Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

A development methodology for adding map-based graphics to the theater war exercise

Darrell A. Quick
Mark A. Roth

Air Force Institute of Technology
Department of Electrical and Computer Engineering
Wright-Patterson AFB, OH 45433-6583

ABSTRACT

The Theater War Exercise (TWX) is a computer-assisted, theater level airpower employment exercise conducted by the Air Force Wargaming Center. Its primary purpose is to give senior level military officers a feel for the intricacies of high level decision-making in a combined air/land conventional warfare situation. Until recently, output from TWX consisted of a large number of tabular hard-copy reports which were difficult to understand and cumbersome to use. This paper discusses the development of a map-based graphical interface to TWX which provides a much more user-friendly interface to the information and provides a baseline for the development of new techniques for user interaction with the exercise. The methodology and toolset developed for this project are of particular interest because of their applicability to a wide variety of other wargaming and simulation systems.

1. THE THEATER WAR EXERCISE

TWX was developed by the Air War College between 1976 and 1977 in response to a directive by the USAF Chief of Staff to develop courses to train senior military officers "in the threat and application of force" (Theater Warfare, 1987). In order to avoid the constraints imposed upon projects involving classified information, the data and algorithms used were taken from unclassified sources. Even so, TWX provides a useful forum for senior officers to study the application of air warfare principles in a simulated environment. Particular attention was given to the goal of allowing students to gain some familiarity with the application of USAF airpower employment strategies and doctrine in a theater level combined air/land conflict. In addition, emphasis was given to simulating the difficulties of coordination between U.S. and allied forces in a credible manner, and of planning and providing logistics support. Finally, the exercise is designed to instill in the participant an understanding of the problems involved in obtaining and processing the information necessary for timely and informed decision-making in a wartime environment (Theater Warfare, 1987).

From its inception in 1977 until last year, TWX was run on a Honeywell 6000 series mainframe computer using application-specific files to hold the database. It was heavily revised last year by two students at the Air Force Institute of Technology (Brooks, Kross and Roth, 1987). The revisions included rehosting the system from the Honeywell to a DEC MicroVAX using Zenith Z-158 microcomputers as remote terminals. In addition, Brooks designed and implemented a relational database management system (DBMS) using the commercial INGRES database product (Brooks, 1987), and Kross developed an interactive menu-driven front end system using the INGRES fourth generation language (4GL) facility (Kross, 1987). These revisions, besides improving the flexibility and maintainability of the system, provided a much friendlier input interface for the user than the hardcopy terminals which had been used previously, and sped up data entry by allowing interactive editing of input.

Unfortunately, output from the system still consisted entirely of a large number of massive, hard-to-decipher reports. The tactical overview, or FLOT (Forward Line Of Troops) Plot, was particularly difficult to use; it consisted of a number of hardcopy sheets which could be separated and laid side-by-side on a table and overlaid with a clear plexiglass sheet marked with country boundaries. The hardcopy sheets were marked with crude symbols indicating the current position and affiliation of military units and bases. To get additional information about a base or unit, or weather status in a country, a player had no recourse but to dig through voluminous reports. These activities were awkward and detracted from the primary goal of the exercise, which was to allow students to make high-level decisions and evaluate the impact of those decisions. The graphical interface described in this paper was developed specifically to counter these shortcomings of the system and to allow the players to concentrate on decision-making and analysis.

TWX is generally conducted over a four-day period as part of the Combined Air Warfare Course at the Center for Aerospace Doctrine, Research, and Education (CADRE). It is also used at the Air War College, Air Command and Staff College, and by the Canadian Forces Command and Staff College and the Royal Air Force Staff College. The Air Force Wargaming Center (AFWC) maintains the exercise and provides run-time support. Two player teams are involved: the Red team, composed of faculty and AFWC personnel familiar with Warsaw Pact doctrine and strategies, and the Blue team, consisting of eight to ten students representing NATO forces. In addition, a third group, the White (or Control) Team, consists of a faculty game director

and data processing personnel who oversee the conduct of the exercise.

A typical cycle of play consists of the player teams analyzing the current scenario, revising their strategy in response to the situation, and entering their responses into the database. The batch portion of the system is then run. The batch system extracts the information from the database and processes it to determine the outcome. The database is then updated with the new scenario and reports are generated.

