Software Design

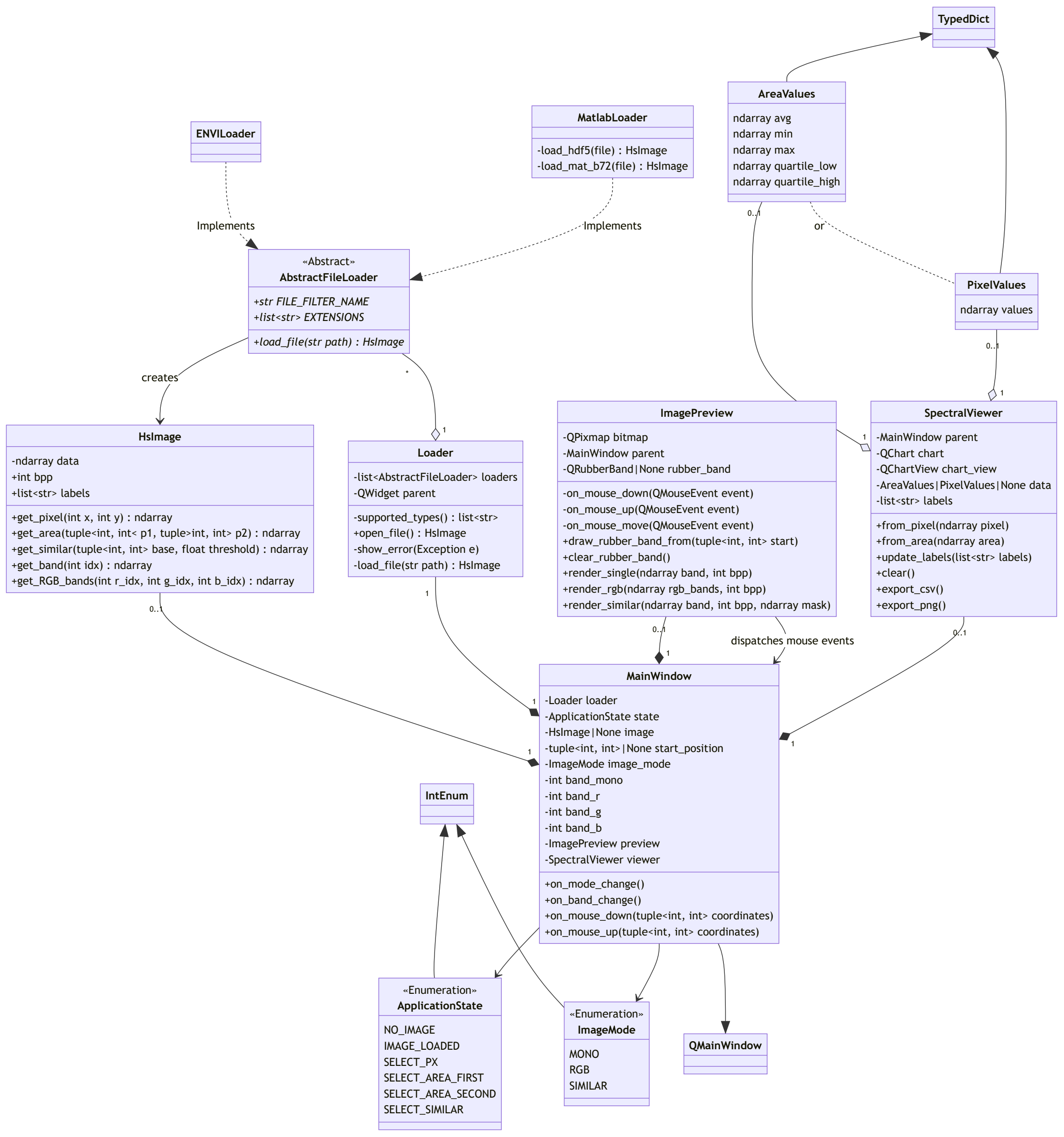
Team 7RS

2022-11-24

Contents

1 Classes

1.1 UML diagram



1.2 Class description

1.2.1 MainWindow

Derived from `QMainWindow`. Contains UI setup, most UI elements and event handlers for buttons.

