DEVELOPING MAP-BASED GRAPHICS FOR THE THEATER WAR EXERCISE

THESIS

Darrell Anthony Quick
Captain, USAF

AFIT/GCS/ENG/88D-16

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY
Wright-Patterson Air Force Base, Ohio

AFIT/GCS/ENG/88D-16

DEVELOPING MAP-BASED GRAPHICS FOR THE THEATER WAR EXERCISE

THESIS

Darrell Anthony Quick
Captain, USAF

AFIT/GCS/ENG/88D-16

Approved for public release; distribution unlimited

AFIT/GCS/ENG/88D-16

DEVELOPING MAP-BASED GRAPHICS FOR THE THEATER WAR EXERCISE

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

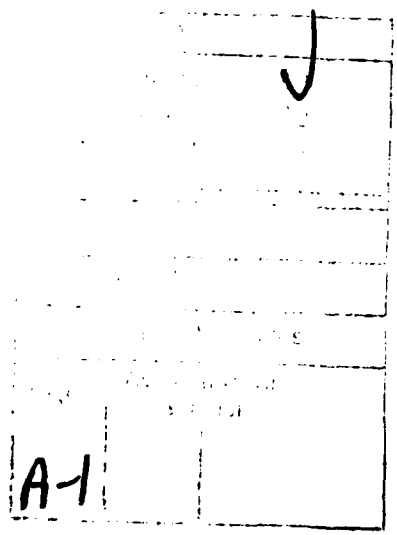
Requirements for the Degree of

Master of Science (Information Systems)

Darrell Anthony Quick, B.S.

Captain, USAF

December, 1988



Approved for public release; distribution unlimited

Preface

The goal of this thesis was to develop a map-based graphical interface to the Theater War Exercise to provide a much more user-friendly interface to exercise information and establish a baseline for the development of new techniques for user interaction with the exercise. A secondary consideration was to design this interface in such a manner that it could be reused to develop similar interfaces for other computerized wargames residing on a variety of hardware and system software configurations.

The Theater War Exercise is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center as part of the Air War College curriculum. 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.

This thesis presents the background of the Theater War Exercise and discusses shortcomings of the old system. The issues involved in developing a map-based graphical interface are discussed, and the selection of a methodology and tools recounted. Finally, the latest version of the system is described in detail.

I am grateful for the support and direction provided by my thesis advisor, Captain Mark A. Roth, and by my thesis committee members, Lieutenant Colonel James N. Robinson and Captain Wade H. Shaw. In addition, I would like to express my gratitude to my fellow thesis students working in the wargaming arena for their support. I particularly want to thank Captain Brian Healy for his help in preparing some of the material herein. Finally, and most importantly, I would like to thank my wife, Melanie, not only for her steadfast emotional support, but also for her very substantial contributions in writing and proofreading this thesis.

Darrell Anthony Quick

Table of Contents

List of Figures

Abstract

The Theater War Exercise (TWX) is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center as part of the Air War College curriculum. 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 hardcopy reports which were difficult to understand and cumbersome to use. This thesis discusses the development of a map-based graphical interface to TWX which provides a much more user-friendly interface to the information and establishes a baseline for the development of new techniques for user interaction with the exercise.

This interface was developed under a hybrid methodology which takes advantage of the best features of both the Classic Life Cycle Methodology and the Iterative Design Methodology, utilizing an underlying object-oriented model to represent graphical entities and classes. With the exception of geographical data, which is extracted from sequential files, all information is retrieved over a network from a remote relational database management system using Embedded Structured Query Language.

Given a set of maximum and minimum latitude and longitude coordinates, this system can display a map of any geographical region. Since the system also performs an automatic conversion to a user-specified game coordinate system, it can be adapted with minor changes to provide map-based graphical interaction with a wide variety of other wargaming and simulation systems. In addition, because of the high level of abstraction and wide product base of the languages and utilities used, the interface software is portable across many different computers and operating systems. The flexibility, portability, and power of this interface make it a useful tool for interaction with map-based computer systems.

DEVELOPING MAP-BASED GRAPHICS FOR THE THEATER WAR EXERCISE

I. Introduction

The Theater War Exercise (TWX) is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center as part of the Air War College curriculum. 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 hardcopy reports which were difficult to understand and cumbersome to use. This thesis discusses the development of a map-based graphical interface to TWX which provides a much more user-friendly interface to the information and establishes 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.

1.1 Background of 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" [3]. 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 [3].

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, the Air Command and Staff College, 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, there is a third group, the White (or Control) Team, consisting of a faculty game director and data processing personnel who oversee the conduct of the exercise. Multiple independent runs of the exercise, called seminars, are conducted simultaneously.

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 battle outcomes. The database is then updated with the new scenario and reports are generated. Other than aircraft movement and rerole, logistics movement, and the player responses, which are stored for batch processing at night, the information in the database is static during the day.

The TWX algorithms use a highly aggregated approach, determining the outcomes of conflicts based primarily 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 currently consists exclusively of their support of ground forces through Offensive Air Support and Air Interdiction missions, although work is in progress on upgrading the land game to a more sophisticated model [17].

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 thesis students at the Air Force Institute of Technology (AFIT) [8, 12]. Both students were involved in rehosting the system from the Honeywell to a DEC MicroVAX using Zenith Z-158 microcomputers as remote terminals.

Captain Brooks' thesis project [8] involved transitioning the air battle software to the MicroVAX and writing a game controller subsystem to allow multiple seminars of the game to be controlled concurrently. In addition, he redesigned the database from application-specific files to a commercially available relational database management system (DBMS) called Ingres. The rewrite of the source code incorporating modern software engineering principles and the transition to a relational DBMS made the system much easier to maintain and modify.

For his thesis project [12], Captain Kross developed a screen-oriented front end interface on the Z-158's using a fourth generation language development facility provided by Ingres. This interface, which replaced the old method of entering and receiving data through an obsolete hardcopy terminal, was designed with special emphasis on human factors to provide users with a comfortable and easy-to-use interface with the computer. In addition, it provided for immediate, interactive validation of input values.

1.2 Shortcomings of the Old System

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. It also 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. Both thesis students recommended the development of a map-based graphical output capability for the TWX system as a follow-on project [8, 12].

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, then 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 about weather status in a country, a player had no recourse but to search 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 thesis was developed specifically to counter these shortcomings of the system and to allow the players to concentrate on decision-making and analysis.

Another important, if somewhat less obvious, disadvantage of the old system was that its conceptual model was largely hidden by implementation-specific details. This made it difficult for players without a computer background to use. The interactive approach adopted for this interface also resolves this problem.

1.3 Enhancements to the System

AFWC personnel requested the development of a map-based graphical interface to TWX to improve its usefulness. Together with AFIT Professor Captain Mark Roth, they put together a set of requirements for such a system. The envisioned graphical interface system would consist of a computer-generated on-screen map of the European theater showing such details as country boundaries, locations of military bases and units, and the Forward Edge of Battle (FEBA). Locations of these items were already maintained in TWX using a Cartesian coordinate system, with each item's position being stored in the Ingres database as a matched pair of X and Y coordinates [17].

Military bases and units would be represented using color-coded standard symbols to indicate their affiliation. Such a computer-generated graphical display of the relative positions of bases and units to each other and the FEBA would be a valuable aid to the players in visualizing their situation [17].

Other proposed features included a method of indicating the weather forecast on the screen, and facilities for "zooming" and "panning" the display. Zooming refers to the process of expanding a portion of the display to fill the entire screen, thus showing greater detail. Panning is the process of moving the display in one direction so that the perspective

of the viewer changes; when panning from right to left, for example, the picture disappears off the left side of the screen, but new parts of the picture appear on the right side of the screen. Used together, zooming and panning are a powerful combination, and their use is becoming relatively standard in graphics applications [5].

Other facilities which would be beneficial to a player included an ability to somehow select a particular base or unit on the map and "pop up" a status report of its current situation (number of planes of each type available, logistics support, etc.). Although this status information can be obtained through Kross' interface, it is obtained as part of the process of entering decision data, and is not very flexible. In addition, obtaining the information by placing the cursor over the base on the map would provide a much better abstraction to the user, letting him feel that he is directly manipulating the data. The importance of this feeling is discussed in more detail in the next chapter.

1.4 Explanation of Terms Used

A number of terms are used repeatedly throughout this thesis. The following list identifies these terms and defines their meanings.

1. *Affiliation* - Which team: white, red, or blue.
2. *Click* - To select an item or invoke a function by moving the mouse cursor over a symbol or token and press the left-most mouse button.
3. *Designer* - The developer/implementor of the Map-Based Graphical Interface.
4. *Developer* - An individual building a Map-Based Graphical Interface for another wargame by reusing this Interface.
5. *Interface* - Short for Map-Based Graphical Interface. Refers to the system produced as a result of this thesis.
6. *Player* - A student using TWX to develop his or her decision-making skills.
7. *User* - The group of individuals responsible for determining the functional requirements for the system. In this case, the faculty and staff of the AFWC.

1.5 Sequence of Presentation

This thesis consists of six chapters. This chapter has presented an overview of the old system, describing its shortcomings and the proposed revisions. The next chapter is a

review of current literature related to the topic. Chapter III describes the selection of a methodology and a group of conventions used throughout the project. Chapter IV discusses the selection and acquisition of appropriate hardware and systems software. Chapter V describes the finished system and discusses lessons learned. The final chapter presents conclusions and recommendations for further research.

II. Review of Literature

> In science, by a fiction as remarkable as any to be found in law, what has once been published, even though it be in the Russian language, is spoken of as known, and it is too often forgotten that the rediscovery in the library may be a more difficult and uncertain process than the first discovery in the laboratory.
>
> Lord Rayleigh, 1884 [7:xiii]

This chapter surveys current literature on topics relevant to this thesis. The subjects examined include graphical interfaces, the use of graphics in simulation, map graphics, and projects similar to this one.

2.1 Graphical Interfaces

In any interaction between man and computer, there is a necessary translation of concepts from one form to another because of the inherently different manner that each retains and processes information. A computer works, at the lowest level, with electrical impulses (called bits) that can be in one of two states: on or off. A human, on the other hand, generally thinks in terms of spoken language or pictures. In order for the two to communicate, an intermediate language must be developed which is somewhere between the two extremes, and each entity must perform the necessary translation between their natural medium and this intermediate form.

Early programming languages, because of the low level of technology, put the brunt of this translation on the human, forcing him to write programs in low level, machine-like code. Largely because of this, the realm of computer science was restricted to those individuals with the time and mental fortitude to perform these translations. As computer technology matured, programming languages closer to the natural language of humans were developed, putting more of the burden of translation on the computer. These "high order" languages encouraged more people to indulge in the art of programming by requiring less investment of time and effort to learn and use.

Technology has reached the point where computers are now being widely used by executives in making high-level decisions. An important factor in the usefulness of computer-based decision support systems is the ease of learning and use.

Although there have been substantial improvements over the years, there has been little formal study of the area of human-computer interaction until recently. Most of the work done on computer languages has been by computer scientists working primarily on an intuitive basis, with little or no reference to psychological principles. Newell and Card attribute this fact largely to their version of Gresham's Law, which basically states that "hard science drives out soft" [14:212–213]. The thrust of their argument is based upon the tendency of "hard" sciences like physics, engineering, and computer science to push into the background "soft" sciences (in this case, psychology). This tendency often manifests itself in the practice of designing computer-human interfaces almost exclusively around hardware and software considerations, with little thought given to the psychological impact on the user. They discuss the possibility of "hardening" the applicable psychological sciences, difficulties inherent in doing so, and the "...prospects for overcoming each of these obstacles..." [14:209]. Despite the rather abstract level at which this article is presented, it is pertinent to this discussion because of its identification of the tendency to overlook human factors, and its vigorous advocacy of the importance of weighing psychological considerations in the design of a system.

MacGregor and Slovic present a series of studies that deal with the more practical aspects of graphical representation of judgmental information [13]. In these studies, volunteers were presented with four types of data on a number of marathon runners to determine how well they could correlate this data to the runners' actual completion times. In the initial study, volunteers were divided into four test groups, and each group was presented the same information represented by a different style of graphical representation.

One type of graph, the "face graph", was a line drawing of a face, consisting of the outline of the face, a mouth, a nose, two eyes, and two eyebrows. The relative positions/shapes of the mouth, nose, eyes, and eyebrows were used to represent the four categories of data. The other three graphical representations were the more conventional bar chart, deviation chart, and spoke chart.

Results of the initial study showed a marked superiority of the face graph in predicting accurately the finishing times of the runners; however, there was some question as to whether the test might have been rigged so that the face graph had an unfair advantage.

"Indeed, the face displays in the current study were constructed to take advantage of the differential attention people tend to pay to some facial features over others by assigning the more predictive cues to the more salient features" [13:193]. Subsequent tests validated that the choice of indicators did affect the accuracy of the judgments made.

Although the types of display used in MacGregor and Slovic's study were significantly different than those to be used in the TWX system, there is an applicable point. By designing your graphical display appropriately, you can communicate its meaning better and even influence the correctness of the decision to be made. This is of particular importance since the results of this thesis project may pave the way for use of graphical interfaces not only in other educational wargames, but also in combat simulations used for actual strategic and tactical planning, as well as in military decision support systems.

In their article on direct manipulation interfaces, Hutchins, Hollan, and Norman discuss the important psychological benefits of designing an interface in which the user can manipulate icons (graphical symbols representing an object or action) interactively [11]. In a typical (indirect) interface, the user types in a command for the computer to perform a manipulation upon an item of data, but he never gets the feeling that he is actually operating upon the data. There is a feeling of having to go through an intermediary, and this introduces a distance between the user and his data which has a negative psychological impact. By designing an interface in which the data is represented as icons, and in which the user can directly manipulate these icons, the feeling of direct control is restored, even though the user is not, in reality, operating directly on the objects. This is accomplished in TWX by allowing the user to bring up additional information on a base or unit by using the mouse to select the icon that represents it.

O'Keefe presents an excellent taxonomy of visual interactive systems (VIS) [15]. He states that a VIS system or application usually provides:

1. Visual Output: portraying the dynamic behavior of the system model.
2. User Interaction: allowing the user to interact with the running model. Interaction can be model determined, where the simulation halts and requests input from the user, or user determined, where the user changes the simulation at will.
3. Visual Input: where a model can be created graphically instead of being programmed or data driven [15:461].

