1999 NASA/ASEE Summer Faculty Fellowship Program

John F. Kennedy Space Center
University of Central Florida

A Visual Editor in Java for Jview

Ryan Stansifer
Associate Professor
Computer Science
Florida Institute of Technology

KSC Colleague: Steve Beltz (Dynacs)

Abstract

In this project we continued the development of a visual editor in the Java programming language to create screens on which to display real-time data. The data comes from the numerous systems monitoring the operation of the space shuttle while on the ground and in space, and from the many tests of subsystems. The data can be displayed on any computer platform running a Java-enabled World Wide Web (WWW) browser and connected to the Internet. Previously a special-purpose program had been written to display data on emulations of character-based display screens used for many years at NASA. The goal now is to display bit-mapped screens created by a visual editor. We report here on the visual editor that creates the display screens.

This project continues the work we had done previously. Previously we had followed the design of the "beanbox," a prototype visual editor created by Sun Microsystems. We abandoned this approach and implemented a prototype using a more direct approach. In addition, our prototype is based on newly released Java 2 graphical user interface (GUI) libraries. The result has been a visually more appealing appearance and a more robust application.

1 Introduction

During the operation of the space shuttle, sensors are monitoring many of the subsystems. This data goes to special hardware at the Launch Control Center (LCC) known as the Common Data Buffer (CDBF). Lots of data is collected, approximately 30,000 measurements. These measurements are continually changing—some of them can change rapidly at certain times. The data is used in monitoring the operation of the shuttle and in analyzing subsystems for safety, performance, technological improvements, etc. Each individual measurement is given a short tag called a function designator (FD) to identify it.

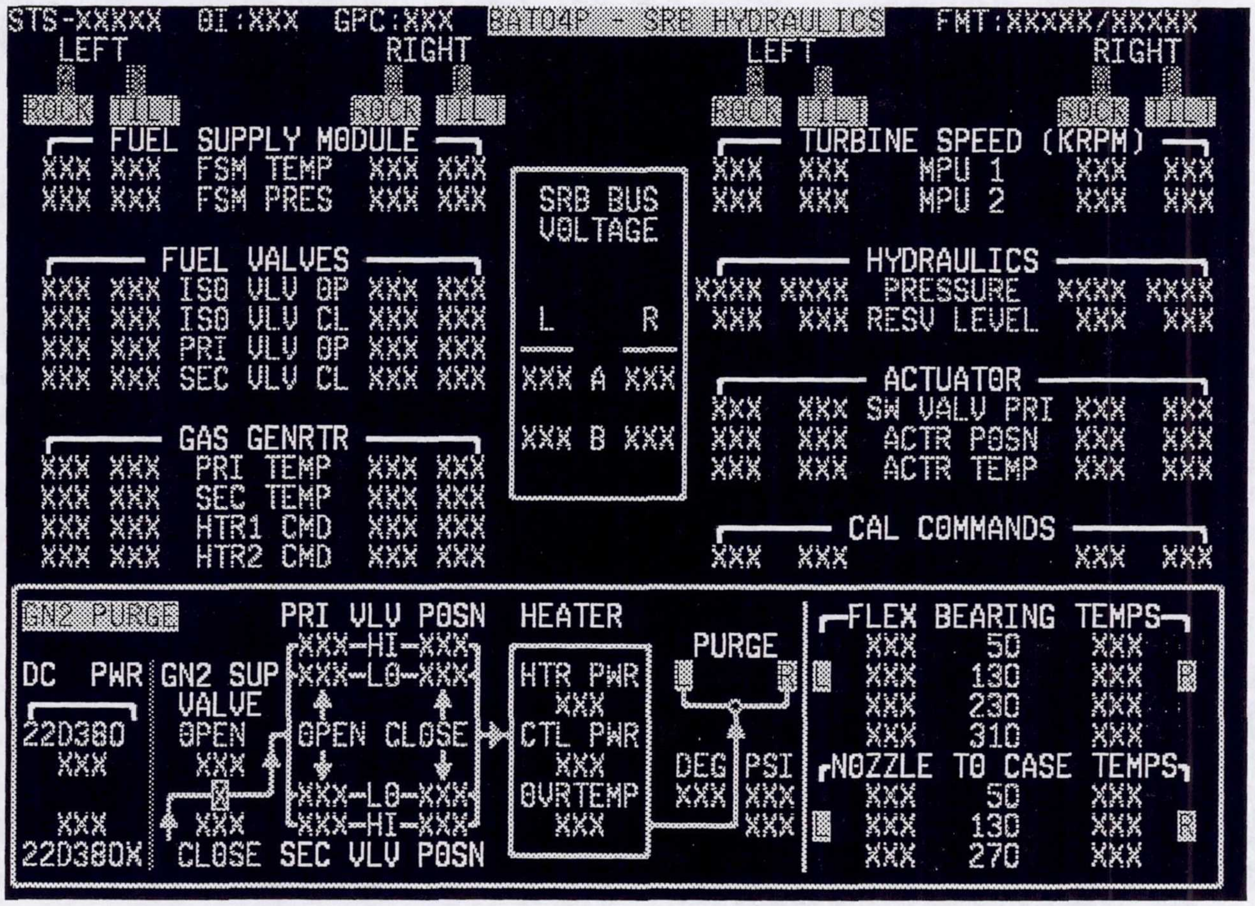
This Real-time data is monitored on system engineering consoles of then the Control Checkout and Monitor System (CCMS) in the Launch Control Center (LCC). The PC GOAL system presents the same data to a wider audience in a format closely resembling the consoles of the CCMS. The CCP (CDBF Communications Processor) scans the memory of the CDMF every 2 seconds and broadcasts the data on the LPS Operations Net (LON). This data (and other data, e.g., FIFO) is relayed to the PC GOAL stations. The PC GOAL stations are PCs with network hardware running DOS and the PC GOAL software.

The PC GOAL system presents shuttle data on schematic-like screens described by character-oriented files known as DSP files. Figure 1 shows one of the hundreds of PC GOAL display screens (no data is been displayed, the character 'X' is filling the character positions that would be occupied by numerals). Each shuttle mission requires substantial effort to organize the CDBF, distribute the DSP files, etc.

2 Jview

The Jview project is motivated by the PC GOAL system to display real-time shuttle data as conveniently and efficiently as possible. The programming language Java was chosen because of the easy of writing both graphical user interface (GUI) code and distributed programs. The Internet is the obvious mechanism to transport the data. The wide-spread availability of browsers for the World Wide Web suggests an obvious user interface for any information system large or small.

Java code can be transmitted as part of the WWW protocol, just like pictures, sound and other data. The extreme interest in Java is caused by the ability of browsers to execute the Java code. These Java programs transported across the Internet and executed locally by a WWW browser are called applets. Applets add interaction to otherwise static WWW documents. The Jview project uses applets to form a connection to a Java data manager that relays the real-time data to the applet.



```
STS-XXXXX  OI:XXX  GPC:XXX  BAT04P - SRB HYDRAULICS   FMT:XXXXX/XXXXX
   LEFT                   RIGHT                  LEFT                   RIGHT
ROCK TILT              ROCK TILT              ROCK TILT              ROCK TILT
 ┌─ FUEL SUPPLY MODULE ─┐                      ┌─ TURBINE SPEED (KRPM) ─┐
XXX XXX  FSM TEMP  XXX XXX    ┌──────────┐    XXX  XXX    MPU 1    XXX  XXX
XXX XXX  FSM PRES  XXX XXX    │ SRB BUS  │    XXX  XXX    MPU 2    XXX  XXX
                              │ VOLTAGE  │
 ┌──── FUEL VALVES ─────┐     │          │   ┌──── HYDRAULICS ─────┐
XXX XXX ISO VLV OP XXX XXX    │          │  XXXX XXXX  PRESSURE  XXXX XXXX
XXX XXX ISO VLV CL XXX XXX    │ L      R │   XXX  XXX RESV LEVEL  XXX  XXX
XXX XXX PRI VLV OP XXX XXX    │ ──    ── │
XXX XXX SEC VLV CL XXX XXX    │XXX A XXX │   ┌───── ACTUATOR ──────┐
                              │          │   XXX  XXX SW VALV PRI XXX  XXX
 ┌──── GAS GENRTR ──────┐     │XXX B XXX │   XXX  XXX  ACTR POSN  XXX  XXX
XXX XXX  PRI TEMP  XXX XXX    │          │   XXX  XXX  ACTR TEMP  XXX  XXX
XXX XXX  SEC TEMP  XXX XXX    └──────────┘
XXX XXX  HTR1 CMD  XXX XXX                   ┌──── CAL COMMANDS ───┐
XXX XXX  HTR2 CMD  XXX XXX                   XXX  XXX             XXX  XXX
┌──────────────────────────────────────────────────────────────────────────┐
│GN2 PURGE        PRI VLV POSN   HEATER            │┌FLEX BEARING TEMPS┐   │
│                 ┌XXX─HI─XXX┐             PURGE   │  XXX      50     XXX  │
│DC  PWR│GN2 SUP  ├XXX─LO─XXX┤  HTR PWR            │L XXX     130     XXX R│
│┌─────┐│ VALVE   │ ↑      ↑ │    XXX              │  XXX     230     XXX  │
│22D380 │ OPEN    │OPEN CLOSE│→ CTL PWR            │  XXX     310     XXX  │
│  XXX  │  XXX    │ ↓      ↓ │    XXX     DEG PSI  │┌NOZZLE TO CASE TEMPS┐ │
│       │         ├XXX─LO─XXX┤  OVRTEMP   XXX XXX  │  XXX      50     XXX  │
│  XXX  │  XXX    └XXX─HI─XXX┘    XXX         XXX  │L XXX     130     XXX R│
│22D380X│ CLOSE SEC VLV POSN                       │  XXX     270     XXX  │
└──────────────────────────────────────────────────────────────────────────┘
```

Figure 1: One of PC GOAL's display screen

2.1 Java applets

The key advantage of using Java applets is that after the applet makes a connection to a data server only the data is transmitted across the network. This is superior to constantly transmitting updated pictures (bit maps) as would be required in other approaches, e.g., common gateway interface (CGI), server push, etc.

The advantages in using Java are even greater than the technical merits suggest. Administratively, the operation of a real-time data service using Java is much better. The Java applet is written once and executed remotely; no porting has to be done. Also, the latest version of the applet is always distributed to the user; there is no version control problem. Finally, the operation of the service is easy as browsers are ubiquitous; no training is required.

2.2 Java Programming Language

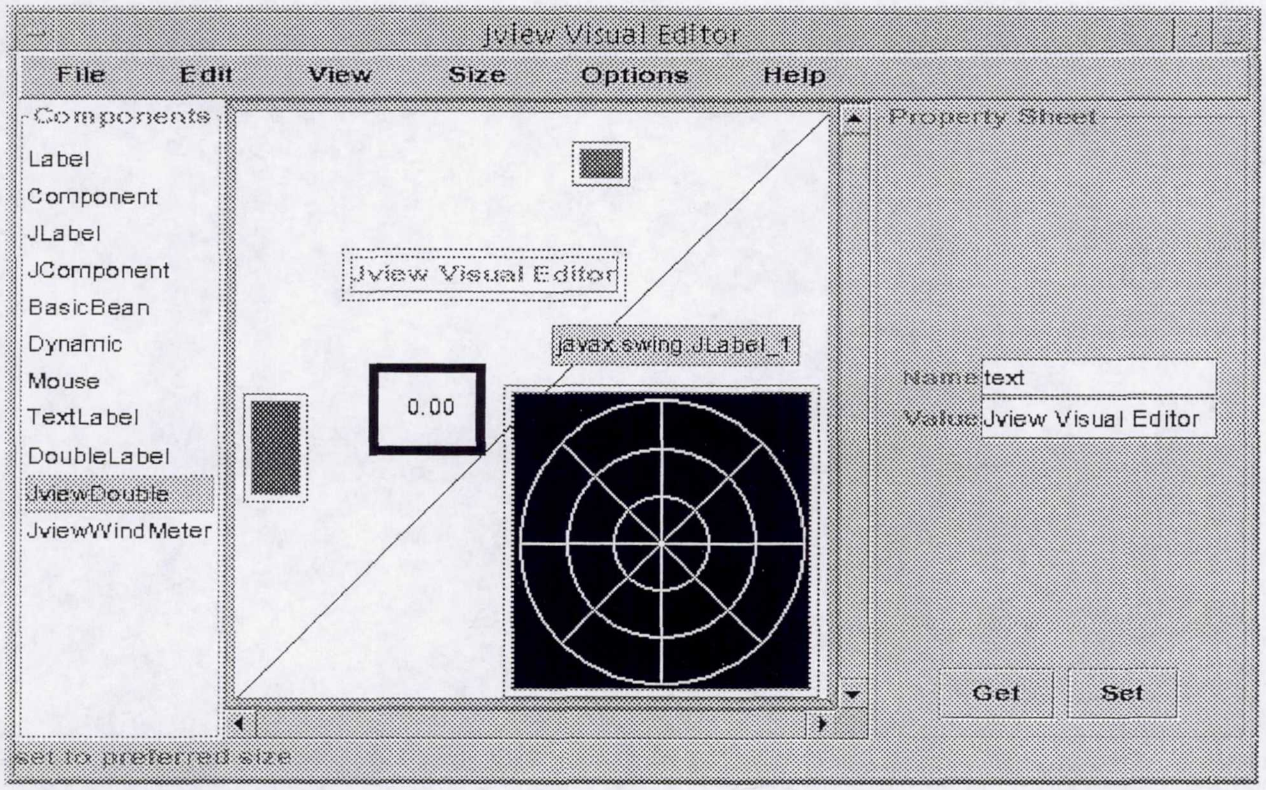
The Java programming language introduced by Sun Microsystems a mere five years ago also provides technological advantages. The good network library has been important to to Jview, as has the good GUI library. This latest version of this GUI library, known as Swing, was used in the visual editor. This has made it easy to produce an application with a pleasing appearance and great functionality. On the other hand, the performance of Swing is noticeably worse than the less sophisticated AWT library used in the previous version of the visual editor.

2.3 Jview Operation

The Jview system has been in operation for about two years. Since it emulates the existing PC GOAL display screens it has been easy for users to use the new system. All the display screens have the same appearance as the PC GOAL display screens, for example Figure 1. But advances in computer systems makes it reasonable to expect more sophisticated displays. It is possible to display the data more realistically with dynamic, bitmapped components: tanks can appear to be filling, analog gauges can be simulated, sound can incorporated, etc. Schematic diagrams can be more detailed.

To create these more sophisticated displays we have designed a visual editor. From an extensible palette of components the user composes a new graphical display for the data. Pictures and schematics (perhaps made from other tools like Adobe's PhotoShop) can be imported. Once a screen has been created, the data viewer displays it as well as updates the components with the individual data values as the data comes in.

It is important to emphasize that the visual editor puts the components together and does not create the components. At the moment, we have few entertaining components



Name text
Value Jview Visual Editor

Figure 2: View of the editor

to use with the editor. It is likewise important to emphasize that the components can be created at anytime and the editor can use them without any change to the editor.

3 Visual Editor

The screen shot of the visual editor is shown in Figure 2. The list of available components appears on the left side of the workspace. In the workspace are five components. One is a `TextLabel` component for a label. Another is the specially created `JViewWindMeter` component. A third is the `JviewDouble` component for numbers; it is currently selected. It has the solid border around it. The current component always has a property editor in a separate little window called "property sheet." In this window the individual values for each property of the component can be set. No real data is reaching the component in the editor. But the display designer may want to see how the component reacts to some particular data

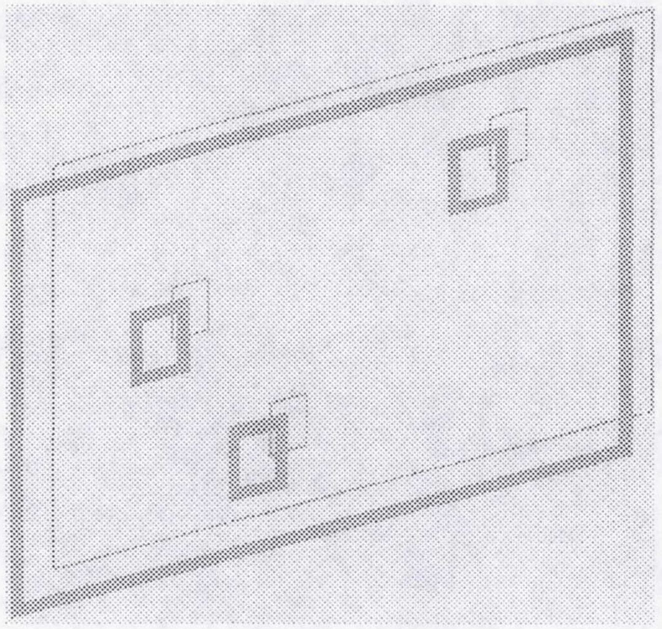


Figure 3: Two layered approach

value. The component may round the value to the nearest degree. Other components might turn a different color for some values, etc.

Good use was made of the Java Swing component `LayeredPane`. Using `LayeredPane` allowed the workspace to be composed of exactly the components that would be used in the display of the finished display screen. The extra effects and functionality of the editor were confined to a second layer that was placed in front. Figure 3 suggests the logical placement of the GUI components in the Jview editor. Displaying borders or capturing mouse events is accomplished by the front-most layer. In it a component (usually invisible) is placed in front of the actually display component.

Components can be widgets from either the AWT library or the new Java Swing library. These components are both "Java Beans." This means they follow certain programming conventions. For instance, `get` and `set` methods. For each property, say, wind speed, there is a method, say, `getWindSpeed` that returns the value of the private variable, and a method `setWindSpeed` that sets the value. My using this naming scheme, it is possible for the editor to manipulate a completely unknown component. The component does not have to be compiled with the editor at all and can be created long after the editor is deployed.

There is a limit to the type of components suitable for use with Jview. All Jview components access the data through FD's and implement an update method that responds appropriately to new data values for the FD's.

4 Acknowledgments

I gratefully acknowledge the support of the DAP group of CLCS headed by Coleman Dugger (NASA) especially the Jview team: Steve Beltz (Dynacs), and Mark Long (Dynacs). Also, deserving thanks are Brad Neal (University of Central Florida) and Tom Beever, president of Netlander. A special thanks is due to Ray Hosler (University of Central Florida), Jane Hodges (NASA), and Greg Buckingham (NASA), who ran the NASA/ASEE Summer Faculty Fellowship Program at the Kennedy Space Center, for being friendly, efficient, and enthusiastic.

The original impetus of this project in 1996 grew out of collaboration with Peter Engrand (NASA) and Charlie Goodrich (Dynacs).

References

[1] Patrick Chan and Rosanna Lee. *The Java Class Libraries: Second Edition, Volume 2.* Addison-Wesley, Reading, Massachusetts, 1998.

[2] Patrick Chan, Rosanna Lee, and Doublas Kramer. *The Java Class Libraries: Second Edition, Volume 1.* Addison-Wesley, Reading, Massachusetts, 1998.

[3] Robert Eckstein, Marc Loy, and Dave Wood. *Java Swing.* O'Reilly, Sebastopol, California, 1998.

[4] Henri Jubin. *JavaBeans by Example.* Prentice Hall, Upper Saddle River, New Jersey, 1997.

[5] NASA, Kennedy Space Center, Checkout and Launch Control System. CLCS project home page. WWW site at `http://clcs.ksc.nasa.gov/`.

[6] NASA, Kennedy Space Center, Launch Processing System. LPS system software PC GOAL home page. WWW site at `http://lpsweb.ksc.nasa.gov/SDC/PCGOAL/`.

[7] Sun Microsystems, Inc. Javabeans. WWW site at `http://java.sun.com/beans`.