SVG Synoptic

Taurus workshop, Soleil 2026

Johan Forsberg (MAX IV)

May 12, 2026

SVG Synoptic

- A PyQt widget that displays interactive SVG images
- Optional Taurus integration.
- Main use case: intuitive overview of a control system, e.g. a beamline
- Gitlab repo
  `https://gitlab.com/MaxIV/lib-maxiv-svgsynoptic`

Features

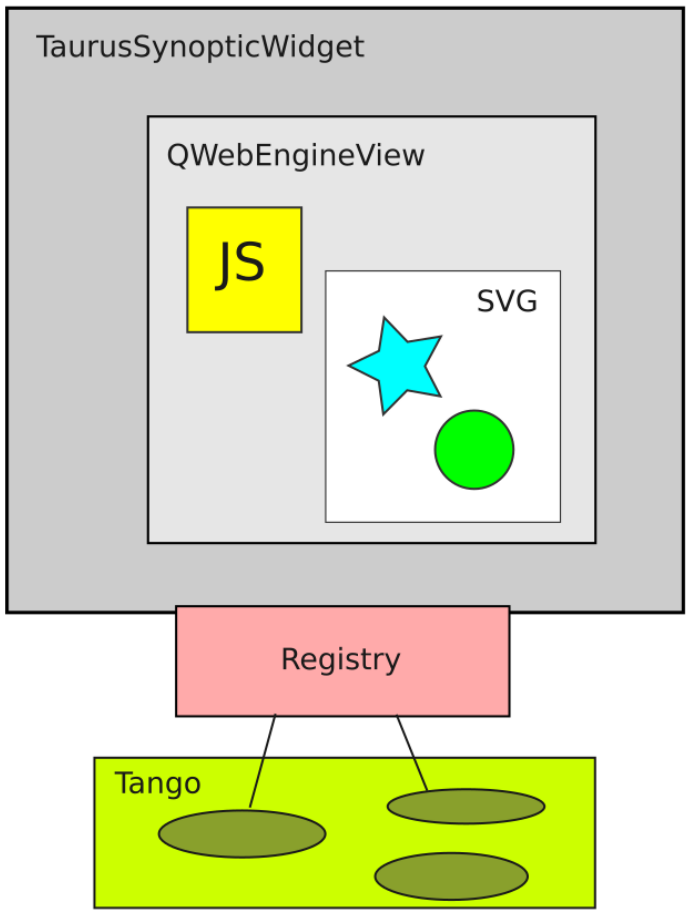
- Uses standard SVGs (but with some special structure)
- Connect to Tango to display e.g. attribute values, device State
- Open popups, panels or other applications on mouse click
- Layers, zooming, thumbnail, quicklinks
- Can handle large systems, by "dynamic" subscription
- Relies on web standards e.g. CSS

Demo

How does it work?

- `SynopticWidget` embeds a web browser using QWebEngine
  - Opens local HTML page with some javascript to load and prepare a SVG file
  - Hooks up mouse events, to send e.g. mouse click events to PyQt
  - Widget executes javascript expressions in the browser context from PyQt to update SVG elements.
- `TaurusSynopticWidget` subclass adds a "registry" for Tango subscriptions
- Uses `d3.js` to handle SVG.

How does it work, page 2



SVG structure

- Main
  - "Vacuum" (arbitrary names)
  - "Optics"
  - ...
  - background (always visible)
  - symbols (never visible, e.g. icons)

Within layers there can be "zoom" levels for different detail

Models

- Hooking up a SVG element e.g. to a Tango device or attribute
- SVG `<desc>` tag inside the element, applies also for child elements

```
<desc>model=sys/tg_test/1/double_scalar</desc>
```

- Just a Taurus model, e.g. `model=eval:1+2`
- Numeric attributes typically attached to `<text>` attributes.
- State, bool affect the `fill` color of the element
- Attribute values also available in `data-value` attribute

Customization

Use CSS to customize e.g. colors

```
/* Override the default bool colors */
.model.boolean[data-value="true"]:not(text) {
    fill: yellow !important;
}

.model.boolean[data-value="false"]:not(text) {
    fill: gray !important;
}
```

- Subclass the `SynopticWidget` e.g. to override the `get_device_panel` method
- Load arbitrary javascript for really custom solutions

Inkscape

- Vector drawing program based on SVG
- Recommended for editing SVGs (but not required)
- Structured with layers
- Set model via the object properties
- Keep a library of icons in "symbols" and clone them (<use> tag)

Generating SVG

- Useful for more complex situations
- Some nice python packages:
  - `svgwrite` (to generate SVG files)
  - `svg.path` (to build SVG paths)

Future

- Hasn't been a lot of development for years
- Works pretty well and easy to extend
- Plans/ideas
  - Qt6 support (via `qtpy`?)
  - Better support for generic "actions" e.g. launching programs
  - Events for easier javascript customization
  - Modernize javascript parts
  - Improve examples

Thanks!

Questions?