DTIC FILE COPY

AD-A204 417

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ENHANCEMENTS TO THE
COMMAND AND CONTROL WORKSTATION
OF THE FUTURE

by

Rex Cobb

December 1988

Thesis Advisor: Dr. Michael J. Zyda

Approved for public release; distribution is unlimited.

DTIC
ELECTE
16 FEB 1989
S D
E

89 2 15 009

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION 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) ENHANCEMENTS TO THE COMMAND AND CONTROL WORKSTATION OF THE FUTURE

12. PERSONAL AUTHOR(S) Cobb, Rex (NMI)

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	1988 December	82

16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Command and control, three-dimensional visual simulation, user interface. thesis

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The modern tactical commander has a flood of sensory and intelligence information at his disposal. A tool is required to sort the information that is most pertinent to the decisions he must make at that time. The Command and Control Workstation of the Future (CCWF) is an interactive, near real-time system that displays multiple windows providing visualization of the terrain in both two and three dimensions. The terrain is drawn with Defense Mapping Agency Digital Terrain Elevation Data using three image resolutions in order to display large areas of terrain. This study consists of enhancements to prior work on the CCWF. The focus of this ~~effort~~ is in two areas. One is enhancing the maintainability of the CCWF system through use of standard software engineering techniques (i.e. modularization, descriptive labeling, etc.). The second focus is on improving the user interface of the CCWF. (KR) ←

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Michael J. Zyda	22b TELEPHONE (Include Area Code) (408)646-2305 22c. OFFICE SYMBOL 52Zk

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete

Approved for public release; distribution is unlimited.

ENHANCEMENTS TO
THE COMMAND AND CONTROL WORKSTATION
OF THE FUTURE

by

Rex Cobb
Lieutenant, United States Navy
B.S., University of Florida, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

December 1988

Author: _____
Rex Cobb

Approved by: _____
Michael J. Zyda, Thesis Advisor

_____
John M. Yurchak, Second Reader

_____
Robert B. McGhee, Chairman
Department of Computer Science

_____
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

The modern tactical commander has a flood of sensory and intelligence information at his disposal. A tool is required to sort the information that is most pertinent to the decisions he must make at that time. The Command and Control Workstation of the Future (CCWF) is an interactive, near real-time system that displays multiple windows providing visualization of the terrain in both two and three dimensions. The terrain is drawn with Defense Mapping Agency Digital Terrain Elevation Data using three image resolutions in order to display large areas of terrain. This study consists of enhancements to prior work on the CCWF. The focus of this effort is in two areas. One is enhancing the maintainability of the CCWF system through use of standard software engineering techniques (i.e., modularization, descriptive labeling, etc.). The second focus is on improving the user interface of the CCWF.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 1

TABLE OF CONTENTS

I. OVERVIEW 1

A. THE COMMAND AND CONTROL WORKSTATION 1

1. Discussion 2

a. Use of color 2

b. Changing viewpoint 2

c. Use of windows 3

d. Use of popup menus 4

B. SCOPE OF THIS STUDY 4

II. FOCUS 5

A. PRELIMINARY RESEARCH 5

B. SYSTEM MAINTAINABILITY 5

C. USER INTERFACE 6

D. SUMMATION 7

III. IMPROVEMENT OF SYSTEM MAINTAINABILITY 8

A. SYSTEM ANALYSIS 8

1. Preliminary Study 8

2. In-depth Study 8

3. Problem Statement 9

4. Defining User Requirements 9

5. Requirements Statement 10

B. PROGRAM STRUCTURE 10

1. Top-Down Design 10
2. Modularity 11
C. DOCUMENTATION 14
1. Importance 14
2. Documentation Techniques 14
3. Charts 18
4. User's Manual 21
D. CONCLUSIONS 22
IV. ENHANCEMENTS TO THE USER INTERFACE 24
A. USE OF MULTIPLE WINDOWS 24
1. Three-Dimensional Terrain Window 24
a. Techniques Used to Draw the Image 26
b. Coloring Terrain by Elevation 27
2. Dial and Button Help Window 27
a. Help Function 27
b. Platform Status Function 29
3. Position Window 30
4. Two-Dimensional Terrain Window 31
a. Piloting Features 31
b. Use as a Radar Repeater 32
c. Implementation 33
5. Performance Data Window 34
B. USE OF POP-UP MENUS 34
1. Max System Menu 35

