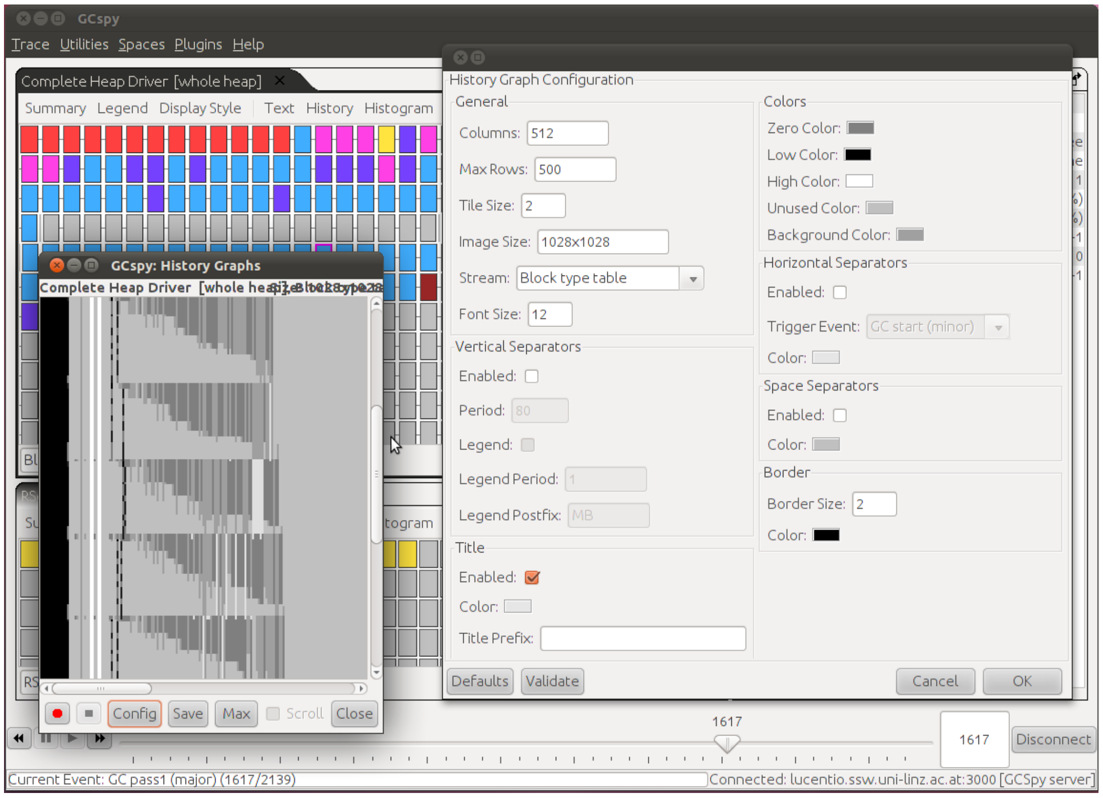
Interactive history graphs for GCSpy

GCSpy [1] is a generic heap visualization framework for collection and replay of memory management behavior. With only a few changes to the memory management system which it is incorporated into, it can visualize various information about memory management systems.

Recent work updated the UI and enhanced it with scripting functionality. Two areas neglected during these changes were the plugin API and the plugins themselves. The former needs to be revisited to take new functionality into account, and the plugins updated. One particular interesting plugin is the “History” plugin which plots values for all data of a single instrument along time, this results in simple color maps as shown in the figure below:


GCspy
Trace Utilities Spaces Plugins Help
Complete Heap Driver [whole heap]
Summary Legend Display Style Text History Histogram
GCspy: History Graphs
Complete Heap Driver [whole heap], Block type table
Config Save Max Scroll Close
Defaults Validate Cancel OK
1617
Disconnect

History Graph Configuration

General
Columns: 512
Max Rows: 500
Tile Size: 2
Image Size: 1028x1028
Stream: Block type table
Font Size: 12

Vertical Separators
Enabled: ☐
Period: 80
Legend: ☐
Legend Period: 1
Legend Postfix: MB

Title
Enabled: ✓
Color:
Title Prefix:

Colors
Zero Color:
Low Color:
High Color:
Unused Color:
Background Color:

Horizontal Separators
Enabled: ☐
Trigger Event: GC start (minor)
Color:

Space Separators
Enabled: ☐
Color:

Border
Border Size: 2
Color:

Current Event: GC pass1 (major) (1617/2139)
Connected: lucentio.ssw.uni-linz.ac.at:3000 [GCSpy server]

Additionally there is a simple configuration dialog for this history graph (on the left), shown on the right. The usefulness of this history graph is very limited: the data points are too small (using a fixed 1x1 pixel square per data point) and there is no way to do explorative data analysis. I.e. it does not allow zooming, there is no possibility to examine values live, there is no way to cross-reference data points to other visualizations, etc.

