Geomajas graphics project

Geomajas Developers and Geosparc

Geomajas graphics project

by Geomajas Developers and Geosparc

1.0.0-M5

Table of Contents

List of Figures

Chapter 1. Introduction

The Geomajas graphics Project is a stand-alone project under the Geomajas banner.

1. Purpose

The Geomajas Graphics project will provide drawing capabilities (SVG, VML) in support of advanced client side redlining, annotation or mark-ups. Basic use cases are dragging resizing, labeling, styling and CRUD. The idea is to create a graphics library that does not necessarily depend on the Geomajas client, but draws directly into SVG or VML. By doing so we could create a general purpose graphics library that might be useful outside Geomajas or even the GIS world as well. The project is will therefore not depend on any a priori geographical constructs like coordinate spaces or transformations. In Geomajas speak, the project focusses on annotating objects in screen space, not in world space (although the latter will emerge from world referencing the screen objects, but this is considered a separate stage).

The idea is to support different types of annotations and interaction mechanisms, some of which are:

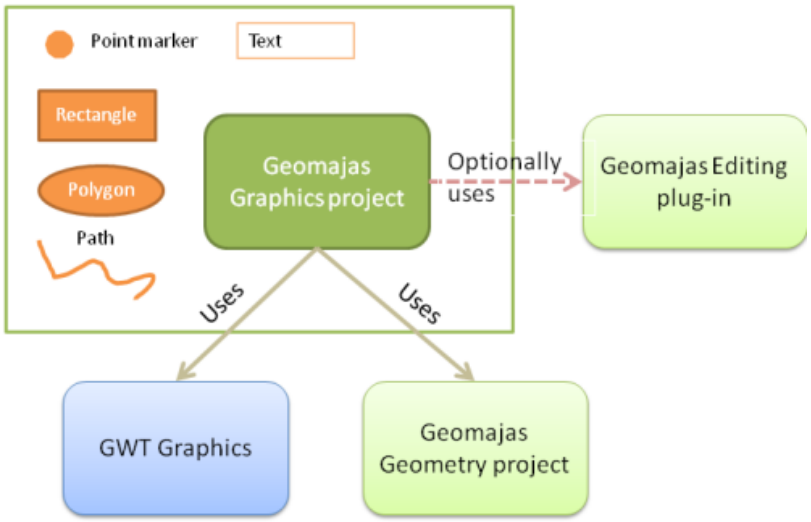
- graphical objects of varying nature, primarily based on what is available in SVG: points/markers, rectangles, ellipses, paths, lines, polygons, images, text
- object attributes like color, font, line style, opacity
- associations of text with objects, also known as labeling
- specific objects like notes, callouts, popups
- basic operations like adding, deleting, dragging, resizing, moving back/front
- advanced operations like styling, coordinate-by-coordinate editing, uploading images, grouping

The goal is to achieve the equivalent of typical drawing applications like PowerPoint or OpenOffice Draw, etc... but with a focus on annotating other objects (think geographical features or imagery).

2. Relationship with other Geomajas projects

The project will of course smoothly integrate with the Geomajas Client and, in combination with the Geometry Editing plug-in, you should be able to reuse the intuïtive, yet powerful ways of changing the Path (add/remove intermediate points, etc.). The following picture illustrates the relationship of the Graphics project with other projects:

Figure 1.1. Project interrelationship



The project depends on the GWT Graphics project for rendering and the Geometry project for simple geometry concepts and geometrical calculations. The Geomajas Editing plugin will be extended with an extra module that hooks up the geometry editor to the graphical object controller framework (see architecture), making sure that the Graphics project itself stays light-weight and independent of the more complex editing mechanisms defined by the Editing plugin.

3. Requirements for the first iteration

The initial version of the project will support the following object types:

- Rectangle
- Ellipse (including circles)
- Image
- Path

The following operations will be available:

- Initial object creation
- Dragging/resizing
- Labeling
- Deleting
- Undo/redo
- Moving objects back and front
- Object selection
- Object persistence: preferably standards, through SVG

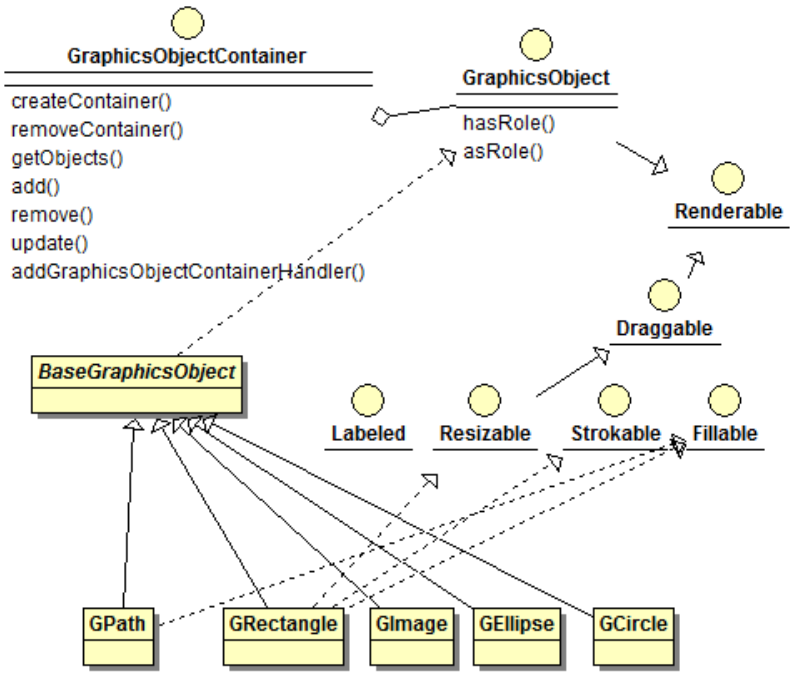
4. Basic concepts and architecture

A `GraphicsObject` is an object that can be rendered in a `GraphicsObjectContainer`. The container supports addition/removal and update of objects and sends out events on the

`EventBus` for each of these operations. The GraphicsObjectContainer wraps the "physical" `VectorObjectContainer` that implements the rendering of the objects in SVG or VML.