He also specifies that the first of these criteria is a necessary condition for a system to be considered a VIS, and that meeting both of the first two criteria is necessary and sufficient to determine inclusion in this category. Condition three is met by some, but not all, visual interactive systems.

2.2 The Use of Graphics in Simulation

There is a large body of literature concerning the use of graphics in simulation, especially in visual interactive simulation (VIS). In VIS, an animated display of the simulation is shown to the user at run time, and the user can interact with the running simulation. In general, however, these articles were not very applicable to this project. Their focus seems to be upon creating general purpose graphical simulation languages for use in evaluating systems and proposed systems. Most simulation languages are designed for use in evaluating system flow and queuing factors, not for wargaming exercises. In addition, these systems are entirely self-contained and would not interface well with other, less specialized, programming languages. Since the actual TWX system was already in existence, and the goal is merely to add graphics to it, these languages were unsuitable.

2.3 Mapping Considerations

The graphical representation of two dimensional, low detail maps is not a subject of great controversy at the present time. What current literature does exist concerns itself largely with such issues as how to associate non-cartographic information with regions on a map, or various methods of achieving photograph-quality renditions of topographic maps showing the contours and textures of mountain ranges. Although perhaps not as recent as could be desired, some of the articles from the 1981 Conference on Data Base Techniques for Pictorial Applications proved to be good sources of information on computerized map data. In particular, the article by Richard L. Vitek of the Defense Mapping Agency (DMA) was of particular interest [22]. The article describes the efforts of the DMA to create a centralized repository of cartographic data, with associated indexes to enable easy retrieval of maps based on user requirements. A phone call was sufficient to determine that the database is not yet functional; however, a referral to the Central Intelligence Agency's

World Data Base II (WDB-II) was very productive. "The full WDB-II is a digitial (sic) map data base produced by the Central Intelligence Agency (CIA) and distributed by the National Technical Information Service (NTIS)..." [16]. A highly compressed version of this database designed specifically for use on microcomputers was located, called Micro World Data Base II (MWDB-II), which contained exactly the sort of data needed for this project [16].

> WDB-II is a digitial (sic) representation of the world coastlines and boundries (sic) suitable for use in automated mapping systems. It contains approximately six million discrete geographic points and was digitized using all available sources of information. Map scales used range from 1:750,000 to 1:4,000,000 with a nominal scale of 1:3,000,000. These points are grouped by and identified as describing (1) coast lines, (2) country boundries, (3) state boundries (USA only), (4) islands, (5) lakes, and (6) rivers. Each of these groupings is further broken down into features and subordinate classifications/ranks. These ranks are hierarchically structured, and are also used for plotting symbol definition [16:2].

The discovery of this map database proved to be fortunate, as it was subsequently chosen for use as the source of geographical data for this project. The other options, which included manually typing in coordinates or using a digitizer to produce a set of points, were highly undesirable. Creating the set of points manually would almost certainly have introduced inaccuracies as well as necessitated a very low level of detail. Digitizing would not have provided the ability to extract different types of geographical data without first manually separating the data, and would have provided no more than what was already available in MWDB. Neither option would have provided the flexibility of MWDB in selecting different levels of detail.

2.4 Graphics in Wargaming

In the realm of computer assisted wargaming, which overlaps with the field of computer simulation, there were still fewer articles of interest, although at least two other theses have been done on related topics at the Naval Postgraduate School. Like most of the simulation articles, the emphasis in recent literature was more on the various algorithms used to implement the models, rather than on their interaction with the user.

Hawkins and Thompson describe a corps-level combat simulation called Quickscreen which was developed by the BDM corporation for the U. S. Army [10]. Quickscreen appears to use much more intricate algorithms than those of TWX. TWX is designed in a highly aggregated fashion so that players make decisions at general officer level, and the results of the decisions are determined directly by algorithms. In Quickscreen, on the other hand, high level decisions are actually implemented in a hierarchical fashion by having subordinate units act out individual scenarios autonomously, and then aggregating the results back up to the higher level. The detailed play of these lower level units can be handled either by the players or by built-in artificial intelligence routines to allow the players to concentrate on high level decision making. There is no mention of graphics capability or of human interface considerations in the article. The impression of this author is that either play is conducted using game pieces on a separate board to keep track of positions, or that the computer periodically prints out a hardcopy of the battlefield [10:pages 576–577].

Battilinga and Grange discuss a wide variety of wargaming simulations, but the emphasis is, again, on the underlying principles and methods, rather than the actual implementations [6].

Glenn D. Simon, in his 1983 thesis [20], discussed the development of an interactive graphical support system for a existing small-unit amphibious operation combat model. This system enabled users to interactively request the generation of line and bar charts graphically depicting attrition information or a contour map of the battle area. Three very basic types of symbols were displayed on the map to indicate the placement of tanks and the primary and alternate firing positions of the defenders. Like the TWX display, this map presented a "snapshot" view of the scenario, not an animated picture in which the movement of units across the terrain is depicted. The time interval for this model was much smaller than the TWX wargame. It was originally set at a ten-second interval, although it was later changed to allow the user to control updates because of the difficulty in assimilating the information in the given interval.

This product was different from the TWX interface in many ways. The interaction was menu-driven, with the map as an optional product, rather than as the main display.

The display was only available in black and white, and it is unclear whether it can be displayed on a screen as well as printed in hardcopy form. Other than the generation of a non-interactive map, the graphics consist of standard business graphics (line and bar charts), whereas the TWX interface concentrates exclusively upon interaction with the map. The TWX interface depicts coast lines and country boundaries, rather than contour lines. Much of this can be attributed to the different goals of each thesis, as well as to the differences in hardware and commercial software packages used. The data-handling and control portions of Major Simon's project were written primarily in Fortran for a specific target application in a mainframe environment, although a proprietary graphics software package called DISSPLA (Display Integrated Software System and Plotting LAnguage) was used for the actual generation of the graphics products.

Another recent thesis from the Naval Postgraduate School, jointly performed by Robert P. Sabo and Remmington G. Bishop [18], also uses DISSPLA (in addition to the Briefing Aid System) to produce maps. The emphasis of their system, Low Cost Graphics (LO-CO-GRAF), is to produce high-resolution, low cost, accurate map graphics suitable for Command and Control use. Like the TWX interface, emphasis was on generating maps showing geographical and political boundaries on a small computer using data accessed from a remote mainframe computer database. LO-CO-GRAF, unlike TWX, applies map projections against the data to convert it from three-dimensional world coordinates to map coordinates. Symbol overlays can be generated, although this must be done interactively by the user, rather than software-controlled like the TWX interface. LO-CO-GRAF, like the TWX interface, uses map data from the Defense Mapping Agency's World Data Bank II. Although the maps produced by LO-CO-GRAF are not used interactively for wargaming, they can be transmitted over communications lines from one location to another.

2.5 Bridging the Gap

Although a substantial body of research exists concerning the role of user-friendly, easy to learn interfaces in computer assisted decision-making, little work has been done to address the inadequacies of current wargaming systems in this regard. The work that has been done is largely application-specific and does not provide a great deal of interaction.

Sabo and Bishop set a commendable example within a particular domain, but do not provide the variety of flexible interface capabilities needed for TWX. There exists a need for a technology which provides not just a reusable map-generating capability, but also a wide range of program-driven, highly interactive functions. This thesis represents the first step in fulfilling that need.

This chapter has examined the body of literature concerning research in areas relating to this project. Chapter III looks at the selection of a conceptual model for the system and a methodology to implement it.

III. Methodology

The next step in the evolution of the interface was to develop a conceptual model of the system, then select a methodology to build it. The selection of an approach was carefully chosen to address the unique goals and constraints of this project. This chapter discusses these goals and constraints, the methodologies considered, and the rationale behind the selection.

3.1 Development Goals and Constraints

This thesis presented many possibilities for enhancement, but the amount of time available was limited, so goals and constraints were established to keep the scope reasonable while still demonstrating thesis-worthy work. The primary goal was to develop the kernel version of a graphical interface for TWX. A strong secondary goal was the development of a set of tools and procedures that could be reused to develop similar interfaces for other, similar wargames across a variety of hardware systems. In effect, the end product would be a generic tool kit for producing portable map-based graphical interfaces.

A number of other constraints were imposed. First, although it would be feasible for the graphical interface to be used to update the database, this project would only concern itself with providing an enhanced output capability. Second, every effort was made to keep the interface as independent of the batch system as possible. This averted the potential problem of the designer becoming overwhelmed by the details of the highly complex simulation algorithms, and provided for easier maintenance by future system managers. Emphasis was placed on providing examples of as many different types of functionality as possible, rather than upon developing each function to its full potential. Once an example of the function was available, it would be easy for later programmers to tailor the system for maximum usefulness.

3.2 Conceptual Model - Overlapping Levels

The conceptual model for the system was relatively straightforward. The map portion of the display would be generated as a series of levels, similar to a group of overlapping

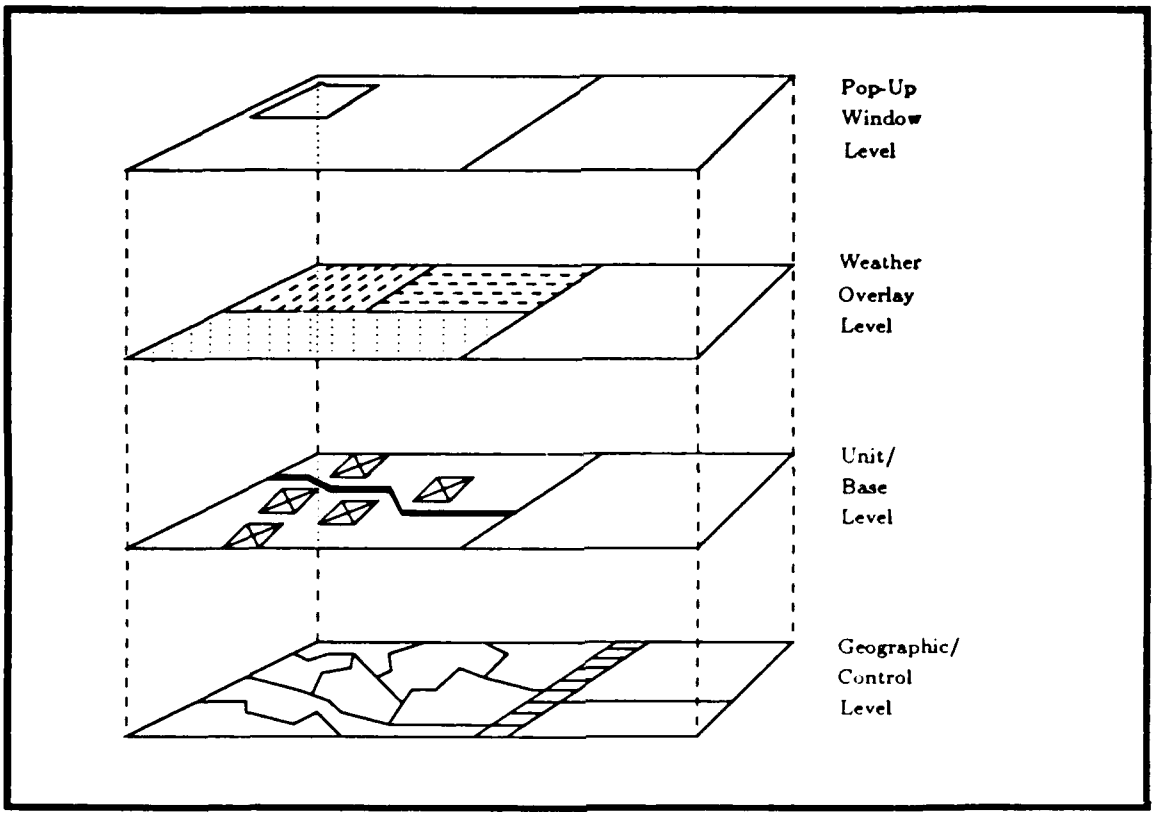


Figure 1. Conceptual Model of Display

transparencies (see Figure 1). The non-map portion of the display would be drawn at the same level as the map, but would take up the rest of the screen. This non-map portion would consist of a set of function "buttons" which the user could select with the cursor to initiate interactive operations, as well as a message area for feedback, a legend area explaining symbols, and a context area showing the user's context. The lowest level of the map display would consist of only the geographical data, including coast lines and country boundaries. This would be overlaid with a unit and base level, followed by a weather overlay level, and finally the pop-up windows level. Each of the graphical entities created would be designed to facilitate later manipulation through the interactive portion of the system. Although this provided a starting point for the design of the system, it remained to be seen how well existing software tools and hardware would support this abstraction.

3.3 Selection of Appropriate Symbols and Colors

A key factor in designing a user-oriented interface was the selection of appropriate graphical representations. This included the choice of both symbols and colors, as well as of innovative methods of presenting them. Verplank and Kim [21] write about a human interface course they put together at Stanford that emphasized invention and new ways of looking at user interfaces. In particular, their discussion of visual representation is of interest; they give examples of the appropriate use of symbols and direct manipulation (for example, deleting a file on the Macintosh by dragging the icon representing the file into a trash can icon). This sort of direct manipulation of a familiar symbol aids users in learning a system.

TWX lends itself strongly to imagery and color. Many of the selections are quite straightforward. For example, since the player teams are designated as blue and red, it is easy to choose appropriate colors for the symbols. Army Field Manual 21-30, *Military Symbols*, lists symbols which are used in actual battle planning [2]. The symbols are also used extensively in popular wargames, such as those produced by Avalon Hill. Since these symbols are standard military and wargaming symbols, they were a logical choice for incorporation into the simulation.

On the other hand, the meaning of many of these symbols are far from intuitively obvious. It might be better to design a new set of icons that are conceptually straightforward. For example, a squadron of A-10's might be represented by a silhouette of an A-10. The final decision on choice of symbols will be up to the end users of the product. Only a subset of the entire set of standard symbols was generated for land units, and two different symbols were created for airbases (see Figure 2). The old symbol, that of a propeller inside a circle, seemed rather outdated. The new, revised symbol was a runway inside a circle. The choice between these airbase symbols will be left to the users and future programmers, as will the generation of a more complete set of symbols.