The task for this thesis is to update the plugin API, and use it to improve the History graph, its configuration UI and add functionality for explorative data analysis.

In particular the resulting plug-in should have the following capabilities:

- improved visualization of “history” data and better integration into the existing UI. Possible visualizations include various 2D and 3D plots (color maps, 3D graphs; see the end of this document for examples)
- redesign of the configuration dialog to allow save/reload of settings. The visualization should be highly configurable for presentation in various circumstances. Changes should propagate directly to existing visualized data.
- enable the developer to explore the existing data by allowing interaction with the visualization: i.e. zooming, panning, and “flights” over the data; it should be possible to cross-reference the data with other visualizations, e.g. showing values below the cursor or view the given data point with other visualizations etc.
  This also means that the visualization itself should be updated in real-time.

The project consists of the following subtasks

- read up on GCSpy and familiarize yourselves with the GCSpy code, in particular the interaction in the plug-in system
- requirements analysis, especially in regards to the UI. Determine use cases. Examine existing data visualization tools (statistics tools). Compare your ideas to similar tools which allow data exploration.
- incorporate your own ideas: how could you extend this visualization for multiple data points at the same time?
- analyze the required changes in the source code; during this, find and compare suitable (graphics) libraries to use.
- implement a plug-in using the plug-in API. The latter might need to be extended to cover the intended use cases.

Basic experience in working with source control systems is useful. In particular, Mercurial will be used for versioning. Basic interest and knowledge in memory management is recommended.

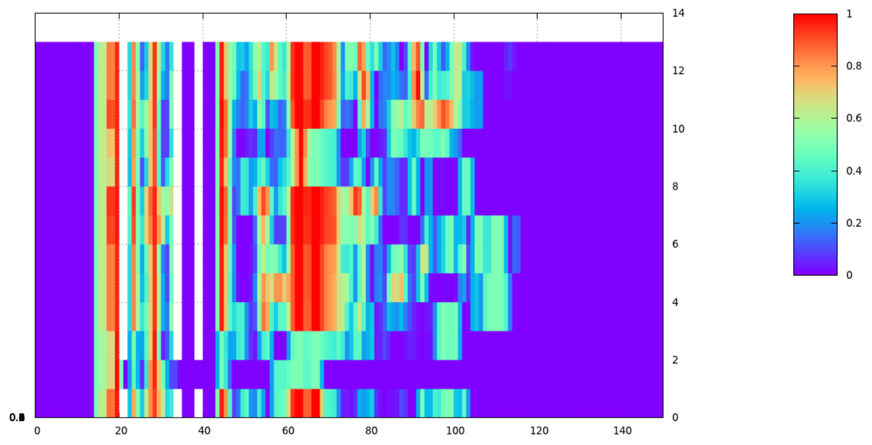
Programming language: Java

Advisor: Dipl.-Ing. Thomas Schatzl (thomas.schatzl@jku.at, room HF305 at the SSW institute)

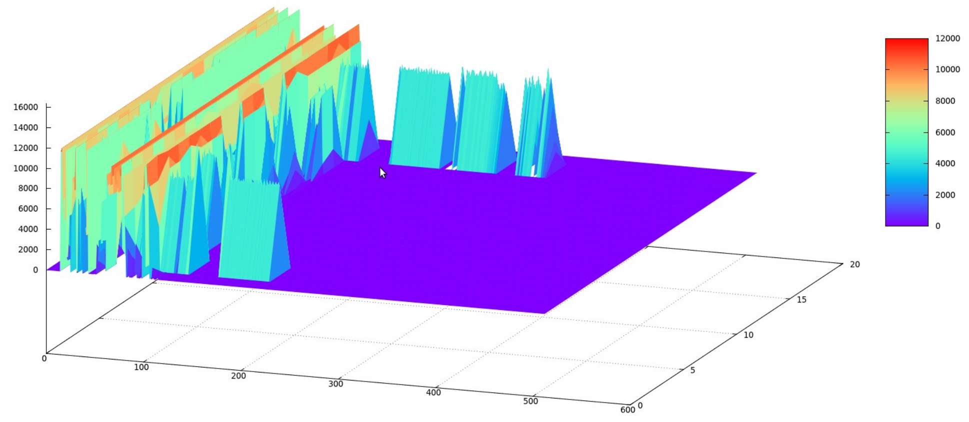
References:

[1] GCSpy homepage: http://www.cs.kent.ac.uk/projects/gc/gcspy/

Color map example:



Possible 3D visualizations



16000
14000
12000
10000
8000
6000
4000
2000
0
0
100
200
300
400
500
600
0
5
10
15
20
12000
10000
8000
6000
4000
2000
0