The TWX algorithms use a highly aggregated approach, determining the outcomes of conflicts based largely upon the relative strengths of the forces involved. In addition, the land battle portion of the game is largely deterministic, since the emphasis in this exercise is upon airpower employment. The students' interaction with the land game consists exclusively of their support of ground forces through Offensive Air Support and Air Interdiction missions.

The remainder of this paper is divided into three sections. Section 2 describes the functional capabilities of a proposed graphical interface for TWX. The third section examines the environment in which the project was conducted and the issues involved. This includes identification of major decisions and the rationale behind these decisions. In the fourth section we describe the prototype as it currently exists. The final section presents conclusions and recommendations for further work.

2. The Graphical Interface

As mentioned in the last section, the old system for displaying the tactical situation was very awkward. This section describes the features of the proposed graphical interface.

The revised output system will display a computer-generated on-screen map of the European theater showing such details as country boundaries, rivers, locations of military bases, and the Forward Edge of Battle (FEBA), or FLOT, on the system's remote terminals. Military bases and military units in the field are represented using symbols color-coded to indicate their affiliation. Locations of graphical items are maintained in the TWX database using a Cartesian Coordinate system, with each item's position being stored as a matched pair of x and y coordinates. A grid is superimposed over the map to assist users in determining their position in terms of these coordinates.

Other features include a method of indicating the weather forecast on the screen and the ability to zoom in and out and pan across the display. A keyboard menu display is shown on the screen to select amongst user desires. A mouse-driven cursor allows the user to select between different keys on the display. As an example of this, consider the use of the "Zoom" button; the user moves the mouse to position the cursor over this button, and then pushes the right or left mouse button to indicate his desire to zoom in on or out from the picture.

A number of user-selectable reports, such as a pop-up display of sortie status for a particular base, could be added to the system and made callable also from the keyboard menu. The system will allow multiple windows to be displayed at once, giving the user the ability to simultaneously view the status of two or more bases for comparison purposes. An advanced version of the system will include the ability to use the position of the cursor on the display to input position data into the database (for target selection, for example).

Figure 1 shows the envisioned final display, complete with an on-screen keyboard showing the user's options. The user will be able to turn on and off the visibility of various classes of items, such as the two different allied air forces, 2ATAF and 4ATAF. Weather conditions in various zones will be indicated by a pattern fill within each zone, which can also be toggled on and off. After selecting a particular base by placing the mouse cursor over it and pressing a button, the user can select between a number of reports (sortie summary, logistics status, etc.). This figure also shows the buttons which allow the user to indicate his desire to zoom in on the display.

The display will include a message area which gives feedback to the user in the form of instructions and error messages, and a status area indicating his current situation. Also included is a "Notes" option, which allows the user to submit a free-form critique at any point during the game. The ability to submit suggestions for improvements as they occur to the user should provide better feedback to the individuals who maintain and enhance the system.

The primary purpose behind these enhancements is to simplify the user's view of the interface with the computer system. This apparent ability to operate directly upon the symbols representing units and bases makes it easier and more intuitive for the user to find the exact information he needs.

3. DEVELOPMENT ISSUES

There are a number of issues involved in the development of a graphical interface that have to be dealt with. These include the identification of development constraints, the selection of an appropriate methodology, a number of decisions about how graphical objects are to be represented, stored, and displayed, and the selection and acquisition of needed hardware and software. The resolution of these issues is addressed in detail in this section.

3.1. Development Constraints

The Z-158s used in the current system as remote terminals are on the low end of the spectrum for graphics work. In addition, it had been discovered late in last year's research that the Z-158 microcomputers were not one-hundred percent compatible with the INGRES/NET software being used to link them to the MicroVAX. It is possible to circumvent this problem, but the

