CLog
(Dynamically pick a logger for each platform)

Made by AiUnity
http://aiunity.com

Table of Contents

Overview

CLog is a flexible Unity logging framework based upon the highly popular open source project .NET Common Logging. CLog provides a powerful and generic wrapper around an included or custom logger, allowing loggers to be freely exchanged. Available loggers include Unity Console, In-Game Console, and the powerful NLog logger (Separate product). The In-Game Console allows you to debug from inside your game while the Unity Console will enhance your familiar debugging experience. From the GUI, loggers can be configured to inject information such as associated class, calling method, timestamp, log level, and exception message.

CLog can be configured to have a different logger for each platform. You could configure CLog to use UnityConsole in editor, the In-Game Console on XBox, and exclude logging for mobile devices. Every facet of CLog can be configured via the intuitive and comprehensive Unity Editor GUI. The configuration is stored an XML, which completely separates the GUI from CLog runtime.

CLog can improve performance by compiling out your logging statements on unselected platforms or logging levels. A log tester is built into to the GUI to help you verify and tweak your configurations. The available logging levels are Trace, Debug, Info, Warning, Error, Fatal, and Assert. CLog extensibility make it easy to create new targets.

Details

The CLog Unity Editor window presents you with global and target based configurations. Each target defines the log adapter used for the specified platform(s) and log level(s). The adapters job is simply to instantiate its logger and accept any customization properties. Valid loggers simply need to implement an interface containing trace, debug, info, warn, error, and assert. Based upon the XML configuration the CLog wrapper forwards log messages to the appropriate adapter. To start you can use the included UnityConsole Adapter which sends log messages to the Unity Console logger. The GUI allows you to choose which platforms should have logging and what levels to log globally. In addition there is a Test Logger built into the GUI so you can quickly see how your messages will look. Your configuration is stored in XML format that can be viewed and directly edited within the Unity Editor GUI. Please take a look at the CLog documentation or give CLog a try from the Asset Store.

At any time and in any environment the GUI or underlying XML can be configured to suite your needs. The Unity console retains the ability to double click log messages to take you to GameObject and source code. With a stunning feature set CLog will lower fustration, provide insight, and speed up development tremedously. CLog is highly extendiable with full source code provided.

With a stunning feature set CLog will lower fustration, provide insight, and speed up development tremedously.

Features

- Total control over logging during development through a comprehensive Unity Editor GUI.
- Log message levels are Trace, Debug, Info, Warning, Error, Fatal, and Assert.
- Log message API overloads allow format arguments to eliminating the need to use string.
- Control what messaging level is enable for each platform independently.
- Ability to assign a different logger for each platform.
- Logging statements on unselected platforms/levels are compiled out of existence.
- Adapters provided for loggers UnityConsole logger, In-Game logger, Null logger, and NLog logger.
- Ideal for Asset or Library projects because it is lightweight, dynamic configurable, and freely distributed.
- Configure CLog via the Unity Editor GUI which includes edit access to the underlying XML.
- A log tester is built into to the GUI so you can verify the configurations.
- Retain the ability to double click log messages to take you to GameObject and source code line.
- GUI themed for light and dark skin.
- Easily extend CLog by adding custom targets for your log messages.
- Highly extendible with full source code provided.
- Dynamically switch between using CLog source code or DLLs.
- Timeless asset in that logging will always be a fundamental debug tool.
- Works with AiUnity products ScriptBuilder, NLog, and MultipleTags.
- Dedicated website and forum available for support issues.
- Tested on .NET 3.5 and 4.6.
- Full source code provided.

Setup

1. Install CLog from the asset store.
2. Add logging statements to your code as described in Usage.
3. Configure CLog using the Unity Editor GUI.
4. Enjoy!

Usage

CLog provides a singleton instance of Log Manager locate at "LogManager.Instance". The LogManager is primarly used to get a logger instance via "LogManager.Instance.GetLogger". GetLogger can be passed the logger name, class instance, or both. The logger name uniquely identifies the logger. The class instance indentifies the associated GameObject to the logger. When class instance is used alone the logger name is assigned the class name. Examples are in the installation directory "Assets/AiUnity/CLog/Examples" or in the CLog Example section below.