In addition to the selection of symbols and colors for representing units, it was necessary to consider carefully the representations of country boundaries, coast lines, grid lines, weather patterns, and the FEBA. These representations had to be distinguishable

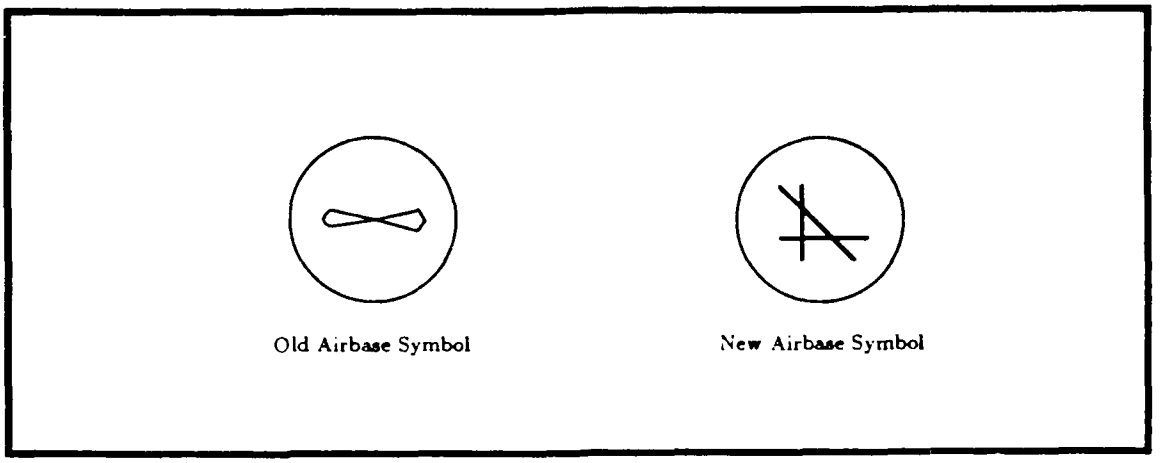


Figure 2. Airbase Symbols

from each other even when printed out in (black and white only) hardcopy form. This was handled by using varying thicknesses and patterns for lines, and patterned area fills for the weather overlay. In addition, the monitor display of each of the lines was a different color, facilitating easy distinction between them.

3.4 Using an Existing Map Database

There were a number of challenges involved in using an existing map database (MWDB II) for this project. For one thing, positional information was stored in the database in terms of longitude and latitude, rather than the Cartesian coordinate system used in TWX. This required a conversion routine to be written to translate between the two frames of reference. This routine had to be flexible enough to provide translation capability not only for the TWX coordinate system, but also for any other Cartesian coordinate system based wargame. The possibility of providing another conversion routine for wargaming systems based on a hexagon grid system was also considered, but rejected due to time constraints and the lack of a suitable system for testing it.

Another anticipated problem stemmed from the fact that MWDB II was organized into a number of files, each containing information on one type of geographical formation (i.e., coastal boundaries, country boundaries, etc.) for the entire world [16]. Since TWX would only need a small portion of information for each of several MWDB II files, it

would be more efficient to extract portions of each of the applicable files so that all the geographical information for a certain region would be stored together. By building a database containing only the subset of map data needed for the particular game, and organized sequentially in the order it would be needed, speed of processing would be improved. Fortunately, the format of the data structures used in MWDB-II was given in the on-line documentation, and the source code for the display routines was provided, making it possible to modify or even replace the code. Another consideration was whether to store the extracted data in a regular sequential file or to load it into the game database with the rest of the data. Placing it in the database would probably slow down retrieval of the information, but it would provide protection against corruption of the data and would make it easier to port the data across implementations.

There was also some question as to whether it would be necessary to apply map projections (such as a Mercator Projection, for example) to the map data. These projections are used to compensate for the distortion caused by mapping points from the curved surface of the Earth to a two-dimensional map representation. It was discovered through experimentation that the amount of distortion for the area used in the TWX simulation was not significant, so no projections were built into this version of the interface.

A final, but very important, consideration in the use of an existing map database was that the display routines that were provided with MWDB II were designed as stand-alone routines. They were written to display the map coordinates when a user interactively invoked them and provided coordinates. It would prove challenging to integrate them with other (graphics) software to keep the map display on the screen while overlaying it with various symbols and menus, as well as providing zoom and pan capabilities. It became evident very early in the project that the display routines would have to be replaced with display routines written specifically for the new interface.

3.5 The Object-Oriented Approach

Graphics systems (and this system in particular) lend themselves conceptually to an object-oriented approach. In an object-oriented scheme, objects and classes of objects are identified. These objects and classes, called entities, are usually related in some hi-

erarchical fashion. Figure 3, which shows a partial hierarchy of graphics objects for the proposed subsystem, illustrates the natural affinity of graphical data for this paradigm. By convention, each of the boxes in the hierarchy represents a class/subclass. Each object derived from a class has associated with it a number of attributes (color, size, and position, for example). By default, the parts of an object that are defined by subclasses inherit the values of attributes set by their superclasses, although they can explicitly override these attributes. This is a relatively difficult concept to convey, so the following example is provided for clarification.

Consider the situation in which the programmer wishes to set the color of a battalion-level infantry unit symbol (BLIUS) to red. Since the BLIUS is ultimately a subclass of the unit symbol, it can be set simply by setting the color attribute of the unit symbol to red, and leaving the color attributes of all subclasses null. The entire BLIUS then inherits the color red.

Alternately, if we wished to make the entire symbol red except for the command-level, we could set the unit symbol color attribute to red (as above), but set the command-level color attribute to blue.

The above examples illustrate the usefulness of the object-oriented methodology for modeling graphics characters. The use of inheritance, especially if it is supported by the graphics package used, can greatly simplify the job of the application designer, as well as potentially prevent the redundant storage of shared data (such as the unit symbol class designation).

3.6 Proposed Methodology

Several software engineering methodologies were considered by Kross for use in his thesis. Three of these methodologies were reviewed for applicability to this project. Each methodology had its own strengths and weaknesses. The approach finally chosen was a hybrid methodology utilizing aspects of each of the three.

The Classic Life Cycle approach involves the following steps: (1) identifying the system requirements, (2) performing a preliminary high-level design that determines the

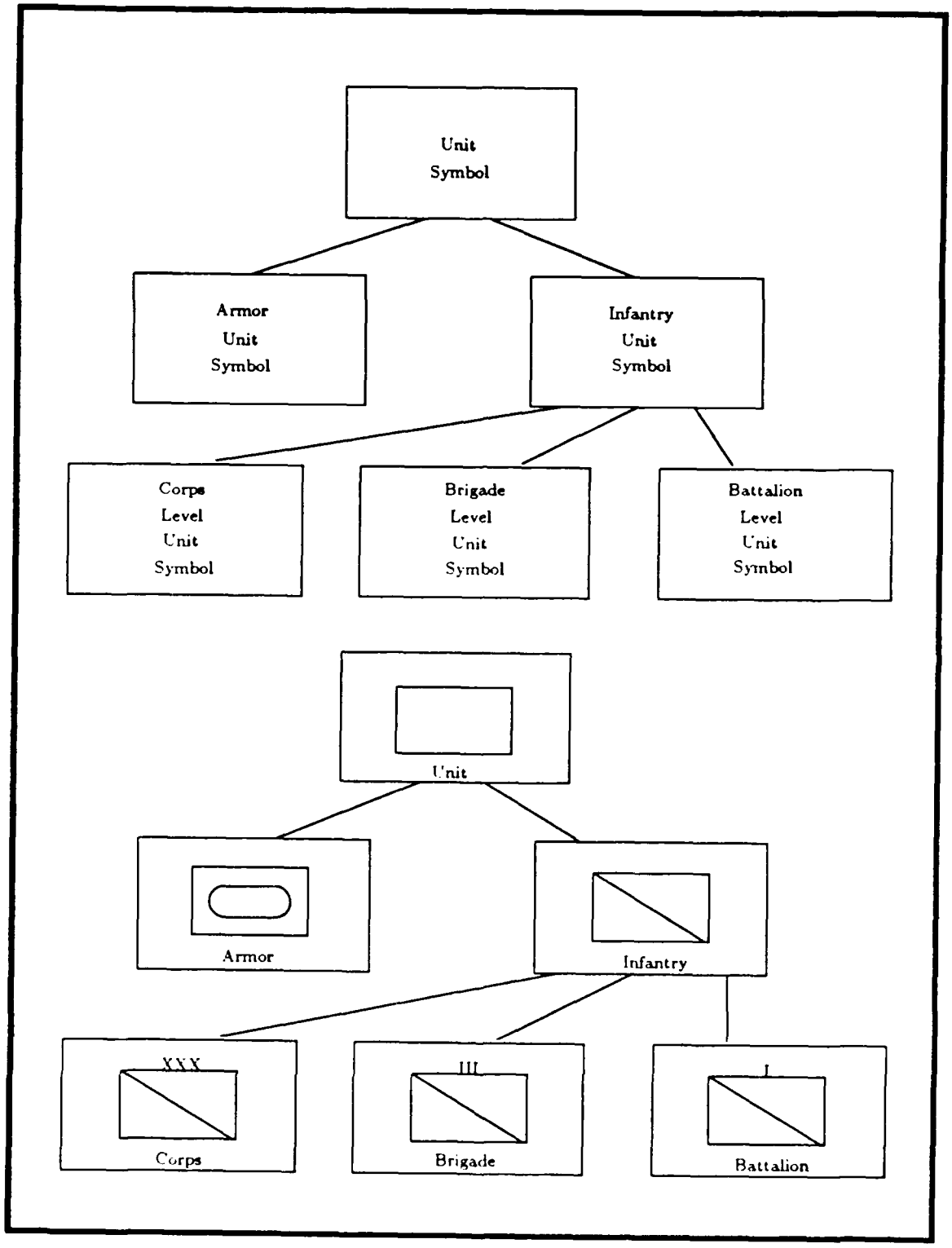


Figure 3. Partial Hierarchy of Graphical Data

system's overall architecture, (3) doing a detailed low-level design in which specific algorithms are identified to accomplish system functions, (4) translating the design into code, (5) testing the system, and (6) installing the completed system onto the user's computer. This is a general purpose methodology which has strong advantages for large-scale software engineering projects involving large numbers of programmers. Also, it supports the entire range of programming languages. On the other hand, it is very laborious and time consuming, and errors made early in the process can become very expensive to correct further down the line.

In the Rapid Prototyping methodology, the designer makes a first cut at his understanding of the system early in the process, and then uses this early model to communicate with the user to refine the system requirements. Changes are made to the model as appropriate, and the process is repeated until the system meets the user's needs. For systems that involve a lot of user interaction and less behind-the-scenes processing, and where a development tool/programming language is available that supports this technique, this is a valuable methodology. By using prototypes to communicate with the user, potential misunderstandings can be resolved early in the project, thereby preventing the problem discussed earlier with the Classic Life Cycle methodology. Also, when using the life cycle methodology, a problem that typically arises with communicating the system requirements is not just that the user cannot express them, but that he does not really have a clear idea of what he wants. Using a prototype in a hands-on manner often helps the user clarify his needs as well as communicate them.

The last methodology considered, that of Iterative Design, provided some particularly strong features for a project of this sort. Under this system, requirements are gathered and prioritized at the beginning of the project, then implemented a version at a time according to the priority scheme. The primary advantage of this methodology is that it breaks the project down into manageable pieces; the disadvantage is that it takes longer to complete the same amount of work. Kross chose Iterative Design as his primary methodology because of its other, less obvious virtue: in a limited time situation such as a thesis, one can implement versions successively until he runs out of time, and still be assured of having a

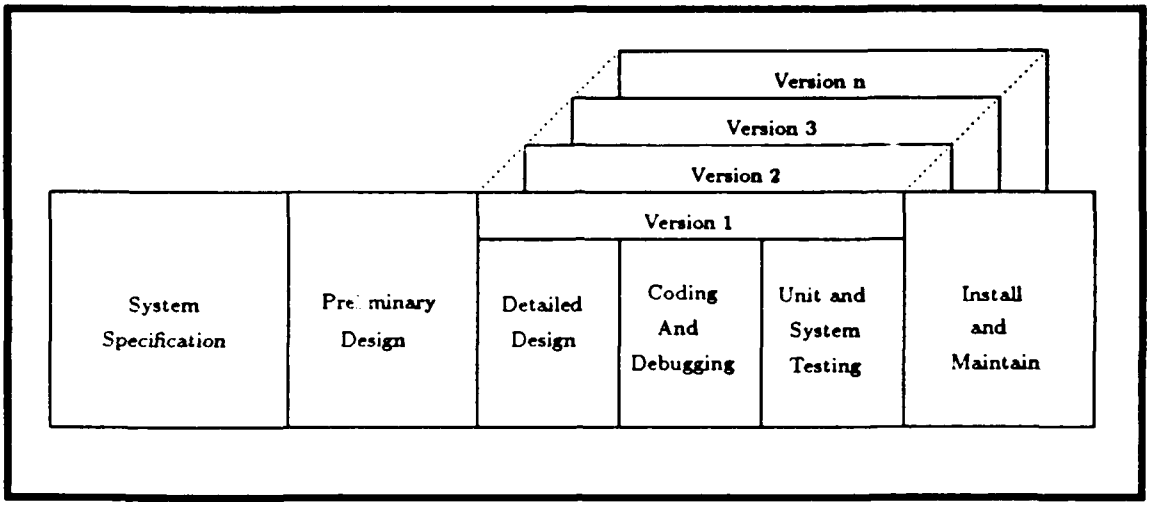


Figure 4. Hybrid Methodology

stopping point to fall back to. This seemed entirely reasonable, and applied to this thesis project equally well.

Another feature of Iterative Design, storyboarding, is used to aid in identifying requirements specifications. In order to get a common understanding between users and developers, a set of crude "storyboards" can be drawn up to depict the workings of the system and then used as a communications tool. This practice generally eliminates a number of potentially unfortunate misunderstandings early in the life cycle. Storyboarding is very similar to rapid prototyping. The primary difference is that Rapid Prototyping requires programming environment support to generate an automated prototype, whereas a paper prototype is used in storyboarding.

It should be noted that the methodologies discussed are not necessarily mutually exclusive. In practice, a combination of these different systems is generally used: a good software engineer tailors his approach to the situation, and employs the combination of tools and methods appropriate to the circumstances.