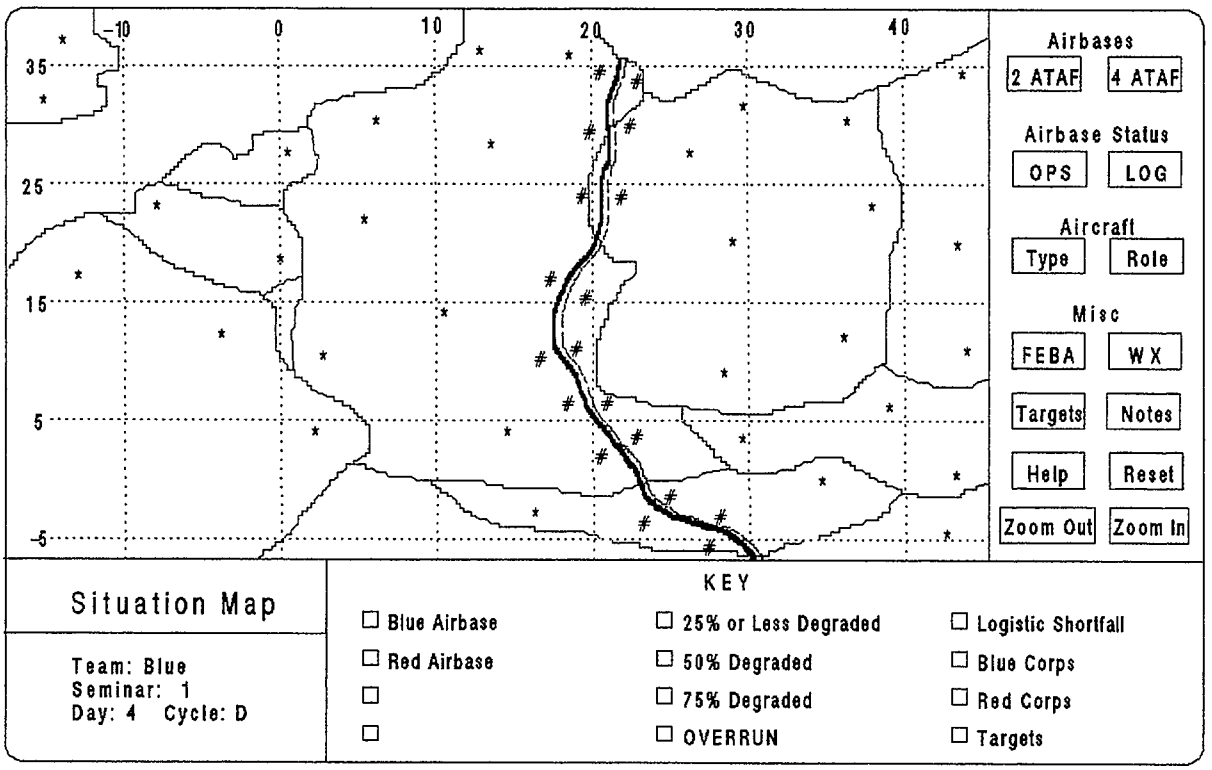

Figure 1: Envisioned TWX System Display

solution is awkward and degrades the performance of the system, as well as making the software more difficult to maintain. In addition, the system was programmed entirely in FORTRAN and INGRES 4GL. The graphics facilities provided by FORTRAN are very rudimentary while the INGRES 4GL is only useful for forms-based input and output. Fortunately, the users were aware of these facts already, and were prepared to provide a reasonable amount of funding for more advanced hardware and graphics software. It was, of course, desired that existing resources be utilized wherever possible.

The initial goal is to modify existing software as little as possible. In particular, no redevelopment is to be done upon the manner in which the system processed the information; the initial modifications are to be entirely upon the presentation of the information to the user. This would involve little or no change to the existing programs and database, since most of the necessary information was already maintained in the current system's database. Minor additions to the database might be needed to store graphical attribute information about existing objects, which might consequently necessitate modest changes to some programs, but these changes would be relatively trivial.

3.2. Development Methodology

TWX is only one of a number of computer assisted wargaming exercises conducted by the AFWC. Few of these systems provide map-based graphical interfaces at this time. AFWC personnel were interested not just in an interface for TWX, but a complete methodology and set of tools which could be used to develop flexible map-based graphical interfaces for existing and new wargaming systems. Therefore, a major goal of this research is to develop a reusable methodology involving procedures and hardware/software tools for developing graphical interfaces for a variety of wargaming systems. Although the primary emphasis was on wargaming graphics, the resulting methodology is general enough to be used for developing graphical interfaces for most simulation systems.

