



# Embedded Training Tools for Large Real-Time Systems

DENNIS PATRICK MCGRODER

*Advanced Techniques Branch  
Tactical Electronic Warfare Division*

September 25, 1995

<b>REPORT DOCUMENTATION PAGE</b>			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE  September 25, 1995	3. REPORT TYPE AND DATES COVERED		
4. TITLE AND SUBTITLE  Embedded Training Tools for Large Real-Time Systems			5. FUNDING NUMBERS	
6. AUTHOR(S)  Dennis Patrick McGroder			8. PERFORMING ORGANIZATION REPORT NUMBER  NRL/FR/5750-95-9749	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)  Naval Research Laboratory Washington, DC 20375-5320				
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Space and Naval Warfare Systems Command Washington, DC 20363-5100			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT  Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT ( <i>Maximum 200 words</i> )  A systematic and organized approach to training new computer operators on large workstation-based real-time systems is in order due to the complexity of advanced multifunctioned software in use and under development today. User friendly graphical interfaces are widely used for the system functions themselves, yet the job of training new operators is still a challenging process. A training module has been designed to provide the tools to create and execute computer-based training sessions that can teach new operators specific functions of a large and complex real-time system. This embedded training module design has been implemented at the Naval Research Laboratory (NRL) in a Navy command and control (C2) environment. The training module uses an X Windows/Motif-based graphical user interface, as found in a typical modern workstation, for operational consistency. For any particular application or module within a large system, a training coordinator can define a series of step-by-step instructions with graphical aids such as highlights, images, and video, plus realistic scenarios, for learning how to use the application. A special scripting language, the Presentation Authoring Language (PAL), has been devised to set a standard method of defining the necessary components of a training session. Actual training sessions have been created for other C2 software modules developed at NRL, as well as sessions for the Fleet training schools for major functions of the Navy Tactical Command System-Afloat (NTCS-A) and Joint Maritime Command Information System (JMCIS).				
14. SUBJECT TERMS  Training                      Real-time systems                      Reconstruction                      Training sessions Command and control                      Software                      Scenario Embedded training                      Computer                      Scenario generation			15. NUMBER OF PAGES  38	
17. SECURITY CLASSIFICATION OF REPORT  UNCLASSIFIED			16. PRICE CODE	
18. SECURITY CLASSIFICATION OF THIS PAGE  UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT  UNCLASSIFIED		20. LIMITATION OF ABSTRACT  UL

## CONTENTS

INTRODUCTION.....	1
REQUIREMENTS.....	2
GENERAL DESIGN APPROACH.....	4
EXAMPLE APPLICATION: THE TRAINING MODULE FOR JMCIS.....	7
RESULTS.....	19
FUTURE DEVELOPMENT AND APPLICATIONS.....	21
SUMMARY.....	23
ACKNOWLEDGMENTS.....	24
REFERENCES.....	24
GLOSSARY.....	25
APPENDIX A—PAL Command Descriptions.....	27
APPENDIX B—PAL As Hypertext Markup Language (HTML) Extensions.....	33

## EMBEDDED TRAINING TOOLS FOR LARGE REAL-TIME SYSTEMS

### INTRODUCTION

Modern real-time computer systems combine sophisticated graphical displays with powerful analytical tools and advanced data manipulation techniques. The advancement of computer technology has occurred at a fast pace, especially in the graphics workstation and personal computer arena. An open-system architecture is available on very fast graphics workstations, and modern window-based graphical user interfaces (GUIs) with a variety of menu structures allow many applications to be integrated, executed, and displayed simultaneously on one machine. The complexity and capabilities of these systems are increasing rapidly, which may seem overwhelming to the first-time operator or the experienced operator who must learn new functions. These modern user interfaces are easier and more intuitive than the character-based commands of operating systems of the past, yet a systematic and organized approach to on-line training of new operators on these large advanced systems is needed in the software development community. User's manuals and on-line help functions are beneficial, of course, and they can aid in deciphering the terminology and functionality of the system being trained, but they rarely offer a step-by-step approach to getting started or example sessions to accomplish specific tasks.

A set of training tools that are embedded in a large real-time system could provide the needed systematic approach, and two basic capabilities are necessary. The first is the capability to execute a training session made up of a series of steps, or "frames," that instruct a new operator on how to start, setup, and use an application or module on the system. The second is the capability to create a training session from scratch by defining a series of these frames made up of textual instructions, graphical aids such as arrows, colored area highlights, images related to the system or function being trained, and graphical feedback of correctly executed system functions, plus realistic scenarios of events handled by the real-time system. Both capabilities are part of the overall set of software that we here call a training module. The first capability would be the only portion accessible by the trainee. The training coordinator, instructor, or training development team would have access to both capabilities.

If the training sessions are designed in a well-thought out manner, and simple procedures are included in introductory sessions and more complex procedures are included in advanced sessions, new operators can follow a systematic approach to learning how to use the system, without having to figure out how to navigate the system by trial and error. A training module like this could decrease the possibility of a new operator giving up or being reassigned to other duties because of the initial overwhelming complexity.

An embedded training module design has been initially applied to the on-line training of new operators of the Navy Tactical Command System Afloat (NTCS-A) and the Joint Maritime Command Information System (JMCIS). An early version of the Training Module has been developed at the Naval Research Laboratory (NRL) and has been successfully integrated within the NTCS-A and is widely deployed. The Training Module has also been deployed with the new JMCIS systems. The development of the Training Module is ongoing at NRL to provide an embedded on-line training capability for the Navy's future command and control systems [1]. The design approach is also

applicable to the on-line training of other military command and control systems, and with some enhancements for the specific application area, many other large real-time systems. This design can also be applied to the creation of general presentations for military briefing purposes or classroom lectures for training school environments, especially when connected to a large screen display.

A special processing language called the Presentation Authoring Language (PAL) has been devised to establish a standard method of textually defining the necessary session components of a computer-based training session or presentation. Training coordinators can define a series of step-by-step instructions, which generate pop-up text windows, and they can determine the title and text for each instruction and set the position on the screen where these text windows will appear during a training session. Along with the instructions, comments can be made about the system function being taught, the current state of the system, or the past event being analyzed. Graphical images may be linked to the text windows, as well as other multimedia elements such as voice, sound effects, and full-motion video. Also, in the case of a map-based Navy command and control display, highlights in the form of arrows, colored geographic areas of interest, and highlights of particular naval track symbols can be made to further enhance the training session or point out specific events or geographic areas (e.g., a colored arrow highlight pointing to a naval port of interest on the background map). In addition, actions such as specific scenarios and events can be driven from the training session, with timing features that control when various components of the session appear or become active (e.g., the arrow stays visible for 10 seconds then disappears). Each session component has a specific command syntax for defining all the information necessary for synthetic creation and display on the screen.

In addition, a scenario generation capability has been developed and integrated with the Training Module to provide the ability to create dynamic training scenarios that simulate ship, air, submarine, and land track movement on the map display and generate events pertinent to the training. This Scenario Generator feature allows for further realism in the training sessions, and timing features allow the simulation to occur at real time, faster than real time, or slower than real time. A scenario is created by the training coordinator in the training session generation portion of the training module, and it is then inserted in the appropriate places in the session. The scenario is played out when the training session is run by a trainee.

## REQUIREMENTS

To establish minimum requirements for the training software module, we should first make some basic assumptions. We assume that we have basically two types of users of this software: the trainer and the trainee. The trainer is a person responsible for training one or more users on the operation, functionality, and use of a software function, module, or entire system. Trainers, or "training coordinators," may base their training sessions on some standard training procedures, laboratory workbooks, or training school guidelines, or on the actual "User's Manual." In addition, they may design their own sessions using their own creativity as a guide. This all depends on the system and its complexity and the organization of which they are a part. Trainees, or students, are people who are required to learn how to use a new software function or module or operate an entire system. We assume that the system is large enough to require at least a user's manual of some sort to get new operators started. Any small, simple, or commonly known software function (e.g., a basic computerized calculator) would not require a training session to learn how to use it. We also assume that the system being trained contains a multitude of options with various possible paths of execution, i.e., it is nonlinear in nature (not a batch system). Typically, in modern systems, the system has a topic-oriented, menu-based structure for navigating through the various functions and options. Working with these basic assumptions, the requirements and general approach to designing an embedded set of training tools can be discussed.

**Training software needs to be embedded, but be clearly distinct from the real-time mode.**

The first requirement of training software is that it is embedded, that is, an integral part of the system for which it will train new operators. The executable software must be resident in the system and must be able to be executed along with other functions and modules of the entire system, thus requiring an open system with simultaneous processing abilities. The training software must not clash with the real-time operations of the system, and a clear distinction needs to be evident between training and real-time modes, such as a banner designating training mode and labeling and color-coding the training windows. Switching between modes must be quick and easy.

**Training software development and maintenance needs to be done on a system with a similar configuration as the destination system.**

The training software must be developed and maintained on a system with similar characteristics as the destination (operational) system, including but not limited to the same operating system, a compatible hardware platform, the same GUI software, and the same or nearly same set of application software that the training module will address. For example, in the case of training for the JMCIS system, the development system must have the same version UNIX operating system, a UNIX-based workstation with a large color graphics monitor, adequate memory and storage capacities, the X Windows and Motif graphical user interface software, and the latest version of the JMCIS software including the tactical chart display, various database managers, and communications processes.

**Training software should have a simple and concise GUI.**

To keep the creation of the training sessions as straightforward as possible for the training coordinators and instructors, the GUI should be simple and concise, as well as user-friendly, and the number of available session components should be small. Standard guidelines should be followed for the user interface. The necessary session components include

- Text windows, for the training instructions and annotations
- Graphical elements such as arrows, rectangles, circles, and symbols, for highlighting specific objects or areas of interest pertinent to the training
- Graphical images such as photographs, diagrams, and system screen copies that illustrate or add to the instruction being given
- Action elements that cause specific events to take place relevant to the training (e.g., processes that drive a scenario or change the background map in a Navy C2 system).

Optional session components that use modern digitized multimedia software techniques, although not necessary to get the job done, can add to the overall presentation of the training. These include

- voice instructions that are produced automatically at each instruction
- full-motion video clips that are run from a session window
- sound clips that are used for special effects or emphasis.

**Training software should have the same GUI as the destination system.**

The training module should have the same or similar GUI as the system for which it is providing training. This guarantees a consistent look and feel and less confusion when a trainee goes back and forth between the system functions and the training module. It is useful, however, to have a visual cue, such as a different background color for training windows, to differentiate the training windows from the real-time system function and application windows. One step further along GUI lines, the

training module should have an equal or higher level of user friendliness than the real-time system. If the embedded training software is more difficult to run than the system itself, then there is really no point in using that embedded training in the first place.

### **Training software should allow branching, or nonlinear, execution of training sessions.**

A training session that is linear would allow only step by step progression, from start to finish, or the reverse. As in the case with user's manuals, the trainee may already know a certain topic or procedure and may skip ahead in the manual to the next topic of interest or different procedure to be learned. In the same manner, the training module must allow a trainee to skip a frame or series of frames that contain already known or previously covered material without losing the continuity of the training session. The trainee may also need to go back to a previous frame to review some material. Providing the option to branch to more detailed information on a particular topic or procedure is also necessary and desirable. Conceptually, this can be thought of as three types of branching: ahead, backwards, and sideways. Branching ahead or backwards should be designed into the training module itself, whereas branching sideways depends on the training topics or procedures and would be set up by the training coordinator. The coordinator would decide which topics would warrant the provision of more details, if desired by the trainee, and would set up these branches by highlighting a word or phrase in a frame that, when selected, would cause a link to the detailed frame(s), with a link back to where the trainee left off.

The various requirements for the training features of a specific system depend on the application area, so the potential list of requirements is endless. But this list of requirements is general enough to apply to all types of embedded training. The real power of the embedded training sessions to be produced comes not from the components themselves, or even how flashy the graphics may be, but from how the training coordinators and training development teams use the advanced features available to them to create cohesive, informative, simple to use, and understandable instructions to train their new operators.