2. CCWF System Menu 35
3. Use of Sub-Menus 37
a. Platform Select 37
b. Visibility Range 38
c. Change Viewpoint 38
C. USE OF DIALS AND BUTTONS 40
1. Button Toggles 40
2. Dial Location vs. Function 40
D. CONCLUSIONS 41
V. CONCLUSIONS 43
A. SYSTEM MAINTAINABILITY 43
B. USER INTERFACE 44
C. FUTURE WORK 45
APPENDIX A - STRUCTURE CHART 47
APPENDIX B - CALLS HIERARCHY CHART 51
APPENDIX C - CCWF MODULES AND THEIR PURPOSE 55
APPENDIX D - CCWF MODULES AND WHO THEY ARE CALLED BY 58
APPENDIX E - CCWF USER'S MANUAL 61
LIST OF REFERENCES 71
INITIAL DISTRIBUTION LIST 72

LIST OF FIGURES

Figure 3.1	Top-down Module Design	12
Figure 3.2	CCWF Makefile	15
Figure 3.3	CCWF Self-documenting code	19
Figure 3.4	CCWF Use of Manifest Constants	20
Figure 4.1	CCWF Window Layout	25
Figure 4.2	Dial and Button Help Window	28
Figure 4.3	Position Window	30
Figure 4.4	Two-Dimensional Terrain Window	31
Figure 4.5	Performance Data Window	34
Figure 4.6	Max System Menu	35
Figure 4.7	Moved and Enlarged 2-D Window	36
Figure 4.8	Main Menu	37
Figure 4.9	Platform Select Sub-Menu	38
Figure 4.10	Visibility Sub-Menu	38
Figure 4.11	View Change Sub-Menu	39
Figure E.1	CCWF Window Layout	61
Figure E.2	Dial and Button Help Window	63
Figure E.3	Position Window	65
Figure E.4	Max System Menu	66
Figure E.5	Performance Data Window	67
Figure E.6	Main Menu	67
Figure E.7	Platform Select Sub-Menu	68
Figure E.8	Visibility Sub-Menu	68
Figure E.9	View Change Sub-Menu	69

I. OVERVIEW

A. THE COMMAND AND CONTROL WORKSTATION

In today's world of supersonic missiles and aircraft, the professional naval officer must be able to assess all sensor inputs in a matter of seconds (not minutes)! Expedient, correct, action is a must. To analyze multiple inputs (NTDS, radar, sonar), in an efficient manner, an integrated method of information presentation must be devised. One way to study the presentation of such information is through the paradigm of the command and control workstation. For that study, we propose a tool that collects and displays inputs from sensors in a manageable form. That tool is called the Command and Control Workstation of the Future (CCWF) [Ref. 1].

This study includes enhancements to the design and implementation of an integrated presentation method that improves the warfighting capability of today's professional naval officer. The CCWF's goal is to provide a prototype visualization tool that is to be used in the Combat Information Center (CIC) to assist in "fighting the ship." The CCWF is also planned as a training tool for a CIC watch team. Additional goals of the CCWF include a user-friendly interface and near real-time, three-dimensional display. The CCWF is implemented on inexpensive workstations available in the Graphics and Video Laboratory of the Department of Computer Science at the Naval Postgraduate School.

1. Discussion

The main advantage of using low cost (less than $100,000) workstations is the future capability of producing a large number of CCWFs for distribution throughout the Navy. At this cost, every ship will be able to integrate the CCWF into its CIC. When this is done, real sensor inputs will be used and the CCWF will begin to show its true potential.

a. Use of Color

The use of color in two and three-dimensional images significantly improves not only the appearance but the effectiveness of the display. The Watch Officer who must monitor several different inputs, including sensor data, communications, message traffic, intelligence information, and ship's location data, needs to be able to analyze a given display quickly. Using a color display assists in quick and accurate information analysis by the Watch Officer.

b. Changing Viewpoint

One of the Surface Warfare Officer's main restrictions while aboard ship is visibility. The lack of extended visibility negatively impacts the ship-bound officer whenever he is on watch. While standing watch in CIC, the officer has to rely on sensors, radar and sonar to form an artificial, interpreted view. As Officer of the Deck (OOD) on the bridge, visibility is limited to about twelve miles on a Destroyer. Visibility can be even less when foul weather is encountered. The OOD is assisted by the same radar that is present in CIC, but frequently this radar is inactive in order to decrease the probability of being located by an enemy warship. If these sensor inputs are removed, CIC is completely without a view and the OOD is limited to the current

visibility. A strategy currently in use to minimize this limited visibility problem is the use of aircraft. To simulate an aircraft, or as a substitute for an aircraft, the CCWF allows the user to change his current viewing position (viewpoint). A new viewpoint can be selected by the user at any time. Once selected, a three-dimensional image of the new viewpoint is presented, as if the user were really there. The capabilities of this feature include over-the-horizon (OTH) targeting, piloting through limited visibility, or recreating the view from another platform in the battlegroup.