At least three popular methodologies offered certain benefits for this sort of development. Since a limited development time period is available for the project (as for most projects) a trade-off exists between incorporating as much functionality as possible within the allotted time and delivering a production quality system. An attractive solution is offered by the iterative design approach, in which successive versions of the system are built, each adding new features to the preceding version. This approach allowed enhancements to be made up to a cutoff date, at which time the product is finalized and packaged up. By building iteratively, there was little risk involved in each version, and there was always a stopping point readily available. On the other hand, the iterative approach takes longer to do the same amount of work because of the additional overhead of tying up loose ends after each version. It was decided that, for this situation, the bene-

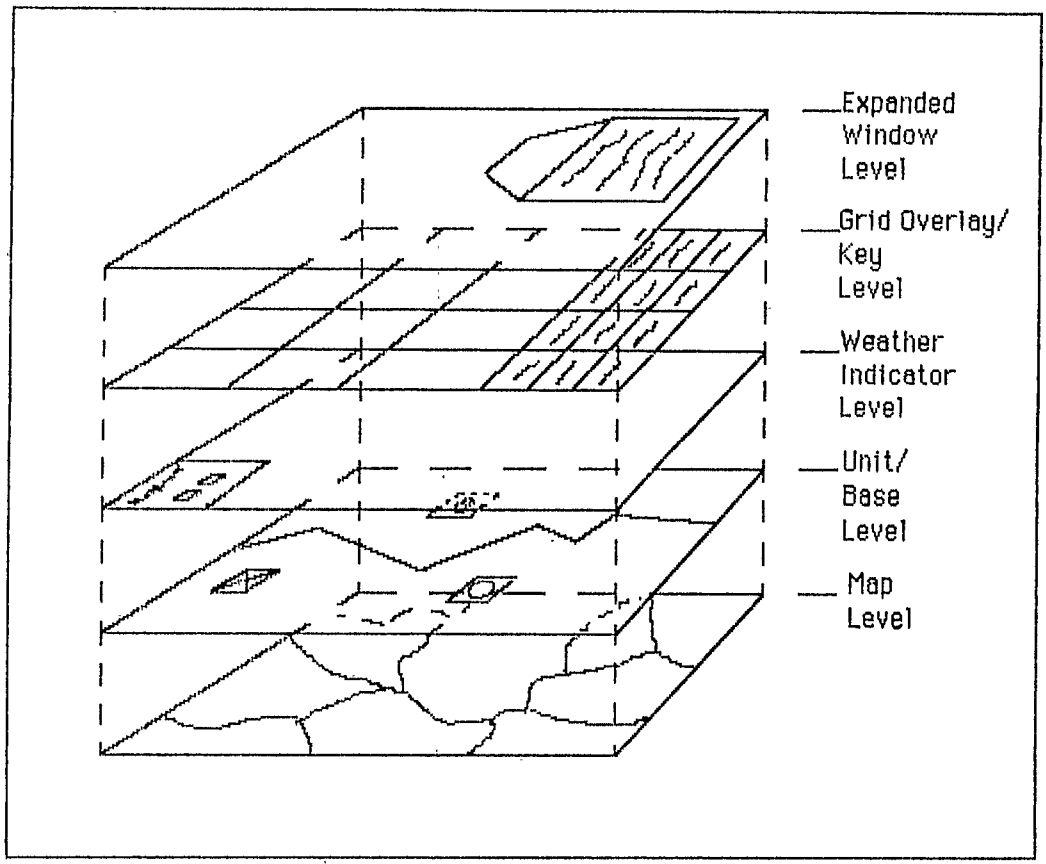

Figure 2: Conceptual Model of Display

fits of reduced risk and frequent stopping points outweighed the detriment of slower development speed, so iterative development was chosen as the overall framework for the project.

Within the context of iterative development, it is still necessary to settle upon a methodology for developing each version. The classical software life cycle approach, which involves functional specification, preliminary and detailed design, implementation, unit testing, integration testing, and system installation, is relatively standard for software projects and was not too difficult to tailor to fit within the context of iterative development. Functional specification and preliminary design are done at the beginning of the project to provide an overall framework. The functions to be implemented within each version are then identified and prioritized. Each version development involves a cycle of detailed design, coding, and testing. While the hybrid of these two methodologies is adequate for the job, it is also evident that certain aspects of this project lend themselves to an object-oriented approach.

Conceptually, it would be nice to think of the display as consisting of a group of objects superimposed upon one another, as opposed to a single two-dimensional picture in which each pixel was set to a certain color. Figure 2 shows a conceptual model of the display in which various graphical objects are aligned in two-dimensional planes according to their categories. These planes are then superimposed upon one another like a sequence of slides to present the overall display to the user's eye.