## **GENERAL DESIGN APPROACH**

### **Distinction Between Training Mode and Real-time Mode**

The design of an embedded training module for large real-time systems began with looking at a particular system, but it can be applied conceptually to any large real-time system. The aspects of the system that are real-time need to be preserved while a training session is being run. In other words, going from training mode to real-time operating mode has to be quick and straightforward, especially in crisis or demanding situations. This means that the training module needs to be embedded (i.e., tightly integrated with the system software) but it must allow the system's real-time operations to continue uninterrupted in the background. Attention must be paid to what real-time processes are going on in the particular system for which training is being developed. System resources cannot be degraded or "hogged" by the training module, and the training module must take advantage of every available "training" or "off-line" feature that is already built into the system. For instance, in the case of NTCS-A and JMCIS, the track database (TDB) manager (Tdbm) process is always running, and it will be processing real-time track data on an event-by-event basis. The database itself has an area specifically designated for "simulated" or "non real-time" tracks that is usable by additional applications, such as the training module. These simulated tracks for training are managed by the same Tdbm that is handling the real-time tracks, but there is no contention, and both track types can be displayed and used in the same manner. Contention or conflicting data could occur if, for instance, a training module were to use the real-time tracks portion of the TDB for its simulations. Numerous problems are possible here, especially with automatic track correlation. So,

in general, it is best to follow as many design guidelines and use as many features applicable to training as possible that are already available in the destination system.

### Components of a Training Session

Before starting to design a training module, we asked what various components would make up a session and how they would be represented. From those answers we designed a sample session using a basic text editor. We first envisioned the conceptual components of a training session, as illustrated in Fig. 1, and translated those components into command elements of a character-based processing language. We realized that a session could be generally applied to on-line presentation and briefing purposes as well as embedded training, thus the name of this language became the Presentation Authoring Language (PAL). A session is a named file made up of a series of PAL building blocks called *frames*. Each frame contains *objects* and *actions*. The possible *objects* are *text windows*, *highlights*, and *images*. The possible *actions* are *timing actions* and *process spawn actions*. The *text window* object is defined by a *title*, *screen position*, *size*, and *text*. The *text window* is the first main element that is seen when starting a session. Every *text window* should have directional buttons for stepping through the session. The initial design has Next and Previous buttons, but a later design will have buttons for branching: Ahead, Backwards, and Branch (sideways to other topics). At a minimum, the *highlight* objects that should be defined by PAL are Arrow, Rectangle, Circle, Polygon, Track, and Symbol. These objects can be used to highlight particular elements on the screen. For example, the arrow highlight could point to a moving ship symbol of interest, and a polygon highlight can enclose a certain geographical area of interest. Others may be added at a later stage. The *image* objects possible are digitized photographs, graphical diagrams and charts, and system screen displays (screen dumps), all possibly from different sources or of different image format types. The *timing actions* are Pause for pausing a session for a number of seconds, Freeze for stopping a specific spawned process during the session, and Timeout for controlling the duration of display of *highlight* and *image* objects. These timing actions would most likely be used for a presentation that is to be run automatically, rather than for a manually executed training session. The *process spawn* actions include a general Spawn action that basically allows any application to be started and run automatically at a certain point in a training session. Some specific *process spawn* actions that were specified for the Training Module include the Replay action and Map action. The Replay action is used for running a Replay File of reconstructed tracks or Scenario File of simulated tracks for a specified time period during a session, that is from the specified start Date Time Group (DTG) to the end DTG. The Map action is for changing the system's background map center and width at a desired point in the session to allow automated pan and zoom functions.

### Run Training Session Capability

The first step in the design process of the Training Module itself was to develop a simple user interface to select the Session File, and to develop the parsing procedure necessary to read and convert the character-based PAL commands into the training module components. All training session components are defined by entering the proper PAL commands in an ASCII text Session File with an editor, with the results being checked by the trainer when test-running the Session File. When the trainer has finished editing and the session is complete, the trainee can run the Training Session File. We call this capability the Run Training Session program. Procedures were developed for sizing and placing text windows on the screen, creating and placing highlights and image graphics on the screen, and linking the frame data together, using an X Windows-based GUI. The basic user interface for the trainee while running a session consists of training windows that contain text instructions plus four buttons: Next, Previous, Images, and Exit. Following intuition, the trainee presses Next to go forward through the session, Previous to go backward to the previous frame, Image to view an image relevant to the current frame, and Exit to quit the current session. In the case of the

Training Module for NTCS-A and JMCIS, an embedded change map procedure was needed, and integration with the JMCIS Chart and the Reconstruction Module [2], developed at NRL, was needed for a fully dynamic C2 system training module.

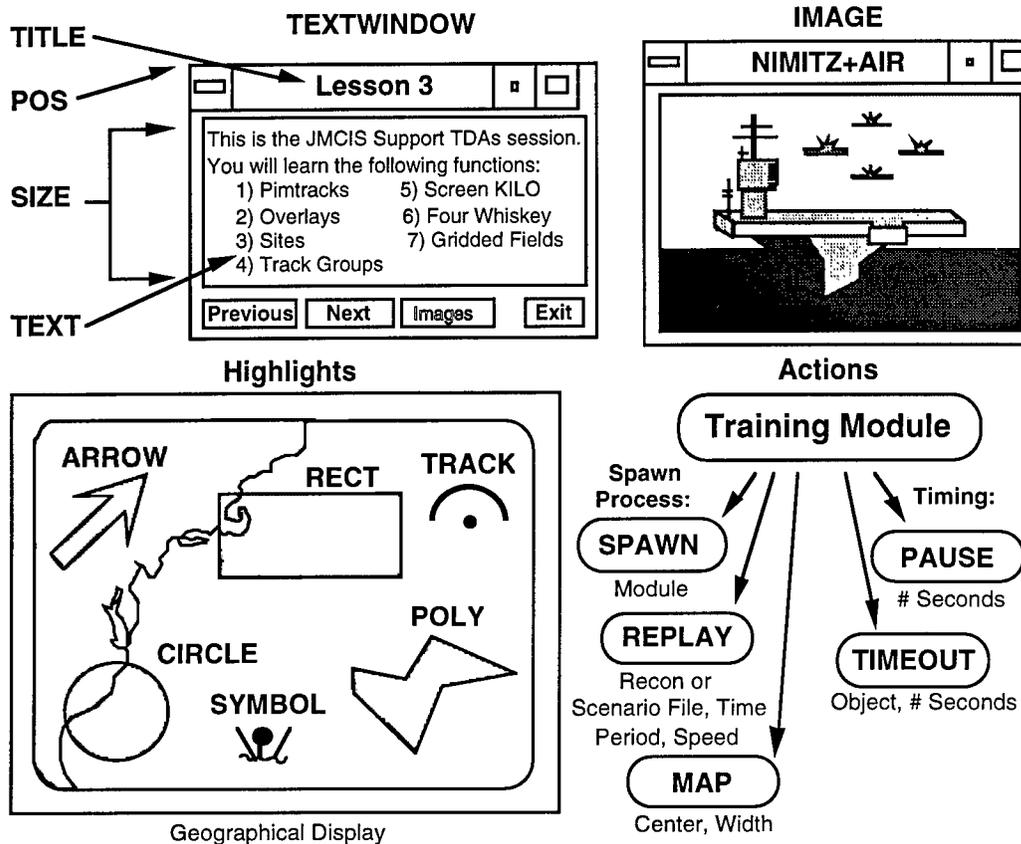


Fig. 1 — Components of a training session

### Training Session Generation Capability

The initial training module design only allowed the execution of a previously created Session File which was typed in by hand in an editor, using the PAL. The next step was to design a Training Session Generation capability that allows a training coordinator or instructor to type the Frame text in a frame definition window, automatically place each frame at desired screen locations with the mouse and cursor, and setup each highlight and graphical image with a GUI that allows easy placement of these items. In the case of the Training Module for NTCS-A and JMCIS, procedures were designed for the trainer to set desired maps for specific frames and define the scenario replay information and constraints, if needed for that session. The session generation software creates PAL commands internally, and when the trainer wants to save a session file, everything defined up to that point will be saved in a PAL Training Session File. To do this, the trainer does not have to know the specific PAL commands. Thus, PAL can become a standard, but does not need to be learned as a language like other third and fourth generation computer languages. In a sense, PAL is similar to the Navy standard Over The Horizon Track GOLD (OTH-T GOLD) message format that is used for intercomputer track data transmission and the scenario generation capability of the Training Module, discussed later in this report.

## **Presentation Authoring Language (PAL)**

A standard set of command key words has been established to represent the various components of an embedded training session or an on-line presentation or briefing. These commands are used to create a Training Session File or Presentation File. The command key words are interpreted by a parsing procedure of a training module or presentation module. The use of capitals for text and slashes as field separators is the text format standard, as used in the Navy standard OTH-T GOLD message format and other Navy computer message formats.

Future versions of PAL and the Training Module will use command syntax similar to the recognized standard Hyper Text Markup Language (HTML) [3] used in Hyper Text applications, and this version of PAL will be presented as an HTML extension. This way, sessions developed in HTML by other applications will be usable in this Training Module, and the Training Module will be enhanced by the extra document formatting and Hyper Text linking features of HTML. These Hyper Text linking features would fulfill the requirement for branching within the Training Module. The trainer could set up these branches through a Hyper Text mechanism by highlighting a key word or phrase in a frame that, when selected, would cause a link to the detailed frame(s). Table 1 is a list of PAL command definitions with their associated parameter names. Appendix A describes the PAL commands in more detail, and Appendix B gives the suggested PAL extensions to HTML.

### **EXAMPLE APPLICATION: THE TRAINING MODULE FOR JMCIS**

The Training Module being developed at NRL for the Navy's future C2 system, JMCIS, serves two purposes. First, it provides the ability for a trainer to create training sessions for the JMCIS system. Second, it provides the ability to conduct these training sessions by a student or trainee who is learning how to use JMCIS. The training sessions may be created within an operational JMCIS at sea or at a shore facility. A broad range of training can be created and conducted, from simple beginner training sessions for basic JMCIS system functions, such as a core JOTS (Joint Operational Tactical System) or track management functions, to dynamic event-driven operational training containing tactics, analysis, and assessment development. Specific sessions may be used for a number of purposes, such as JMCIS functionality training, JMCIS User's Guides, Navy course material (student guides and lab workbooks), course lecture material using large-screen projection, and command briefings and presentations. An early version of the Training Module has been integrated, tested, and released with the NTCS-A and JMCIS systems, and has begun to be used at a number of shore sites [4]. Further discussion of the Training Module in this report refers to that being developed for JMCIS.

### **Training Module Integration with JMCIS**

The Training Module follows the JMCIS integration guidelines inherent in the Common Operating Environment (COE) and Integration Standard (IS) that has been determined for all JMCIS software developers [5, 6]. The training sessions are created within JMCIS, and all functions and modules of JMCIS are available to the trainees and the training developers as they create the sessions. The Training Module is available from a top level menu of JMCIS; JMCIS core software such as the Chart service, the Tdbm service, and the X Windows and Motif GUI with the standard Navy User Interface Specifications (UIS) [7] are used by the Training Module via the Application Programmer Interface (API) [8]. API library functions for using the Tdbm and Chart services, plus the Motif and X Windows libraries, are called by the Training Module. Therefore, the Training Module is a true embedded module. Figure 2 shows the separate levels of the JMCIS COE.

Table 1 — PAL Commands and Parameters

COMMAND	DEFINITIONS
<b>Object: Frame</b>	
FRAME	FRAME/
ENDFRAME	ENDFRAME/
<b>Object: Text Window</b>	
TEXTWINDOW	TEXTWINDOW/TextWindowNumber/
TITLE	TITLE/TextWindowTitle/
POS	POS/ScreenX/ScreenY/
SIZE	SIZE/NumberChars/NumberLines/
TEXT	TEXT/TextWindowLine[1] TextWindowLine[2] ... TextWindowLine[n] /                    where n <= 20
<b>Object: Highlight</b>	
ARROW	ARROW/ObjectNumber/Direction/Lat/Lon/Color/ where Direction = N, E, S, W, NE, SE, SW, or NW
CIRCLE	CIRCLE/ObjectNumber/CenterLat/CenterLon/Radius/Color/
RECT	RECT/ObjectNumber/Lat/Lon/Length/Width/Color
POLY	POLY/ObjectNumber/NumPoints/Lat1/Lon1/.../LatN/LonN/Color
TRACK	TRACK/ObjectNumber/TrackName/Color/
SYMBOL	SYMBOL/ObjectNumber/SymbolType/Lat/Lon/Color/
<b>Object: Image</b>	
IMAGE	IMAGE/ImageType/ImageFilename/ImageFileFormat/X/Y/ where ImageType 0 = Stored System Slide (Screen Image) ImageType 1 = Graphical Image (Photo, Illustration)
<b>Action: Timing</b>	
FREEZE	FREEZE/
PAUSE	PAUSE/NumberSeconds/
TIMEOUT	TIMEOUT/Object/ObjectNumber/NumberSeconds/
<b>Action: Process Spawn</b>	
SPAWN	SPAWN/ModuleName/Arg1/Arg2/.../ArgN/
REPLAY	REPLAY/Filename/FreezeDTG/Ratio/Delay/TimeOrder /UseTdbm/UpdateTdbm/PlotSymbols/PlotAltitudes /PlotHistories/HistoryLength/UseBitmaps/ShowLOS/
MAP	MAP/Lat/Lon/Width/
CHANGEMAP	CHANGEMAP/On_or_Off/

See Appendix A for more detailed descriptions of these PAL commands and examples of their use, and Appendix B for the suggested PAL extensions to HTML.

