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.

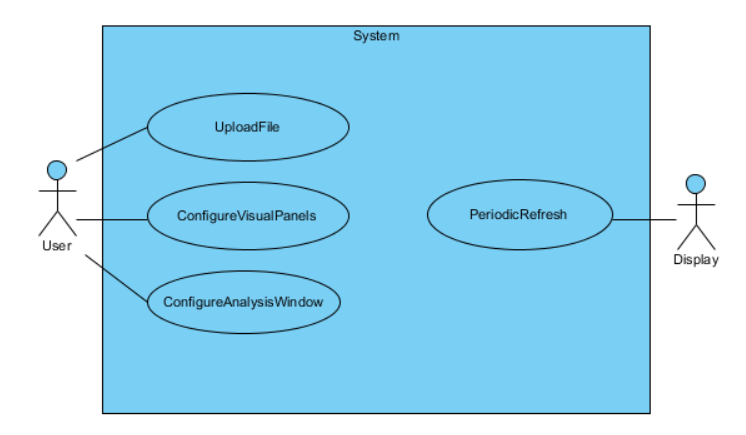
# Functional Requirements

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 (scrolling or zooming in/out) a viewing window in which analysis is done.
6. The User can enable/disable the visual panels available on screen.
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 toggle their visibility.
4. User confirms options.
5. System updates the Display.

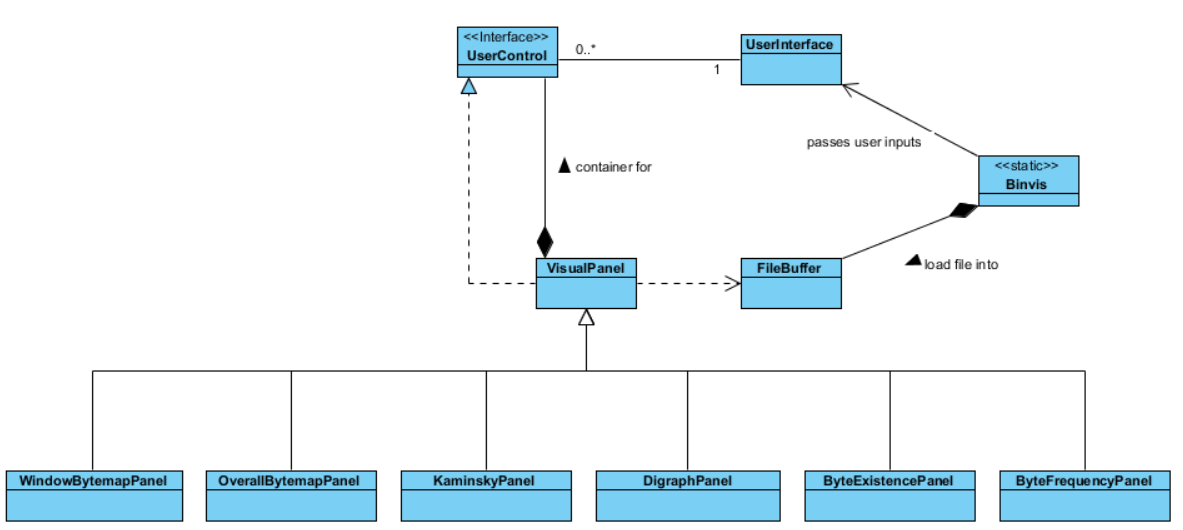
## Configure Analysis Window

1. User holds and drags the Viewing Glass in the Overall Bytemap to change the offset but keeping the same range of analysis.
2. User holds and drags the extents of the Viewing Window to change the offset and range independently.
3. While the User is holding and dragging, the Display is updated in real-time to reflect the changes in offset and range.

## 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 Window attributes to be used by any instance that requires them.

## FileBuffer

1. When a File path is specified, this class will be instantiated.
2. This class holds the File’s attributes.
3. The File is only read (at the specified offset and range) upon request from any dependent instances.
4. Read data is cached.
5. Duplicate requests (ie, same offset and range) will be ignored and the cached data is returned instead for performance.
6. 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 UserControl, which is updated periodically.

## UserControl

1. A general interface describing the common operations of an interactive control.
2. Upon instantiation, it is registered as a child of either the UserInterface, or a VisualPanel.
3. Upon receiving an update request from the UserInterface/VisualPanel, it is updated.
4. Upon receiving a render request from the UserInterface/VisualPanel, it performs its rendering operation.
5. It can listen for events from the browser client.

## VisualPanel

1. A concrete implementation of a UserControl.
2. It is also a container class consisting of other UserControl. Hence, a UserControl can register as a child in this class.
3. Upon receiving an update from the UserInterface, all children are updated.
4. Upon receiving a render request from the UserInterface, it performs its own rendering and sends a rendering request to all children.

## OverallBytemapPanel

1. Displays the overall bytemap of the loaded file.
2. It caches its own bytemap so that it need not request from the FileBuffer often.
3. Contains the Viewing Window, with adjustable extents.
4. The Viewing Window properties are stored globally in the Binvis class.

## WindowBytemapPanel

1. Shows a more fine-grained view of the bytemap within the extents of the Viewing Window.
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

## ByteExistencePanel

1. Shows the existence (and normalized frequency) of each byte in within every scanline
2. Only applicable if WindowBytePanel is enabled and is on Scanline mode.

## ByteFrequencyPanel

1. Displays a frequency histogram of the bytes within the Viewing Window.

# 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 viewfinder determining the extents of the file to analyse. |