In light of the advantages of each methodology, a hybrid paradigm was adopted which incorporated the best characteristics of each. The use of storyboarding was used instead of Rapid Prototyping to provide the same benefit - clarification of user needs. The overall structure of the hybrid methodology, shown in Figure 4, resembles the Classical Life Cycle

model. Like this model, the system specifications and preliminary design are done at the very beginning. However, detailed design, coding, and testing are done in a version-wise manner like that of Iterative Design. Finally, after the last version of the system has been produced, the product is installed and begins the maintenance portion of its life cycle. The advantage of this methodology is that the designer does not incur as much overhead in finishing up one version and starting up the next as he would using the Classical Life Style Methodology. At the same time, he retains the benefits of low risk and frequent stopping points associated with Iterative Design.

The kernel system was developed first, consisting only of a map (with the FEBA), the grid, and bases/units. Subsequent versions incorporated enhancements such as graphical indication of weather conditions and interaction with the symbols. Development of these subsequent versions continued as long as time permitted, but were discontinued in time to put all documentation and code in final form for delivery.

This chapter reviewed design considerations and constraints specific to this project. In addition, the selection of methodology was discussed. The next chapter relates the process of selecting appropriate hardware and systems software.

IV. Selection of Hardware and Software

Selection of appropriate hardware and software was important not only because of its bearing on the success of this thesis, but also from the point of view of obtaining resources which will not immediately become obsolete. In the realm of automation, such a choice typically involves balancing the required level of functionality against the cost. In situations like this one, where future outlays of funds for upgrades are likely to be few and far between, it is important to purchase equipment that will meet not only current needs but also those of upcoming years. In addition, it is useful to extrapolate trends in order to judge which hardware and software will still be supported in the future, and hopefully to anticipate which ones will become so widely-used as to become a standard. As demonstrated by the success of the IBM PC and MS-DOS, for example, this can lead to a wide variety of products that support, or are supported by, the standard. This chapter summarizes some of the choices contemplated, and gives our conclusions.

4.1 Hardware Considerations

The Zenith Z-158's used by Kross and Brooks were used primarily because they were already available in large numbers at the AFWC; however, they have a number of undesirable qualities, particularly for graphics work. First, while they were advertised as being IBM PC XT compatible, they did not fully live up to this claim. There were some problems with the use of the Ingres local area network on these machines which Captain Kross had to work around. In addition, the Z-158's are rather slow compared to the newer Z-248's which are now relatively in common use in the U. S. Government; speed of processing is generally a more critical consideration in computer graphics than in non-graphics programming. Finally, the Z-158's were only supplied with the standard IBM CGA graphics capability, which has relatively low resolution (640 x 200 pixels), and a small selection of colors.

Resolution refers to the number of pixels (short for picture elements) that the screen is divided into; the more pixels, the less grainy the picture will be. The IBM EGA graphics standard, which allows significantly higher resolution (640 x 350) and a much wider

selection of colors [19], is available for Z-248's. At the present time, EGA graphics are the highest quality graphics used extensively enough on PC's to be considered standard. Although EGA graphics is currently in widespread use, a new graphics standard for PC's, called VGA, is just hitting the marketplace. VGA provides a resolution of 640 x 380, and can simultaneously display 256 different colors out of a selection of 262,000. While VGA costs more, there is a good chance it may become the next standard on the market.

Although there are few graphics software packages available on the market for use with VGA at this time, there are many for EGA. Many VGA monitors are downward compatible; that is, they allow the software to issue VGA, EGA, or CGA commands, depending upon which mode they are in. By purchasing VGA hardware and using an EGA graphics software package, this project could be accomplished using currently available technology, and the AFWC would be prepared to take advantage of VGA software when it became available.

A quantum leap in graphics capabilities can be obtained by looking at graphic workstations such as the Sun or the DEC VAXStation GPX. Workstations generally provide resolutions of around 1024 x 1024 and a tremendous range of colors. In addition, workstations often have a number of the most common graphics processes built into their hardware: this enables these processes to be done quickly and easily, and without using up memory space to store routines to do them. There are two primary disadvantages of workstations over PC's: they are very expensive, and they tend to be very specialized. On the other hand, they tend to have much more widespread software support than PC's for three widely-acknowledged sets of graphics standards: Core, Graphical Kernel System (GKS), and Programmer's Hierarchical Interactive Graphics Standard (PHIGS).

Considering that the AFWC is likely to use whatever computers they purchase next for some time, it is important that the selection of hardware to be purchased for this project be done judiciously. Buying ahead in the computer market is always a tricky business; there is a subtle tradeoff between buying technology that will not become obsolete within a short period of time, and avoiding the purchase of a system that never really catches on (and is, therefore, less standard). Buying VGA technology would be a relatively conservative move, but would allow the AFWC to hold their own in the mainstream of technological progress

for several years. The ideal solution would be to locate a graphics software toolkit that provided support across PC's and a variety of popular workstations, and then implement the system on a PC. The question of software selection is discussed in the next section before discussing the final solution to this problem.

4.2 Software Considerations

A plethora of graphics software is available for PC's, including paint programs, business graphics packages, computer aided design (CAD) packages, general graphics toolkits, and mouse driver packages, as well as many others. Unfortunately, most of these are not appropriate for this project. Paint programs generate images, but these images generally can't be used outside of the paint program. Business graphics packages generally produce standard business charts, like bar and pie charts, given input data. CAD programs are useful for producing highly complex three-dimensional images, but like the paint programs, are completely self-contained.

Graphics toolkits, on the other hand, provide calls to routines that perform such diverse operations as drawing, rotating, scaling, or moving figures. These calls can be invoked from within a regular program, thus providing the capability to integrate these functions with each other and provide interactive control via the program. These graphics packages are to graphics what high-level programming languages are to programming; they make the programmer's work easier by handling the minute details and allowing him to concentrate of the overall objectives of the system.

A mouse driver would also be a valuable tool for this system. A mouse is a hand-held sensing device with a tiny ball embedded on its underside. The user moves the mouse over a surface (such as a desk top), causing the ball to roll; the computer translates the movement of the ball into movement of the cursor on the screen. A mouse also has a number of buttons (typically, one to three) on its upper surface which the user can use to signal the interface.

The mouse driver software allows a programmer to associate sequences of commands with these buttons. In addition, most mouse drivers provide facilities for easy generation

of menus or displays of information. A menu consists of a list of possible actions; the user can select from these options with the mouse. A mouse could be used in the TWX system, for example, to move the cursor across the map onto a base and select the base by pushing a button, whereupon information about the base selected would be displayed on the screen. If there were several categories of data the user might wish to view, a menu might be generated to allow him to select which category he wanted.

A number of graphics toolkits were examined. Core, GKS, and PHIGS were examined, but few implementations of these were available for PC's. Another toolkit, Hierarchical Object-Oriented Picture System (HOOPS), met the majority of our selection criteria [1]. In particular, HOOPS provides support not only in the workstation environment, but also on PC's, and source code developed on one machine can be ported to any other supported configuration with no changes. Another selling point was that HOOPS routines could be called from a variety of host languages, including Microsoft C. This was particularly fortuitous since the database calls were to be written in PC Ingres, which also used Microsoft C as a host language. This would allow the graphics to be written in C using calls to Ingres routines to retrieve data and HOOPS routines to display it. The C language does not lend itself well to software engineering practices, but it does afford the programmer tremendous flexibility and power in accomplishing his goals. Besides, although HOOPS supported several other language bindings, the Ingres Embedded Structured Query Language (ESQL) being used for data retrieval did not, so C was destined to be the language used.

Another fortunate coincidence was that PC and DEC VAXStation GPX versions of HOOPS had already been purchased by a local unit under a site license, so it was available for use without any additional expenditure, although the AFWC would have to purchase HOOPS to maintain the software. In addition to the above-mentioned benefits, HOOPS provided a very important feature not found in any of the other toolkits: direct support of the object-oriented approach discussed in Chapter III. Other important features of HOOPS not found in most of the other toolkits included a Hewlett-Packard Graphics Language (HPGL) plotter driver, a PostScript laser printer driver, mouse drivers for each supported terminal device, and two high-resolution graphics drivers for the PC version. A

minor disadvantage was that HOOPS does not conform completely to any of the major graphics standards, although it provides capabilities very similar to those of PHIGS. On the other hand, because of the wide base of configurations and operating systems supported, and the portability across these systems, this factor is not really a problem.

4.3 Decision

In light of its numerous advantages, HOOPS was selected as the graphics toolkit. The initial decision on hardware was to use a Z-248 with one of the high-resolution display drivers supported by HOOPS. This would provide higher-quality graphics than would have been possible using EGA or VGA. There was some concern about the possibility of running out of main memory on the PC. The memory-resident HOOPS routines in demonstration programs took up a large part of the 640K accessible through MS-DOS calls. The PC Ingres driver routines in sample Ingres ESQL programs also took up most of main memory. It was postulated that a combined HOOPS/ESQL program might need more than the available 640K just to load their memory-resident routines. Early experimentation proved this to be the case, so development was shifted to a GPX workstation, which provided virtually unlimited main memory. HOOPS software developed up to this point was transferred to the GPX with a minimum of difficulties, substantiating the vendor's portability claims. Development was completed on the GPX. The combination of the upcoming OS-2 operating system and the corresponding HOOPS version should eliminate the problem of insufficient main memory on the PC. When they are released, the graphical interface can be ported back down to the PC if desired.

Various issues involved in the selection of hardware and graphics software were discussed in this chapter. The latest version of the system is described next.

V. Description of the Prototype System

This chapter describes the latest version of the interface. The final configuration of the system is recounted, followed by a detailed description of the initial display. Each of the interactive functions is characterized. Finally, the attributes of the interface that promote reusability with other wargames are summarized.

5.1 Configuration

The finished kernel system runs under the VMS operating system on a DEC VAXStation GPX networked via DECNET to a DEC MicroVAX. The Map-Based Graphical Interface software runs on the GPX, accessing the TWX database on the MicroVAX for positional coordinates, weather predictions, and detailed information about particular airbases or units. A separate database is generated from a template database for each seminar in order to permit multiple games to be conducted at once. At this time, geographical information is still maintained in sequential files on the GPX, rather than in the TWX database.

When the interface is run, it first asks which device the seminar database is on, and then prompts the user for a seminar number (to determine which database to access). After the user has provided this information, the system generates the display shown in Figure 5[1] in the appendix. The initial display takes one minute and fifteen seconds to generate.

Most of the data retrieval is done during the initial display generation. All geographical and positional data on units and bases is read in. In addition, each land unit's command level (division, corps, etc.) and type (armor, infantry, etc.) is obtained, and all weather forecasts are applied to generate the weather overlays, which are initially invisible. Under the current system, none of this information changes except during the nightly batch runs, so no screen update facility is necessary. The display simply uses the current information in the database each morning. The only information retrieved on an

[1] All figures showing the screen display have been placed in the appendix. Since there were so many of these, and they each take up an entire page, it would have been disruptive to place them in the text where they were referenced.

"as-needed" basis is the detailed information on specific units and bases displayed by the "Show Detail" function.

5.1.1 Initial Display The initial display (shown in Figure 5) consists of a context area, a message block, a function keyboard, a map legend, and a map display. The context area shows the seminar number, current game day and cycle, and the team affiliation. This is important because, although the white (control) team has access to all data, each player team is privy only to certain information. The members of the blue and red teams are entitled only to certain information about the other team's forces, although they have unlimited access to status of their own resources. Only the white team's display is implemented at this time.

The message block is used to give feedback, prompts, and error messages to the user. The function keyboard consists of a set of fourteen purple-colored keys matching the positions of the function keys on a GPX keyboard. Although currently interaction with this keyboard display is limited to use of the mouse, it is anticipated that future versions will allow the user to choose between invoking the function with the mouse, the function keys, or the button corresponding to a highlighted letter in the command name.

For example, the "Change Vis" function could be selected by any of the following methods: (1) position the mouse cursor over the appropriate key on the display and "click" on it (press the mouse button), (2) press the left-most function key on the GPX keyboard, or (3) press the "V" key on the GPX keyboard. Whenever a function is invoked, the key color changes to yellow to confirm the selection and the appropriate message is displayed. After the function has been performed (or in the case of the weather overlay and pan capabilities, turned off), the key changes back to its original color. Buttons for each of the following operations are displayed at the start-up level: "Change Vis", "Show Detail", "Weather", "Print Screen", "Make Screen", "Help", "Exit", and "Zoom In". In addition, there are six "Not Used" buttons. The function of each of these keys will be discussed in more detail in Section 5.1.2.

The map legend explains the symbology used on the map. The initial display shows the symbols for coast lines, country boundaries, the FEBA, the grid, and airbase/unit marker symbols. Additional symbols can be added and removed as needed.

The map itself consists of coast lines, country boundaries, the FEBA, airbase and land unit markers, and a superimposed game grid. "Markers" are a HOOPS-specific feature used to represent the bases and units at this level, and represent a necessary compromise. At the initial and second level of detail, units are in very close proximity to each other on the map. The NATO symbols are so complex as to be indistinguishable at these levels; the resolution is not high enough. However, markers contain enough detail to indicate the position and affiliation of the bases and units at the top level, and are easy to use in HOOPS. Airbases are therefore initially represented by markers in the form of asterisks, and units by markers which look like squares with two diagonals.[2]

If blue and red units are in contact (on the same location), the system indicates this by showing the red unit marker superimposed on, and slightly offset from, the blue marker. The grid is numbered and consists of dashed horizontal and vertical lines. Only one grid line is displayed for each ten X or Y coordinates. Grid lines could have been displayed for each position on the board, but this would have overcrowded the map display and slowed down the initialization process.

With the exception of bases and units, each type of symbol is color-coded and distinguishable from others by differences in width or pattern so that the information content is preserved even when printed out in black and white. Grids are black dashed lines. Coastal boundaries are represented with thin solid orange red lines, while country boundaries are shown as slightly thicker solid forest green lines. The FEBA is a solid violet red line which is thicker than any of the other lines. Units and bases are color-coded either red or blue; their affiliation can generally be distinguished on the hardcopy by their positions relative to the FEBA. Land masses and oceans are not filled in; they take on the background color (white).