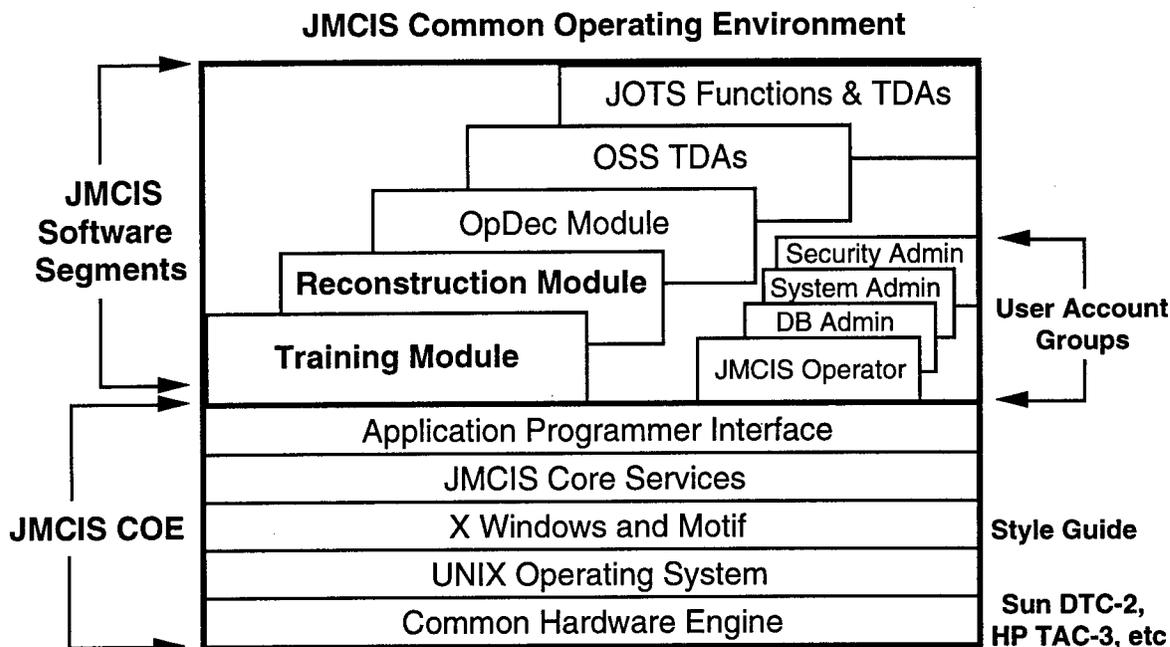


Fig. 2 — JMCIS Common Operating Environment

To create dynamic training scenarios for more realistic training sessions, the Training Module uses Replay, a submodule of the Reconstruction Module, and Scenario Generator, two other JMCIS-based applications developed at NRL. The events that drive the system during the more complex training sessions originate from real track data archived by the Reconstruction Module or through simulated track data created with the Scenario Generator. The embedded Training and Reconstruction modules are referred to as JMCIS Segments when integrated. Figure 3 shows how these modules are related and fit in with the overall JMCIS Client/Server Architecture.

Currently, the JMCIS system is learned by new operators through whatever paper bound user's manuals [9] that are provided by the developers, plus those materials that are developed at Fleet training schools. The goal of the Training Module is to enhance these forms of training. Through the use of the Training Module, training developers can easily provide a complete on-line manual of training sessions for all JMCIS functions that would be similar to the bound versions, plus additional sets of sessions that take advantage of the dynamic real-time aspects of the systems for the various types of operators and tactical areas of interest.

**Running a Training Session**

The trainee or student needs to know the name(s) of the training session to be run. A selection window comes up first when the Run Session program of the Training Module is started, and a Training Session File is selected from a list of sessions. After that, the trainee merely reads the information in the training windows and follows the instructions given in each frame of the training session. Figure 4 shows the Run Training Session window (Fig. 4 (a)) plus a sample training window (Fig. 4 (b)). The instructions may be to select specific JMCIS options, input proper data, and perform desired actions. When all tasks of one frame are completed, the trainee presses the Next button and continues. Highlighting aids such as arrows, symbols, icons, annotations, photographic imagery, or graphic windows may be placed on the JMCIS display as part of the session. If there is a scenario or replay file attached to the session, it will play according to predetermined timing settings at particular relevant frames in the session. The trainee watches the trainee scenario play out, and waits for the

next frame window to appear for instructions. At any point, the trainee can go to the Previous frame for review (as far back as desired). The trainee can also exit at any point by pressing the Exit button and may start the session over again or start a new session.

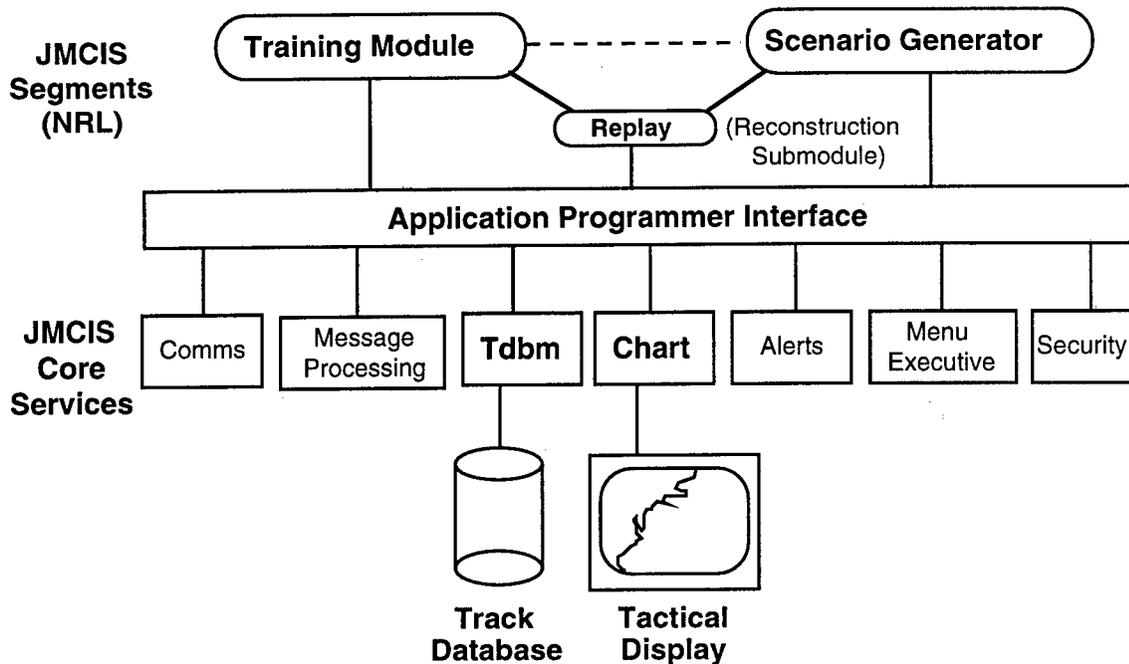
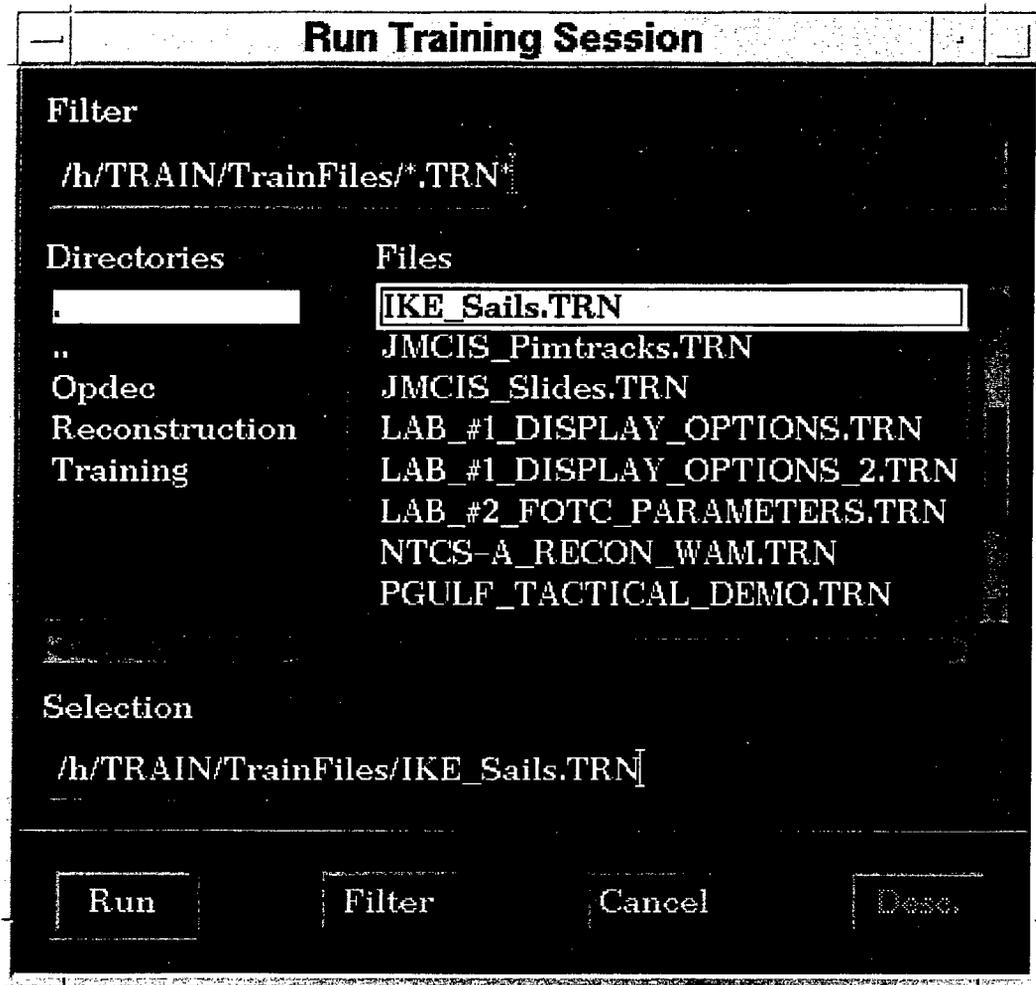


Fig. 3 — JMCIS Client/Server Architecture

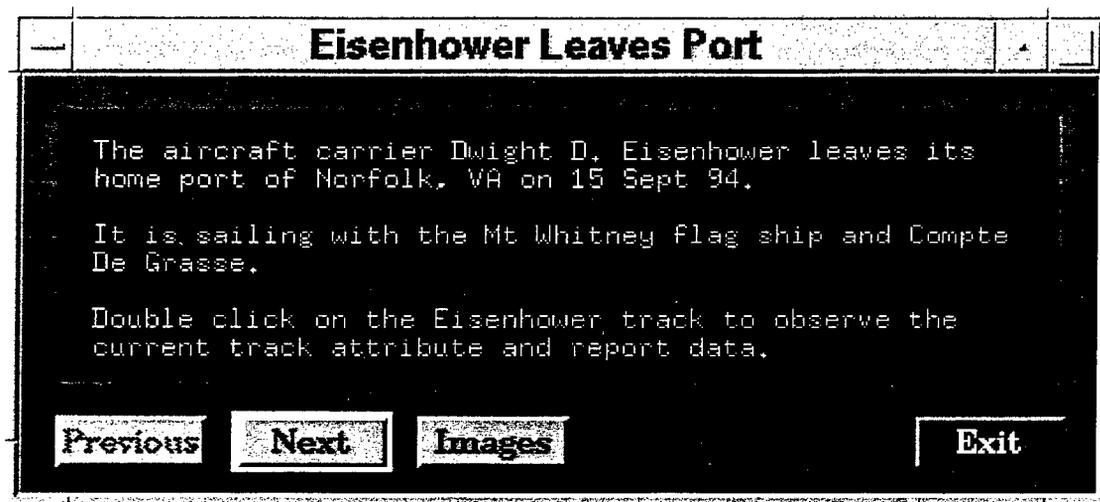
### Creating a Training Session

The trainer first determines the content of the training sessions and scenarios to be used. The training instructions are prepared in advance, preferably in a rough outline form on paper, inserted into the system with the Training Module's Training Session Generation program, and presented on the standard JMCIS tactical display through a series of orderly text instruction windows and illustrative graphics, optionally linked to a dynamic training scenario. These training windows will provide specific information and instructions that direct the trainee to select specific JMCIS options, input data, and perform certain actions. Groups of training sessions can be organized in various ways, but it is intended that each session be devoted to one unique system function or module, or general topic.