A `GraphicsObject` supports different types of roles, depending on the context and the operation that the object is subjected to. The following roles can be distinguished: `Draggable`, `Resizable`, `Fillable`, `Strokable`, `Labeled` and so on. Possible `GraphicsObject` implementations are `GRectangle`, `GEllipse`, `GImage`, etc...Graphics objects usually do not directly implement these roles but use role-composition instead, making them dynamically extensible. Graphics objects can be rendered by supporting the `Renderable` role.

Figure 1.2. Graphical objects and roles

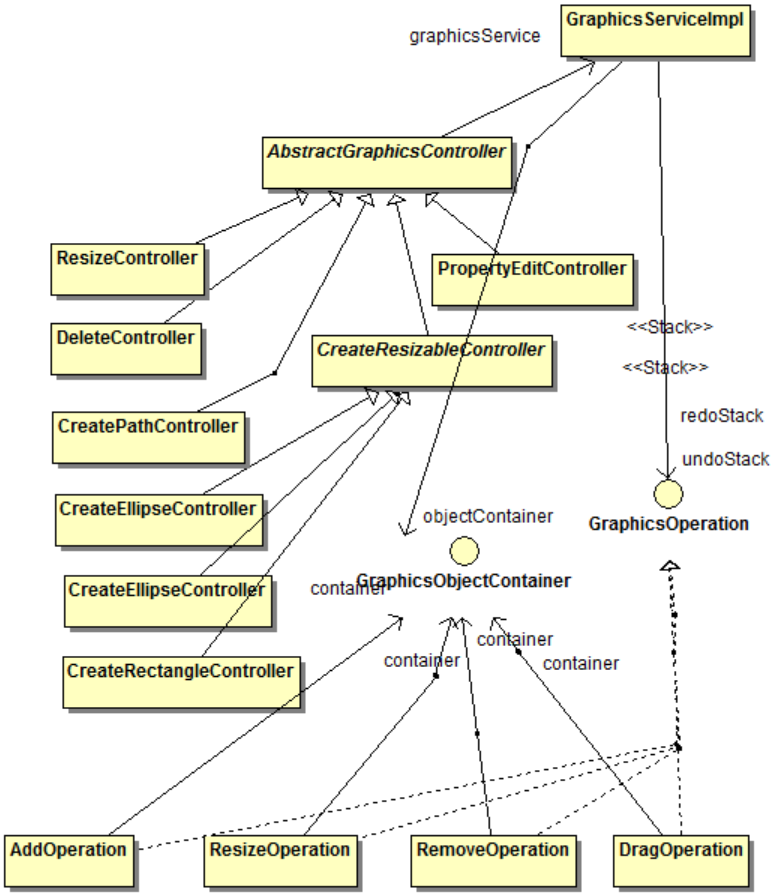


A `GraphicsObjectController` is responsible for interpreting user events and converting them into operations that will eventually be applied to the graphics objects by the `GraphicsService` (see further). The controller has access to the `VectorObjectContainer` to render helper objects like handlers (for resizing), drag lines (while creating a path) or drag masks (while dragging) and can register for whatever mouse events or key events it is interested in. Possible `GraphicsObjectController` implementations are `DragAndResizeController`, `DeleteController`, `EditGeometryController`, `PropertyEditController`, etc...Controllers act on one object only (controller-per-instance or a single controller for all instances with varying object scope) and can be activated and deactivated.

Activation can be achieved by defining a special meta-controller that activates the controller(s) for a specific object. Initially a single-selection meta-controller that activates all controllers at once by clicking on the object should be sufficient. All controllers that act on a single object should be able to coexist. This implies that controllers do not register for all `VectorObjectContainer` events but instead delegate their work to specific helper objects like resize handlers, drag masks or special icons (e.g. to activate a specific editor).

The `GraphicsService` is a manager class that manages the `GraphicsObjectContainer`, instantiates the `GraphicsObjectController` objects (and acts a registry for their factories) and executes `GraphicsOperations` on the `GraphicsObject` instances. A `GraphicsOperation` is undoable and redoable by keeping sufficient information to restore the object state to what it was before the operation.

Figure 1.3. Graphics service and controllers



Chapter 2. Configuration

In order to use this project through Maven, use the following dependency:

```
<dependency>
    <groupId>org.geomajas.project</groupId>
    <artifactId>geomajas-project-graphics</artifactId>
    <version>1.0.0-M5</version>
</dependency>
```

Note

For those not using Maven, this project depends on the Geomajas annotation project.

Chapter 3. How-to

This section covers some examples of how to use the services that are provided on top of the graphics model.

TODO