c. Use of Windows

A primary goal of the CCWF is to provide as much information as possible in an integrated fashion. The best method of accomplishing this goal is through the effective use of windows. Windows are separate displays combined onto a single monitor. Each window contains a different type of information. The "big picture" is generated by combining all the data displayed in each window.

The CCWF includes windows that display the following information: three-dimensional visualization from the current platform; two-dimensional display of the entire area, similar to a color-coded (by elevation) chart; simulator performance data -- number of polygons drawn; position data -- latitude and longitude; dial and button help -- includes course, speed, altitude, and other pertinent information about the current platform from which the viewpoint is determined. Another important characteristic of a window is its ability to be moved and/or resized. In the CCWF the two-dimensional chart window can be moved around and/or resized to better fit the needs of the operator.

d. Use of Popup Menus

A secondary goal of the CCWF is to provide a user-friendly environment. The importance of this objective is usually underestimated in software system development. The lack of proper human factors engineering is usually due to cost-cutting efforts or inadequate time spent developing the system. This can turn a good system into a failure when implemented if the user rejects it due to an unsatisfactory user-interface.

A technique used to increase the ease with which a user masters a given tool is the use of popup menus. A completely menu-driven interface is easier to master mainly because there is nothing a user has to remember. By pressing a mouse button a menu appears which states available options at any given time. The user selects an option by highlighting his choices and releasing the mouse button.

B. SCOPE OF THIS STUDY

The scope of this effort is to enhance previous work on the CCWF. The focus of these enhancements is in two main areas. First, to significantly improve the maintainability of the system through application of standard software engineering techniques. These techniques include increased use of modularity, comments, and descriptive labeling of variables and modules. Second, provide for a friendlier user-interface which is not only easier to use, but provides more comprehensive information to the user. The goal of this system is to improve upon already existing data structures and algorithms in order to provide a more realistic display of sensor information and the sea/land environment.

II. FOCUS

A. PRELIMINARY RESEARCH

One computer graphics research project at the Naval Postgraduate School laid the groundwork for the current CCWF [Ref. 1]. This project concentrated on generating a three-dimensional image of a sea/land environment that could be traversed in near real-time. To draw three-dimensional land images, Harris used Digital Terrain Elevation Data (DTED) from the Defense Mapping Agency (DMA). In order to decrease the number of polygons being drawn in each frame, three different levels of resolution were used. In the foreground, polygons are drawn using data points at 100 yard intervals. In the mid-ground, data points are 1200 yards apart and in the background data points are 12,000 yards apart [Ref. 1: pp 49-51]. Using three levels of resolution significantly increases the rate at which images can be generated. This increased image generation rate provides for more realistic animation of movement.

B. SYSTEM MAINTAINABILITY

One of the largest expenses of a software system over the long term is software maintenance. This maintenance includes correcting "bugs" that occur as the system is used over time. Also included in maintenance cost are system upgrades. These corrections and/or upgrades can be even more costly if the original system implementation team is no longer available to assist.

Before corrections and/or upgrades to the system can be made, the current implementation team must spend many hours studying the existing system. After a

lengthy, and therefore costly, study phase the current team must determine if the desired system changes are feasible. Once determined feasible, the correction and/or upgrades may be implemented.

During the implementation phase, additional time must be spent rewriting the original system. This rewriting must be done in strict accordance with standard software engineering principles to ensure that when future system corrections and/or changes need to be made, the time spent studying the system is minimized.

Standard software engineering principles include emphasis on modularization, self-documenting code (numerous comments), and descriptive labeling of modules, variables, and constants. These principles are no more difficult to apply during the initial system coding than during the second or third system revision .

C. USER INTERFACE

It is important throughout the design and implementation of any system that the needs of the ultimate user are considered. If the user's needs are ignored, no matter how revolutionary a new system is, it may ultimately fail due to rejection by the user.

