15.12

PLUG AND PLAY SCIENTIFIC GRAPHICS WITH SGT

Donald W. Denbo*

Joint Institute for the Study of Ocean and Atmosphere, University of Washington, Seattle, WA

1. INTRODUCTION

The Scientific Graphics Toolkit (SGT) (Denbo, 2001; Denbo, 2000; Denbo, 1999; Denbo, 1997) is a library of java graphics classes which facilitates the development of platform independent Java applications and web applets to produce highly interactive, flexible, publication quality, object-oriented graphics of scientific data. Plot types supported include contour, line, point, and vector, plot features include user settable or automatically scaled axes, sophisticated, automatically self-scaling time axes, mouse-movable objects (labels, line keys, icons), customizable objects, automatic generation of legends to explain the data being displayed. Although use of SGT requires detailed expertise in object oriented java programming, it has proven popular with these specialists in developing graphics applications. Since March 2000, there have been over 11,000 downloads of SGT source, class, and javadoc jar files (Java archives) by 3955 unique sites in 65 countries, and was featured in JavaWorld magazine in March 2001.

We are making this powerful graphics toolkit significantly more accessible to a far wider community of non-expert users by creating easy-to-use SGT JavaBeans. SGT JavaBeans is easier for an expert programmer to utilize, and moreover, it gives beginner and intermediate Java developers access to these powerful scientific graphics in their Java applets, applications, and servlets, without the steep learning-curve required to use SGT directly. See http://www.epic.noaa.gov/java/sgt for more information and download links.

2. DESIGN GOALS

We have begun the task of creating easy-to-use SGT JavaBeans from the Scientific Graphics Toolkit (Figure 1). SGT JavaBeans are be easier for an expert programmer to utilize, and moreover, gives beginner and intermediate Java developers access to these powerful scientific graphics in their Java applets, applications, and servlets, without the steep learning-curve required to use SGT directly. Developing JavaBean components from SGT classes involves adding support for property management, introspection, event handling, persistence, and application builders. Property management involves handling all interactions relating to the state of the bean (a JavaBean class) and constitutes the data part of a bean's structure. The introspection facilities are the mechanism by which components make their internal structure readily available to the outside world. The event handling facilities specify an event-driven architecture that defines the interactions among beans and applications. The persistence facilities in the JavaBean API specify the mechanism by which beans are stored and retrieved to a specific internal state and appearance. Finally, the JavaBeans API provides the facilities necessary to edit and manipulate beans using visual application builder tools.

We will include “wizard” support for incorporating and customizing the scientific graphics classes from an Integrated Development Environment (IDE), Rapid Application Development (RAD) tool, or the freely available Bean Builder (see below). This means that a novice programmer will be able to create applications by using a simple wizard user interface, reducing the amount of detailed knowledge required to effectively use the scientific graphics Java Beans. A wizard is a user interface that takes a user through a step-by-step process to customize a bean. The “wizard” will be used for incorporating and customizing the scientific graphics classes. By providing wizards, we reduce the amount of detailed knowledge required to effectively use the scientific graphics beans. With the wizard, SGT JavaBeans becomes “plug and play” java graphics.

JavaBeans were chosen to implement SGT components because they are a Java standard, platform independent, and supported by most Java IDE and RAD software. Presently Java is the only programming environment that is both cross platform and has multiple vendors providing development software.

3. JAVABEANS OVERVIEW[1]

JavaBeans brings component technology to the Java platform. With the JavaBeans API you can create reusable, platform-independent components. Using JavaBeans-compliant application builder tools, you can combine these components into applets, applications, or composite components. JavaBean components are known as Beans.

Beans expose their features (for example, public methods and events) to builder tools because feature names adhere to specific design patterns. A “Java-

* *Corresponding author address:* Donald W. Denbo, NOAA/PMEL/OCRD, 7600 Sand Point Way NE, Seattle, WA 98115; e-mail: dwd@pmel.noaa.gov