The natural hierarchy existing between symbols further supports an object-oriented approach. Conceptually, for example, the unit and base symbols can be thought of as a class of graphical objects, all possessing certain common characteristics such as a square outline. This class can possess subclasses each having common characteristics above and beyond those of the parent class. These subclasses will all, by default, inherit the characteristics of the parent class unless they specifically supercede them. This property, called inheritance, is one of the central characteristics of an object-oriented design. Inheritance will greatly simplify the design of this system, and if a programming language can be found which directly supports it, coding becomes much easier. Manipulation of graphical objects will be facilitated by the ability to perform operations upon complex objects, as opposed to individual pixels. Because of these benefits, it was decided to tailor the life cycle approach with certain characteristics of object-oriented design, including the identification of a hierarchy of types and inheritance of attributes. Thus, the final methodology was a medley composed of elements from the

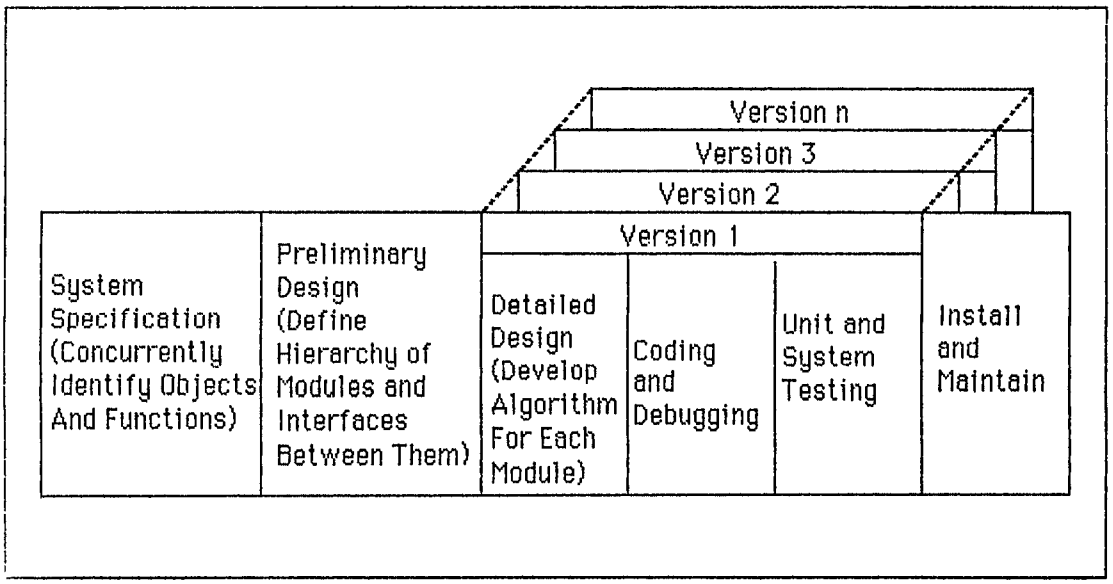

Figure 3: Integrated Development Methodology

classical life cycle approach and object-oriented design within a framework of iterative design (see Figure 3).

3.3. Software/Hardware Issues

Settling upon a methodology did not address all the issues. There were a number of questions very specific to this sort of undertaking that still needed to be answered. Many of these dealt with how to develop the map image upon which everything else was to be superimposed. Would map data have to be created specifically for this project, or was there a suitable source of cartographic data already available? How would the data be translated into an image? Would any sort of mapping transformations (such as a Mercator projection) have to be performed upon the data before generating the image? How would the positions of geographic locations on the map be associated with the cartesian coordinate system used by the existing system? How would panning and zooming be accomplished?

Fortunately, a ready source of data for the proposed environment is available. Microworld Data Bank II (MWDB II) is an extract of the digital map database produced by the Central Intelligence Agency; it includes data and code placed in the public domain by Fred Pospeschil and Antonio Riveria (Pospeschil). After close examination, we determined that MWDB II contained exactly the level of detail of information needed for this project. A simple routine can be written to extract only the information pertaining to the theater of interest. The data was stored using latitude and longitude coordinates, which are converted to screen locations via some relatively simple computations in the display routine. It was found (based on a simple trial run) that the data could be translated directly to the display without significant distortion (projection was not necessary). Base locations from the cartesian coordinate system in the TWX database are cross-referenced with their longitude and latitude coordinates to develop a set of transformation routines for the two systems. Finally, we decided that the implementation of the zoom and pan capabilities would be postponed until after the selection of hardware and graphics software, since these might handle these features for the programmer.

