Requirements Overview

– Kiko –

February 2024 - May 2024

TophUwO

Contents

Rationale

Data analysis and visualization is an integral part of today's software on any imaginable platform. While many web-based solutions exist that allow the pretty display of data series in a variety of different types, each designed for a very specific purpose, the options in traditional desktop applications and related technologies are limited to non-existent. The Qt framework, for example, provides the Qt Charts module as an in-ecosystem solution to this specific problem. However, due to Qt Charts being tied to the Qt framework on top of its not-so-liberal licensing policy, a data visualization software in the form of an extensible software library, codenamed *Kiko*, will be implemented to alleviate heavy dependencies and allow for a lightweight implementation of data visualization functionality on any platform.

General Information

Unlike existing solutions, the software is meant to be integrated into a desktop application using either a statically or dynamically linked library. To maintain compatibility with main desktop application frameworks, the software shall support multiple output formats such as *Scalable Vector Graphics* (SVG) (for example, allowing graphics output via Qt's *QSvgWidget* component), a host of raster graphics formats such as *Portable Network Graphics* (PNG), *Windows Bitmap* (BMP), and *Joint Photography Group* (JPG), as well as direct rendering to any *OpenGL*-, *Direct2D*-, *Direct3D*-, or *Vulcan*-compatible surface. The core of the software should be a generic data model that is to be extended and refined by specialized chart types. The software will implement numerous chart types, ranging from well-established and frequently used (*bar/column chart*, *line/spline chart*, *area chart*, and *pie chart*) to more obscure and specialized (*heatmap*, *radar chart*, *bump chart*, *histogram*, *donut chart*, and *scatter/bubble plot*). Inside each of these data visualization variants, multiple subtypes are to be implemented that facilitate the display of more complex data collections (e.g. *stacked/grouped bar chart*, etc.). Charts are grouped in a view, consisting of a layout tree and entry cells. Cells hold (supplemental) chart elements such as chart visualizations which contain axes and data series. Each chart can contain multiple axes and data series. Supplemental chart elements are explained further down in this document.

Layouts

Kiko uses a layout tree inspired by Qt's layout system. There is a *top-level layout* which is by default initialized to a 1x1 grid layout, housing one chart view as direct descendant.

The top-level layout may be initialized to be a *grid layout* of any positive size, and the more specialized *horizontal* and *vertical* layouts. Each layout contains *layout cells*, which can either hold one *child layout*, a chart view (containing one chart), or a *spacer item*. The top-level view, layouts, and chart views, as well as additional visual elements use a box model akin to the *CSS box model*. To allow for more customizable visual arrangement of nodes inside layouts, layout cells can be merged (and unmerged). Additionally, each layout cell has a content *alignment mode*, aligning the cell's contents to specific *edges* (e.g. *top*, *bottom*, ...) or *positions* (*center,* ...) inside the cell. Furthermore, spacer items can be inserted which have a fixed size to allow for custom cell spacing inside any layout.

Axes

Each chart can hold a (theoretically) unlimited number of *axes*. There are four axis positions inside each chart view: *top*, *left*, *bottom*, and *right*. Each position can hold multiple axes. Axes, by default, are aligned to the corners of the orthogonal axes, forming a rectangular plot area skirted by axes. Axis alignments can be defined in order to align an axis to a specific data-point on an orthogonal axis, facilitating the creation and management of more complex axis configurations such as four-quadrant charts, and more. Axes consist of a series of data-points, aligned to either a horizontal or vertical (hair-)line. Data-points can be arbitrarily spaced and support automatic axes modes for numeric data-points (*linear*, *exponential*, *logarithmic*, ...).

Supplemental Chart Elements

Kiko supports a host of additional chart elements intended to enhance the appeal and information density of the data visualization. The main supplemental chart elements are *legends*, and *titles*. Legends can be placed inside or outside the axes area. There can be multiple legends, and legends can have custom *content arrangements*. Titles are text areas, usually (but not necessarily) placed at the top edge of the chart view. They can, like all other chart elements, be added to any area in any layout. Other chart elements allow the inclusion of vector- or raster-based graphics into the view. These resources can optionally be embedded into the output file if one is generated. Apart from more complex chart elements, simple chart elements like rectangles, n-polygons, paths, etc. can be added to the chart view to facilitate data visualization. There is also the option to add a label, optionally with an icon or path, to any empty layout cell. This should help in the creation of more complex or split legends.

Output Compatibility

When SVG is chosen as the output format, the library can generate code in various SVG versions, based on the feature-set required. This is to make sure that the files are usable across platforms, even those that only offer limited SVG support. To do that, there will be presets to select from (SVG versions) and more fine-grained control that allows toggling certain SVG tags. The result of the disabled tags is then simulated by still enabled tags. If all tags capable of simulating a generated element are disabled, a warning is produced prompting the user to review the settings.

Input Methods

By default, data is supplied via a programmatic API, abstracting away internal mechanisms specific to a certain chart type. This way, modifications can be made without substantially changing the API for each chart type, making it easier to use and less complex at a user level. Outside of programmatic data supply, the library can process various simple data distribution formats such as CSV, XML, or JSON for automatic data feeding.