Stores:

- loaded image
- state

- image mode
- selected bands
  - mono
  - red
  - green
  - blue
- tolerance for the *magic wand* functionality
- starting position of area selection

Event handlers:

- open file menu action - request loading file from `Loader`
  After the file is loaded:
  1. Save the `HsImage`
  2. Set state to `IMAGE_LOADED`
  3. Reset selected bands to default values
  4. Clear `SpectralViewer` or update it with the whole image and update its labels
  5. Clear the rubber band on `ImagePreview`
  6. Render the new image in `ImagePreview`
- mode selection - display appropriate band settings and update the preview
- band change - save the new value and update the preview
- ESC button clicked - set state to `IMAGE_LOADED` (or `NO_IMAGE`) and clear the rubber band
- select pixel/area/similar - update the state according to the clicked button

Actions triggered by `ImagePreview`:

- mouse down - ignore if not in `SELECT_AREA_FIRST` state, save `start_position`, request drawing a rubber band and change state to `SELECT_AREA_SECOND`
- mouse up - action depends on current state:
  - `SELECT_PX` - get selected pixel and update `SpectralViewer`
  - `SELECT_AREA_SECOND` - get area between `start_position` and received coordinates and update `SpectralViewer`
  - `SELECT_SIMILAR` - get similar pixels, change image mode to `SIMILAR` and update `ImagePreview`

  After any of those actions state must be restored to `IMAGE_LOADED`.

1.2.2 `Loader`

Provides an interface for loading files. Manages errors during file loading (I/O, invalid format, expectations not met, etc.). Has a list of available file loaders.

1.2.2.1 Notes for implementer

1. Use static `QFileDialog` methods - it is simpler to implement. Example:

   ```
   file_path, used_filter = QFileDialog.getOpenFileName(
       parent,
   ```

```
        "Open image",
        "",
        "Matlab file (*.mat);;ENVI .hdr labelled image (*.hdr);;All
        ↪  supported types (*.mat *.hdr)",
    )
```

2. Keep (or get as a function parameter) a reference to parent, to block the parent (main) window, while a dialog (open file, file loader *settings* or error) is open.

3. Error messages can be displayed using `QMessageBox` with static `.warning` or property-based API if informative or detailed text should be set.

1.2.3 `AbstractFileLoader`

An abstract base class for specialised file loaders. Each loader must have a getter for a friendly *category* name and a list of supported extensions. `load_file(path: str) -> HsImage` abstract method provides an universal interface for loading files.

1.2.4 `MatlabLoader`

Derived from `AbstractFileLoader`. Supports `.mat` files. If multiple three-dimensional variables are available prompt the user for selection.

1.2.4.1 Notes for implementer

1. `QInputDialog` can be used for variable selection:

```
variable_names = ["data", "something", "a_name"]
selected_var, ok = QInputDialog.getItem(
    parent,
    "Select variable",
    "Variable containing image data",
    variable_names,
    editable=False,
)
```

2. For MAT-files <= 7.2 use `scipy.io.loadmat`
   For 7.3 use `h5py` (a HDF5 library for Python). `h5py` is not available for Python 3.11 yet.

3. **Warning:** Some datasets have negative values instead of using unsigned integers.

1.2.5 `ENVILoader`

Derived from `AbstractFileLoader`. Supports ENVI `.hdr` labelled files. Should report compatibility with `.hdr` files, but a file with the same name, but no `.hdr` extension must exist in the same directory.

1.2.5.1 Notes for implementer

1. ENVI files can be loaded using `GDAL`, but its installation may be tricky. Windows packages can be found here, for Linux it probably will have to be built from source.

1.2.6 `HsImage`

Stores image data:

- raw pixel data as a 3D array [height, width, bands]
- bits per pixel
- band labels

If band labels are not provided in the image file a sequence `1..=bands` should be used instead.

Must provide:

- pixel data given its coordinates
- subarray with pixels bounded by given coordinates
- a mask with pixels similar to one with given coordinates and a threshold
- a single band of the image given its index
- three bands in specified order given their indices

1.2.7 `ImagePreview`

Displays the image preview using specified mode. Proxies events to `MainWindow` after translating mouse position to pixel coordinates. Draws a `QRubberBand` when selecting an area.

Optional: Manages zoom and panning.

1.2.8 `AreaValues` and `PixelValues`

Are derived from `TypedDict` (PEP 589). Both store data displayed in `SpectralViewer` - either *aggregate values* (`AreaValues` - avg, min, max, quartiles) or a single pixel value (`PixelValue`). Their properties are vectors (1D `ndarrays`).