To properly design a new system in accordance with user's needs, complete analysis must be done. This analysis would include learning how the current system operates, defining user requirements, and evaluating alternative solutions [Ref. 2: pp 142-151]. Once proper systems analysis is completed, design and implementation may begin.

During system design and implementation, the underlying theme must be that the system is for the user. Keeping this in mind will ultimately lead to the delivery of

a system that will be well-received by the user. With the user's acceptance the long-term success of the system is much more likely to occur.

D. SUMMATION

This study takes existing groundwork on the CCWF and adds several enhancements. These enhancements include significant improvement to the maintainability of the software system and an even friendlier user-interface. Improving system maintainability is a particularly important focus for any project that has future potential for growth. Because of this importance, the use of proper software engineering principles should be emphasized.

III. IMPROVEMENT OF SYSTEM MAINTAINABILITY

A. SYSTEM ANALYSIS

1. Preliminary Study

Before any work begins on an existing software system, time must be spent becoming familiar with the current system. The amount of time this familiarization process takes is dependent upon how well prior design and implementation teams applied the elements of good programming style. Elements of good programming style include: writing clearly, making programs read from top to bottom, choosing a data representation that makes the program simple, use of modularization, making every comment count, and using variable names that mean something [Ref. 3].

After a preliminary familiarization period has taken place, the feasibility of upgrading the current system must be determined. If the system is completely lacking the elements of good programming style, the current implementation team could decide to start from scratch. On the other hand, if the current system does make use of the elements of good programming style, it will be considered feasible to upgrade the current system. If the current system is usable, a significant amount of implementation time will be saved.

2. In-depth Study

Once feasibility is determined, the current design and implementation team must perform an in-depth study of the existing system. Objectives considered during this detailed study include: determine who the system is being designed for, analyze

any limitations or constraints the project has, identify what functions the current system provides and to what extent these functions relate to the overall system goals, and learn how the current system operates. After the in-depth study has been completed, a problem statement is generated [Ref. 2: pp 182-190].

3. Problem Statement

The problem statement used for enhancing the CCWF is as follows:

(1) Significantly increase use of the elements of good programming style.
(2) Add lighting -- compute the normal vectors for all polygons.
(3) Convert preprocessed DMA data file to be on-line with the display program (no file or data duplication).
(4) Improve user-interface through use of multiple windows.
(5) Improve control interface -- currently too sensitive.
(6) Draw the ocean as polygons instead of an underlying plane.
(7) Integrate ship hulls and bottoms (underwater view) into three-dimensional image.
(8) Provide for changing of user's viewpoint.

4. Defining User Requirements

Once the current system is completely understood, taking the time to determine what the intended user wants the system to do is critical. If this system analysis step is not performed, an otherwise effective system could fail because it doesn't do what the user wants.

The objectives of the user requirements study include: identifying all users of the new system, defining the functions that must be provided by the new system, ensuring that the user's needs are in accordance with the overall goals of the system, and defining the data and other information that must be processed. After the user requirements study is completed, a requirements statement is generated [Ref. 2: pp 191-197].

5. Requirements Statement

The requirements statement used in the CCWF is constantly expanding as more system features become possible. The following is a current list of system requirements:

(1) Generate a three-dimensional image using DMA data.
(2) Traverse over three-dimensional image using near real-time animation.
(3) Allow user to define traversal view changes with course, speed, and altitude inputs.
(4) Maintain and display current position in terms of latitude and longitude
(5) Provide system performance data using the number of polygons being drawn as a unit of measure.
(6) Make use of multiple windows to present as much information as possible.
(7) Provide a friendly user-interface with which a new user can become proficient in a short period of time.
(8) Provide a User's Manual to assist the new user in learning the system.
(9) Generate a two-dimensional image of entire area of database.
(10) Display user's current position in two-dimensional image.
(11) Update user's position as it changes through application of course and speed.
(12) Allow user to change viewpoint to another position within the database.
(13) Allow user to return to original viewpoint.
(14) Allow user to choose which type of platform to operate (ship or aircraft).
(15) Allow user to select from a range of visibilities.

B. PROGRAM STRUCTURE

1. Top-Down Design

After system analysis is completed the project enhancements are completely understood. With the problem statement and requirements statement in hand, design of enhancements may begin.

Through the use of top-down design methodology, each enhancement to the CCWF is implemented. After a specific enhancement has been selected as our

objective, the first step is to write a high-level pseudo-code statement describing the objective. We then begin refining the high-level statement into stages which when accomplished will meet our objective. We continue refining and filling in specific tasks, within each stage, until a final pseudo-code version is generated.

