Obtaining data

Traceable data: Which data will be traced?

Allocating and releasing memory

The most important part of this Analyzer module is tracing all allocation and free calls within the code. That includes normal allocation/free calls as well as allocations in the Backing Heap.

The used terminology within the Analyzer is:

- Allocate memory <-> Alloc
- Free memory <-> Free
- Allocate in Backing Heap <-> Create
- Free Backing Heap Space <-> Destroy

Informational data

> This feature is not added to a public interface yet.

Additional to these four existing types are several information types.
These informations are used to provide more information and are generally used as orientation and filter helpers.

List of information types:

- Allocation failed message: Contains information of the failed request
- Warning message
- Error message - Possible types: textual message or error code
- Info message: Supports different priority levels additional to the message
- Start and end marker: Defines a named time range

Start and end marker

These marker define a specified timespan within application runtime.
Example: Encapsulate the render loop. Based on these markers it is easy to find out which memory changes happened every single loop iteration.

Retrieving data

Binary memory logging

The standard way to log memory changes is to enable it with CMake.
To enabled it you have to complete the following steps:

- Enable MemoryPools

```
FEATSTD_MEMORYPOOL_ENABLED true
```

- Enable monitoring

```
FEATSTD_MONITOR_ENABLED true
```

- Enable memory binary logging

```
MONITOR_MEMORYPOOL_BINARY_LOGGING_ENABLED true
```

Optional:

- Change filename

```
MONITOR_MEMORYPOOL_BINARY_LOGGING_FILENAME "filename.membin"
```

> Only logging into a file is currently supported. Turning off Performance Recording whilst logging memory changes is currently not supported yet. Using TCP/IP or Serial without an established connection (and therefore Performance Recording is ignored) is working though.

Textual memory logging

It is possible to load a trace-log of your Logger into the Analyzer. There are a few required steps to make it work.

- The logger has to be enabled
- The messages are sent as DEBUG messages. So the logging level has to be DEBUG as minimum.

```
FEATSTD_LOG_SET_LOG_LEVEL(Debug);
```

> Set the log level as soon as possible. All allocations before will not be traced, this also includes the memory pool preinitialization.

- The messages have to keep their default format:

```
// sample for heap alloc
0000000.025 [0x00000000] DEBUG FeatStdMemoryManagement FeatStdDefaultMemoryPool(0) heap all
// sample for alloc memory
0000000.026 [0x00000000] DEBUG FeatStdMemoryManagement \featstd\src\FeatStd/Container/Vecto
// sample for free memory
0000000.027 [0x00000000] DEBUG FeatStdMemoryManagement \featstd\src\FeatStd/Container/Vecto
// sample for heap free
0000000.028 [0x00000000] DEBUG FeatStdMemoryManagement FeatStdDefaultMemoryPool(0) heap fre
```

