Minimal Debug Console

Installation

Installing the Unity package is all you have to do, the package is set-up to automatically run on start-up.

Using Debug Console

You can open the debug console at any time by pressing the backquote (`) key, this can be changed either at the top of the “DebugConsole” script file or by changing it in the DebugConsole instance during runtime.

In the same way, you can also change the maximum number of lines in the console buffer.

Built-in Commands

log “message”

Logs message to the debug console in white

log “message” Color

Logs message to the debug console in given color

clear

Clears the debug console

fullscreen <FullScreenMode>

Changes the fullscreen mode to the given mode

quit

Exits the game

killTaggedObjects <tag>

Kills all objects of the given tag

killObject <name>

Kills the first object with the given name

Built-in Type Parsers

UnityEngine.Color

Takes 3 bytes representing the r, g and b color values in that order

UnityEngine.Vector2

Takes 2 floats that represent the x and y values of Vector2 respectively

UnityEngine.Vector3

Takes 3 floats that represent the x, y and z values of Vector3 respectively

UnityEngine.Vector4

Takes 2 floats that represent the x, y, z and w values of Vector4 respectively

UnityEngine.Rect

Takes 4 floats representing the x, y, width and height values of Rect respectively

UnityEngine.Bounds
Takes 2 Vector3s (6 floats) representing the center and the size values of Bounds respectively

UnityEngine.Quaternion
Takes 4 floats representing the x, y, z and w values of the Quaternion respectively

UnityEngine.Vector2Int
UnityEngine.Vector3Int
UnityEngine.RectInt
UnityEngine.BoundsInt
Same as the float counterparts except that it takes ints instead of floats

Adding Commands
Commands can be added by putting the [DebugConsoleCommand(“command”)] attribute above a static function.

Example:
```
[DebugConsoleCommand("log")]
public static void Log(string message, Color c)
{
    DebugConsole.Instance.AddMessage(message, c);
}
```

Adding Type Parsers
Type parsers can be added by implementing the IDebugTypeParser interface, the console will automatically discover the type parser during startup.

The interface needs to implement the following functions:

Type GetTarget()
Return the type of the object you want to support

void AppendArguments(List<Type>)
Add the individual primitive variables to the list, currently, type parsers can only use C# primitives and strings to build the object

object GetValue(Queue<object>)
Return the value, the queue contains the three types requested in AppendArguments in their given order

Example

```
public class ColorParser : IDebugTypeParser
{
    public Type GetTarget()
        => typeof(Color);

    public void AppendArguments(List<Type> arguments)
    {
        arguments.Add(typeof(byte));
        arguments.Add(typeof(byte));
        arguments.Add(typeof(byte));
    }

    public object GetValue(Queue<object> arguments)
    {
        return new Color((byte) arguments.Dequeue() / 255F, (byte) arguments.Dequeue() /
255F, (byte) arguments.Dequeue() / 255F);
    }
}
```