1.2.9 `SpectralViewer`

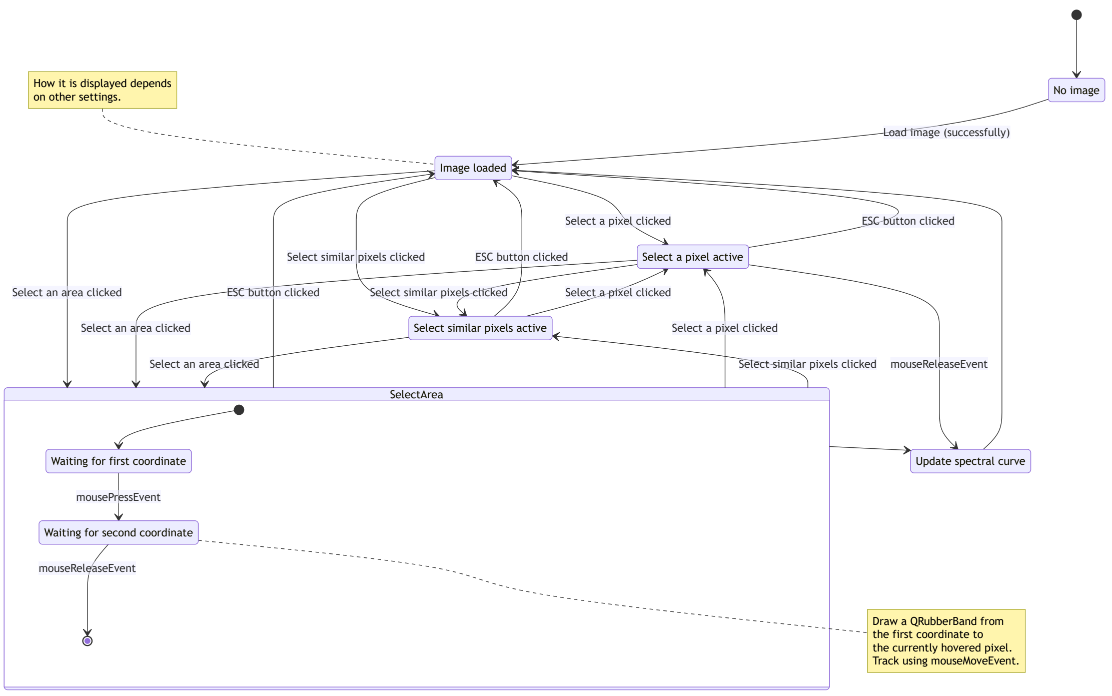
Manages the spectral curve chart. Chart is labeled using stored labels, which should be updated when a file is loaded. When created from a pixel or an area it should calculate appropriate values, store them as `PixelValues` or `AreaValues` and update the chart.
Two methods for exporting must be provided:

- exporting to CSV - simply write `PixelValues or AreaValues` and labels to CSV
- exporting as PNG - render the `chart` or `chart_view` to a `QPixmap` and save it

Note: If needed exporting as JPEG can also be provided just by allowing to save `QPixmap` with `jpg/jpeg` extension.

