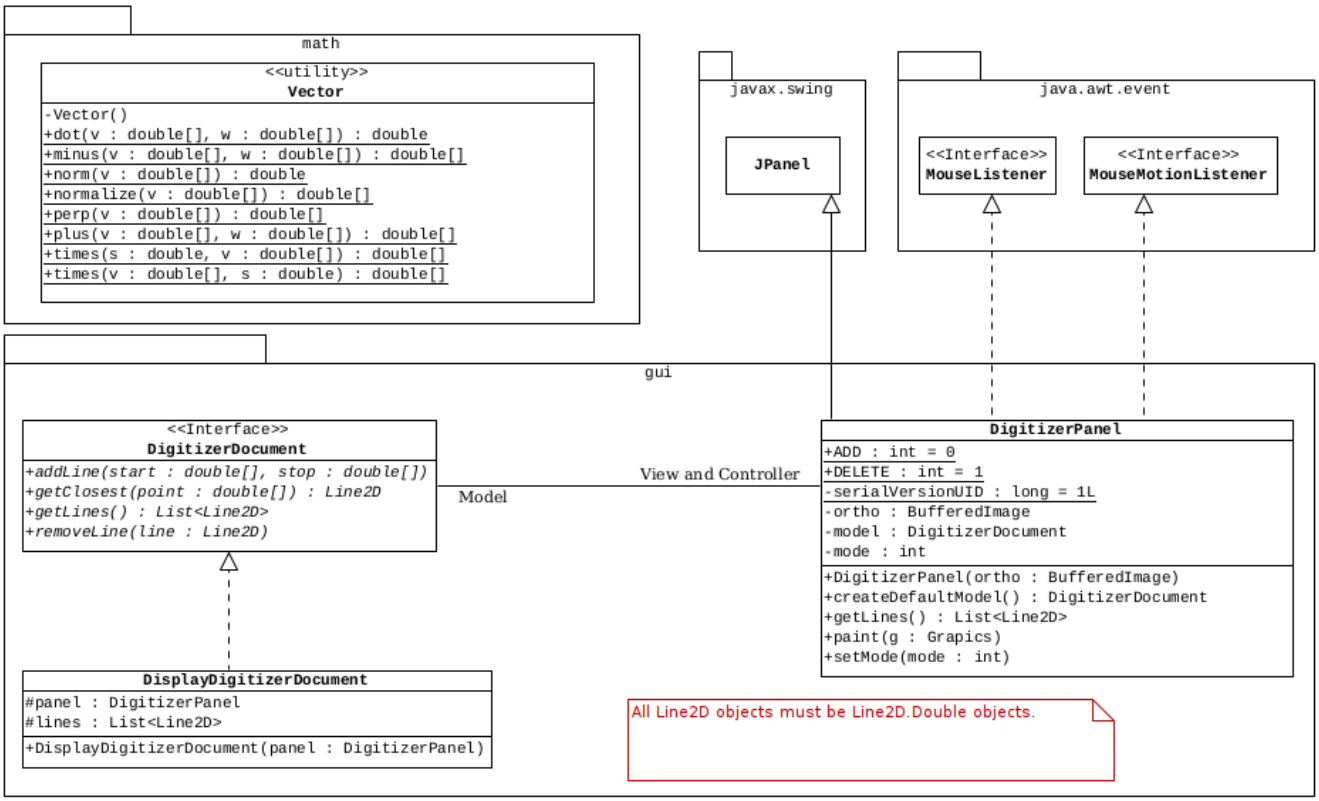
Digitizer v1

Purpose

Version 1 of the Digitizer is a collection of components that can be used to enter coordinates. This version will only support display coordinates (i.e., linear coordinates in which the horizontal coordinate increases from left to right and the vertical coordinate increases from top to bottom). Future versions will also support Euclidean coordinates and geographic coordinates (i.e., longitude and latitude).

Design

The design of the system is summarized in the following UML class diagram:



Note that the details of the components that are part of Java's Swing API have been omitted from this diagram.

Specifications

This section contains design specifications for the components above.

The Vector Class

The Vector class is a utility class that can be used to perform operations on 2D vectors (i.e., arrays with 2 elements).

The DigitizerDocument Interface

A `DigitizingDocument` is the model component (in the sense of the Model-View-Controller Pattern) of the system. The functionality of most of the methods should be apparent from their names. All `Line2D` objects must be instances of `Line2D.Double`.

The `getClosest()` method must return the "approximately closest" line segment to the given point. The approximate distance between a point and a line segment is the minimum distance between the point and the two end points of the segment. (The definition of closest will be improved in future versions of this component.)

The DisplayDigitizerDocument Class

A `DisplayDigitizerDocument` is a `DigitizerDocument` that uses display coordinates (in which the horizontal coordinate increases from left to right and the vertical coordinate increases from top to bottom).

The DigitizerPanel Class

The `DigitizerPanel` is the view and controller component (in the sense of the Model-View-Controller Pattern).

The `paint()` method must:

1. Clear the component (e.g., by invoking `super.paint()`).
2. Draw the ortho photo that was passed to the constructor at the coordinate (0,0).
3. Draw all of the line segment in the `DigitizerDocument` (in `Color.GREEN`).
4. Draw the line segment that is currently being drawn (in `Color.YELLOW`).

The methods in the `MouseListener` and `MouseMotionListener` interfaces must enable the user to add and remove line segments.

- When in "add mode", the user must be able to add a line segment by: clicking the mouse, dragging the mouse, and releasing the mouse. The component must show the progress of this process (which is referred to as the current line segment).
- When in "delete mode", the user must be able to delete the closest line segment by clicking the mouse and must be able to cancel the deletion process by dragging the mouse before releasing it.

You must write the code necessary to provide this functionality. You may need to add both private attributes and private methods to accomplish this.

The "add mode" is illustrated in the following screenshot: