A Java-based Authoring Tool for Developing Power Systems Labware

P. Jayanetti, J. Olcott, J. Johnson, J. Patton
Department of Electrical and Computer Engineering
University of Maine

This paper describes our efforts in developing Java class libraries to provide multimedia authoring capability similar to many high level commercial programming environments while also providing a rich mathematical simulation capability. The tools are being used to develop multimedia based simulation labware to augment junior and senior power systems labs. This authoring system will allow us to easily integrate C++ or Java models of protective relays, three phase transformers, synchronous machines, and other equipment. A World Wide Web page was created that contains additional material showing the user interface. The Web page can be accessed at http://www.eece.maine.edu/Power/Java.

1 Introduction

Recently there has been a great deal of interest in developing multimedia based courseware and labware [1, 2, 3,4, 5,6]. This project involves the creation of a Java based, multimedia power plant simulator, to be used in coordination with a power system lab course [7]. The simulator emulates Bangor Pacific's West Enfield Hydro-Electric Power Plant, located on the Penobscot River in Maine.

Junior engineering students at the University will augment traditional labs with the multimedia power plant simulator. The simulator will enable students to "connect" lab experience using scaled down equipment to real-world power production and control situations.

Initially, we attempted to use commercially available packages to develop the simulator. Most multimedia authoring tools such as Apple's Apple Media Tool, Macromedia Director, and mFactory's mTropolis, do not provide the ability to easily integrate simulation models written in C or C++. Furthermore, most of these tools are written for the Apple Macintosh platform, and problems inevitably arise during porting of code to a PC platform. Many platform and programming options were explored before we chose Java as the development vehicle.

Java not only gives the ability to create a multimedia development environment, it provides the ability to integrate C++ models into the project. Java is platform independent, and because its syntax is similar to C++, it allows easier porting of existing code. Because network capabilities are also built into the Java language, we obtained the added benefit of being able to make portions of the simulator available over our LAN and the Internet.

Two major efforts were required in the development of the authoring environment. The first focused on the Navigator, a class that defines the interaction between all the other classes (and is the primary component the user must write). The Navigator is further described in Section 2.

2 The Multimedia Authoring Tool

Using Java, we created the multimedia authoring tool, mmBrewer. This tool is comprised of a collection of multimedia components all of which begin with the prefix “mm.”

The structure of the mmBrewer is analogous to a story book. Just as a story book contains different pages, mmBrewer contains screens which serve as the framework. Each screen contains images, text, and interactive components capable of initiating a variety of responses. A collection of components comprise a screen, and a collection of screens make up a title or the entire story book.

Class Library

The first objective was to create a multimedia class library. These classes are fairly simple code snippets that access much more detailed Java code, and supply the variables needed to initiate an action. Although Java libraries already exist that provide a range of functionality from basic data types to graphical user interface toolkits, they do not directly facilitate multimedia programming. The classes developed in the mmBrewer package make multimedia programming in Java as quick and easy as nearly any package offered commercially.

The classes of mmBrewer extend other classes which share the same characteristics. In Figure 1, note how the classes extend from one another. With the mmBrewer authoring tool to build from, the simulation labware can be created easily without much knowledge of Java. For example, one line of code can be substituted for pages of code and reused. This saves time and minimizes the amount of redundant code.

Classes are placed within the tree structure based on information which is passed to them. For example, mmImage is a class which contains a screen name, an image file name, and placement coordinates. If a new class is developed which requires an image and placement coordinates, its appropriate location is below mmImage. Furthermore, the new class can contain inputs in addition to those provided by mmImage. This inheritance property is a strong feature of object-oriented programming.

At the root of a string of classes is a superclass. All characteristics of a superclass are inherited by the subclass. For example, a capability built into mmPainter (a subclass of mmImage), such as the recognition of HotSpots (an area which warrants a specific response), will be acquired by all other classes extended from mmpainter. However,

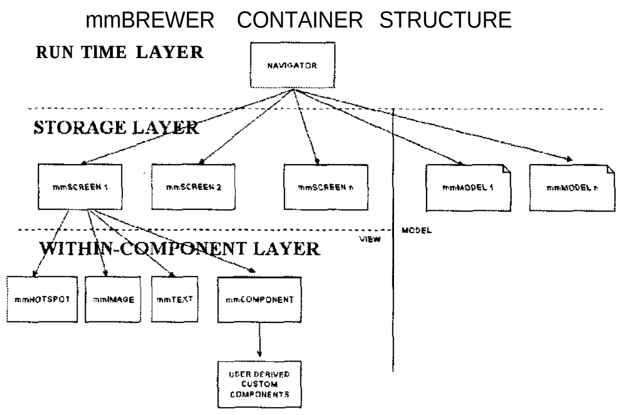


Figure 1: Structure of mmBrewer

if mmPainter includes the ability to change shades of color, another branch such as mmImage does not inherit this ability.

Navigator

The navigator acts as a "binding" of the multimedia title, connecting all pages and defining all interaction. Typically, the user is not concerned with defining classes below this root class. The user simply uses these sub-classes.