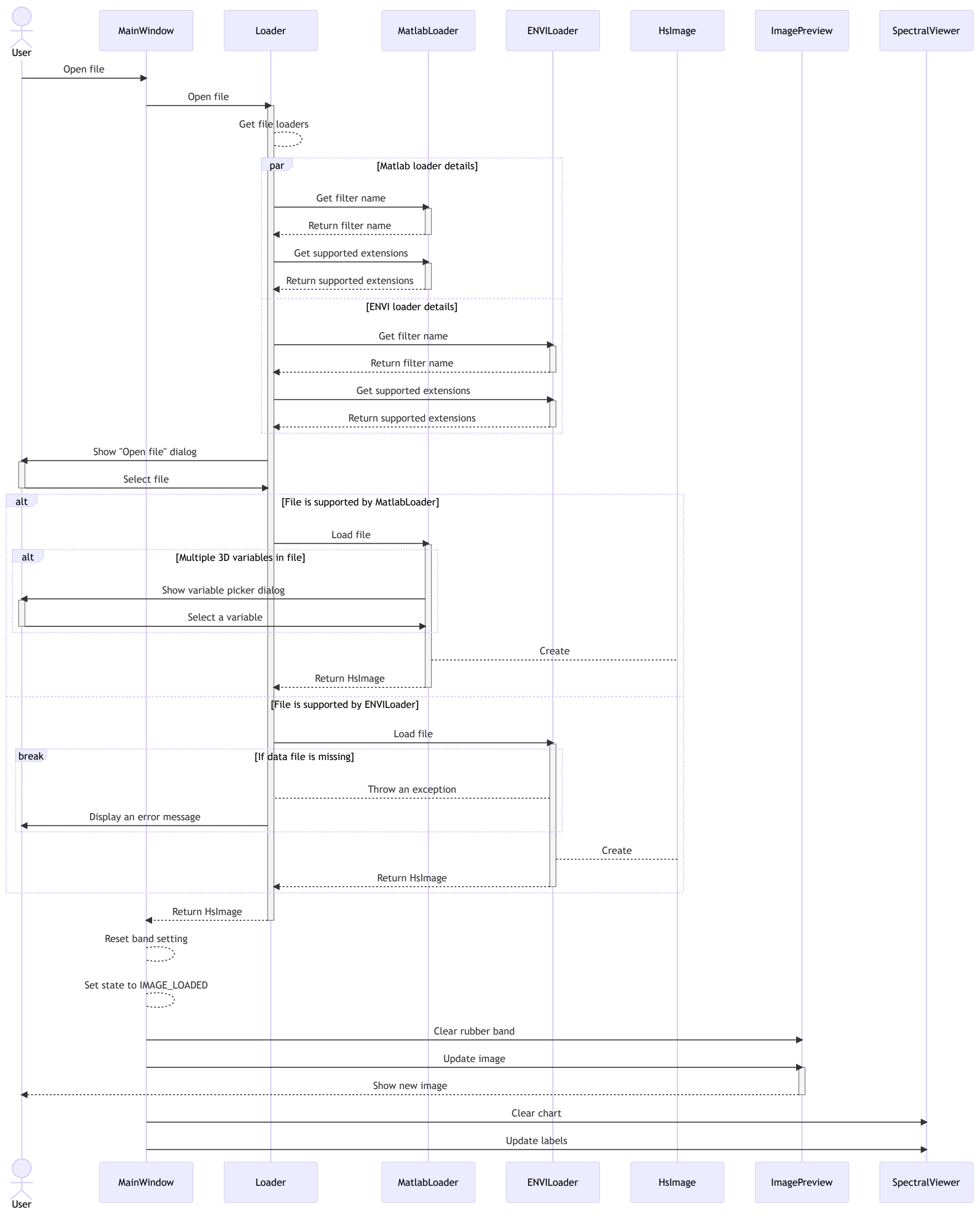
2 High level overview

2.1 State diagram

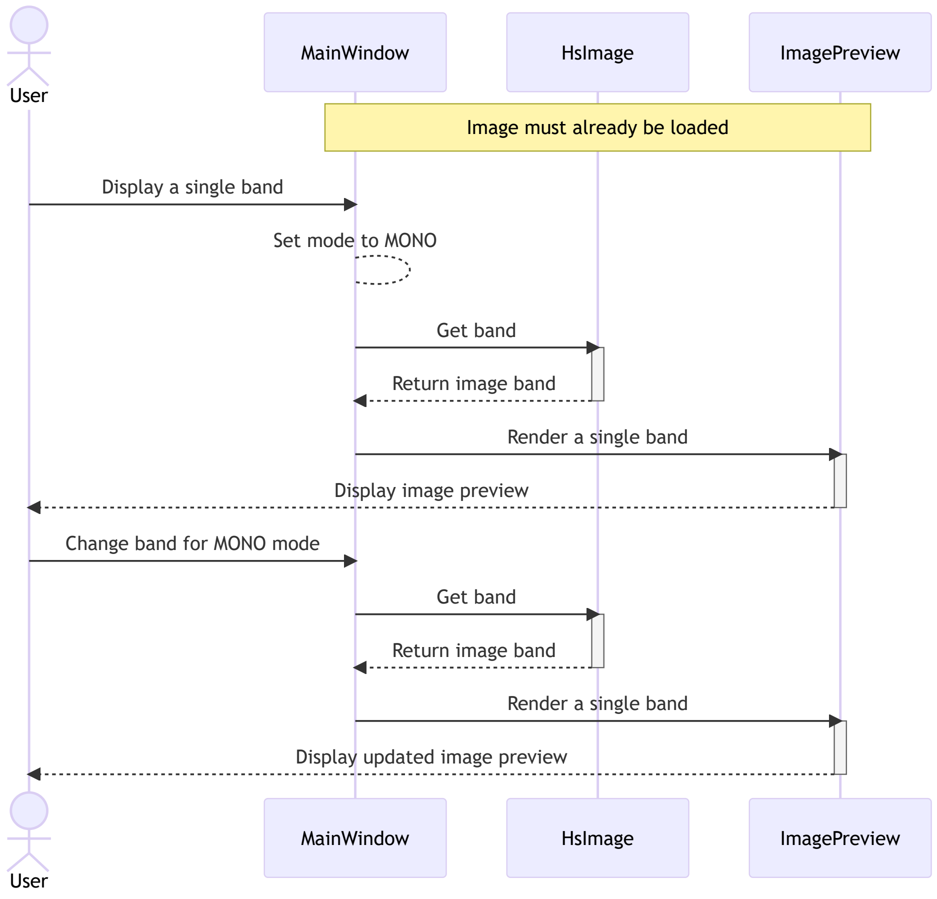

No image
Load image (successfully)
Image loaded
How it is displayed depends on other settings.
Select a pixel clicked
ESC button clicked
Select similar pixels clicked
ESC button clicked
Select a pixel active
Select an area clicked
ESC button clicked
Select similar pixels clicked
Select a pixel clicked
Select an area clicked
Select similar pixels active
Select a pixel clicked
Select an area clicked
Select similar pixels clicked
mouseReleaseEvent
SelectArea
Waiting for first coordinate
mousePressEvent
Waiting for second coordinate
mouseReleaseEvent
Update spectral curve
Draw a QRubberBand from the first coordinate to the currently hovered pixel. Track using mouseMoveEvent.