Figure 5 shows the Training Session Generator window. The file management options under the File pull-down menu allow the trainer to create a new session, edit a session, run a session to test it, save a completed session in the session file directory, and delete a session. The editing options under the Edit pull-down menu allow the trainer to insert a new frame after the current frame, remove the current frame, and clear the frame text and associated files (to start over). The Find pull-down menu is still under development; it will allow the user to find any word or group of words in the text of the session. The tools available to the trainer under the Tools pull-down menu are Replay, Scenario Generator, and Slide Archiver. Replay and Scenario Generator are explained in the next section. The Slide Archiver allows the user to save JMCIS screen slides used in training sessions on a tape, for easy transfer between systems.



(a)



(b)

Fig. 4 — Run Session File Window and Training Window

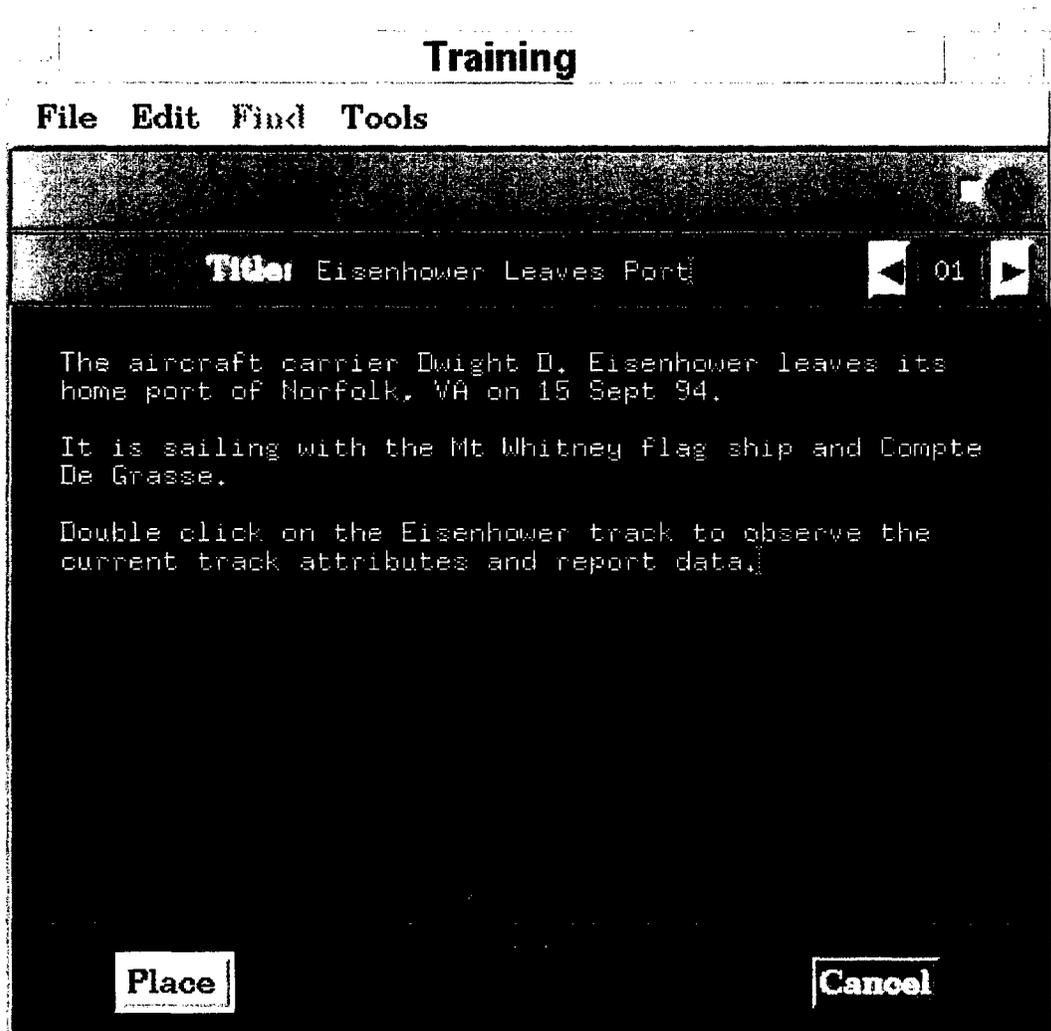


Fig. 5 — Training Session Generation Window

### Placing the Text Window, Highlights, and Images

When a training frame is completed by entering from 1 to 20 lines of text, the trainer needs to place the text window for that frame at the desired screen location. This is the location at which the window will appear when the session is run by the trainee. The trainer also may want to add highlights in the Chart display area and include pop-up images that enhance the individual training step. The trainer merely presses the Place button, and the Text Placement window with four buttons (Ok, Highlights, Imagery, and Cancel) appears on the screen, as shown in Fig. 6. Pressing the Highlights button brings up the Highlights window, which the trainer can use to place any number of arrow highlights on the JMCIS Chart, as shown in Fig. 6. Other types of highlights are still under development and will be added at a later stage as requested by users. By pressing the Imagery button, the Image File Setup window pops up, which allows the selection of JMCIS Slide images and graphic images from various sources. The trainer selects all images to be displayed in the Available Imagery list, and presses the Add button to put them in the Images to Display list. When the Place button is pressed, all images selected appear on the screen, and the trainer places them in appropriate positions on the screen, and presses the Ok button to complete the Imagery setup (in Fig. 6, only one image was placed). When the Ok button is pressed in the Text Placement window, it freezes the placement of the text window, arrow highlights, and imagery. These elements will all appear exactly where they were placed when the training session is run.

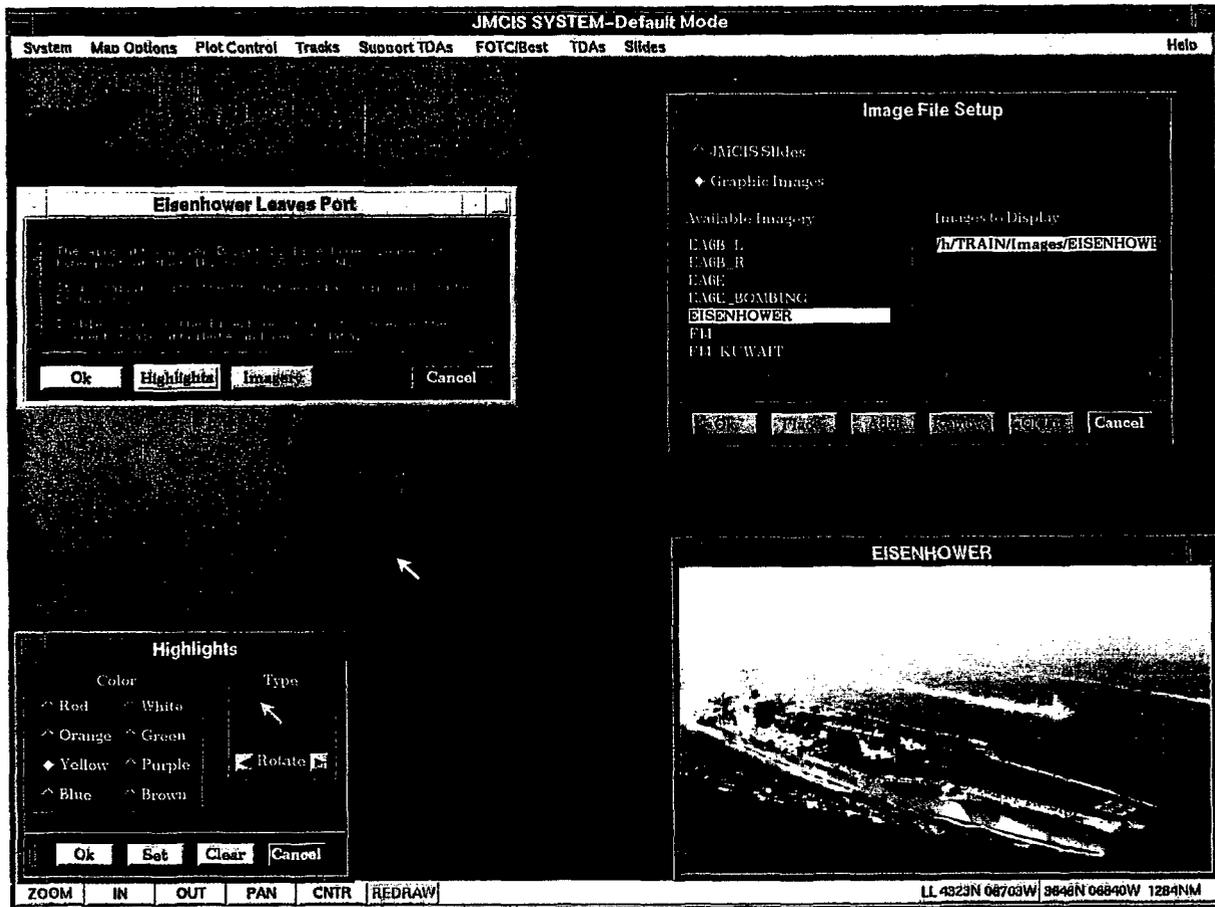


Fig. 6 — Text Placement window with arrow highlights and image placement

### Replay Within a Training Session

The Replay tool allows the trainer to associate an existing replay file of dynamic OTH-T GOLD track data with the current frame. The replay file may be obtained from the Tdbm Logger, which can be run during JMCIS real-time operations to obtain large amounts of actual real-time track data. Replay files may be merged, sorted, and edited by means of various options of the Reconstruction Module [2]. The replay file may also be a scenario file created using the Scenario Generator, which is accessible in the Training Module. With the Scenario Generator, the trainer can create simulated tracks to generate a scenario from scratch or provide a simulation enhancement to an actual Fleet exercise by merging real and simulated tracks. Either way, when a frame that has a replay associated with it is reached in the session, the tracks will replay on the Chart display according to predetermined timing parameters, then will "freeze" at the Freeze DTG previously specified by the trainer. Then the next training window will be displayed.

Selecting Replay from the Tools pull-down menu brings up the Replay File Setup window, shown in Fig. 7, which allows the trainer to select a replay file from a list of all files currently archived in the replay file directory. The trainer then establishes display and timing parameters for the replay. Display options include a Use Track Data Base Tracks option, which causes the tracks to be updated in the JMCIS track database. Other display options affect the display of the replay tracks. The Plot Histories option allows the track histories to be displayed during replay. The Line of Sight option

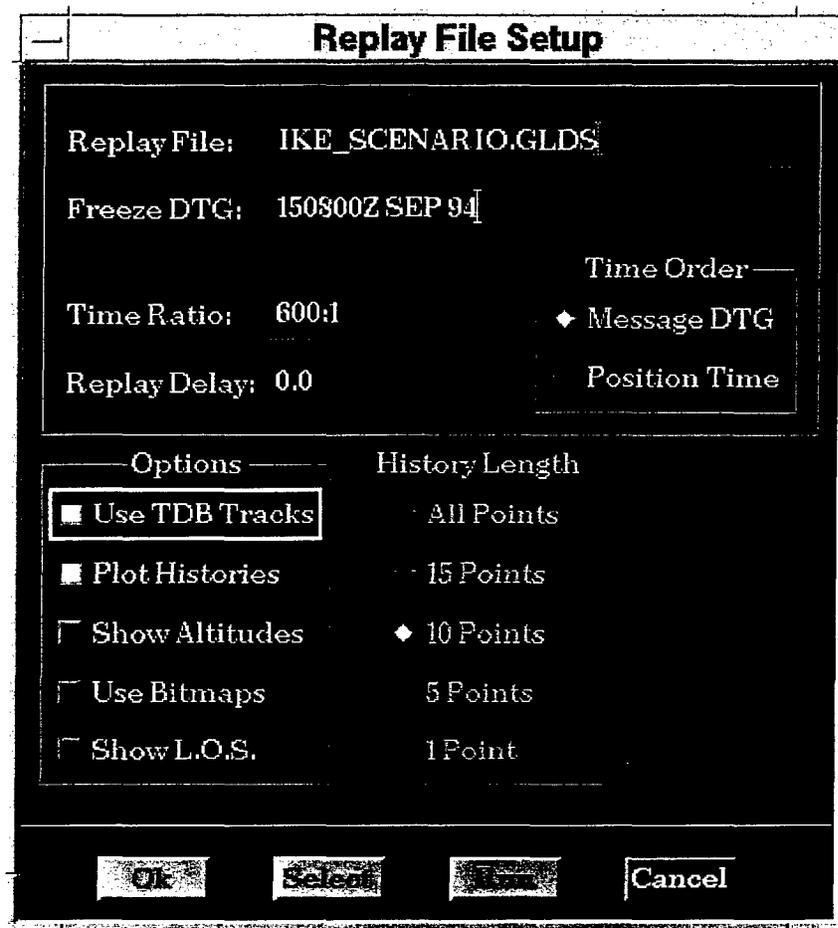


Fig. 7 — Replay File Setup window

displays line of sight range circles around tracks that have altitude data. The Show Altitudes option displays altitude lines of air tracks. The Use Bitmaps option allows the display of realistic track symbol icons during replay. The timing options allow the replay to be done at real-time speeds (1:1 replay ratio) for short, intense scenarios or careful examination of specific events, or faster than real-time (e.g., 60:1 replay ratio) for a quick review of positions. The trainer must determine the Freeze DTG as a point in time that the replay will stop for the current frame, running from the Freeze DTG of the previous frame. Otherwise, the replay file will run from start to finish, which may be desired in some cases.