[2]The hardcopies of the screen image show the unit markers as circles. This is not how they are displayed on the terminal. It is a quirk of the HOOPS routine that converts the screen information to Postscript.

5.1.2 Functions This section discusses the functions performed by each of the function keys on the keyboard display.

The "Change Vis" function is short for change visibility. When this function is selected, an intermediate menu pops up on the map window (see Figure 6), and the query "What do you want to change the visibility of?" is displayed in the message area. When the user selects one of the proffered menu items (for example, blue bases and units), the pop-up window disappears, a message appears telling which item's visibility is being changed, and the visibility of that class of objects is changed (see Figure 7). Essentially, the function toggles symbols on and off. If a class of objects is visible, then this function causes it to disappear from the screen; if it is currently invisible, then it is again displayed on the screen. At this time, the usefulness of this function is limited; however, it may used in conjunction with later functions to great advantage. For example, if a function is added that makes blue bases with logistics shortfalls blink, and the user wants to use it to concentrate specifically on logistics shortfalls on blue 4ATAF bases, he can simply turn off the visibility of the blue 2ATAF bases. With the 2ATAF bases invisible, the player can easily identify the bases he is interested in.

The "Show Detail" function brings up the message "Place cursor on base or unit you want detail on". When the user clicks on a unit or base symbol, the symbol turns black to confirm its selection, and a pop-up window appears in the opposite quadrant of the map window (see Figure 8 and Figure 9). This window contains detailed summary information about the item selected, including ID number, name, affiliation, and coordinates. If the selected object is a base, then the ATAF and status are also displayed. If the object is a unit, corps and current combat power are displayed. When the user has finished reading the information, he can remove the window by pressing the mouse button again. If multiple units are on the same location, the information on the next unit is displayed when the user clicks, and the process continues until all units at that location have been examined. If red and blue units are in contact, all of the units of one side are displayed first, then the units of the opposing side.

The "Weather" function, like "Change Vis", brings up an intermediate menu when invoked (see Figure 10). This intermediate menu prompts the user to select between the

day and night cycle weather forecasts. When the user selects one, the menu disappears, the weather overlay for that cycle is turned on, and keys for the three types of weather are added to the map legend (see Figure 11). Although only three weather zones are shown in this figure, up to eighteen different weather zones can be represented for each cycle. A zone can have one of three types of weather: good, fair, and bad. These weather forecasts affect mission planning in the game by influencing the chances of a mission being successful. Good weather is represented by a peach-colored diamond pattern, fair by a chartreuse-colored slant line pattern, and bad by a melon-colored vertical line pattern. Because the representations are different in pattern as well as color, the weather information can be obtained in hardcopy as well as on the terminal. When the overlay is turned off by clicking on the "Weather" key again, the overlay becomes invisible and the weather symbols disappear from the legend.

The "Print Screen" function was used to print the figures in the appendix. When invoked, it displays the message "Printing display - please wait" for about five seconds while HOOPS converts the screen information to Postscript and sends it to the printer. According to the HOOPS manual, this function could also be used to produce a hardcopy on a HPGL plotter with minor changes in environmental variables, although this capability was not explored in this thesis.

The "Make Suggest" feature is not implemented at this time. Its eventual purpose will be to allow the players to make comments and suggestions on-line without leaving the game environment. This is described in more detail in the last chapter. The "Help" feature also is not implemented in this version.

The "Zoom In" function brings up the message "Place cursor on center of area to be zoomed in on, then click". When the user does so, the display zooms to twice the scale, centering on the location designated by the cursor. At the same time, the two left-most "Not Used" keys are replaced by a "Zoom Out" key and a "Pan Across" key. These keys will remain here until the display is zoomed out to the original scale, at which time they will be replaced with the "Not Used" keys. When the display is zoomed in to the third level of detail, all airbase and unit markers are replaced with their NATO symbol counterparts showing unit type and level (see Figure 12). At this level, the units are far enough apart

to allow discrimination between the different symbols. Multiple units at the same location are represented by placing a smaller box inside the regular unit box. Opposing units in contact are represented with a blue unit enclosing a red box. The display can be zoomed in on indefinitely.

"Zoom Out" is simply the reverse of "Zoom In", except that the user need not specify a location to become the new center. The new center of the screen is automatically selected in such a manner that the center of the TWX map will be at the center of the screen when it is zoomed back to the original (and lowest) level of detail. When the display passes from the third level of detail to the second, the base and unit symbols are replaced with markers.

"Pan Across" brings up a window with four red arrows as shown in Figure 13. By clicking on one of these arrows, the player can pan the display a preset distance in the direction indicated by the arrow. Neither this function nor the "Zoom In" will allow the player to wander off the map. The panning window can be turned off either by clicking on the function key again, or by zooming out to the lowest level of detail.

All of the "Not Used" buttons, if clicked upon, display the message "Not implemented at this time". The "Exit" function clears the display window and exits the system.

5.2 Applicability to Other Wargames

The interface was designed to be reusable. It can be employed, with little modification, to build map-based graphical interfaces for other wargaming systems. A utility program prompts the developer for specific details, extracts the appropriate data from the MWDB II files based on the developer's answers, and then places the information into a map data file which can be accessed by the interface program. The developer must specify several parameters: minimum and maximum latitude and longitude of the area of the world to be represented, minimum and maximum X and Y coordinates of the game system, and which of five levels of detail to use. This information is stored in a header record in the map data file. When the interface is invoked, it automatically displays the map and converts the game system coordinates to the corresponding point of the map.

This system will work for wargames based upon a cartesian coordinate system, but not for those using a hexagon system. Also, since the aspect ratio of the geographic area chosen will probably differ from that of TWX (2:1), the user will have to modify the placement and shapes of the other windows to fit the screen. For example, if the wargame to be represented had a square (1:1) aspect ratio, the developer would probably want to move the function keyboard alongside the map, instead of below it. Because of the structured manner in which the system is written, these changes can be made relatively easily. Only the positions of the corners of the windows (and some text) need be changed; any functions associated with the windows will not need to be modified.

This chapter has described the graphical interface developed in this thesis. The next chapter presents this researcher's final conclusions and recommendations for follow-on work.

VI. Conclusions and Recommendations

This chapter describes the evaluation of the completed system, and proposes an approach for implementing the system at the AFWC. It then discusses the strengths and weaknesses of the system and of the approach and tools employed in developing it. In addition, it discusses a number of potential future enhancements to the interface which were not implemented in the initial version due to time constraints but which are recommended as follow-on projects.

6.1 Evaluation

A relatively informal evaluation method was used to rate the final product. The primary goal of this evaluation was to determine weaknesses of the system with an eye towards shoring them up, rather than to give a quantification for purposes of comparison with other systems. First, the system was used through several entire runs of the exercise in the development environment. This identified a number of minor problems caused by misunderstandings about movement and positioning of the units.

After these problems had been corrected, a number of fellow students and faculty members unfamiliar with the system were invited to try it out. Again, this brought to light a number of weaknesses of the system. Most of these were corrected, although some were outside the scope of this project and were therefore deferred for future work.

The final, and most important evaluation, was that of the users. The finished system was demonstrated at the AFWC in November. The users were pleased with it, and were quick to grasp the implications in terms of future enhancement. In light of this response, the system was deemed a success.

6.2 Implementation at the Air Force Wargaming Center

After the addition of a couple of additional features described below and the final selection of a hardware configuration by the AFWC, the system will be ready for implementation. A full set of documentation, including a user's manual, maintenance manual, and an installation manual, has been developed and will be delivered to the AFWC with

the software. The AFWC has a VAXStation GPX available, so it is anticipated that an environmental test will be conducted by running several a seminar using the interface on the GPX in parallel with seminars using the old system. If this test proves successful, licensing agreements will be obtained for versions of HOOPS and Ingres for the hardware configuration selected by the AFWC and the software will be ported to these systems.

6.3 Strengths and Weaknesses of the System

The Hybrid Methodology worked quite well for this project. The development of a kernel system, followed by successively enhanced revisions of the system, produced very satisfactory results. In addition, the use of "storyboards" early in the requirements specification process enabled us to avoid several misunderstandings concerning the users' requests. The identification and use of a graphics software environment that supported object-oriented graphics programming was also quite fortunate. The ability to operate upon graphical objects and classes of objects as named entities rather than having to deal with each of their constituent elements allowed a much higher level of abstraction, making the code much more easier to read and write. In addition, the inheritance of graphical attributes from classes to subclasses of objects saved a great deal of potentially redundant coding.

The primary advantage of the interface is its ability to graphically represent a variety of information that was previously available only in tabular form. Users are able to zoom and pan on the display, bring up additional information on specific bases and units, and graphically display the weather forecast. In addition to these specifically-requested abilities, the system is able to selectively toggle the visibility of classes of objects and to print a hardcopy of the screen without loss of information. Units and airbases can be represented with standard NATO map symbols, although this is only feasible when the display has been zoomed in to display a smaller portion of the map. This constraint is a limitation of the TWX wargame, rather than of the interface. Applied to a wargame with a more localized battle area, the markers could be dispensed with entirely, and entities represented exclusively with symbols. Alternately, the display could be replaced with a larger screen with higher resolution. It is unlikely, though, that such a monitor would be feasible at this

time, since a 19" diagonal screen with a resolution of 1024x864 was used for this thesis. The use of direct manipulation upon the symbols, combined with an appropriate selection of icons to represent objects, supports a strong and highly abstract user view of the system. The details of the implementation are no longer imposed upon the user.

A second strength of the interface system is its portability. HOOPS, Ingres, and C are supported across a tremendous variety of computers. This is important not only because it allows more flexibility in selecting a target configuration for TWX, but also because it increases the reusability of the package. The system can be ported to a variety of computers, enabling the addition of map-based interactive graphics capability to other wargaming systems in their native environment.

A third benefit of the system is its reusability with wargames other than TWX. This system can be adapted with very minor changes to provide the same functionality for a large number of other wargames. The inherent limitations are that the wargame must use a Cartesian coordinate system and must run on a configuration that will support HOOPS and C. The software could be adapted to run against a database composed entirely of sequential files rather than a DBMS, but this would require lots of additional programming, reduce the portability of the system by tying it to a specific file structure, and hurt flexibility, diminishing the ease of future enhancements.

A fourth major benefit is the flexibility of the system. HOOPS provides a large variety of high-level functions that can be performed upon graphical objects with a minimum of difficulty. For example, zooming and panning were relatively straightforward, as was changing the visibility of entities.

On the other hand, generating entities was much more rigorous than manipulating them. "Paint" facilities to interactively draw symbols, screens, and menus for HOOPS manipulation would be a definite plus. Currently, developing such objects is a matter of programming in the endpoints of lines, starting points of text, and corner points of windows, as well as specifying the color, size, and visibility of each. In particular, the absence of circle or ellipse-drawing primitives was sorely missed. Much of this could be automated, freeing the developer to specify program-driven transformations upon the objects.

6.4 Suggested Enhancements

A large number of additional enhancements to the interface were possible, but it was not feasible to attempt them as part of this project due to time constraints. A number of these potential enhancements are discussed in this section.

One additional feature that must be added to the interface before it will be ready for use in actual TWX seminars is a password system to discriminate between members of the different teams. Although the control team is privy to all of the information displayed by the current system, members of the player teams should only be allowed to retrieve certain information, particularly about the other team's resources. Available information about location, type, and status of the opposing team's airbases and land units is based upon the number and type of intelligence-gathering missions flown. The interface currently allows users to call up all information in the database about both teams. The system needs to be expanded to require players to log in with team-specific passwords, and to allow them access to only the appropriate information.

Another feature that should be implemented before releasing the system into production is the "Help" function. This function would bring up a context-sensitive help page describing in layman's terms what is expected at this time. Fortunately, the system was designed using modern human interface principles to be easy to learn and use, with maximum feedback provided to the user at each stage, so the help system need not be elaborate.

The "Make Suggest" function, like "Help", was not actually implemented in this version of the system. This is basically just an on-line suggestion/comment form. A form would be brought up soliciting identification information (initiator, time, date) and allowing the user to write a narrative description of the suggested fix or enhancement. In addition, the capability to attach a copy of the current display to this suggestion might aid in describing a specific situation. These suggestions should be stored as separate entities, and a system developed for managing them.

At this time, the map data extracted from MWDB II is stored in a sequential file on the GPX terminal. This has two disadvantages: it forces redundant storage of the

information and it reduces the portability of the system. Since the file must be located on the GPX terminal, a copy of the file must be made and stored on each terminal along with the program. If a change is made to this file, each copy must be replaced. The portability is reduced because the file is stored in binary (operating system dependent) form, rather than ASCII form. This problem was encountered when transferring the system from the PC to the GPX early in the project. The reason for this choice of storage format was that the volume of map data was very large, and binary form is much more compact than ASCII text.

Storing this data in the central Ingres database would eliminate both of these problems. The data would be stored once in the exercise master database. Whenever a game is run, a game-specific copy of the database is made, and then deleted after the seminar is completed. In addition, translation of Ingres data from one operating system to another is handled automatically by existing Ingres routines, and is invisible to the user. On the other hand, retrieving the map data directly from a sequential file on the same machine is probably the fastest method of getting the data. Retrieving it from a remote Ingres database will introduce not only database retrieval overhead, but also networking overhead. The best way to determine whether this is feasible is to try it and see how well it works.

Enhancing the conversion routine to allow use with hexagon-based grid systems would increase the applicability of the toolkit to include a much larger group of wargames, although there is some question as to the usefulness of a hexagon-based system on computerized wargames. The original idea behind using a hexagon-based game board was to allow easy correlation of movement allowances of specific units to number of grids moved. Since movement distances can be automatically calculated in computer-based wargames, there may no longer be a valid need for using a hexagon-based coordinate system.

Another enhancement that would increase the reusability of the system is to build in a set of map projections. These were not needed for TWX because the area to be mapped was relatively close to the equator, but this would not be the case for many wargames.

It was suggested in the last chapter that the interface be modified to allow the player more variety in selecting functions. The current system allows interaction only through the use of the mouse, but future versions should also allow the user to chose a function by pushing a function key or the letter key on the keyboard corresponding to the function name.