Another issue was how the graphical data was to be stored and represented. In particular, the representation of the hierarchy of classes of objects might have proved to be difficult; however, this consideration was taken care of by the graphics software package selected. This will be discussed in more detail later.

Besides the difficulties of maintaining the hierarchical relationships of objects, there was some question of whether it was appropriate to store the graphical data in the main TWX database, or to store it locally on the microcomputers. Advances in technology over the last decade have made moderately fast displays available and affordable, but speed of display update is still a significant factor in most graphics systems. Although the need for animated display effects are not anticipated for TWX, other exercises dealing with movement and combat of individual units might be unduly constrained by the inability to perform animation. Storing and retrieving graphical data in the central database on the host computer would probably introduce a rather large delay in response time due to network overhead. However, it might be feasible (if a suitable representation could be found) to store graphical data in the main database for long-term storage, and download it to local storage on the microcomputer when the interface is first invoked. Although this will involve some delay in the initial display, subsequent interaction can proceed at

an acceptable pace. In addition, graphical representations will exist in a single, centralized repository. If a change is made to the map, for example, the data will only have to be changed in one place.

Another, less difficult decision was the choice of the most appropriate set of symbols for military units and bases. NATO map symbols were selected for this project both because of their almost universal use in military tactical situation displays and in wargames.

Perhaps the most important decisions to be made involved the selection of additional hardware and graphics software. These are addressed together here because the factors involved in the selection are heavily intertwined. As pointed out earlier, the Z-158 microcomputers are not completely compatible with INGRES/NET, and they do not provide very sophisticated graphics capabilities. They are equipped with CGA graphics adapters, which provide a maximum resolution of 640x200 pixels, and they are slow in performing graphics calculations. Workstations such as the Sun, the Silicon Graphics IRIS, and the VAXstation GPX, which provide an order of magnitude leap in graphics capabilities, were briefly considered, but the decision was made to stay within the personal computer (PC) realm. Potentially available, the AT-compatible Zenith Z-248 provides EGA graphics, with a maximum resolution of 640x350 pixels, and is substantially faster than the Z-158. However, taking into account the rapid pace at which current capabilities are becoming obsolete, it was decided to try to get ahead of the game by purchasing hardware which would not fall behind the state of the art too quickly. A number of graphics driver boards are available which provide PCs with almost workstation-level graphics functionality. We determined that one of these boards, installed in a Z-248 and combined with an appropriate monitor, would provide the desired graphics capabilities at minimal cost. It remained to be seen whether the main memory limitation of 640K bytes would impose too much of a constraint, since it would be necessary to have INGRES routines and the appropriate graphics routines resident at the same time. Selection of a specific graphics driver would depend upon the graphics software package selected.

While PC compilers for most standard programming languages provide facilities for graphics programming, these facilities are typically very low-level primitives dealing with individual pixels. What is needed is a language which will allow the definition, storage, and manipulation through high-level function calls of graphical objects and classes of objects, hopefully with some degree of support for object-oriented characteristics.

There are a number of different categories of graphics software available commercially. Many of these do not permit interaction, and many of those that do allow interaction do not allow control by a user-written program. There are several graphics application standards and proposed standards which address the sort of graphics functionality needed for this project, including Core, Graphical Kernel System (GKS), and Programmer's Hierarchical Interactive Graphics Standard (PHIGS). Each of these standards allows the separate modeling and viewing of graphical objects through high- level function calls. These function calls usually either constitute a programming language unto themselves, are implemented as functions or subroutines of another language, or can be embedded into a standard programming language and translated by a preprocessor. Unfortunately, most implementations of these standards are geared towards workstations, and there are relatively few available for PCs. A graphics application development system called Hierarchical Object-Oriented Picture System (HOOPS) was evaluated which runs on a variety of systems, including AT-compatible microcomputers. HOOPS, a PHIGS-like graphics package, provides many of the desired features. It allows definition, storage, and manipulation of graphical objects and classes of objects through calls to a library of C functions or Fortran subroutines. Inheritance of attributes is directly supported, as well as high-level functions to change attributes (such as color) and perform transformations upon objects (such as panning and zooming). The PC version of HOOPS supports EGA graphics as well as several popular graphics drivers. Theoretically, source code developed with HOOPS is portable across the entire range of hardware systems supported (limited by the portability of the host language in which HOOPS routines are imbedded). HOOPS was given a hands-on evaluation to determine its suitability, and subsequently chosen for use. The HOOPS Fortran binding was newly released (and therefore untried), and existing HOOPS documentation was written entirely for C applications. In light of these considerations, as well as the fact that the version of INGRES being used only supported a C binding, it was initially decided to program the graphics subsystem in C using HOOPS function calls and embedded INGRES commands. Experimentation with the Z-248 showed that the main memory limitation was indeed too constraining, so development efforts were switched to a Vaxstation GPX. Once OS-2 becomes available, with its ability to access large amounts of main memory, the system can be ported back onto microcomputers.

