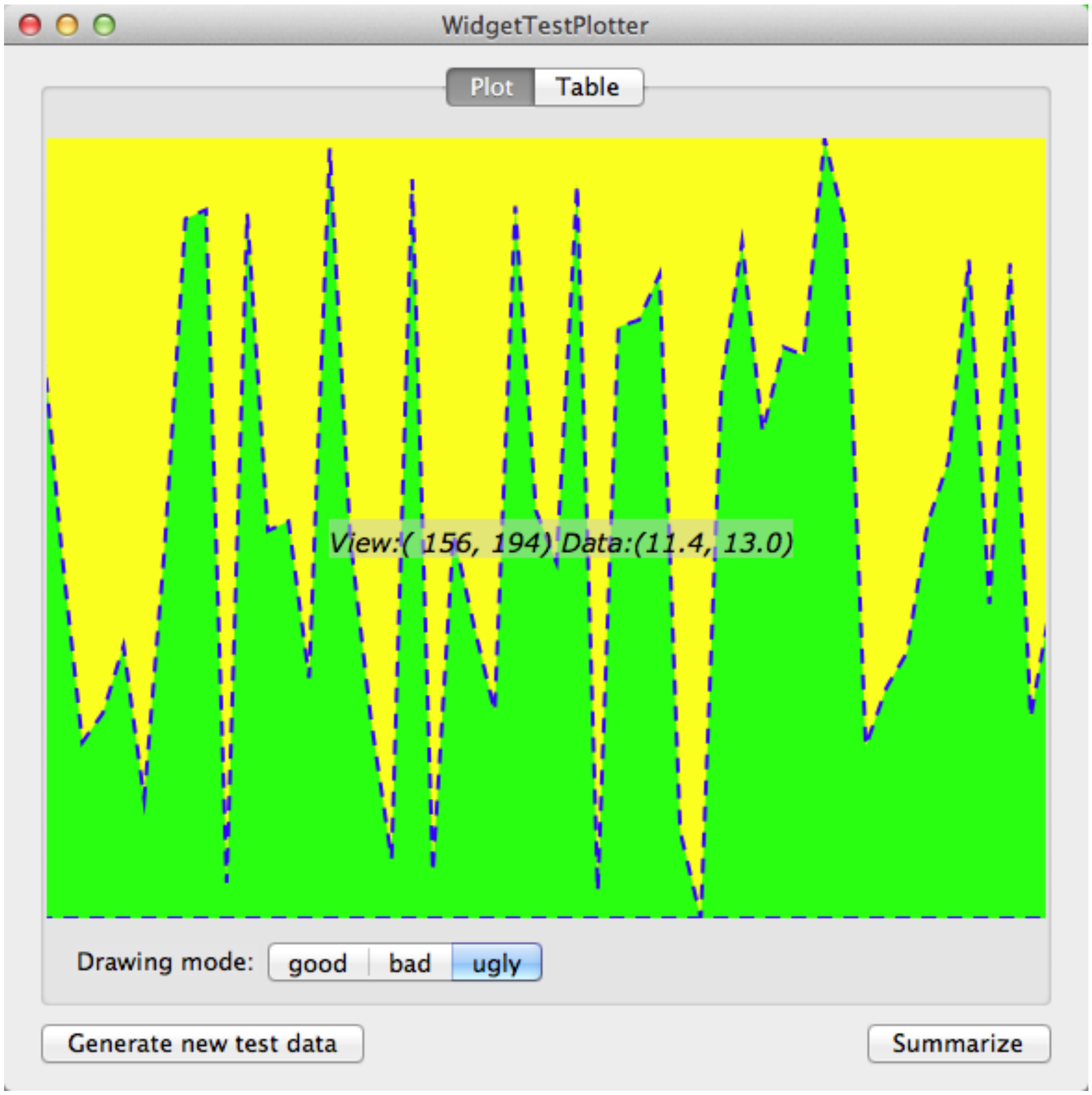
Voltage/Time Test Plotter

Scenario: you have a piece of test equipment that analyzes a widget and provides a time series of measurements of the voltage observed on the widget. The time series is stored in the WidgetTester class as NSMutableArray of WidgetTestObservationPoint instances. See the .h files for those two classes for the interface.

Your first task is to graph the points. Provide multiple rendering styles. Rendering styles can have different background and foreground colors, different line width, and whatever other stylings you'd like to tinker with.



Once you have the basic drawing implemented correctly, add these features:

Track mouse movements across the WidgetTestRunView. Draw attributed text (programmatically specify a font, style, point size) to display mouse position in view coordinates (NSAttributedString). Draw the text at the mouse position.

When the Summarize button is clicked, place the WidgetTester's -summary into the system general pasteboard (the normal copy/paste buffer). Test that you can paste that summary into a TextEdit document.

Optional challenges:

- display both the view coordinates and the data coordinates for the mouse position.
- embed your WidgetTestRunView in an NSScrollView. Vary the zoom factor, using a suitable UI.
- modify the mouse coordinate drawing attributes based on keyboard modifier state (control, option, shift, and command keys). For instance, you might choose to use italics if the Shift key is down, and use a completely different font and color scheme if the Control key is down.

Starting Points: you have your choice of two starting points.

I have provided a WidgetTester class. It generates simulated test data, and can also report the max/min sensor values observed, the start/stop times, and a text summary of the run. This class uses WidgetTestObservationPoint instances to store its data. I have also provided an entire project, including these classes. The segmented controller and Generate New Test Data button each send -setNeedsDisplay:YES to the custom NSView subclass, and data is generated on demand. The project includes unit tests for the WidgetTester class. An alternate View (using NSTableView) for the widget Model is also done for you. I used autolayout for this XIB instead of springs and struts.

If you want to make this exercise as simple as possible, start with WidgetTestPlotter_starter.zip. All you need to do then is write the -drawRect: method on WidgetTestRunView, modify the summarize action, and implement the mouse event handling.

If you want more of a challenge, you can start with only the WidgetTester and WidgetTestObservationPoint classes, and write the rest of the project yourself.

Note that the range of the simulated test data does not line up with typical graphics coordinates, so you'll have to do some projecting to get the data to fit the window and to appear full scale. I strongly recommend that you write two methods, one to project data points (time/voltage) into graphics space (x/y) and the other to project graphics coordinates into data space.

Grading focus:
path drawn multiple ways, full scale
mouse info is displayed
summary text is in copy/paste buffer

Don't panic that we haven't covered mouse events or copy/paste yet. Those are first on the list for next week.