Online Binary Visualisation Tool

# Mission Statement

To create an efficient and cross-platform binary visualisation tool (binvis) based on the HTML5 standard so that it can be viewed and utilised on modern internet browsers. The binvis tool will incorporate the multitude of visualisation techniques learnt from literature and some standalone executable binvis tools currently available.

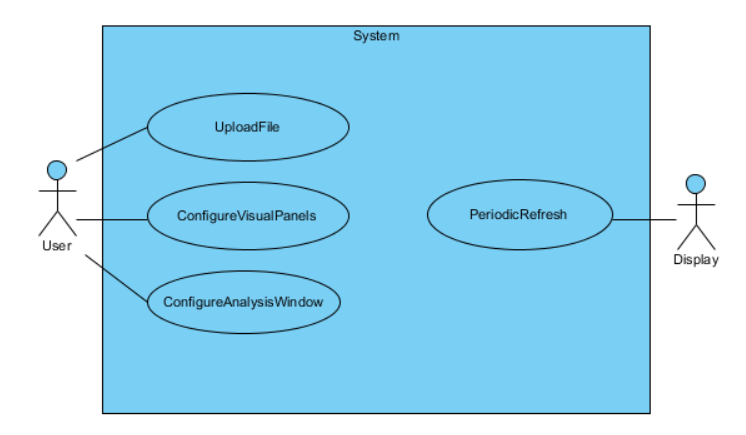
# Functional Requirements

1. The User must upload a file for analysis.
2. Files loaded are locally cached.
3. Processing of File is done on local CPU.
4. The User can see the Overall Bytemap of file upon successful upload.
5. The User can choose a 256KB viewing window in which analysis is done.
6. The User can choose the Visual Panel to be shown.
7. All visual panels are updated in real-time upon User input.

# Non-Functional Requirements

1. The visuals must update at frame rates of 30-60 FPS. This requires update logic to finish under 16 ms or less.
2. User interface has to be highly intuitive – it takes 10 minutes or less to familiarise with interface and navigations.

# Use Case



## Upload File

1. User clicks on the Load File button.
2. System prompts User for File path.
3. User confirms File selection.
4. System reads File loaded as a sequence of bytes.
5. System invokes refresh on Display.

**Alternative: File path invalid/not found**

1. System displays error via Display.

## Configure Visual Panels

1. User clicks on a Panels button.
2. System displays the Panel selection GUI via Display.
3. User clicks on the appropriate Panel buttons to choose the panel to be displayed.
4. System updates the Display.

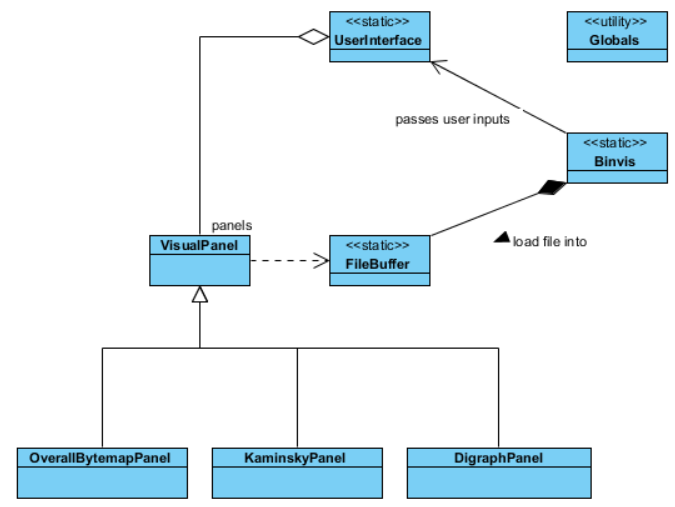
## Configure Analysis Window

1. User can change the offset of the viewing window by means of a slider or through manual input.
2. The Display is updated to reflect changes in the offset.

## Periodic Refresh

1. The System must post a refresh call to the Display every 16 ms or less.

# Conceptual Model



## Binvis

1. A static instance that is the interface between the underlying System and the browser client.
2. It will contain the necessary event handlers to listen for events from the browser.
3. Parameters from events fired from the browser client (such as mouse coordinates and key press) will be passed over to the UserInterface for further processing.
4. Invokes the FileBuffer to cache the loaded file.
5. Stores the viewing offset to be used by other instances.

## FileBuffer

1. When a File path is specified, this class will be instantiated.
2. This class holds the File’s attributes.
3. Read data is cached.
4. File reading operation is asynchronous. Hence, dependent instances have to register the appropriate callback function.

## UserInterface

1. Holds the reference to the drawing canvas on screen.
2. Contains many helper functions to aid rendering.
3. All rendering calls are made on this class to output to the Display.
4. Updates itself automatically via a timer.
5. Contains a collection of VisualPanel, which is updated periodically.

## VisualPanel

1. Contains the rendering logic and event handlers needed for interactivity.
2. Upon receiving an update from the UserInterface, it runs its update function.
3. Upon receiving a render request from the UserInterface, it performs its own rendering.
4. Events fired from the UserInterface can be passed down to this VisualPanel.

## BytemapPanel

1. Shows a more fine-grained view of the bytemap within the extents of the Viewing Window defined by the offset and the 256KB block limit.
2. Scan mode is adjustable between: Scanline, Hilbert curve, Snake

## KaminskyPanel

1. Shows a Kaminsky dot plot for the range of bytes indicated by the Viewing Window

## DigraphPanel

1. Shows a digraph plot for the range of bytes indicated by the Viewing Window

# Performance Optimisations

1. Use of an auxiliary canvas to buffer the visuals. They are only re-rendered when an update is issued.

# Glossary

|  |  |
| --- | --- |
| **Term** | **Description** |
| User | An external entity that interacts with the browser client. |
| Browser (client) | The internet browser used to view this web application. |
| System | Encompasses the components managing the user interface and the processing of byte data. In short, the binvis tool itself. |
| Panel | The screen shown on the browser interface. Different panels show different visuals depending on its use. |
| Viewing Window | The window of analysis defined by the global byte offset, and ranges for 256KB. |

# Notes

## Load File Process

1. User clicks on load file.
2. User selects file.
3. FileBuffer is updated to use selected file.