Finally, as pointed out above, one of the shortcomings of the current system is the labor-intensive nature of defining the graphical objects. Development of a set of tools for interactively defining graphical objects, screens, and menus would greatly increase the usefulness of this system as a generic toolkit for adding map-based graphical interfaces to existing wargaming systems. The current method used for panning works well, but using elevator bars similar to those in most commercial paint programs would eliminate the problem of the panning window obscuring part of the map, while keeping the user aware of his current position relative to the rest of the map. Development of a generic routine for making windows with automatic elevator bars would be particularly useful. Extending this idea further, the selection menus used by the visibility, weather, and panning routines all use a icon-selection convention similar to hypertext. The details of setting up this sort of submenu could be encapsulated in such a manner as to provide a generic menu-building capability, while hiding the details from future developers. Incorporation of algorithms for automatically resizing and repositioning the context area, legend area, keyboard display, and message area according to the aspect ratio of the selected geographical area would take some of the burden off of future developers.

6.5 Conclusion

This thesis detailed the development of a map-based graphical interface system for the Theater War Exercise. In addition to the primary goal of producing a working interface for TWX, two secondary design criteria were adhered to. One of these was that the interface would be reusable with other wargaming systems. The other was that the tool would be portable across a wide variety of computers. The system met all of these criteria. In view of the warm reception it received, and since there is still plenty of room for enhancement, it is anticipated that this project will spawn many future projects.

Appendix A. *Display Figures*

This appendix contains figures showing the display. These figures, although referenced in Chapter 5, are placed here to avoid disrupting the flow of the text.

These images were produced with the "Print Screen" function. At this time, there are several slight bugs in the HOOPS routine for producing hardcopies of the screen. One of these bugs causes unit markers to be displayed differently on the hardcopy than on the terminal.