### Scenario Generation Within a Training Session

The Scenario Generator is a tool that can be accessed from the Tools pull-down menu within the Training Module's Training Session Generation program to create dynamic training scenarios containing definition and movement of ship, air, sub, and land tracks over time, plus descriptions of events pertinent to the training. The main application window is shown in Fig. 8.

The Scenario Generator is integrated with the JMCIS Track Data Base Manager, and an option allows all simulated tracks to be updated in the JMCIS track database as the scenario runs. These tracks are updated independently in the system as "Terminal Simulated" tracks, a separate area from the "Local" or "Global" "Real-Time" tracks, so there is no clashing. Also, an option is available to place an "X" in front of all simulated track name labels to designate them as exercise or simulated tracks to avoid confusion. The Scenario Generator is also integrated with the JMCIS Chart to draw the simulated track symbols, the line of sight range circles, and formation range rings.

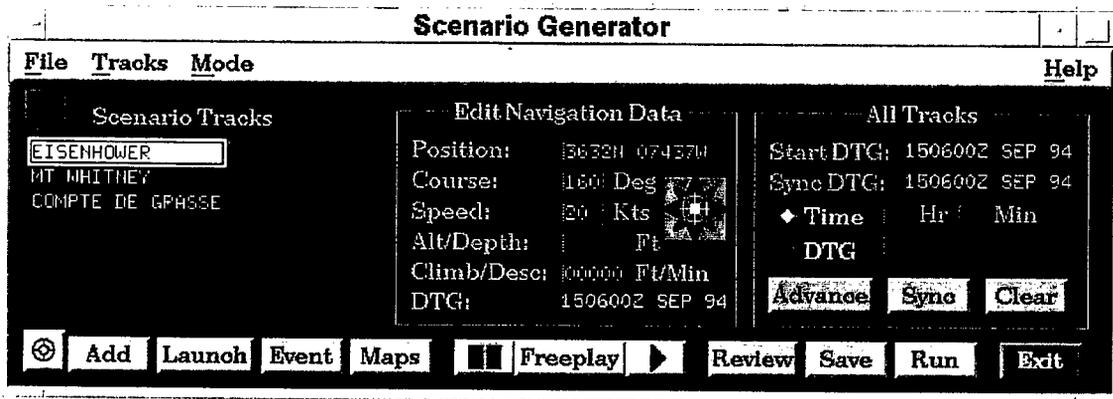


Fig. 8 — Scenario Generator Main Application window

Multiple categories of simulated tracks can be created with the Add Scenario Track Window, shown in Fig. 9. The possible force designations include Friendly (Blue), Neutral (Green), Hostile (Red), and Unknown (Yellow). The available platform types include Ship, Air, Sub, Land, and Unknown Air or Missile. A scenario is created by first defining the starting time and positions of all the units to participate in DTG format. Simulated tracks are added one at a time, where initial position data is entered by clicking a position on the chart or by typing in the latitude-longitude values. Other

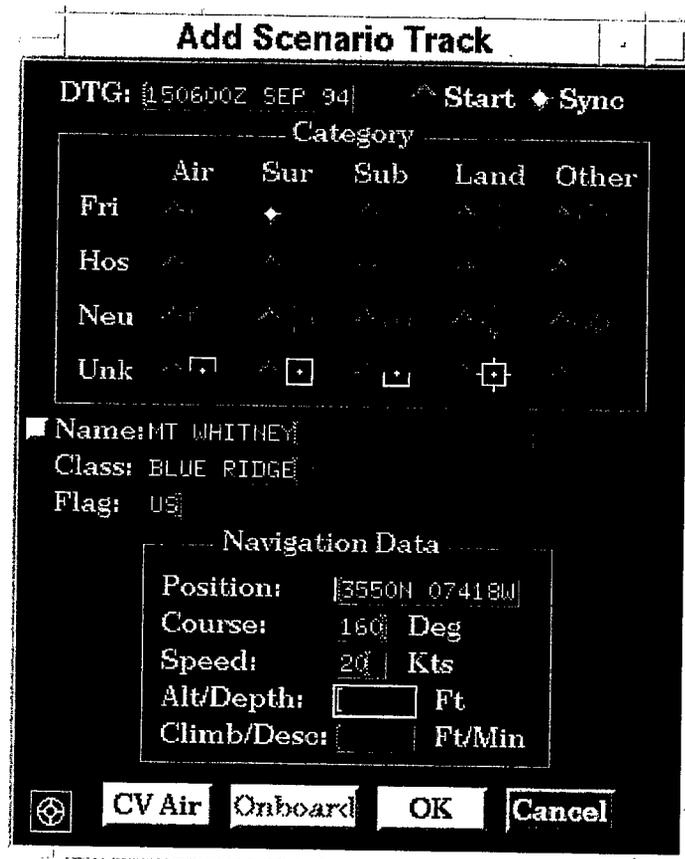


Fig. 9 — Adding a Scenario Track

initial navigation data including course, speed, altitude or depth, and climb rate or descent rate are also entered for new tracks. The CV Air button brings up the window shown in Fig. 10. An aircraft carrier (CV) can be selected as the next scenario track to be added, plus a set of any number of various classes of aircraft to be included on that CV in the scenario. The CV and its aircraft tracks will appear in the Scenario Tracks List of the main window, but the associated aircraft will not appear on the JMCIS Chart until they are "launched" from the CV with the Launch option.

CV And Onboard Aircraft	
CV	Onboard Aircraft
ABRAHAM LINCOLN	A-6E SWIP
AFSU KUZNETSOV	A-6E TRAM
AMERICA	A-6E WCSI
CARL VINSON	A-7E
CONSTELLATION	A-7H
DWIGHT D. EISENHOWER	A-7P
ENTERPRISE	E-2C 1
<b>GEORGE WASHINGTON</b>	EA-6A MOD
INDEPENDENCE	EA-6A RECAP
JOHN C. STENNIS	EA-6B 2
JOHN F. KENNEDY	EA-6B ICAP-1 (M)
KITTY HAWK	EA-6B MOD
NIMITZ	EAV-8B
RANGER	F-14A 2
SARATOGA	F-14A MOD
THEODORE ROOSEVELT	F-14A PLUS
UNITED STATES	F-14B
VARYAG	F-14D

Number Of AC: 1

Fig. 10 — CV Air window

A Formation Editor can be used to set up a formation among several tracks, where one track in the group whose initial position is already defined is designated as the reference track. Then each track is added to the formation by giving a relative bearing and range from the reference or by merely clicking on the chart, thus defining initial positions. Bearing and range are easily set and modified with the aid of a formation range rings display, which is similar to a maneuvering board, as shown in Fig. 11. Each track in the formation inherits the course and speed of its reference track. In the example in Fig. 11, the *Mt Whitney* is the reference track, and the *Eisenhower* and *Compte De Grasse* are in formation with the *Mt Whitney* as they leave Norfolk, each having the same course and speed as the *Mt Whitney* (160, 020). Formations can be made or broken at any point during the scenario generation by using the Join and Break formation editor options. It is envisioned that this option could be extended to allow simulated tracks to be in formation with actual real-time tracks that are being updated in the system, to provide a simulation enhanced Fleet exercise capability.

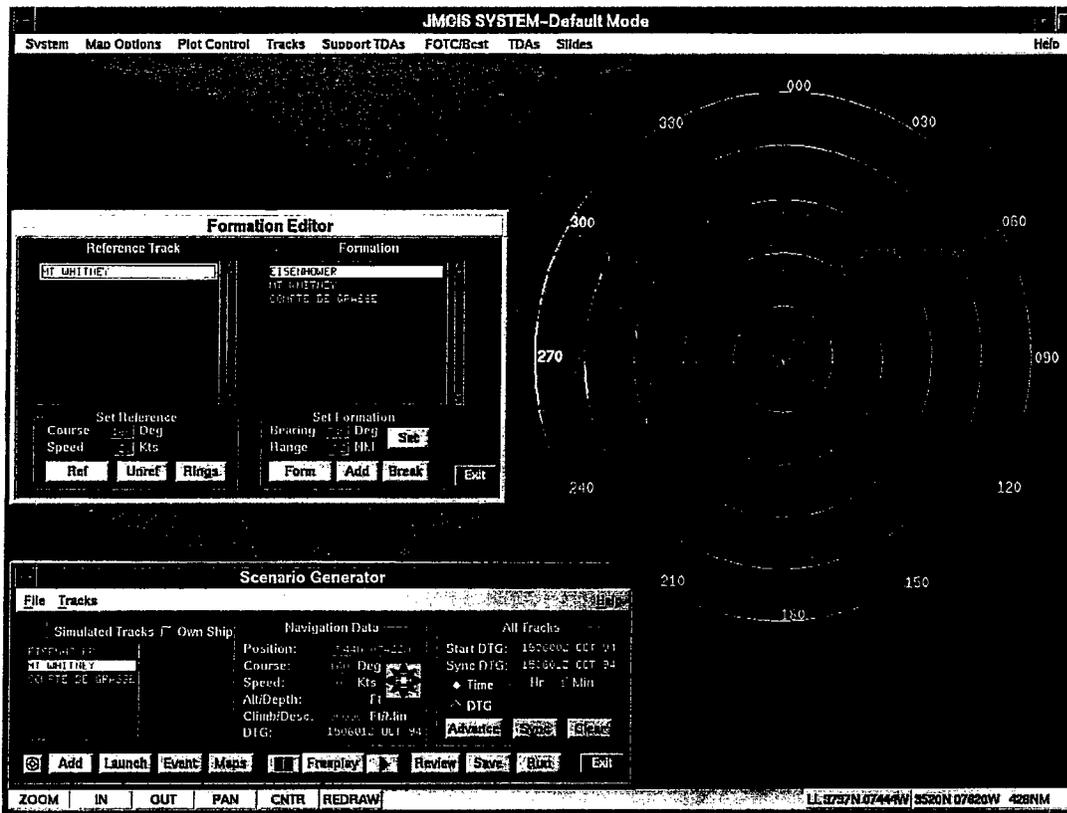


Fig. 11 — Formation Editor with tracks in formation

The scenario is then advanced and new positions are generated by using the Advance options. Two navigation methods, Great Circle and Rhumb Line, are possible for advancing tracks, and the method is set once in the beginning. The tracks can be advanced to their next positions by clicking a position on the chart, entering latitude-longitude values, or by various time projection options, as illustrated in Fig. 12. All tracks can be advanced by a particular time amount (e.g., 15 minutes) based on their current course and speed, to a particular point in time (such as 150800Z JUL 95), or to the latest time that will synchronize all tracks to the same point in time (Sync option). Also, new formations may be made and current formations can be broken during the scenario advancement phase. The trainer can also add "narrative/opnote" or "event" text for time stamping events into the scenario.