4. PROTOTYPE

An early prototype of the system has been developed already. The prototype system (see Figure 4) consists of a map of the NATO Central European Region, showing coastal and country boundaries, and an asterisk representing a base location. In addition to being drawn in different colors, coastal and country boundary lines can be distinguished by their different patterns; coast lines are solid and country boundaries are dotted. This use of redundant distinguishing features is especially useful when looking at a hardcopy of the map, since the laserprinter used renders only black and white pictures. Asterisks will be used for the majority of bases and units at the overview level because of the constraints of space. As the user zooms in on a specific area,

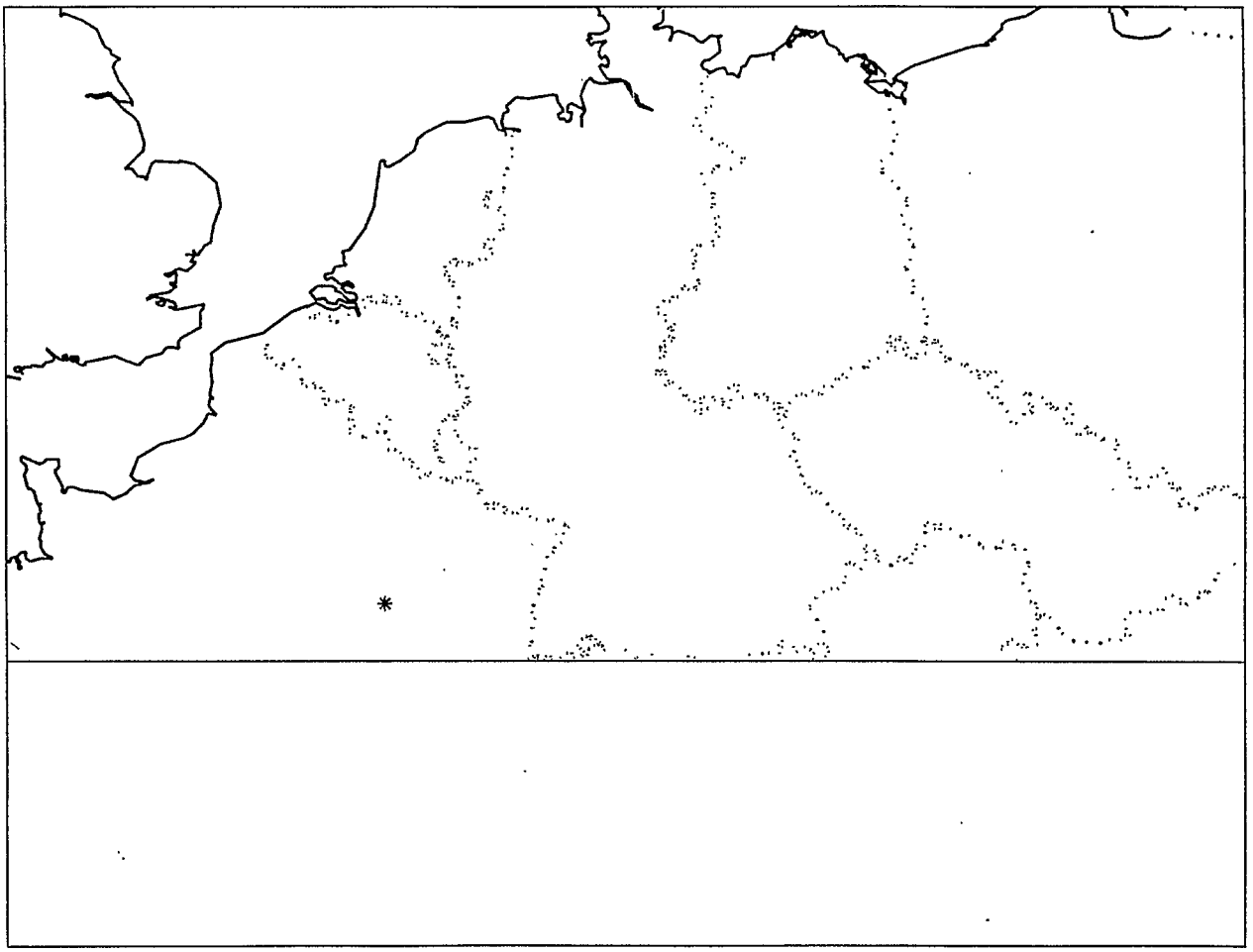

Figure 4: Prototype TWX System Display

these asterisks will be replaced with NATO symbols, beginning with the higher-level units first.

5. CONCLUSION

In this paper we have proposed an improved interface for the TWX system, and a reusable approach for adding a map-based graphical interface to an existing wargaming system. Although the system is still being implemented as of the writing of this paper, it seems appropriate to forward some observations and draw some conclusions on lessons learned.

It is already evident that the hybrid methodology consisting of elements of iterative design, the classic software life cycle, and object-oriented design is a viable approach to this sort of project. Each of these approaches imparts certain strengths to the development.

A number of potential enhancements to TWX suggest themselves. The most obvious enhancement addresses the inability to interactively generate and store new graphical objects. An icon-driven interactive "paint" program, similar to those pioneered by the MacIntosh computer, but with the ability to generate HOOPS-compatible images and store them in the central INGRES database, would be an excellent follow-on to this project. Another potential area for exploration might be the integration of the menu-driven INGRES 4GL front end system with the HOOPS graphical interface. Currently, each subsystem operates separately, with the 4GL system doing textual input, and the graphics system handling output. The integration of these two subsystems would greatly simplify use and maintenance.

ACKNOWLEDGEMENTS

This research is supported by a grant from the Air Force Wargaming Center, CADRE/WG, Maxwell AFB, AL, 36112.