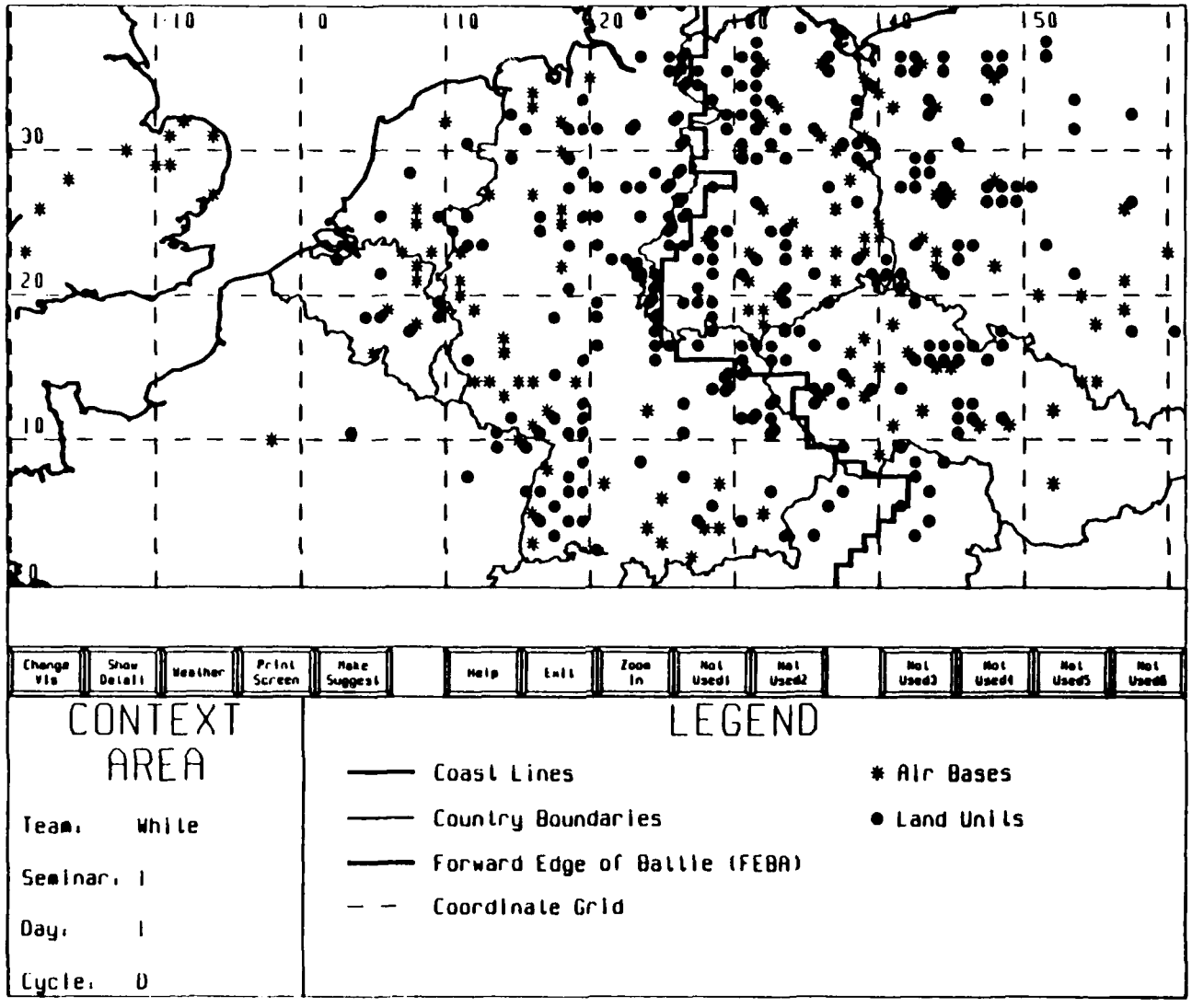


Figure 5. Initial Display

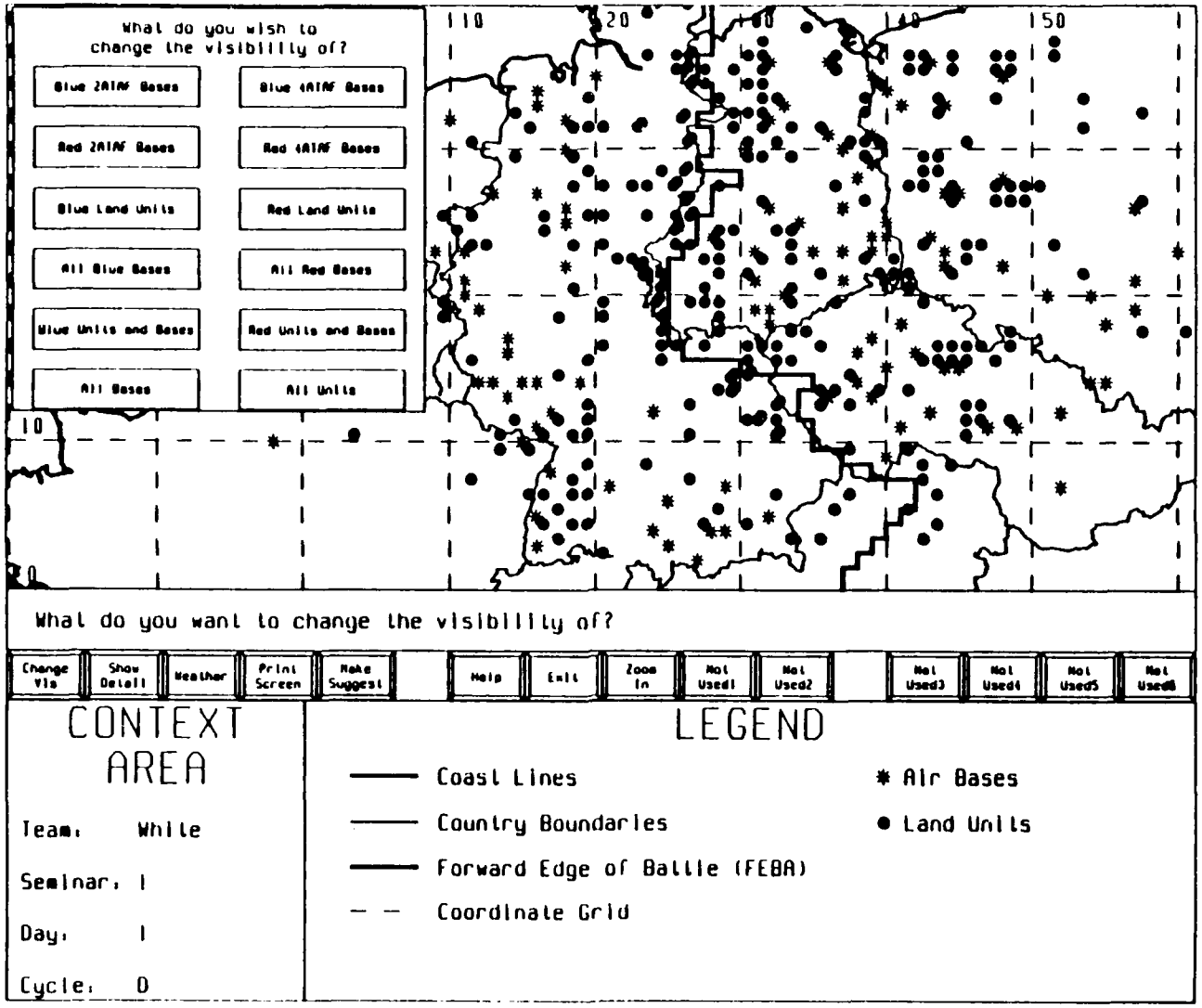


Figure 6. Visibility Selection Menu

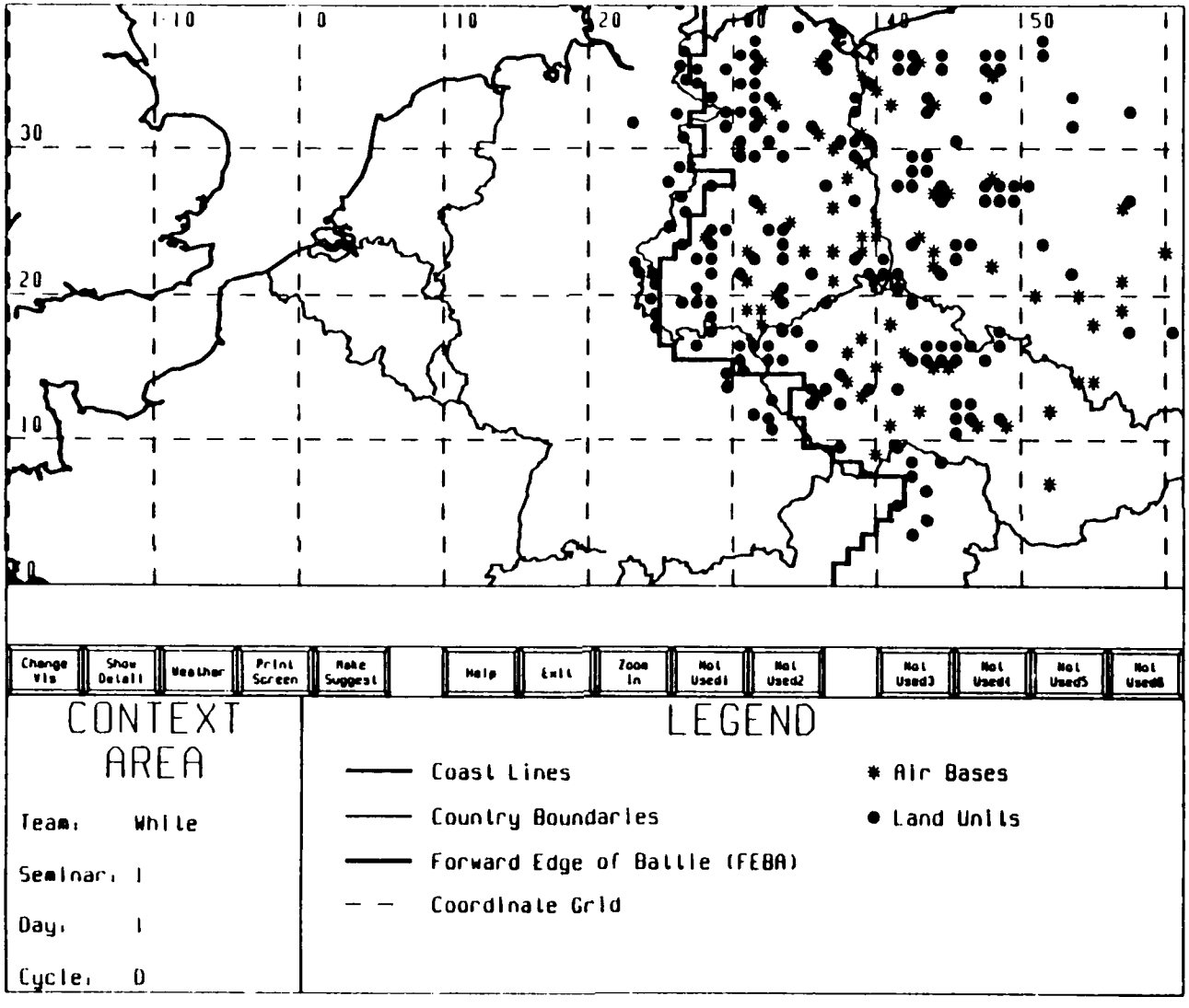


Figure 7. Visibility of Blue Bases and Units Turned Off

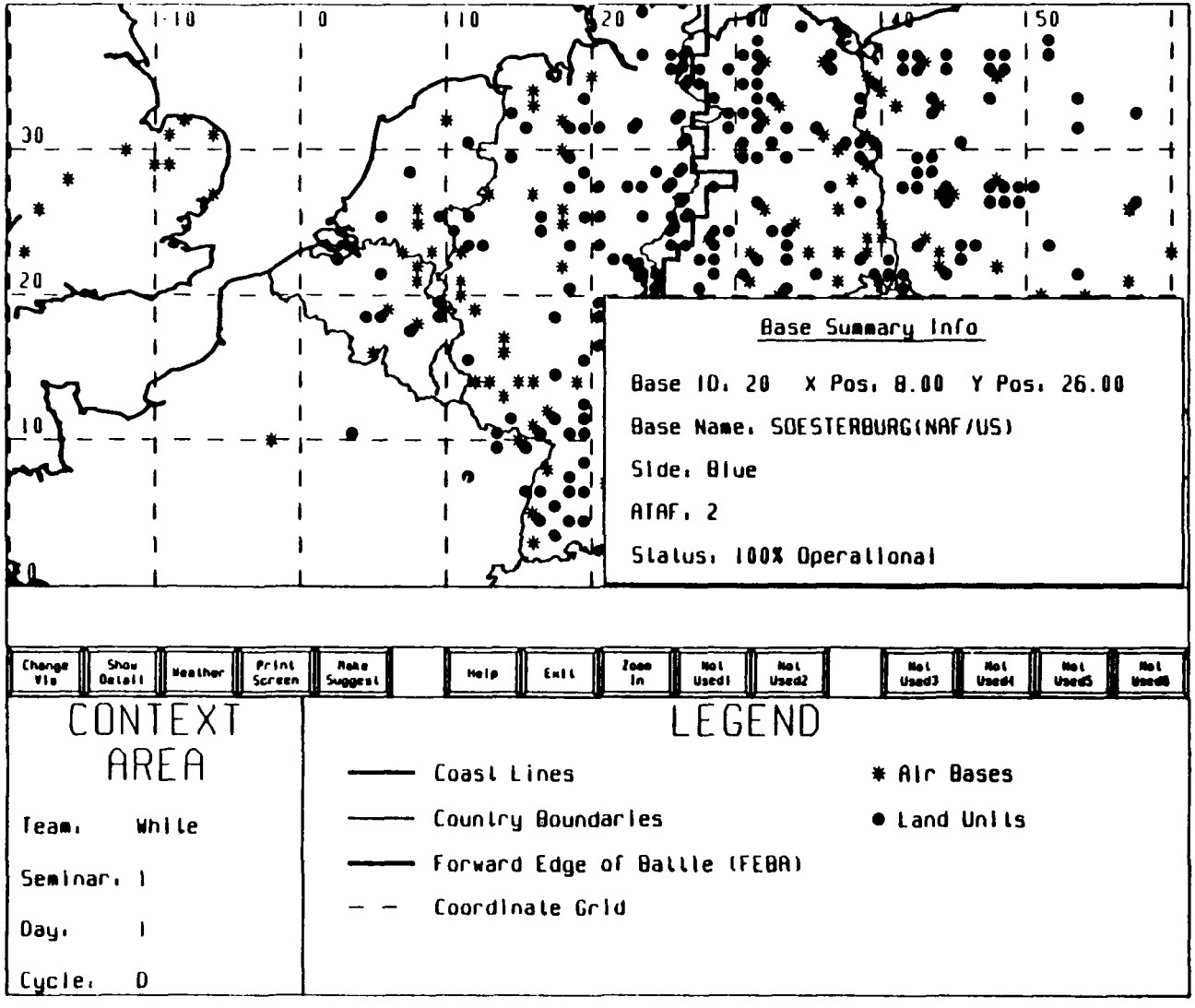


Figure 8. Show Detail Window: Base

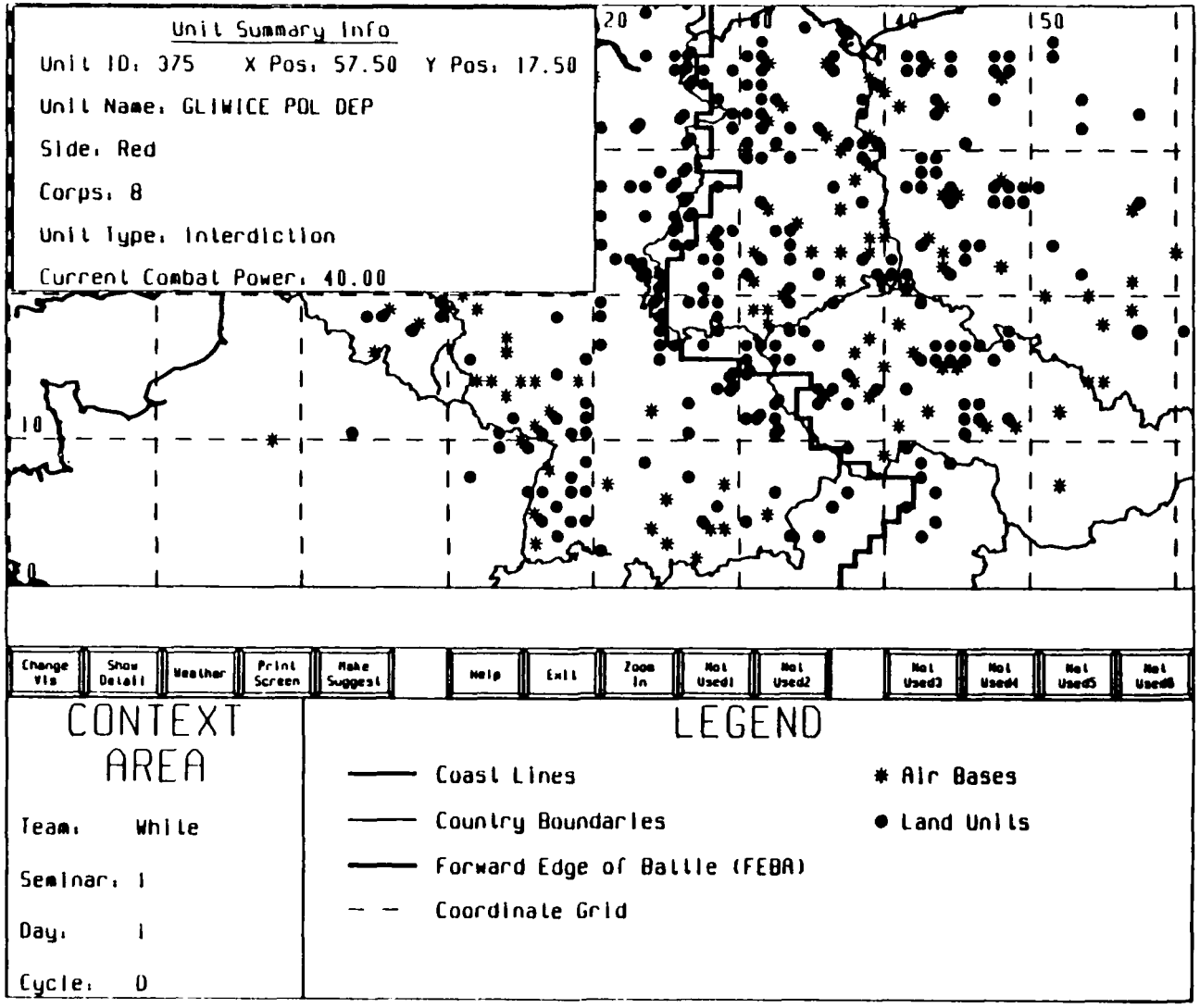


Figure 9. Show Detail Window: Unit

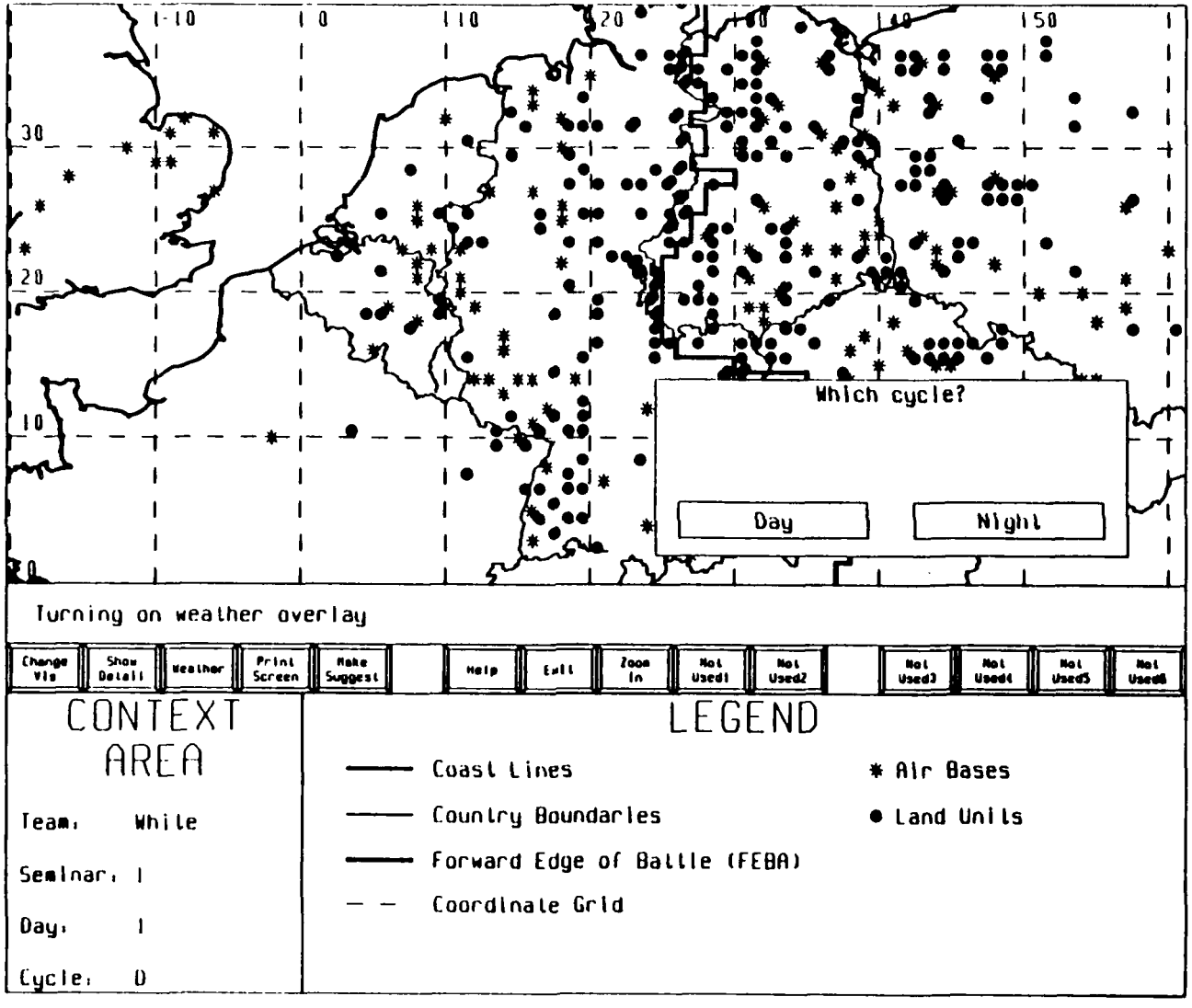


Figure 10. Weather Overlay Menu

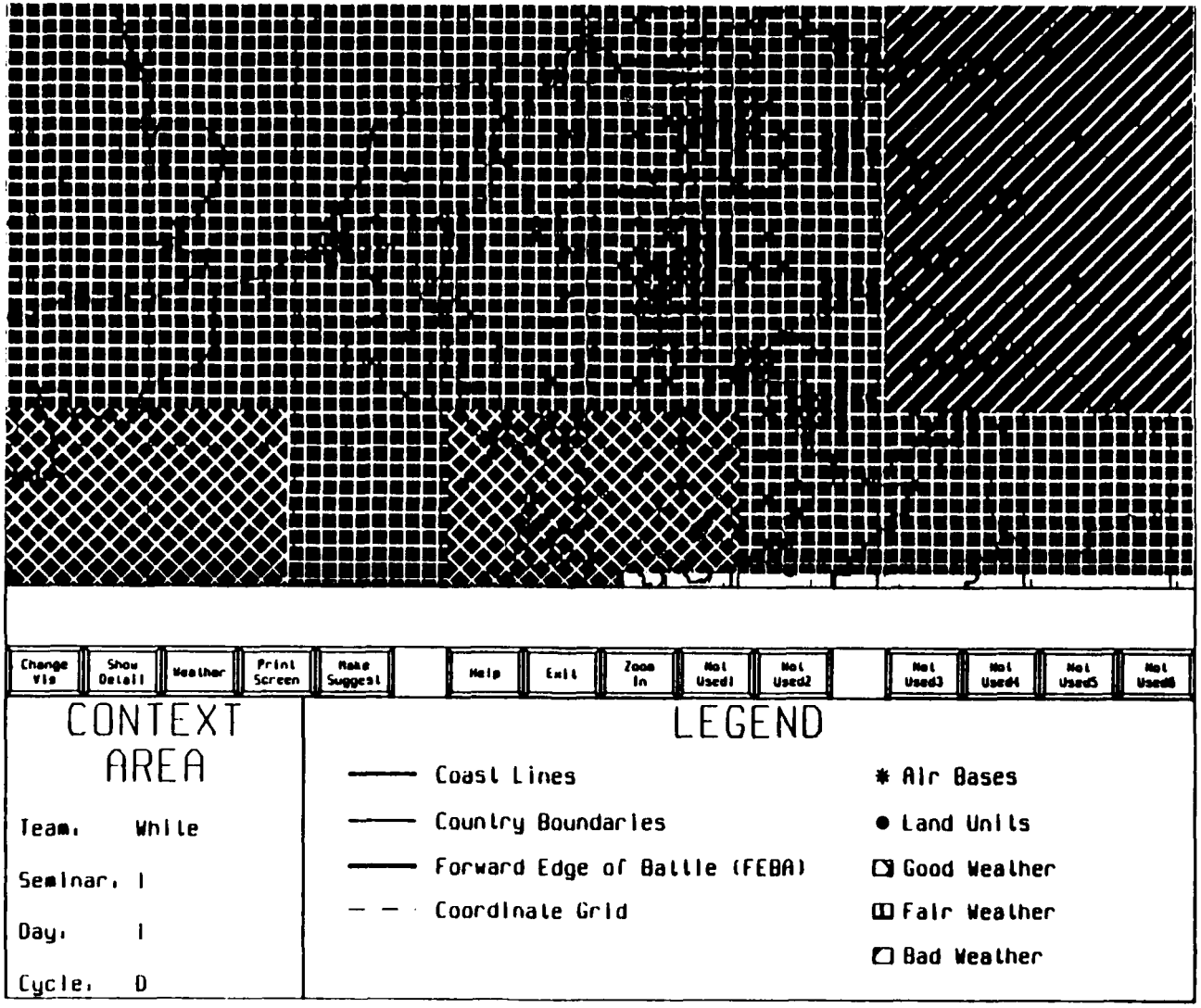


Figure 11. Weather Overlay: Night Cycle

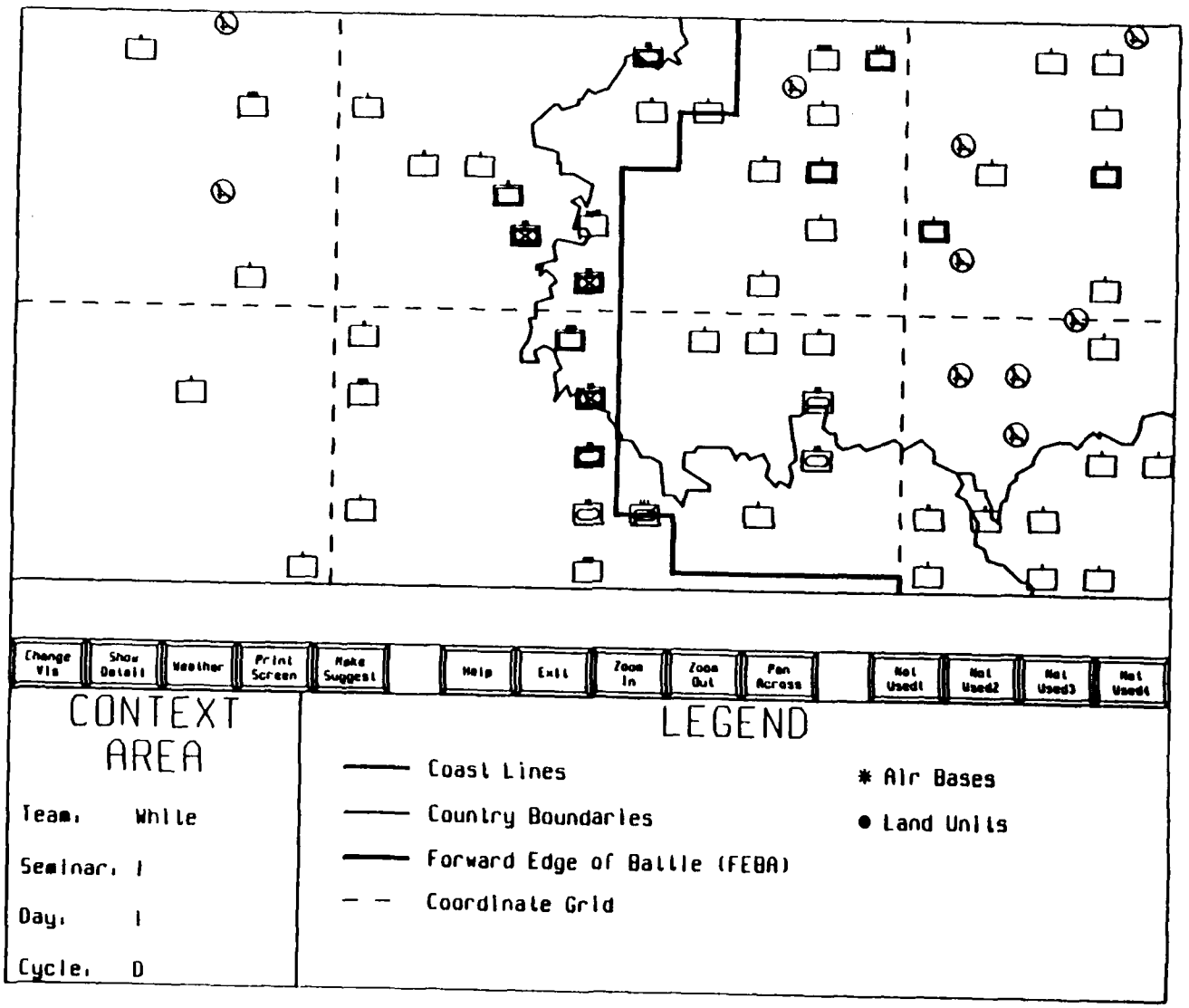


Figure 12. Third Level of Detail

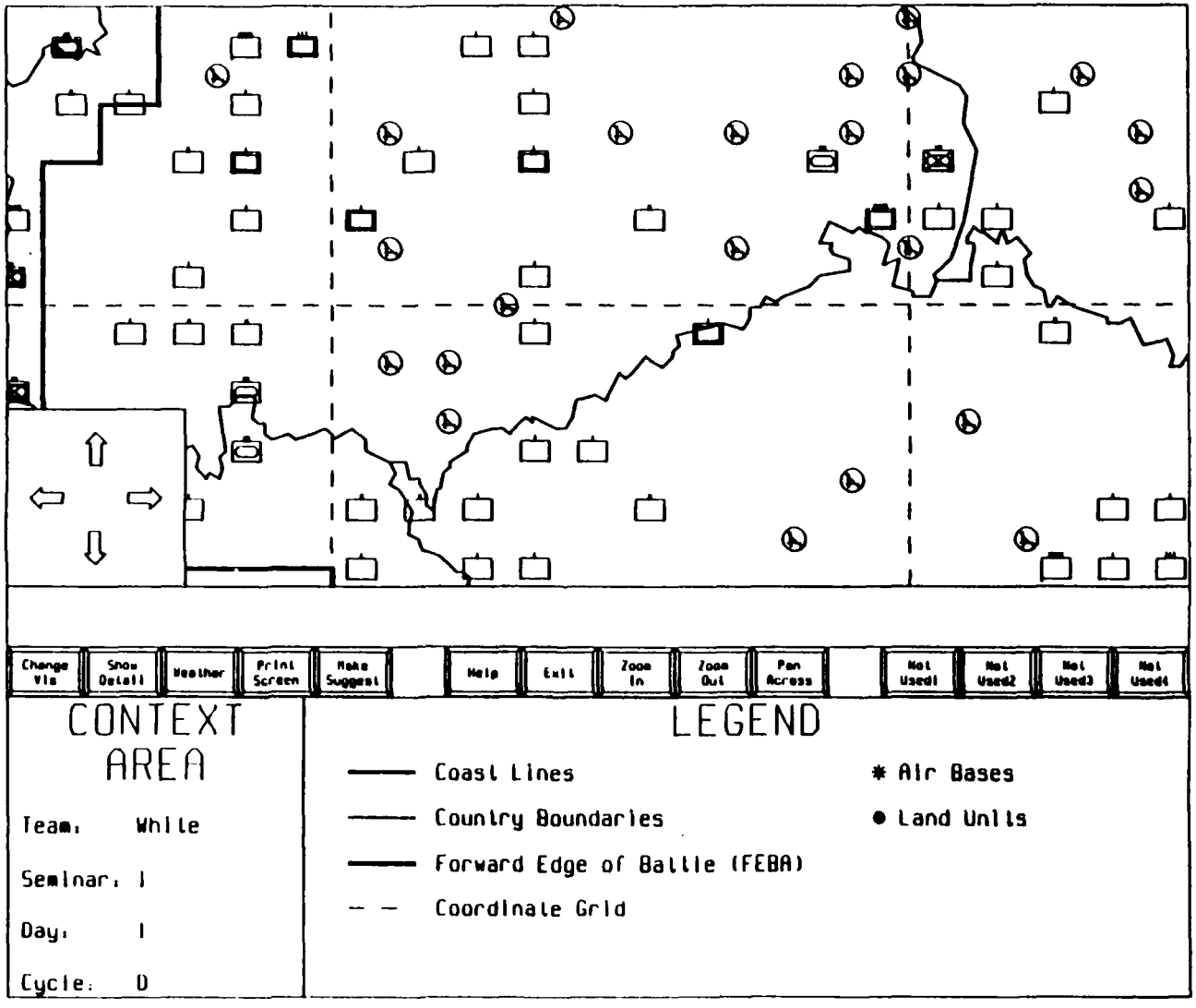


Figure 13. Panning Across the Display

Bibliography

1. *HOOPS Graphics Software.* Ithaca Software, The Clinton House, Ithaca, NY, 1988. Company Brochure.
2. *Military Symbols, AFM 21-30.* Department of the Army, Washington: HQ USA, May 1970.
3. *Theater Warfare Exercise Users Handbook 1987.* Air Force Wargaming Center, Maxwell AFB, AL, 1987. Unpublished Manual.
4. Major Phil Amburn. Instructor of Computer Systems, Electrical and Computer Engineering Department, School of Engineering. Personal Interviews. Air Force Institute of Technology, Wright-Patterson AFB, OH, 1 February 1988 through 12 February 1988 .
5. Bruce A. Artwick. *Applied Concepts in Microcomputer Graphics.* Prentice-Hall, Inc., New Jersey, 1984.
6. John A. Battilinga and Judith K. Grange, editors. *The Military Applications of Modeling.* Air Force Institute of Technology Press, Wright-Patterson AFB, OH, 1984.
7. Kenneth R. Boff, Lloyd Kaufman, and James P. Thomas. *Handbook of Perception and Human Performance, Vol. 1.* John Wiley and Sons, New York, 1986.
8. Captain Michael D. Brooks. *Developing a Database Management System and Air Simulation Software for the Theater War Exercise (ADA189681).* Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87D-6.
9. Lt. Col. Daniel B. Fox. *A Conceptual Design for a Model to Meet the War-Gaming Needs of the Major Commands of the United States Air Force.* Technical Report AU-ARI-84-8, Airpower Research Institute, Air University Press, Maxwell AFB, AL, July 1985.
10. Timothy C. Hawkins and G. Grank Thompson. Quickscreen. In *Proceedings of the 1985 Winter Simulation Conference*, pages 575–585, IEEE Press, New Jersey, 1985.
11. Edwin L. Hutchins et al. Direct manipulation interfaces. *Human-Computer Interaction*, 1:311–338, 1985.
12. Captain Mark S. Kross. *Developing New User Interfaces for the Theater War Exercise (ADA189744).* Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87-19.
13. Donald MacGregor and Paul Slovic. Graphic representation of judgemental information. *Human-Computer Interaction*, 2:179–200, 1986.
14. Allen Newell and Stuart K. Card. The prospects for psychological science in human-computer interaction. *Human-Computer Interaction*, 1:209–242, 1985.
15. Robert M. O'Keefe. What is visual interactive simulation? (and is there a methodology for doing it right?). In *Proceedings of the 1987 Winter Simulation Conference*, pages 461–464, IEEE Press, New Jersey, 1987.

16. Fred Pospeschil. On-line documentation file for micro world data bank ii. Bellevue, NE.

17. Captain Mark A. Roth. Assistant Professor of Computer Systems, Electrical and Computer Engineering Department, School of Engineering. Personal Interviews. Air Force Institute of Technology, Wright-Patterson AFB, OH, 1 February 1988 through 12 February 1988 .

18. Robert P. Sabo and Remmington G. Bishop. *LO-CO-GRAF: Generating Maps to Support Command and Control/Crisis Management Using Small Computers (ADA193964)*. Master's thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1987. AFIT/GCS/ENG/87D-6.

19. Kari Scott. Buyers bewildered by variations in color monitors. *PC Week*, 4:51–52, 1987.

20. Major Glenn D. Simon. *Interactive Graphical Support for a Small-Unit Amphibious Operation Combat Model (ADA128561)*. Master's thesis, Naval Postgraduate School, Monterey, CA, March 1983.

21. Bill Verplank and Scott Kim. Graphic invention for user interfaces: an experimental course in user-interface design. *SIGCHI Bulletin*, 18:50–66, January 1987.