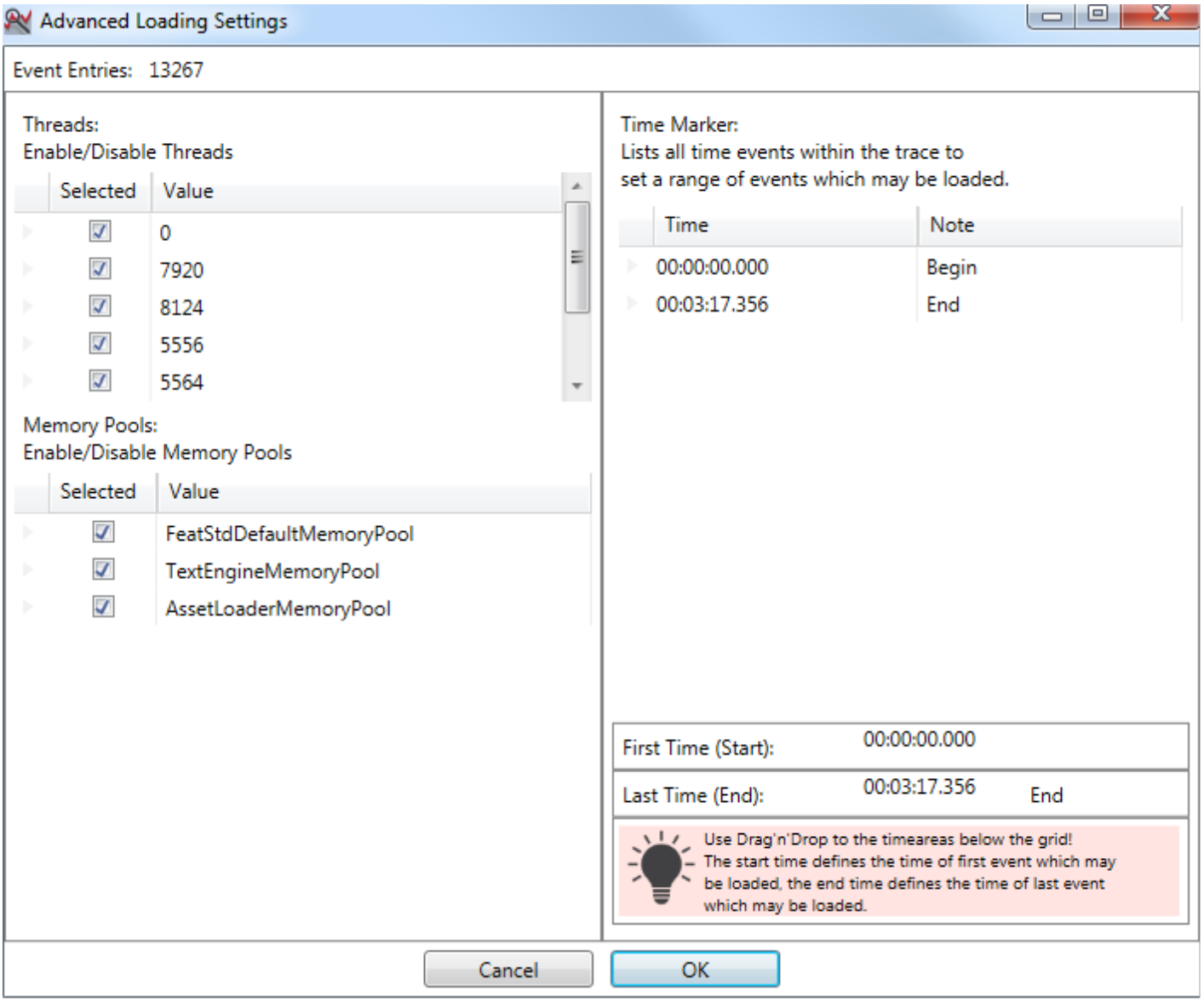
> CgiAppLogger replaces the default logger and changes the format for DEBUG messages.

- To trace all possible data, it is important to create the LogAppender before initializing memory pools and destroy it afterwards.
- The file extension for these files has to be '*.memlog'

> All the other message types: info, warning, error, start/end marker are not available when using these textual logs.

Loading data

All logfiles can be loaded by clicking the application menu button -> open -> Open Memory-Log or by Drag'n'Drop.
When loading a binary log file, a loading window will open after preanalyzing the content of the log file.



This window gives information about the logfile and allows to prefilter the dataset.
It is possible to:

- Disable threads
- Disable memory pools
- Set a time range based on all recorded time marker. Use Drag'n'Drop to place a time marker on 'Start' or 'End'.

> Large log files may slow down the analyzer drastically. The MemoryPool-Analyzer settings contains a property to set a cap for maximum loaded entries

Revision #6
Created 2023-02-16 05:41:36 UTC by Tsuyoshi.Kato
Updated 2025-01-10 09:12:52 UTC by Elisabeth Zauner