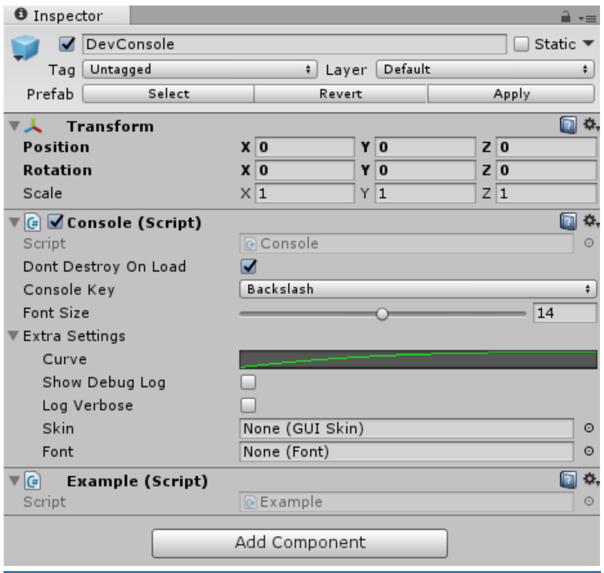
Latest Patch Notes

- The command system has been upgraded to the new feature-rich system used in DevConsole2
- The default "Command" has been replaced by either "ActionCommand" or "FuncCommand", depending on whether the method returns any value or not
- The RemoveCommand method has been marked as obsolte. The functionality will be restored in the near future
- Invidual command help has been disabled. This will be re-enabled in the near future

How to

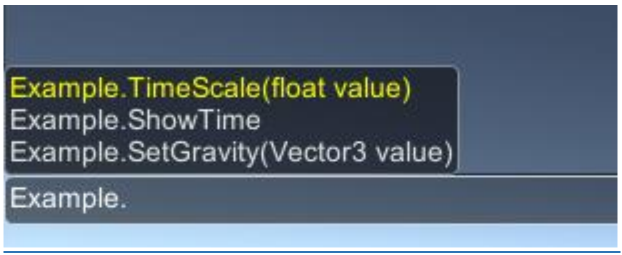
Using the Console



```
DevConsole                    Static
Tag Untagged          Layer Default
Prefab   Select   Revert   Apply

Transform
Position    X 0   Y 0   Z 0
Rotation    X 0   Y 0   Z 0
Scale       X 1   Y 1   Z 1

Console (Script)
Script                Console
Dont Destroy On Load  ✓
Console Key           Backslash
Font Size             14
Extra Settings
  Curve
  Show Debug Log
  Log Verbose
  Skin                None (GUI Skin)
  Font                None (Font)

Example (Script)
Script                Example
```

- To add the Console to your project, just add the DevConsole prefab to the scene. It comes with an Example script, remove it as needed.
- The “Console Key” is the key used to both open and close the console.
- The command “.clear” clears the current console log.
- The command “.help” shows a list of available commands.

Using the Autocomplete Window



```
Example.TimeScale(float value)
Example.ShowTime
Example.SetGravity(Vector3 value)
Example.
```

- When typing, the autocomplete window will automatically appear showing a list of suggested commands.
- To select the current highlighted command, either press Tab or Return/Enter.
- Navigate through using up and down keys.
- Close: ESC key.
- Show: F1 key.

History Navigation

- When there is no text in the input field, you can access the history by pressing UP or DOWN arrow keys.

Logging

- Calling the Console “Log” method logs a message to the console
- You can also specify a color for the text.
- Alternatively, calling “LogWarning”, “LogInfo” and “LogError” will log a light blue, yellow and red text, respectively.

Creating Commands

Commands can be created using two different procedures.

Manually adding the Command

The method AddCommand accepts a CommandBase as a parameter, which, in turn, needs a specific delegate to a method to be called when invoked.

There are two built-in usable versions of CommandBase: ActionCommand and FuncCommand. The first is used for methods with no return type or “void”, while the second is used for the rest of functions.

This creates a command with the same name as the method and in a group with the same name as the class the method is declared in. Alternatively, a group name and an alias can be issued to the command to override the default values.

Using the Command attribute

Adding the attribute [Command] to a static method will automatically make the DevConsole recognize that method as a command. Akin to the first method, optional parameters can be passed on to the Command attribute to specify both group and alias of the command.

The Example file uses this second method.

Calling Commands

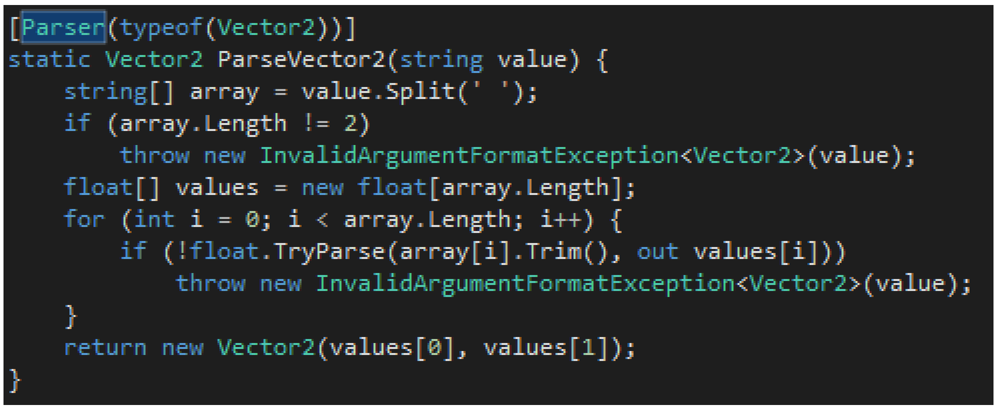
In order to call commands, the list of parameters should follow the command after a space. Arguments should be separated by spaces, and either “ or ‘ can be used to gather multiple elements into a single argument.

Thus, the command “Example.SetGravity (Vector3 value)” should be called this way:

Example.SetGravity “0 4.9 0”

Custom Argument Parsers

When calling a command, the arguments passed get parsed into the corresponding type for that argument. In order to do so, the [ParserAtrribute] is used.



```
[Parser(typeof(Vector2))]
static Vector2 ParseVector2(string value) {
    string[] array = value.Split(' ');
    if (array.Length != 2)
        throw new InvalidArgumentFormatException<Vector2>(value);
    float[] values = new float[array.Length];
    for (int i = 0; i < array.Length; i++) {
        if (!float.TryParse(array[i].Trim(), out values[i]))
            throw new InvalidArgumentFormatException<Vector2>(value);
    }
    return new Vector2(values[0], values[1]);
}
```

The static method marked with the Parser attribute must convert the given string in an object of the specified type.

Even though many types already have built-in parsers, custom types might not. In that case, a custom Parser method should be made.