2.2 User interactions

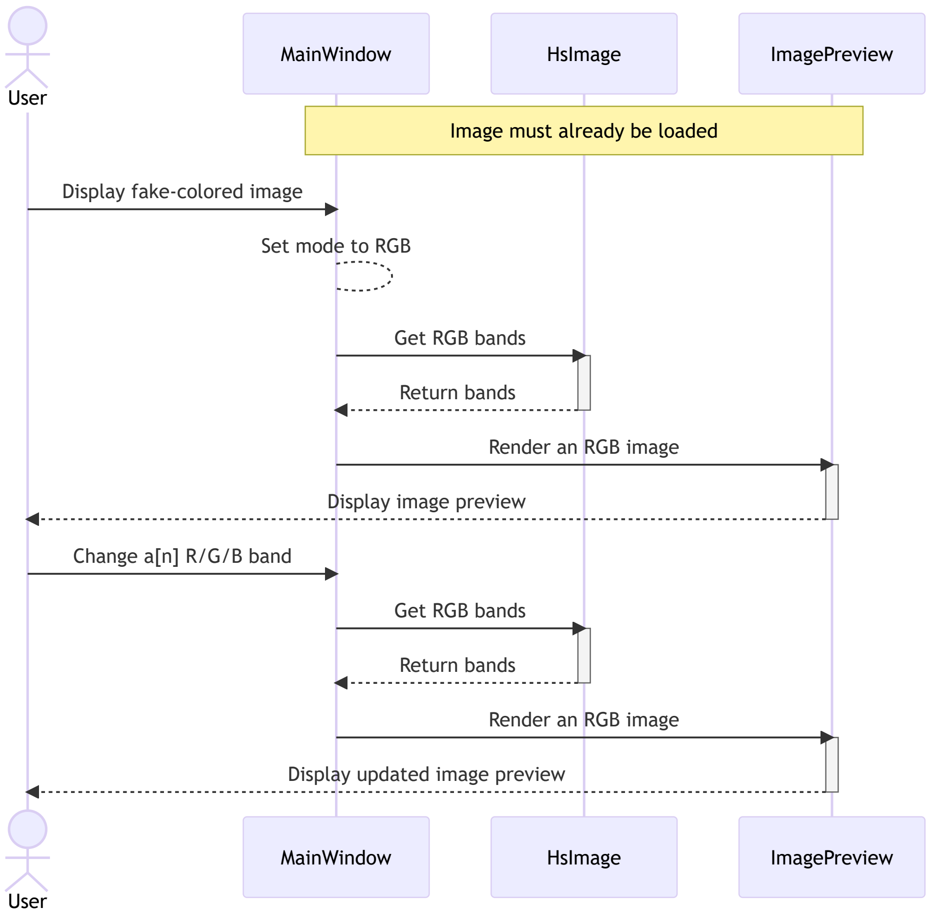
2.2.1 Open image



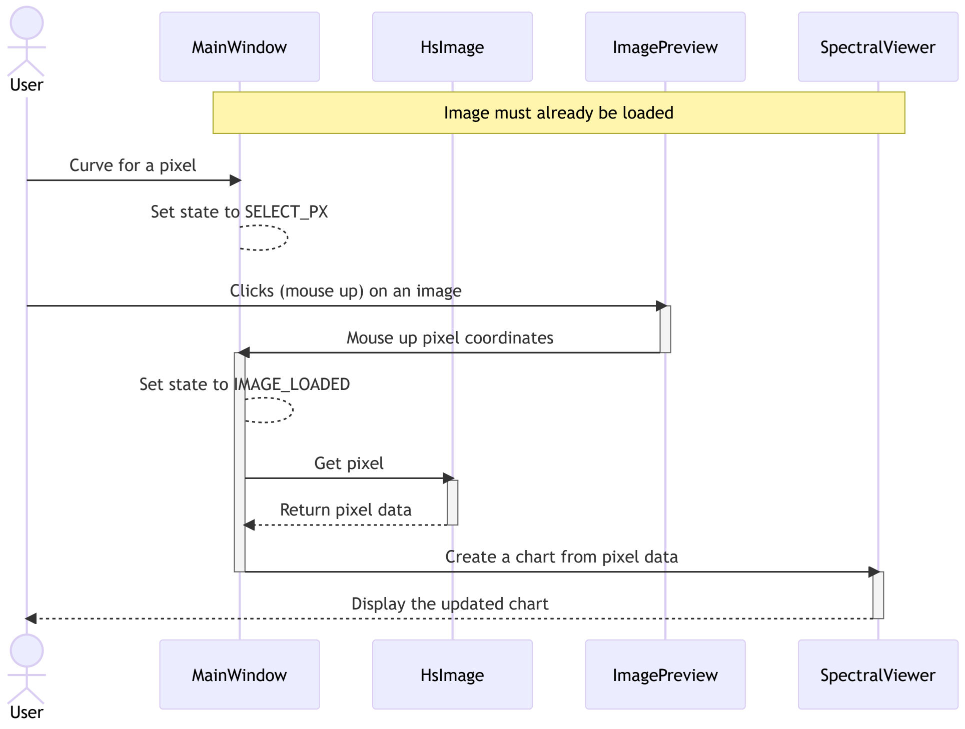
2.2.2 Display a single band



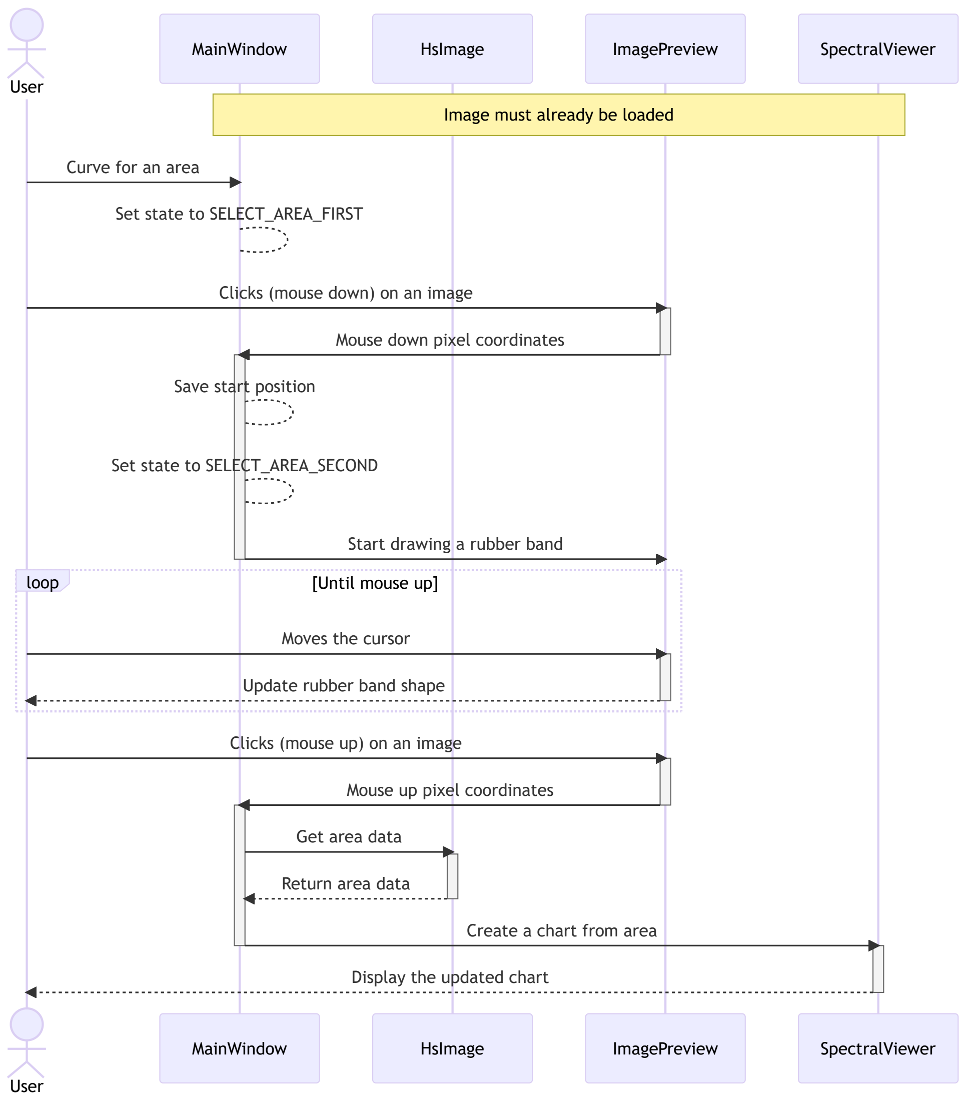
2.2.3 Display fake-colored image