22. Richard L. Vitek. Data base requirements for geographical mapping. In A. Blaser, editor, *Data Base Techniques for Pictorial Applications*, Springer-Verlag, Berlin, 1980.

Vita

Darrell A. Quick [redacted] Immediately after graduation, he enlisted in the Air Force Reserves as an Avionics Instrument System Specialist assigned to the 433 Tactical Airlift Wing at Kelly AFB, Texas. He was granted an Honorable Discharge from the Air Force in 1979 to take advantage of an Air Force ROTC Scholarship he was awarded. He attended Southern Methodist University in Dallas, Texas, for a year, and then transfered to Southwest Texas State University (SWTSU) in San Marcos, Texas. Darrell graduated with a bachelor's degree in Computer Science from SWTSU and was commissioned as a second lieutenant in the United States Air Force in December 1983. His initial assignment at Randolph AFB, Texas, was as an analyst/programmer, and later as the Database Administrator for the Air Training Communications Division. He is currently a M.S. student in the School of Engineering at the Air Force Institute of Technology and a Captain in the United States Air Force. His next assignment is to the Cheyenne Mountain Complex in Colorado Springs, Colorado. His current interests include graphics, software engineering, and database management systems.

[redacted]

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/88D-16	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AU CADRE/WG Maxwell AFB, AL 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
DEVELOPING MAP-BASED GRAPHICS FOR THE THEATER WAR EXERCISE

12. PERSONAL AUTHOR(S)
Darrell A. Quick, Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988 December	54

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Computer Graphics, Interactive Graphics, War Games
12	05		
15	06		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Thesis Advisor: Mark A. Roth, Capt, USAF

Approved for release in accordance with AFR 190-1
12 Jan 1989

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Mark A. Roth, Capt, USAF	22b. TELEPHONE (Include Area Code) 513-255-3576 / 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.*

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Block 19 Continued.

The Theater War Exercise (TWX) is a computer assisted, theater level, airpower employment exercise conducted by the Air Force Wargaming Center as part of the Air War College curriculum. 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 hardcopy reports which were difficult to understand and cumbersome to use. This thesis discusses the development of a map-based graphical interface to TWX which provides a much more user-friendly interface to the information and establishes a baseline for the development of new techniques for user interaction with the exercise.

This interface was developed under a hybrid methodology which takes advantage of the best features of both the Classic Life Cycle Methodology and the Iterative Design Mehtodology, utilizing an underlying object-oriented model to represent graphical entities and classes. With the exception of geographical data, which is extracted from sequential files, all information is retrieved over a network from a remote relational database management system using Embedded Structured Query Language.

Given a set of maximum and minimum latitude and longitude coordinates, this system can display a map of any geographical region. Since the system also performs an automatic conversion to a user-specified game coordinate system, it can be adapted with minor changes to provide map-based graphical interaction with a wide variety of other wargaming and simulation systems. In addition, because of the high level of abstraction and wide product base of the languages and utilities used, the interface software is portable across many different computers and operating systems. The flexibility, portability, and power of this interface make it a useful tool for interaction with map-based computer systems.