To implement the modules required to meet our objective, first make use of the pseudo-code to review and debug the algorithm. Once the algorithm is correct, rewrite it using C. C is the programming language used by the Silicon Graphics IRIS graphics workstations.

Another important step in top-down design is choosing a data structure. Throughout the design process of enhancements to the CCWF, data structures have been chosen that will make the program as simple as possible. The main reason for this being the ease with which future students will be able to familiarize themselves with the current system. Figure 3.1 demonstrates the use of top-down design in the CCWF project.

2. Modularity

Due to the large size of many programs, it is basically impossible to understand how they function as a whole. The program must be separated into many small pieces, which can be implemented individually. These small pieces become the building blocks, or "modules" of large programs. Every module is to be designed to do one thing well. If a module is designed to do too many things, its usefulness will be limited. The over-tasked module will be more complex and harder to maintain [Ref. 3: pp 59-64]. A positive side-effect of problem simplification through modularization is increased program efficiency [Ref. 4: pg 105].

Figure 3.1 Top-down Module Design

Objective: Generate a two-dimensional image of entire area of database

High-level psuedo-code: Draw two-dimensional image

First Refinement:
Read in data
Draw image
Display image

Second Refinement:
Read data into data structure
Calculate altitude breakpoints
Generate image relative to altitude
Display image

Final Refinement:
Read data into array
Calculate altitude breakpoints
Select which window to display image
Establish world coordinates of window
Draw ocean as background
Use breakpoints to establish altitude contours
Color in points relative to altitude contour
Display image

The final refinement is implemented using four separate modules:

Read data into array	=>	*read_terrain_data*
Calculate altitude breakpoints	=>	*calc_breakpoints*
Select which window to display image	=>	*setwindow*
Establish world coordinates of window	=>	*draw_2D_terrain*
Draw ocean as background	=>	*draw_2D_terrain*
Use breakpoints to establish altitude contours	=>	*draw_2D_terrain*
Color in points relative to altitude contour	=>	*draw_2D_terrain*
Display image	=>	*draw_2D_terrain*

Each module can be written and compiled separately from the rest of the program. After the module is tested and found to perform its function in accordance with the system requirements, it is linked to the rest of the modules. This linking process integrates the newly generated module into the main program and adds the newly created feature to the system.

An underlying objective of many computer programs is to automate the repetitiveness of an action performed by a user. When modeling the user action in the form of a function, the repetitiveness will remain in the form of code. Modularization allows for the removal of the repeated code section by separating it into an individual module. If any irregularities do exist in the repeated action, they can be summarized in the module's argument list.

In order to improve system maintainability, a good rule of thumb is to minimize coupling of modules. Coupling is the existence of relationships between modules. Minimizing coupling provides for the maximum independence between modules. If coupling does exist, it should be made as explicit as possible [Ref. 3: pg 62]. Explicit coupling is demonstrated in the relationship between the modules, *setwindow()* and *billboard()*. The only relationship between the two modules is the window number *setwindow()* needs to perform its function. This relationship is explicitly stated in the argument list of the module call statement. The call statement used by *billboard()* is "setwindow(BILLBOARDWIN)," where BILLBOARDWIN is a manifest constant equal to the billboard's window number.

In addition to performing only a single function, each module is to hide its own method of operation from the rest of the program. This principle of information

hiding will improve the system maintainability by allowing the individual module to be changed independently of others.

The software tool used by the CCWF to implement modularization and linking is *Make*. *Make*, an Unix operating system software tool, takes a programmer generated *Makefile* as input. The *Makefile* is a listing of all the modules contained in the CCWF (see Figure 3.2). *Make*, checks the time stamp on all the listed modules. Any module that has been modified since *Make* was last executed is recompiled. Once all modules are compiled, *Make* automatically links the entire list of modules. Output from *Make* is a single executable program.

C. DOCUMENTATION

1. Importance

The significance of complete documentation to the future maintainability of a software system cannot be understated. Without accurate and comprehensive documentation, upgrading a system is infeasible. Such a large amount of time will be spent learning the current system, the costs of enhancements will far outweigh any benefits gained. Because of the excessive familiarization costs, the enhancement design and implementation team will be better off to start from scratch.