The scenario can also be played in "freeplay" mode, that is, advanced automatically by the computer, using the computer clock to generate time-stepped advances. We call this option Scenario Freeplay, and tracks can be allowed to move along their courses at a real-time pace based on the computer clock, and faster or slower than real-time based on a timing ratio and update time step (Fast Motion and Slow Motion options). Changes in navigational data (course, speed, altitude, depth) can be easily entered as the freeplay is running, and their effect will take place at the next update time step. The freeplay can be paused and restarted at any time in the scenario generation. In addition, at any time, the scenario generated so far can be quickly reviewed (Review option) and saved in a file (Save).

When a scenario is determined to be complete, the simulated track data can be saved in an OTH-T GOLD formatted Scenario File and then replayed when the training session is run. The scenario filename must be given, along with any timing constraints, in the Replay Setup portion of the Training Session Generation program. Here, the scenario is linked to a particular frame in the session, and if the training is to be done at a real-time pace, then real-time replay is set. Otherwise, a faster than real-time pace is "set" for track movements between the frame "freeze" points, where further instructions are followed or other trainee actions can be taken. Also, scenario files can later be manipulated by editing the simulated track data, merging multiple scenarios, or merging a scenario with real reconstructed track data occurring in the same time period. These merges of reconstructed tracks with additional tracks in the scenario can be used to present, during exercise debrief, the actual events and ground truth track movements that occurred, where the track data that were reported in real-time may have been incomplete or in error.

### **Automated Training Sessions and Presentations**

When a completed training session or presentation is going to be observed by a large group of people, such as in a classroom situation or staff briefing, it may be desirable to have the session run automatically without operator intervention. A single operator or trainee may also use this automatic capability to get a quick overview of the material without actually following the session instructions. This option is available in its initial stages in the Training Module's Run Training Session program. Each frame is displayed for a fixed amount of time before the session advances automatically to the next frame (as if the user pressed the Next button). Images have their own designated amount of time for display. These time amounts are easily changed as the session is running by using the slider bars to speed up or slow down the running of the session.

### **Recording of Operator Actions**

For a trainee to be "graded" on his/her performance in completing a training session by comparing actions with the set of correct responses for that session, it is useful to be able to record all the operator interactions while on the system. In addition, this is useful for the trainer, who can record all correct actions for a session and then play back this "recording" during a training session to show the trainee what should be done. This recording could be used by the trainer in the beginning of a session to introduce the trainee to the material, or it could be used at the end of the session for the trainee to check his/her results or actions against the "correct" ones, just as correct answers are at the end of the chapter in text books.

The JMCIS Macro Recorder, as it is currently known, is under development to fulfill these requirements. Similar to Macros on personal computers, this recorder can save all operator actions, from menu selections to button presses to keystrokes, from the moment the recorder is started to the

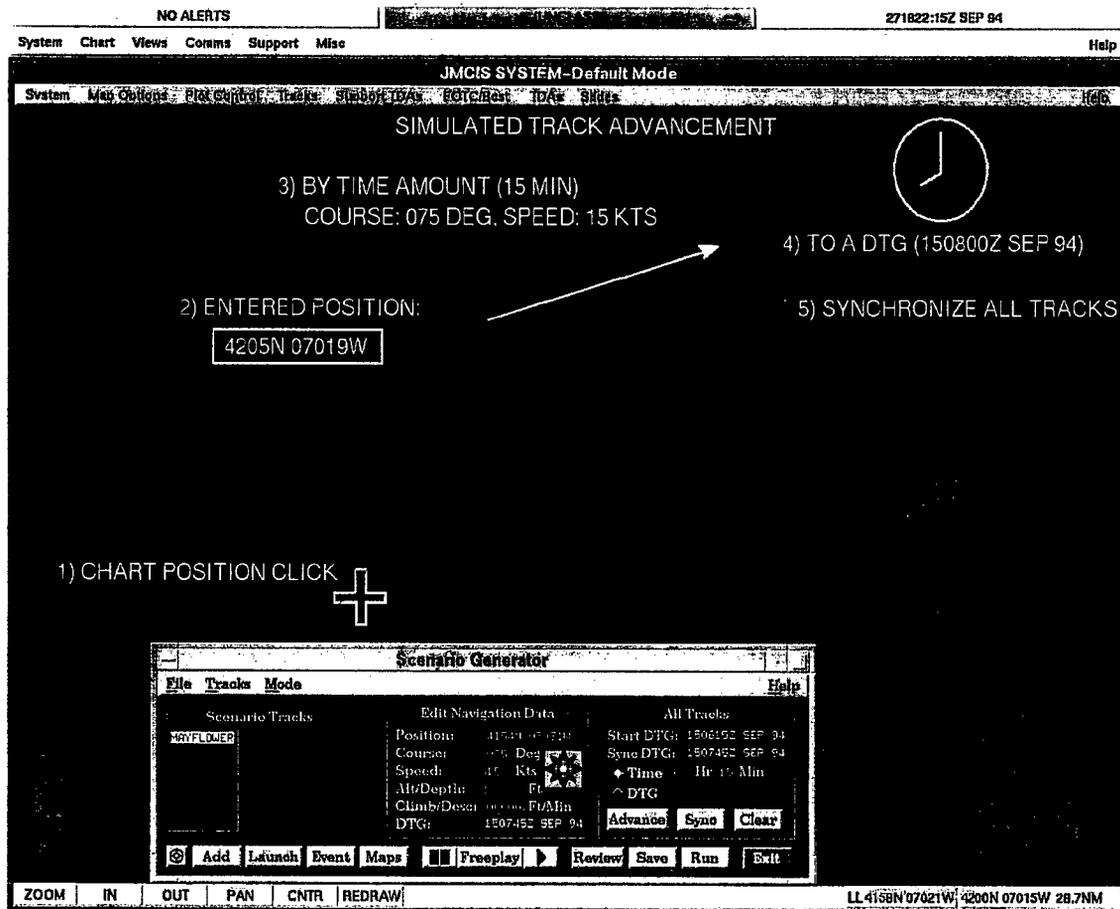


Fig. 12 — Scenario Advancement

moment it is stopped. The recorded actions are saved in a file in a special recorded information format, and that file is later selected to play back the recorded actions. During this play back, all JMCIS and module functions that were executed during the recording are actually re-executed automatically in the play back, with realistic pauses and response times between actions.

## RESULTS

The Training Module has been used to create many presentations and demonstrations to the Navy Fleet and C2 software development community, with very positive feedback. Some have taken the task of creating usable training and presentation sessions for various activities, such as Navy exercise analysis and reconstruction, laboratory training at Fleet training centers including real-time track management, coordination, and data fusion, and for the C4I Staff Watch Officer Course. In addition, the Training Module has been integrated and tested with the NTCS-A and JMCIS, along with the Reconstruction Module, and it is currently installed aboard most Navy combatants, including all that have NTCS-A version 2.0.10.5 or later, or any version of JMCIS.

### Training Session for the Reconstruction Module

A training session has been developed for those involved with exercise reconstruction and analysis. The session will introduce the analyst to the Reconstruction Module in NTCS-A and JMCIS, and demonstrate its usefulness and interaction with the Warfare Assessment Module (WAM), developed at the Naval Warfare Assessment Division in Corona, CA. Both the Reconstruction Module and WAM use archived track data and present a dynamic replay of track movements on the map with

event annotations. However, the most common input data of each program is different, so conversion software is used to allow WAM data to be replayed and displayed by the Reconstruction Module, and to allow archived OTH-T GOLD data from NTCS-A and JMCIS to be replayed and displayed by the WAM. This training session will show the analyst how to use the two modules together to get a more complete reconstruction, based on track data collected in real-time, and track data determined later to be "ground truth," based on more complete information. By running the Reconstruction Module and the WAM side by side, comparisons can be made between the analyzed "ground truth" track data and the track data seen by the Battle Group Commander in real-time during the exercise. Instructions are given for how to collect, merge, and edit the track data, how to convert from one input data format to the other, and how to use the various display and track and event history options in the Reconstruction Module. Also included are some lessons on basic NTCS-A and JMCIS features and how to create a final presentation for a post-exercise debrief.

### **Training Sessions for NTCS-A and JMCIS Functions**

Training sessions have been developed for teaching how to use various functions of the NTCS-A and JMCIS systems. The user's manuals for these systems have been used as a guide to get started, as well as the training manuals written and used at Fleet training centers, such as the Tactical Training Group Atlantic and Pacific, at Virginia Beach, VA. Sessions for the Force Over-the-horizon Track Coordinator (FOTC) have been developed to instruct operators on the use of the many FOTC-related functions in NTCS-A and JMCIS. Training sessions have also been created for other modules and applications being developed at NRL, namely, the Reconstruction Module, the Operations Deception Module, and the Scenario Generator.

### **Training Sessions for the Staff Watch Officer Course**

Training sessions are currently being developed for the Staff OTH/C4I Watch Officer Course (SWOC), a training course given at the Tactical Training Group Atlantic. This course has been used to train staff watch officers on the NTCS-A system (and later JMCIS). It is taught with a lab workbook, with five labs on general system topics, which describe the instructions for the relevant system functions. The lab workbook is being converted into a series of on-line training sessions that will be done in a particular order. The first completed lab session, "Lab #1 Display Options," will familiarize the SWO with some basic NTCS-A and JMCIS display options, focusing on Map Options and Plot Control Options. The second lab session, "Lab #2 Support Options," focuses on the battle management support features and the tactical decision aids (TDAs) available under the Tracks, Support TDAs, and TDA menus of NTCS-A and JMCIS. Dynamic training scenarios, not available with the lab workbook, are being included with these sessions to simulate a more realistic operating environment. For example, the second lab session contains a small simulated battle group that crosses the Mediterranean. A search and rescue (SAR) plan using the Expanding Square pattern has been created by the SWO trainee with the SAR Planning TDA, as shown in Fig. 13. In the scenario, a search aircraft is launched by the *Kennedy* to assist the *Mississippi* in locating a missing helicopter. The aircraft follows the search pattern, which was set by the SWO trainee, to locate the helicopter before returning to the *Kennedy*, thus adding an element of dynamics and realism to an otherwise static session. Dynamic scenarios like this will be used in the rest of the labs for the SWOC, as well as for other relevant training materials that will be converted. The addition of dynamics to training sessions makes them more interesting and requires more focused trainee involvement.

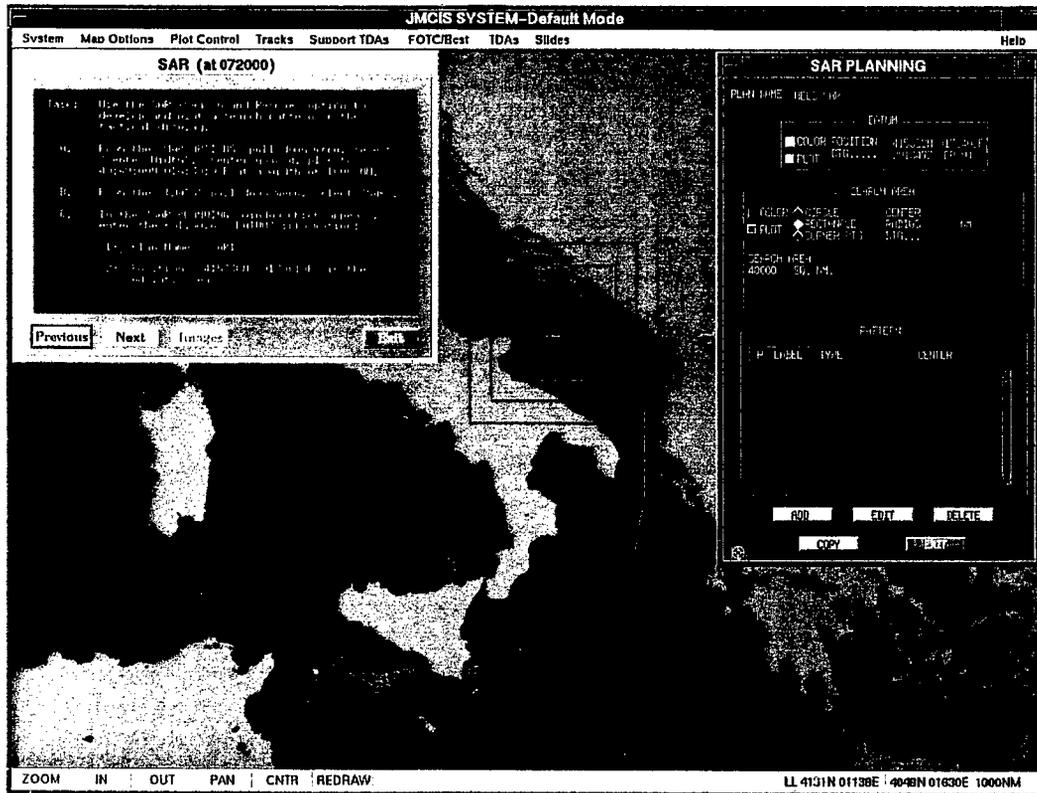


Fig. 13 — SAR Planning in the Staff Watch Officer Course Training Session

## FUTURE DEVELOPMENT AND APPLICATIONS

### Training Module Enhancements

It has become clear with the initial use of the Training Module that occasions will exist for trainees to stop working in the middle of sessions, either to take a break, or at the end of the work day. There needs to be a way of making any point in the session directly accessible, to allow the trainees to come back to where they left off. The trainees will merely open the sessions they were running previously, and instead of starting by pressing Next in the first frame, they will select from a list of frame titles to go directly to their desired starting point. This enhancement has been designed and is relatively straightforward to implement.