1. Material excerpted and edited from http://java.sun.com/docs/books/tutorial/javabeans/index.html

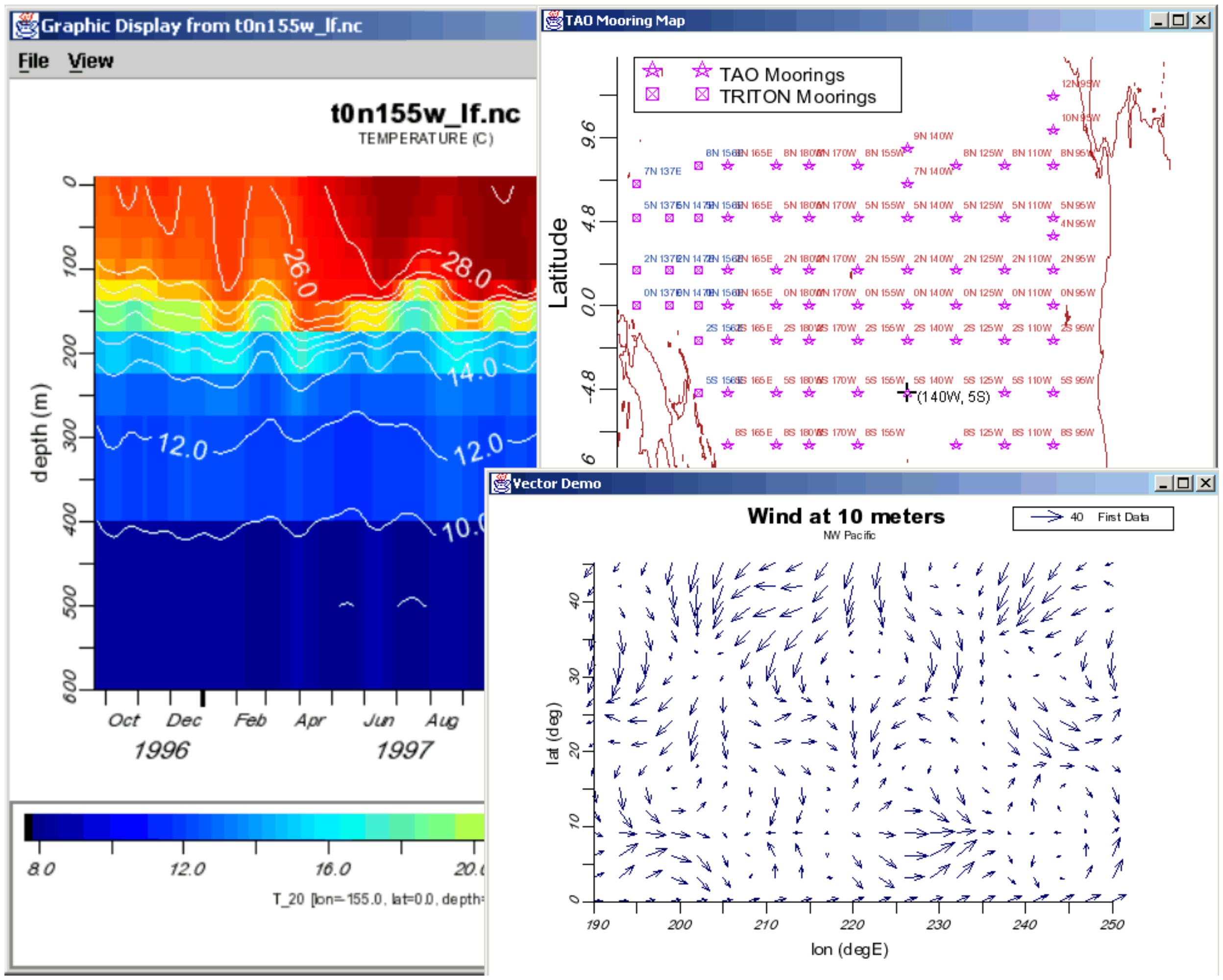

Figure 1. Examples of SGT capabilities.

Beans-enabled" builder tool can then examine the Bean's patterns, discern its features, and expose those features for visual manipulation. A builder tool maintains Beans in a palette or toolbox. You can select a Bean from the toolbox, drop it into a form, modify it's appearance and behavior, define its interaction with other Beans, and compose it and other Beans into an applet, application, or new Bean. All this can be done without writing a line of code.

The following list briefly describes key Bean concepts.

- Builder tools discover a Bean's features (that is, its properties, methods, and events) by a process known as introspection.
- Properties are a Bean's appearance and behavior characteristics that can be changed at design time.
- Beans expose properties so they can be customized at design time.
- Beans use events to communicate with other Beans. A Bean that wants to receive events (a listener Bean) registers its interest with the Bean that fires the event (a source Bean).
- Persistence enables Beans to save and restore their state.
- A Bean's methods are standard Java methods, and can be called from other Beans or a scripting environment. By default all public methods are exported.

Although Beans are designed to be understood by builder tools, all key APIs, including support for events, properties, and persistence, have been designed to be easily read and understood by human programmers as well.

Bean Builder. The Bean Builder is a freely available tool which allows the visual assembly of an application by instantiating and setting the properties of components based on the JavaBeans component architecture. The dynamic behavior of the application is specified by "wiring" relationships that represent events handlers and method calls between the objects in an application. The state of this application is saved to and restored from an XML format. An application can be constructed using the Bean Builder without having to write a line of source code.

4. FUTURE DIRECTIONS

We will also include “wizard” support for incorporating and customizing the SGT Beans from a IDE or RAD tool. This means that a novice programmer will be able to create applications by using a simple wizard user interface, reducing the amount of detailed knowledge required to effectively use the SGT Beans. With the wizard, SGT Beans become “plug and play” java graphics.

Acknowledgment. This publication was supported by the Joint Institute for the Study of the Atmosphere and Ocean (JISAO) under NOAA Cooperative Agreement #NA67RJ0155, Contribution #946. PMEL contribution 2522. The views expressed herein are those of the author(s) and do not necessarily reflect the views of NOAA or any of its subagencies. This work was funded by NOAA’s HPCC program.

5. REFERENCES

Denbo, D.W., 2001. Interactive graphics toolkit for Java applications and web applets. *17th Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, AMS, 14-19 January 2001, Albuquerque, NM, 372-375.

Denbo, D.W., 2000. The Scientific Graphics Toolkit. *Proceedings of Oceans '99 MTS/IEEE Conference*, 13-16 September, Seattle, WA (on CD).

Denbo, D.W., 1999. Using Java graphics to display ocean observations in NOAAServer. In *15th International Conference on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology*, AMS, Dallas, TX, 10-15 January 1999, 442-444.

Denbo, D.W., 1997. NOAAServer Graphics Engine Architecture. Presented at *NOAA WebShop97*, October 22-23, 1997, Silver Spring, Maryland.