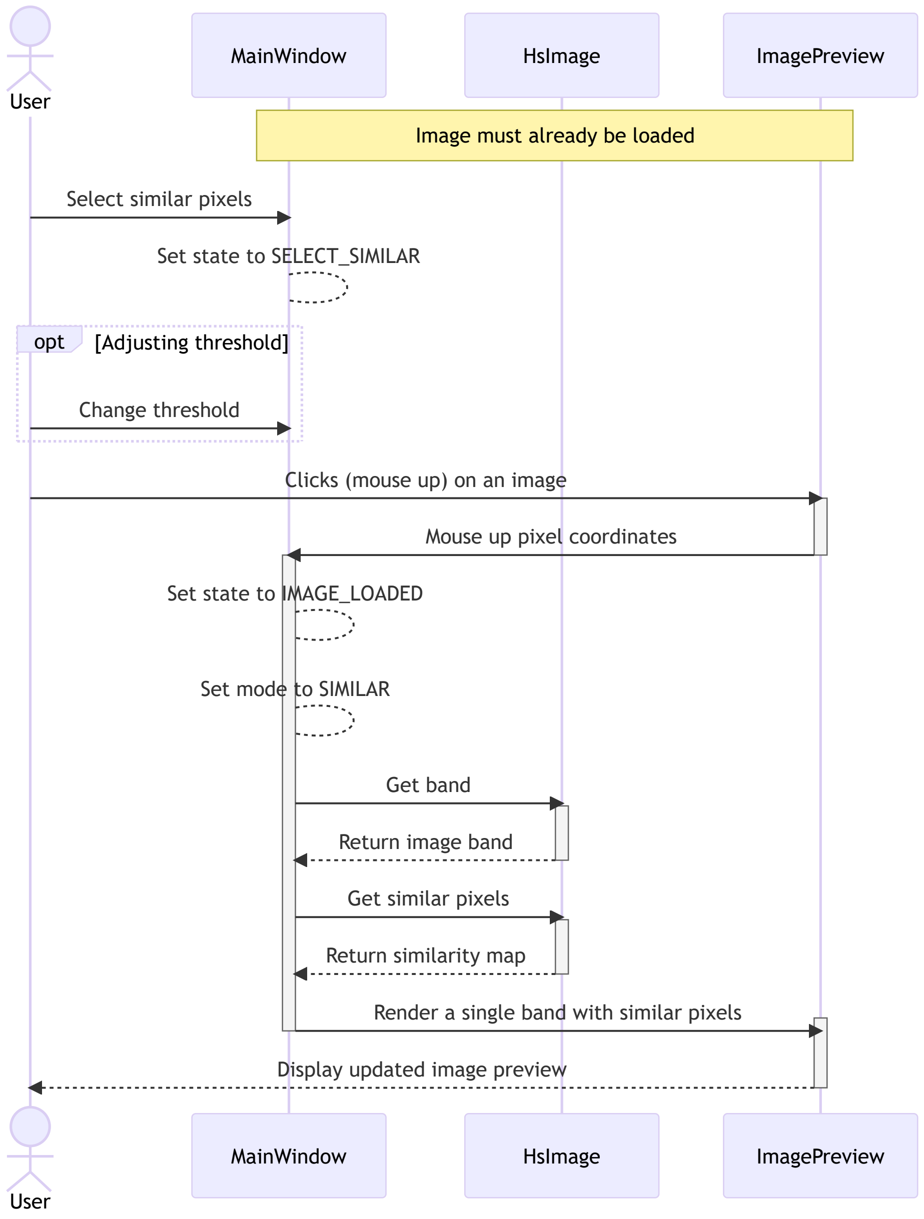
2.2.4 Display curve for a single pixel


User
MainWindow
HsImage
ImagePreview
SpectralViewer
Image must already be loaded
Curve for a pixel
Set state to SELECT_PX
Clicks (mouse up) on an image
Mouse up pixel coordinates
Set state to IMAGE_LOADED
Get pixel
Return pixel data
Create a chart from pixel data
Display the updated chart
User
MainWindow
HsImage
ImagePreview
SpectralViewer