REFERENCES

Brooks, M. D. (1987) *Developing a Database Management System and Air Simulation Software for the Theater War Exercise.* MS Thesis, AFIT/GCS/ENG/87D-6. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH.

Brooks, M. D., Kross, M. S. and Roth, M. A. (1987), Re-Hosting a Computer Assisted Wargame Exercise from a Mainframe to a Micro: Database and User-interface Issues, *Proceedings of teh 1987 Winter Simulation Conference*, A. Thesen, H. Grant, W. David Kelton, eds., 870–875.

Kross, M. S. (1987), *Developing New User Interfaces for the Theater War Exercise.* MS Thesis, AFIT/GCS/ENG/87-19. School of Engineering, Air Force Institute of Technology (AU), Wright- Patterson AFB, OH.

Pixelworks, Inc. (1987) Clipper Graphics Series. Company Brochure. Pixelworks, Inc., Hudson, NH.

Pospeschil, Fred. On-line documentation file for Microworld Data Bank II. Bellevue, NE.

Theater Warfare Exercise Users' Handbook (1987). Unpublished Manual. Air Force Wargaming Center, Maxwell AFB, AL.

AUTHORS' BIOGRAPHIES

DARRELL A. QUICK is an M.S. student in the School of Engineering at the Air Force Institute of Technology and a Captain in the United States Air Force. He received a B.S. in computer science from Southwest Texas State University in 1983. His previous assignments included systems analyst/programmer and database administrator for Air Training Command. His current interests include graphics, software engineering, and database management systems.

Darrell A. Quick
Air Force Institute of Technology
AFIT/ENA
Wright-Patterson AFB, OH 45433-6583

MARK A. ROTH is an assistant professor of computer systems in the School of Engineering at the Air Force Institute of Technology and a Captain in the United States Air Force. He received a B.S. in computer science from Illinois Institute of Technology in 1978, an M.S. in computer systems from the Air Force Institute of Technology in 1979, and a Ph.D. in computer science from the University of Texas at Austin in 1986. His previous assignment included systems analyst and programmer for Headquarters Air University, Data Automation Directorate in the area of computer assisted wargaming exercises. His current research interests include wargaming simulation and database management systems. He is a member of ACM and IEEE Computer Society.

Mark A. Roth
Air Force Institute of Technology
AFIT/ENG
Wright-Patterson AFB, OH 45433-6583