The navigator performs important tasks, which include:

- loading of images and text
- displaying screens as required
- sending user interactions to the current screen
- initializing start up
- navigating between screens
- ordering shut down upon user-initiated quit

Each screen can be thought of as a "container" of media components, such as the power plant background image, levers, switches, and lights. Once a screen is initiated, Java renders the screen and its media objects. Some of the media objects, such as levers and lights, can respond to events and messages.. Messages are coded into each screen not only to act upon a certain event, but also to look for an action on any other screen. For example, if there is a mouse down event indicating a breaker closed, the Java program sends messages such as "show-red" to the breaker lights and "show-next-frame" to the breaker lever to animate the action. When such events occur, a message is sent to other objects and screens. For example, a “breaker-closed” message can be sent to the generator model. Figure 2 describes the message structure of mmBrewer.

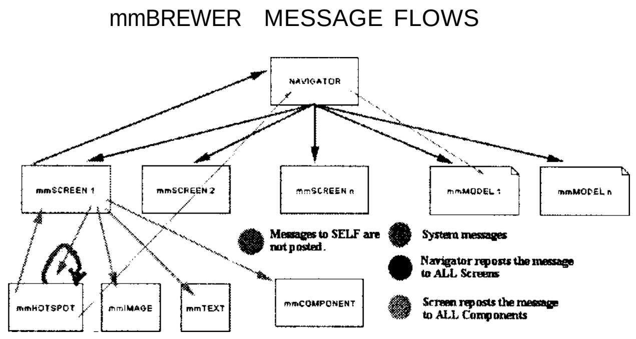


Figure 2: mmBrewer’s Message Structure

During the preliminary construction of the simulator, images were collected from a variety of sources. A portion of the images were rendered on a Macintosh using Alias Sketch and Adobe Photoshop 3.0, and later imported to the PC platform. A number of images were digitized from actual photographs of the plant. Other images were created and rendered using MicroStation.

Two forms of animation are possible and were tried. The first method involves running a Quicktime Movie to produce the desired animation. The second method is a process of combining multiple images with Java painting actions. Painted animations were discovered to have an advantage over Quicktime Movies. Quicktime Movies tended to slow the program down, especially when animating several meter dials simultaneously. The needle movement became jerky. Using Java, this problem was solved by simply rendering a needle dynamically, similar to a clock’s second hand. The class responsible for this animation runs on a different execution thread. Upon receiving a message, such as “turn needle 40 degrees,” the thread keeps erasing and redrawing the needle until it reaches its final angle. The erase and redraw (called double buffering) is fast enough to provide smooth animation. This technique also reduces required storage for external media (e.g. Quicktime Movies).

Interaction Modes

There are three modes of interaction with the simulator: control mode, exercise mode and information mode. The control mode is the primary way students interact with the simulator. Figure 3 depicts a screen in control mode. This mode stimulates learning through trial and observation. The most obvious indication of control mode is the ability to manipulate switches and adjust equipment settings and parameters. Meters, lights,

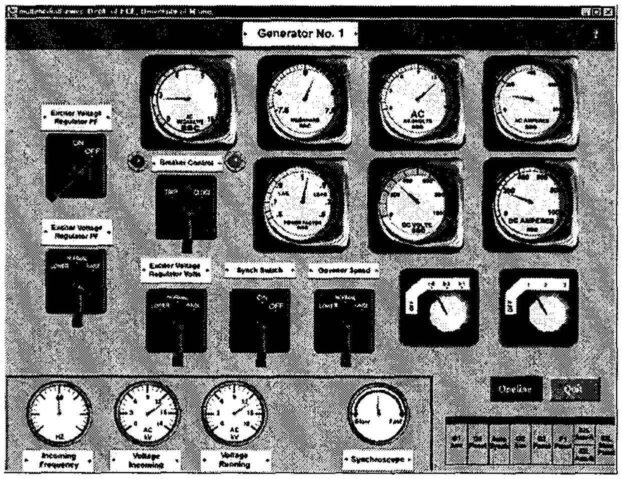
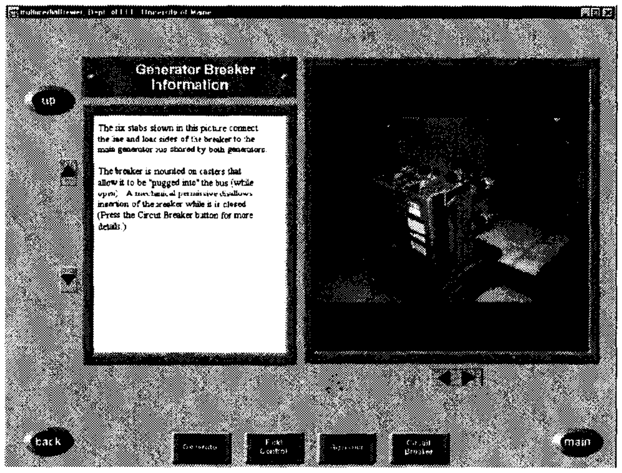


Figure 3: Control Screen

live text, flags, and audio indicators provide feedback comparable to that obtained in the actual plant.

