Getting started

Go to Tools menu and select 'GreedyLogger'->'Instantiate GreedyLoggerInitializer' to instantiate initializer in your current scene. This game object is required for GreedyLogger to work. Make sure you place this object into the **first** scene that will be opened in your game. After that, you can go to settings asset by selecting 'Tools'->'GreedyLogger'->'Open Settings' in menu. Available settings:

- **Logging Enabled** - this checkbox allows you to enable and disable all logging through the GreedyLogger.
- **Write Logs To Files** - allows you to write a logs to a new file each time you start the game.
- **Max Files Count** - specifies how many files can be created in target directory. If the number of files exceeds the maximum, the oldest file will be deleted.
- **Log File Directory** - in which directory you want to store your files. If not specified, all files will be placed into "Application.persistentDataPath" directory.
- **Contexts** - here you can enter any number of contexts for your logs. All entries from this list will be generated into enum. By default there are 4 contexts: *Gameplay*, *UI*, *Meta*, *Infrastructure*.
- **Contexts Filter** - allows you to filter your logs by selecting contexts you want to be shown in Console.
- **Log Levels** - this list contains logging levels. By default there are 3 levels: *Default*, *Warning*, *Error*. You must have at least 1 logging level. Each logging level consists of these elements:
    - Name - the name of your logging level. It will be used in the logging method name.
    - Color - color of the text in logging level.
    - Colorize Context Only - checkbox that allows you to colorize only context, not all text. If a message is sent without context with this checkbox enabled, the whole message will not be colored at all.
    - Emphasis - here you can choose what style your log message will have. You can select**bold**, *italic* and underline style or ***combine*** them.
    - Type - you can choose whether your log message will have average *Log*, *Warning*, *Error* or *Assert* type.
- **Log Exceptions** - check this box if you want to generate a method that will allow you to log exceptions (similar to the default Debug.LogException(Exception ex) method). Buttons:
- **Generate Code** - press this button to generate contexts, importance levels and custom logging methods.
- **Reset Contexts Filter** - resets contexts filter to default value. By default this filter passes all logs with or without context.
- **Reset To Default** - restores all logging settings to default values. Do not forget to press **Generate Code** button after restoration to apply changes!

Basic usage

Let's imagine you've just imported this package into your project. After instantiating GreedyLoggerInitializer into your scene you will be able to use GreedyLogger without any issues. To gain access to GLogger class with logging methods add this using statement:

```
using GreedyLogger
```

Do not forget to add a reference to GreedyLogger assembly to your working assembly defenition in case you need it.

To log a message with default logging levels simply do this:

```
GLogger.Log("Hello world!"); // Logs a message with a default logging level
GLogger.LogWarning("Hello world!"); // Logs a message with a warning logging level
GLogger.LogError("Hello world!"); // Logs a message with an error logging level
```

Use this method to log an exception:

```
GLogger.LogException(new Exception());
```

Let's say you added a new logging level named 'Success' and pressed 'Generate Code' button. To log a message with this logging level and all its settings (Color, Emphasis, Type) just simply call this method

```
GLogger.LogSuccess("Hello world!");
```

If you want to add a context to your log, you need to add it as a second parameter to the logging method:

```
GLogger.Log("Hellow world!", LogContext.Gameplay)
```

This message will look like this: [Gameplay] Hello world! If you want to add a custom context, just open the settings asset, go to Contexts list and add a new context (for example, AI). To be able to use it from code just press 'Generate Code' button below. After that you can call any logging method with your new context:

```
GLogger.Log("Hellow world!", LogContext.AI)
```

Logging to files

GreedyLogger allows you to write all your logs to files (if needed). To enable this feature simply toggle **Write Logs To Files** checkbox. You can select target directory and maximum files count in it in the settings asset. This feature is disabled by default.