Another feature being discussed is the organization of Training Session Directories that will store sets of training sessions according to the application or type of system module or function to which they pertain. JMCIS is organized into two major menu structures, and it would make sense to have the training sessions for JMCIS functions and TDAs organized in the same way. For example, if the trainee is going to run a session on the "Pimtracks" option under the "Support TDAs" Chart level menu, then to select the correct session, the trainee would select the "Chart Menu" directory, then the "Support TDAs" directory, and finally, the "Pimtracks" training session. This way, as more training sessions are created for the system, the trainee will be able to go quickly to the desired training topic without having to scroll through a long list of sessions for the whole system.

Probably the most important enhancement that has been requested by users is some form of remedial checking or grading system for the training sessions. Just as the lab workbooks of the SWO Course have Skill Tests at the end of each lab session, the Training Module needs features that allow the trainees to be tested on their knowledge of the material just covered. This implies that either a set

of "results" or "correct answers," depending on the training topic or function, needs to be set up in advance by the trainer (i.e., creator of the training sessions) and stored in the system. Also, trainees need a way of checking their work against these results. One method currently available in the system is to provide visual feedback through the use of the JMCIS slides. Here, a JMCIS function window with correct entries or an image of the results of trainee actions on the Chart are displayed to the trainee with a previously created JMCIS slide. However, this does not provide any real automatic checking within the system; it is merely visual. Eventually, a thorough method of testing the trainees at the end of sessions will be necessary, with direct feedback on their performance, and these methods are currently being designed.

### **Related Training Applications**

Several more advanced applications come to mind when considering all the features available with the Training Module and Scenario Generator. A team training application is an obvious next step. Applications for simulated enhanced exercises and dynamic plan simulation are also natural outgrowths of the embedded training module concept. In most of these areas, it is necessary to have a computer-linked network for training, separate from the network for real-time operations, plus a way to have multiple scenario generators running simultaneously to simulate actual movements and actions.

The Scenario Generator developed at NRL can be set up to allow all simulated tracks to appear on all displays in one local area network (LAN) mode, which is useful for team training aboard one platform. Taking this one step further to a wide area network (WAN) mode, several players on several platforms could generate simulated tracks that would appear on all displays of the WAN. Plans are currently under way to design a Virtual Training Network within the JMCIS architecture to allow these types of simulations. All track updates and tactical training events would be communicated to all the players in a team training environment, with automatic synchronization of time critical events across the network.

### **Multiple Scenario Generators for Team Training**

For team training applications to be possible for a single platform, it will be necessary to have multiple scenario generators running simultaneously on the LAN aboard ship and synchronized to one master clock. A team member at each display terminal could be controlling a certain group of assets in the training scenario, such as Blue Ships or Red Air, as in the over-simplified illustration in Fig. 14. For team training among a group of platforms, a high-speed training WAN needs to be implemented for synchronizing all the Scenario Generators on the network and passing the track updates and the array of possible scenario events among the various platforms. The referees, or neutral members in this training environment, can be generating a predesigned scenario of simulated tracks, or Background Scenario, and all the team members can then respond interactively as events unfold. This background scenario will be the one that controls the master clock (in case the clocks on the separate computers on the training network are slightly off), and will provide for the synchronization of all team members' events and actions. The team members will have their own Scenario Generator (SG1, SG2, SG3, SG4 in Fig. 14) in Freeplay mode, and will enter changes in navigation data (course, speed, altitude, exact position, etc.) as the scenario advances. A lot of design work needs to be done before realizing these team training goals, but the ground work has been laid with the Training Module and Scenario Generator.

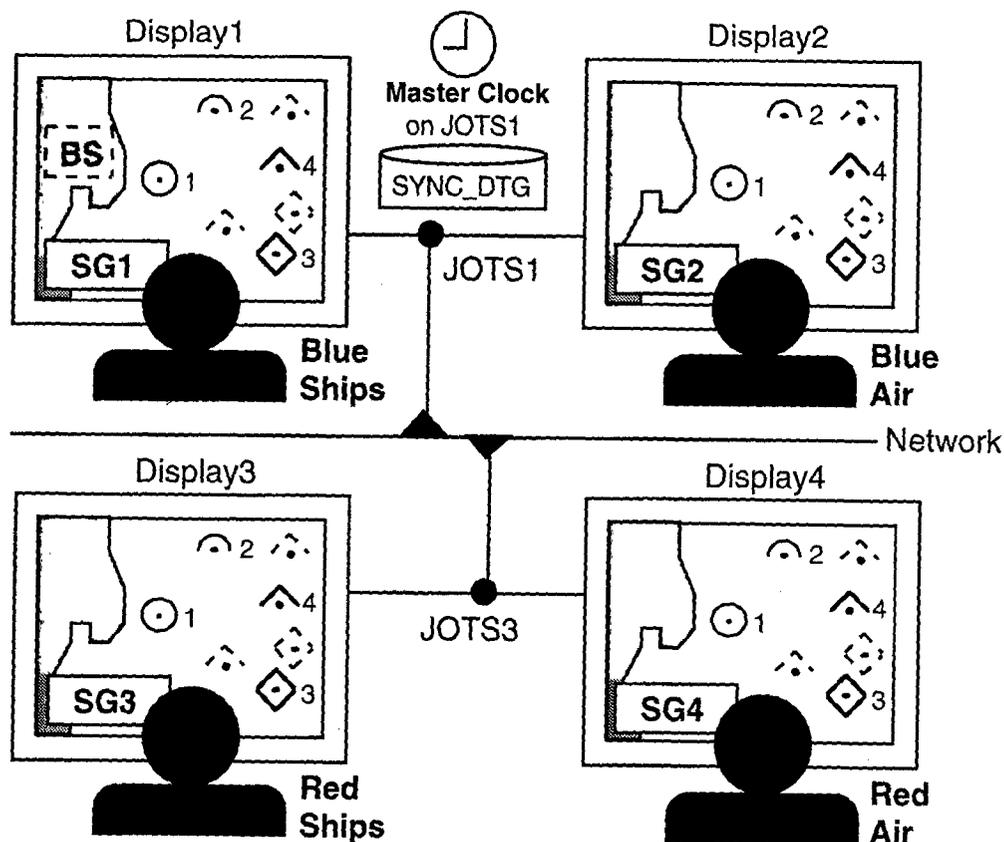


Fig. 14 — Multiple Scenario Generators for team training

### SUMMARY

Embedded training tools are necessary in the large real-time systems of today and the future. It is evident that embedded training will eventually be very commonplace in these systems. An important criterion is that the embedded training itself be part of the design process when building new large software systems. The designers should have in mind how the system will be used and how system operators will be trained so that the training tools can be embedded intelligently and the user interfaces for training will be user-friendly and straightforward. As long as the training module software does not interfere with the real-time operations, complex systems will be easier to learn. Because of the student interaction on an actual operational system, the learning curve will be greatly reduced. This should effectively lower the overall costs of training new operators.

In the Navy C2 arena, we have developed a comprehensive package of training and presentation tools that will enhance how the Navy does business in many areas, from Fleet exercises and operational preparedness to training schools and command briefings. The Joint Maritime Command Information System and the Navy Tactical Command System-Afloat are complex systems, with new functionality and tactical decision aids being planned and added. The training of these new options needs to keep pace with the speed of development and with the advancement of computer technology. The Training Module developed at NRL is a very big step in this direction, with room for extension built into the module's design and its associated command scripting language PAL, to accommodate more sophisticated training requirements of the future.

## ACKNOWLEDGMENTS

The author acknowledges the contributions of NRL colleagues George M. Bulgin and Dennis T. Baba of Kaman Sciences for their insight, discussions during design, and software development expertise in the implementation of the Training Module and Scenario Generator software. The author also acknowledges Gene E. Layman, John R. Robins, and John T. Egan for their help in design and development and for their overall knowledge and expertise in the areas of Navy C2 and system training requirements.

## REFERENCES

1. G.E. Layman, "Embedded Training Methods For Command and Control Systems," *NRL Review*, 1994, pp. 168-171.
2. D.P. McGroder, "Graphics-based Software for Embedded Event Reconstruction in Navy Command and Control Systems," NRL Report 9365, November 1991.
3. "A Beginner's Guide to HTML," National Center for Supercomputing Applications / pubs@ncsa.uiuc.edu, September 1994. See the Internet World Wide Web (computer-based) reference: <http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>.
4. "Training Module User's Manual," developed by NRL, compiled and prepared by Command and Control Warfare Group Atlantic (CCWGRULANT), Norfolk, VA 23520, November 1994.
5. JMCIS Common Operating Environment (COE), Inter-National Research Institute, Inc. (INRI), Reston, VA 22091, February 1994.
6. JMCIS Integration Standard (IS), Navy Command, Control and Ocean Surveillance Center Research, Development, Test & Evaluation Division (NRaD), San Diego, CA 92152-5000, February 1994.
7. K. Fernandes, "User Interface Specification (UIS) for Navy Command and Control Systems," Version 1.3, Navy Command Control and Ocean Surveillance Center, San Diego, CA, December 1993.
8. JMCIS Application Programmer Interface (API) Manuals, Inter-National Research Institute, Inc. (INRI), San Diego, CA 92131, December 1993.
9. JMCIS User's Manuals, Inter-National Research Institute, Inc. (INRI), San Diego, CA 92131, January 1994.

**GLOSSARY**

<b>API</b>	Application Programmer Interface
<b>COE</b>	Common Operating Environment
<b>C2</b>	Command and Control
<b>DTG</b>	Date Time Group
<b>GUI</b>	Graphical User Interface
<b>HTML</b>	Hyper Text Markup Language
<b>JMCIS</b>	Joint Maritime Command Information System
<b>JOTS</b>	Joint Operational Tactical System
<b>LAN</b>	Local Area Network
<b>NTCS-A</b>	Navy Tactical Command System - Afloat
<b>OTH/C4I</b>	Over The Horizon / Command, Control, Communications, Computers, and Intelligence
<b>OTH-T GOLD</b>	Over The Horizon Track GOLD
<b>PAL</b>	Presentation Authoring Language
<b>SAR</b>	Search and Rescue
<b>SWO</b>	Staff Watch Officer
<b>SWOC</b>	Staff Watch Officer's Course
<b>TDA</b>	Tactical Decision Aid
<b>TDB</b>	Track Database
<b>Tdbm</b>	Track Database Manager
<b>UIS</b>	User Interface Specification
<b>WAM</b>	Warfare Assessment Module
<b>WAN</b>	Wide Area Network



## Appendix A

### PAL COMMAND DESCRIPTIONS

#### Frame Definition Commands:

**FRAME**            FRAME/

Command to indicate start of a frame, which may include TEXTWINDOW annotations, IMAGE graphic overlays, highlights, replay actions, map settings, and timing functions. The possible "Objects" within a FRAME include TEXTWINDOW, ARROW Highlight, Area Highlights such as CIRCLE, RECT, and POLY Highlights, TRACK Highlight, SYMBOL Highlight, and IMAGE. The possible "Actions" within a FRAME include REPLAY, MAP settings, SPAWN of other modules, and timing functions such as FREEZE, PAUSE, and TIMEOUT.

**ENDFRAME**        ENDFRAME/

Command to end a frame. Everything between FRAME and ENDFRAME will be displayed on the screen at one time.

#### Text Object Commands:

**TEXTWINDOW**      TEXTWINDOW/TextWindowNumber/

Command to start a text window definition. This command must be followed by a TITLE, POS, SIZE, and TEXT command. Give the Text Window Number.