2.2.5 Display curve for a region

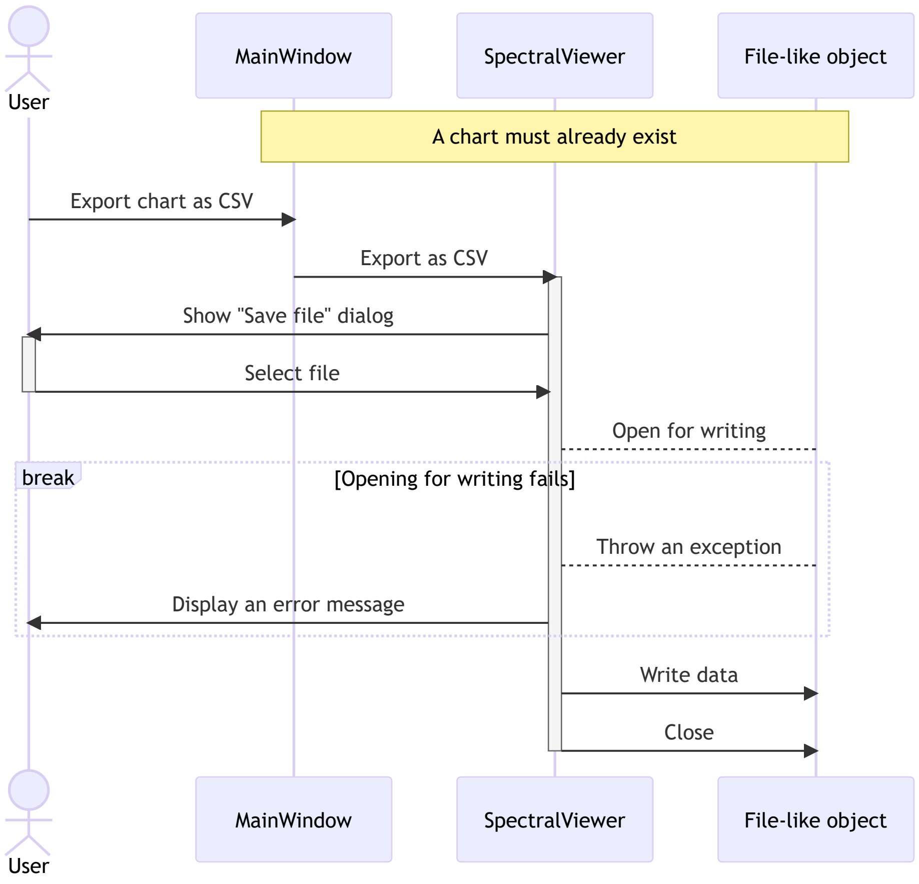

User
MainWindow
HsImage
ImagePreview
SpectralViewer
Image must already be loaded
Curve for an area
Set state to SELECT_AREA_FIRST
Clicks (mouse down) on an image
Mouse down pixel coordinates
Save start position
Set state to SELECT_AREA_SECOND
Start drawing a rubber band
loop
[Until mouse up]
Moves the cursor
Update rubber band shape
Clicks (mouse up) on an image
Mouse up pixel coordinates
Get area data
Return area data
Create a chart from area
Display the updated chart
User
MainWindow
HsImage
ImagePreview
SpectralViewer

2.2.6 Select similar pixels

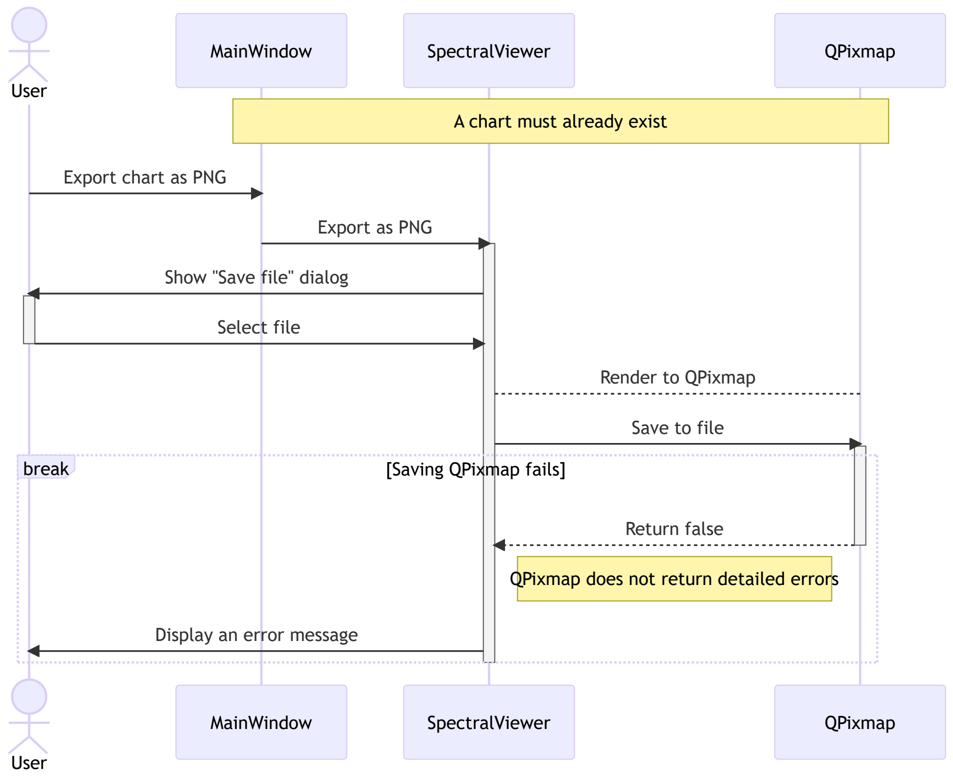


2.2.7 Export spectral curve(s)

2.2.7.1 Export as CSV



2.2.7.2 Export as PNG


User
MainWindow
SpectralViewer
QPixmap
A chart must already exist
Export chart as PNG
Export as PNG
Show "Save file" dialog
Select file
Render to QPixmap
Save to file
break
[Saving QPixmap fails]
Return false
QPixmap does not return detailed errors
Display an error message
User
MainWindow
SpectralViewer
QPixmap