The exercise mode provides reinforcement of theory and principles learned in lecture that are not immediately obvious from controlling the electrical plant simulation. For example, students are able to view phasors related to synchronizing operations and the effects of unlike phase sequence on each side of a connecting breaker. Although these actions are illustrated in the simulator, the exercise provides more theoretical background. A goal of the simulator is to model the (many times) limited feedback a plant operator has in observing the plant operation. No such limitation exists in exercise mode. Its purpose is to help explain the concept and can show phasors or other mathematical modeling tools.

These exercises may or may not be directly required for plant simulation control, but the simulator provides information or background on any such inquires.



The six stabs shown in this picture connect the line and load sides of the breaker to the main generator bus shared by both generators.

The breaker is mounted on casters that allow it to be "pugged into" the bus (while open). A mechanical permissive disallows insertion of the breaker while it is closed. (Press the Circuit Breaker button for more details.)

Figure 4: Information Screen

Information mode provides photographic and text information constituting a physical description and context for the simulation. A sample information screen is shown in Figure 4. A “drag and drop” question mark automatically acquires information about any component upon which it is dropped. By means of two classes called mmInfoBtn and mmInfoSc, information screens are reduced to a single line entry, a list of needed images, and the appropriate text file. Associating each information screen with a component allows the navigator to open the correct screen when the “drag and drop” message is detected.

3 Simulator Model Description

With the multimedia authoring environment in place, the next step is to incorporate the mathematical plant model. This is the component that calculates all values presented to and received from the interactive screens. The simulator models the control and operation of the West Enfield hydro-electric power station.

The model is sufficiently detailed to illustrate the fundamental concepts of the Junior level Power System Lab course, but not so detailed as to overwhelm the student. Each plant subsystem of interest is represented as a simulation module that is interconnected to other simulation modules or physical devices. Where possible, an interface is made to physical devices. In particular, programmable logic controllers (PLCs) can be programmed separately by students and interfaced to the power system simulator in much the same way they are interfaced to the real world plant.

The heart of the model is a dual synchronous generator representation written in Java. The output terminals of the synchronous machines contain three-phase voltage and current information that is fed to the metering and protective relays. The relay outputs are interfaced to the laboratory PLCs. Manual or PLC-operated generator field and governor controls determine the machines’ behavior. A synchronizing switch enables manual synchronization. The wye-grounded/delta power transformer is connected between the main bus and the equivalent system. The equivalent system represents an infinite bus behind the system impedance and draws or supplies real and reactive power based on the synchronous machine field and governor controls.

4 Conclusions and Future Goals

At this point, the multimedia authoring tool has been completed. In addition, the major components of the simulation model and several exercises have been completed. Present activities include networking the PLC’s and simulator and integrating all the systems. We anticipate using the simulator for the first time in our lab class beginning in January of 96.

The mmBrewer toolkit has drastically decreased component development time (e.g. meters, levers, and breakers). Not only has the creation time been expedited, but expansions and modifications involve minimum adjustments. The integration of existing C++

models has not only decreased the completion time of the project, but has created a truly interactive environment. Although mmBrewer is currently being used for the power plant simulator, this tool is capable of developing a wide range of multimedia applications.

Our future goals include:

- high quality 16-bit audio support, Java currently handles only 8-bit format which has proven inadequate
- a video class, although Java does not support video, we hope to develop this capability through native code
- expand the simulator to include mechanical engineering functions and use the simulator in mechanical engineering undergraduate labs

Acknowledgements

The authors wish to thank the National Science Foundation Division of Undergraduate Education Instrumentation and Laboratory Improvement Program, Bangor Hydro-Pacific Associates, and the University of Maine Power Research Association.

References

[1] S.I. Mehta and S.M.Gronhovd, “Instrumentation and Communication Modules on CD-ROM’s for Enriching Engineering Education”, IEEE Transactions on Education, August 1996, pp 304-309

[2] P.J. Mosterman et al, “Design and Implementation of an Electronics Laboratory Simulator”, IEEE Transactions on Education, August 1996, pp 309-310

[3] B. Aktan et al, “Distance Learning Applied to Control Engineering Laboratories”, IEEE Transactions on Education, August 1996, pp 320-327

[4] R. Ybarra, J. Glatz, and M Becvar,“Animations, Simulations and Other Learning Stimulations: An Electronic Laboratory Tour” ,ASEE 1995 Annual Conference June, 1995 pp 1713-1717

[5] A. Oloufa, “Bringing the Real World to the Classroom with Multimedia” ,ASEE 1994 Annual Conference June, 1994 pp 2742-2745

[6] R. Abbanat, K Gramoll, J.Craig “Use of Multimedia Development Software for Engineering Courseware” 1994 Annual Conference June, 1994 pp 1217-1222

[7] J.B. Patton, P. Jayanetti, “The Making of Multimedia Power Systems Control and Simulation Labware”, IEEE Transactions on Education, August 1996, pp 314-320