Example: TEXTWINDOW/5/

**TITLE**            TITLE/TextWindowTitle/

Command to set the Title of the current text window.

Example: TITLE/Introduction to Scenario Generation/

**POS**              POS/ScreenX/ScreenY/

Command specifying the X,Y position at which to place the current training text window on the screen.

Example: POS/500/200/

This will cause the training text window to pop-up near the top and center of the screen.

**SIZE**                    SIZE/NumberChars/NumberLines/

Command to set the size of the current text window, as the Number of Characters (width) by the Number of Lines (height).

Example:    SIZE/55/12/

**TEXT**                    TEXT/TextWindowLine[1]  
                              TextWindowLine[2] ...  
                              TextWindowLine[n] /            where n <= 20

Command for (a) a text window of instruction steps in a training session or (b) annotations to the events of a reconstruction or presentation. The text is created from the succeeding lines in the command, and terminated on the last line with a "/" (slash). The position in which to place the text window during the training session or presentation is an X,Y screen position as given in the preceding POS command, which is set by the user when window placement is done during session creation. The size, in number of characters by number of lines, is given by the preceding SIZE command.

Example:    TEXT/This Scenario Generation capability of the Training  
                  Module provides the ability to create dynamic training  
                  scenarios that simulate ship, air, sub, and land track  
                  movement on the map display and generate events pertinent  
                  to the training./

#### Highlight Object Commands:

**ARROW**                    ARROW/ObjectNumber/Direction/Lat/Lon/Color/  
                              where Direction = N, E, S, W, NE, SE, SW, or NW

Command to draw a filled arrow of given Color and Direction at a given Lat/Lon position on the Chart. Give arrow Object Number.

Example:    ARROW/3/NE/3800N/07700W/blue/

This will draw a blue arrow pointing in the North East direction at the chart position 3800N 07700W. The object number 3 indicates that it is the third object defined in the object list for the current frame in the session.

**CIRCLE**                    CIRCLE/ObjectNumber/CenterLat/CenterLon/Radius/Color/

Command to draw a filled circular area centered at a given Lat/Lon with given Radius in nmi (nautical miles) and Color. Give circle Object Number.

Example:    CIRCLE/2/3400N/12030W/500/orange/

This will draw an orange circle with a center at 3400N 12030W (near Southern California) with a radius of 500 nmi. This circle is the second object within the object list for the current frame.

**RECT**           RECT/ObjectNumber/Lat1/Lon1/Lat2/Lon2/Color

Command to draw a filled rectangular area with the top left at Lat1/Lon1 and the bottom right at Lat2/Lon2. Give rectangle Object Number and Color.

Example: RECT/2/3400N/12030W/3200N/13000W/cyan/

This will draw a cyan rectangle with the top left corner at 3400N 12030W and the bottom right corner at 3200N 12030W.

**POLY**           POLY/ObjectNumber/NumPoints/Lat1/Lon1/.../LatN/LonN/Color

Command to draw a filled polygon area with given Number of Points and given Lat/Lon positions and Color. Give polygon Object Number.

Example: POLY/1/5/3800N/07700W/3800N/07600W/3700N/07530W/3700N/07400W  
/3730N/07700W/yellow/

This will draw a yellow polygon with five points and five sides. It is the first object in the object list.

**TRACK**           TRACK/ObjectNumber/TrackName/Color/

Command to highlight a particular named track with a larger track symbol font, which may be a different color. Give track highlight Object Number and Color.

Example: TRACK/4/MOUNT WHITNEY/green/

This will highlight the MOUNT WHITNEY track on the chart with a green track symbol (larger than the MOUNT WHITNEY symbol). This highlight is the fourth in the object list.

**SYMBOL**           SYMBOL/ObjectNumber/SymbolType/Lat/Lon/Color/

Command to draw a specific symbol at a given Lat/Lon position, which may be any color. Give symbol Object Number, SymbolType, and Color.

Example: SYMBOL/7/MarineHelicopter/4200N/01600E/white/

This will draw a white Marine Helicopter symbol at 4200N 01600W. It is the seventh object in the list.

#### Image Object Command:

**IMAGE**           IMAGE/ImageType/ImageFilename/ImageFileFormat  
/ScreenX/ScreenY/

where ImageType 0 = Stored Slide (JMCIS Screen Image)  
ImageType 1 = Graphical Image (Photo, Illustration)

Command to refer to any graphics drawn in a JMCIS window from any application, such as photographs, charts, diagrams, geographical areas, coverages, track histories, PIM tracks, and overlays,

that are saved in an image file (graphics file). Give the Image Type, the Image Filename, and the Image File Format (e.g., TIFF, GIF, xwd, bitmap, etc.).

Example: IMAGE/1/E2C\_WITH\_ESCORT/xwd/700/400/

This will cause a digital photograph stored in the xwd-type image file "E2C\_WITH\_ESCORT" to pop-up on the screen at X=700, Y=400 (near the lower right of the screen).

### Timing Action Commands:

**FREEZE**            FREEZE/

Command to tell the replay process within the training session or presentation to "freeze" (stop running). It could cause a window to pop up that says "PRESS TO CONTINUE" with a CONTINUE button in order to play through the rest of the presentation or session.

**PAUSE**            PAUSE/NumberSeconds/

Command to tell the replay process within the training session or presentation to pause for a given Number of Seconds, before continuing.

Example: PAUSE/10/

**TIMEOUT**        TIMEOUT/Object/ObjectNumber/NumberSeconds/

Command to make a particular object in this frame be displayed, such as a TEXTWINDOW comment, or an ARROW, TRACK, or Area Highlight, for a fixed Number of Seconds and then disappear (time out). "Object" is the type of object followed by the Object Number as specified in object creation.

Example: TIMEOUT/ARROW/3/20/

This will cause the arrow highlight defined in the ARROW command example above (object 3 in the object list) to disappear after 20 seconds.

### Process Spawn Action Commands:

**SPAWN**            SPAWN/ModuleName/Arg1/Arg2/.../ArgN/

Command to spawn another module or program (process) from within the current frame. Give the Module Name and any command line Arguments.

Example: SPAWN/TRAINscenarioPART/-role/BlueShips/-file/BLUESHIPS.GLDS  
          /-dtg/041200Z NOV 94/-nd/

This will spawn the TRAINscenarioPART program (Training Scenario Participant) and pass the arguments for -role as "Blue Ships," -file as "BLUESHIPS.GLDS," -dtg as "041200Z NOV 94," and -nd (which the spawned program defines as NO DATA BASE).

**REPLAY**           REPLAY/Filename/FreezeDTG/Ratio/Delay/TimeOrder  
                  /UseTdbm/UpdateTdbm/PlotSymbols/PlotAltitudes  
                  /PlotHistories/HistoryLength/UseBitmaps/ShowLOS/

Command to spawn the Reconstruction Module's Replay option from within the Training Module. This command is inserted in a training session file or presentation file to do a replay of real operations or an exercise or a replay of a scenario of synthetic tracks, or a combination thereof. Replay Ratio and later arguments are optional. Give the command line arguments for: the Replay Filename or Scenario Filename, and optionally a Freeze DTG, and the three replay timing arguments: Replay Ratio (e.g., 60.0 for 60:1), Replay Delay, that is, the Number of Seconds to delay between each message (e.g., 2.0 for 2 seconds), and Replay Time Order: 0 (by MSG DTG) for replay by message date-time-group or 1 (by Position Time), which causes sorting according to the position time. Also give the TDB options: UseTdbm: 1 if the tracks are to be saved in the TDB at all, 0 otherwise, UpdateTdbm: 1 if the tracks are to be updated and plotted by the Tdbm for each report in the replay, 0 otherwise, and PlotSymbols: 1 if track symbols are to be plotted by the REPLAY process during replay, 0 otherwise (i.e., letting Tdbm do the plotting). Then give the other display arguments: PlotAltitudes: 1 if altitude lines are to be drawn for air tracks with altitude data, 0 otherwise, PlotHistories: 1 if track history lines are to be drawn during the replay, 0 otherwise, HistoryLength: 0 if ALL history points should be included, 1 if 15 points, 2 if 10 points, 3 if 5 points, and 4 if only one most recent point, UseBitmaps: 1 if bitmap-based icons are used for track symbols, 0 otherwise, and ShowLOS: 1 to display the Line of Sight circles for air tracks, 0 otherwise.

Example: REPLAY/Filename/FreezeDTG/Ratio/Delay/TimeOrder  
          /UseTdbm/UpdateTdbm/PlotSymbols/PlotAltitudes  
          /PlotHistories/HistoryLength/UseBitmaps/ShowLOS/

**MAP**               MAP/Lat/Lon/Width/

Command to tell the Chart Server to change the map during a training session or presentation. The Latitude and Longitude values define the center of the map, and the Width value defines the width (in nmi) of the map to be displayed on the screen.

Example: MAP/4200N/07600W/600/

**CHANGEMAP**       CHANGEMAP/On\_or\_Off/

Command to toggle the CHANGEMAP flag On or Off for the session, to tell whether subsequent frames will cause a changed map when a new MAP command is given; 1 for On, 0 for Off is.

Example: CHANGEMAP/1/

## Appendix B

### PAL AS HYPER TEXT MARKUP LANGUAGE (HTML) EXTENSIONS

What follows is a list of the commands, given in Appendix A as converted to the suggested PAL extensions to HTML. This will provide compatibility between the Training Module's sessions and presentations and the sessions and documents produced with HTML by presentation, document production, or other multimedia applications (such as Mosaic). By replacing the slashes of the original PAL with spaces and surrounding the commands with the left and right brackets ( < and > ) of the HTML, the basic command syntax will be consistent. Any PAL extensions would be considered "foreign" to other HTML-based applications (at least initially) and would simply be ignored without causing any errors.

Add to these the commands for Hyper Text Links, or Anchors, plus those for Video and Sound, then all other text formatting tags in HTML, and we will have a comprehensive set of commands to define any presentation or training session that has multimedia capabilities, using Hyper Text functionality, with map and replay dynamics. In addition, after formatting enhancements are made to the Training Module, the HTML-based output from other applications will be readable. Take note of the special comments (in *italics*).

#### COMMAND

KEY WORDS:           USAGE:

#### Frame Definition Commands:

FRAME               <FRAME>  
ENDFRAME           <ENDFRAME>

#### Text Object Commands:

TEXTWINDOW               <TEXTWINDOW TextWindowNumber>  
TITLE                    <TITLE>TextWindowTitle</TITLE>    (*already defined in HTML*)  
POS                      <POS ScreenX ScreenY>  
SIZE                     <SIZE NumberChars NumberLines>  
TEXT                     TextWindowLine[1]  
                          TextWindowLine[2] ...  
                          TextWindowLine[n]                    (*Free Format Text*)

#### Highlight Object Commands:                    (*Assuming a system map display*)

ARROW                    <ARROW ObjectNumber Direction Lat Lon Color>  
                          where Direction = N, E, S, W, NE, SE, SW, or NW  
CIRCLE                   <CIRCLE ObjectNumber CenterLat CenterLon Radius Color>  
RECT                     <RECT ObjectNumber Lat1 Lon1 Lat2 Lon2 Color>  
POLY                     <POLY ObjectNumber NumPoints Lat1 Lon1..LatN LonN Color>  
TRACK                    <TRACK ObjectNumber TrackName Color>  
SYMBOL                   <SYMBOL ObjectNumber SymbolType Lat Lon Color>

**Image Object Command:** *(already defined in HTML)*

**IMAGE** <IMG SRC = ImageFilename or URL>  
where URL = Universal Resource Locator  
We may need to add our own extension for these fields:  
ImageType, ImageFileFormat, ScreenX, ScreenY  
where ImageType 0 = Stored System Slide (Screen Image)  
ImageType 1 = Graphical Image (Photo, Illustration)

**Timing Action Commands:**

**FREEZE** <FREEZE>  
**PAUSE** <PAUSE NumberSeconds>  
**TIMEOUT** <TIMEOUT ObjectType ObjectNumber NumberSeconds>

**Process Spawn Action Commands:**

**SPAWN** <SPAWN ModuleName Arg1 Arg2 ... ArgN>  
**REPLAY** <REPLAY Filename FreezeDTG Ratio Delay TimeOrder  
UseTdbm UpdateTdbm PlotSymbols PlotAltitudes  
PlotHistories HistoryLength UseBitmaps ShowLOS>  
**MAP** <MAP Lat Lon Width>  
**CHANGEMAP** <CHANGEMAP On\_or\_Off>