Example

```
private void BuildFields()
using UnityEngine;
using AiUnity.CLog.Core;
public class CLogTest : MonoBehaviour
{
    private CLogger logger;
    void Awake() {
        // Passing in the instance will result in logger name being CLogTest and
        // double clicks linked to the GameObject associated with this script.
        logger = LogManager.Instance.GetLogger(this);
    }
    void Start()
 {
        Debug.Log("Test standard Unity debug message", this.gameObject);
        logger.Trace("Test CLog TRACE message");
        logger.Info("Test CLog INFO message");
        logger.Debug("Test CLog Debug message");
        logger.Error("Test CLog ERROR message");
        logger.Assert(false, "Test CLog ASSERT message");
        logger.Trace("Test CLog DEBUG message (Explict Context)", this);
    }
}
```

GUI

The CLog GUI is a Unity Editor window that can be accessed via Tools->AiUnity->CLog->ControlPanel. CLog configuration is stored in an XML file that is maintained by the GUI. The GUI contains tooltips, selection boxes, and references making learning intuitive.

Config file

Shows fixed location "Assets/Rources/CLog.xml" of CLog XML configuration file. The XML configuration can be modified via individual GUI controls below (recommended) or directly via the XML Viewer section. The CLog framework is solely configured by reading the resulting XML file, leaving it independent to the Unity Editor. The +/- icon next to Config File allow for the creation/deletion of CLog.xml respectively. While CLog.xml is absent all log statements are dropped, which may be ideal for asset store deployment.

Source

Specifies if CLog DLLs (recommended) or Source Code is used durring compilation. DLLs compile faster and enable double-click of log messages to bring up corresponding IDE editor line. Source code allows you to investigate, modify, and extend the internal functionality of CLog. This magical feat is accomplished by manipulating global defines and DLL import enable flags.

Platforms

Specifies what Unity platforms will build with CLog logging. All logging statements on unselected platforms will be compiled out of existance. Note the standalone editor platform cannot be unselected.

Build Levels

Specifies active CLog logging levels in Unity builds. These globaly effect which build log levels are accepted and do not effect editor. Logging statements on unselected levels will be compiled out of existance.

Internal Levels

Specifies active CLog logging levels for CLog internal debug messages. Adding Levels Info, Debug, or Trace can be immensily helpful if CLog behaves unexpectedly.

Test Logger

The test logger is used to test your configuration. When the play icon is pressed a logger is initiated and preview test log statements are excuted. This replicates the procedure that would be done in your classes.

Loger Name

Name of logger to be tested. In code the logger name can be set at logger instantiation or defaults to class name.

Logger Context

GameObject associated with log message (Optional). The gameObject gains focus when console log message are selected.

Logger Levels

A test log statement is generated for each selected level.

Logger Message

The message body of the test log statement.

GUI Targets

The Targets section of the GUI provides the ablility to add/remove targets. The + icon next to Targets brings up a add a target adapter. Conversly the - icon will remove the respective target. See Targets for a description of available targets and coresponding settings.

XML Viewer

View and edit CLog XML configuration file directly. Changes will be reflected in GUI Controls once focus is lost. For brevity target attributes that match the default value are removed. See XML File for a description of XML file.

Targets

The target(s) define the log message destination. All target types and associated parameters provide tooltips to minimize documentation referencing. For additional reference see the Targets in the .NET CLog project

A target has a set of configuriable parameters in the GUI that governs its behavior. All targets have parameters Name, Type, Platforms, and Log levels. The Name is the means it which rules reference targets. The Type is a read-only parameter that indentifies the target type. The Platforms determines what platforms will use this target. The Log levels determines what log levels will use this target.

UnityConsole

UnityConsole logger writes log messages to the Unity editor console and could be your only target. Internally UnityConsole uses Unity Debug log APIs so the console always behaves as expected.

GameConsole

GameConsole logger writes log messages to a provided console assest instantiated inside your game. The Console contains a settings menu to configure Font and Log Levels. When not it use it can be disabled or minimize to icon form. Rich text markup is accepted by Game Console.

NLogLogger

NLogLogger writes log messages to the extremely powerful NLog logger located in the [assets store]. Both CLog and NLog were developed together and are very complimentary. CLog primary purpose is logger abstraction and NLog is intended to deliver advanced logging features.

Null

Null Logger drops all log messages.

GameConsole

GameConsole logger writes log messages to a provided console assest instantiated inside your game. The Console contains a settings menu to configure Font and Log Levels. When not it use it can be disabled or minimize to icon form. Rich text markup is accepted by Game Console.

XML File

The XML configuration file is called CLog.xml and must be located in the Assets/Resources folder. The CLog framework is configured by CLog.xml and works independent of the GUI. The GUI is the recommend method of modifying the configuration, but th XML can be edited directly. Altering the CLog.xml configuration after build/deployment could be a valuable debug feature. If CLog.xml file is not present the default behavior is to drop all log messages.