2. Documentation Techniques

The primary technique used by the CCWF is that of self-documenting code. Self-documentation is accomplished through the application of several different principles. First, the program is written with a clean structure. A well structured program follows from the use of a top-down design methodology. The program is read from top-to-bottom as the system actually works. When this is true the familiarization

Figure 3.2 CCWF Makefile

```
CFLAGS = -Zg -g

OBJS = 3d.o\
        adjust_bounds.o\
        billboard.o\
        calc_breakpoints.o\
        clear_overlays.o\
        colorramp.o\
        do_menu.o\
        draw_2D_terrain.o\
        draw_cell.o\
        draw_dials.o\
        draw_ocean.o\
        draw_poly.o\
        draw_skirt.o\
        draw_terrain.o\
        gamma.o\
        get_terrain.o\
        init_controls.o\
        init_graphics.o\
        init_queue.o\
        init_terrain.o\
        init_view.o\
        initialize_3d.o\
        mainmenu.o\
        make_level1.o\
        make_level2.o\
        make_level3.o\
        maxmin.o\
        name_conv.o\
        open_2D_terrain_wind.o\
        open_3D_terrain_wind.o\
        open_billboard_wind.o\
        open_dial_help_wind.o\
        open_performance_wind.o\
        open_posit_wind.o\
        open_windows.o\
        platform_crashed.o\
        process_DIAL0.o\
        process_DIAL4.o\
        process_DIAL5.o\
```

Figure 3.2 CCWF Makefile (continued)

```
                process_view_select.o\
                read_terrain_data.o\
                relocate_mouse.o\
                setwindow.o\
                timer.o\
                update_2D_position.o\
                update_dial_data.o\
                update_elevation.o\
                update_performance_data.o\
                update_posit.o\
                update_position.o

HDRS1 = get_terrain.o read_terrain_data.o

HDRS2 = adjust_bounds.o\
                draw_cell.o\
                draw_ocean.o\
                draw_poly.o\
                draw_terrain.o\
                process_view_select.o

HDRS3 = 3d.o\
                billboard.o\
                calc_breakpoints.o\
                clear_overlays.o\
                colorramp.o\
                do_menu.o\
                draw_2D_terrain.o\
                draw_dials.o\
                draw_ocean.o\
                draw_skirt.o\
                gamma.o\
                get_terrain.o\
                init_controls.o\
                init_graphics.o\
                init_view.o\
                open_2D_terrain_wind.o\
                open_3D_terrain_wind.o\
                open_billboard_wind.o\
                open_dial_help_wind.o\
                open_performance_wind.o\
                open_posit_wind.o\
```

Figure 3.2 CCWF Makefile (continued)

```
                platform_crashed.o\
                process_DIAL0.o\
                process_DIAL4.o\
                process_DIAL5.o\
                read_terrain_data.o\
                relocate_mouse.o\
                setwindow.o\
                update_2D_position.o\
                update_dial_data.o\
                update_elevation.o\
                update_performance_data.o\
                update_posit.o\
                update_position.o

HDRS4 = 3d.o\
                do_menu.o\
                draw_2D_terrain.o\
                get_terrain.o\
                init_terrain.o\
                init_view.o\
                mainmenu.o\
                make_level1.o\
                make_level2.o\
                make_level3.o\
                process_DIAL0.o\
                process_DIAL4.o\
                read_terrain_data.o\
                relocate_mouse.o\
                update_2D_position.o\
                update_dial_data.o\
                update_posit.o\
                update_position.o

3d : $(OBJS)
                cc -o 3d $(OBJS)  $(CFLAGS)

$(HDRS1): filenames.h
$(HDRS2): terrain.h
$(HDRS3): constants.h
$(HDRS4): typedef.h
```

process is much shorter than when no structure exists in the program (see Figure 3.3).

Another principle used by the CCWF to increase comprehension while reviewing the program is the use of code formatting. Proper use of "white space" and indentation makes the program's logical structure much easier to determine.

The best technique of documenting expressions, variables, modules, and constants is to use identifiers that have meaning. The C programming language promotes this technique through the use of manifest constants and unlimited length identifiers. The CCWF has two header files, *constants.h* and *terrain.h*, which list all the manifest constants used throughout the system (see Figure 3.4).

Any additional documentation needed is in the form of comments to the source code. Comments are included in the CCWF whenever it is anticipated that a future reader may have difficulty understanding what the code is doing at that point. Comments are not in the form of a line-by-line explanation of what is happening.

3. Charts

Another method of describing a program is through the use of charts. Charts used to provide additional documentation to the CCWF are a Structure Chart, Calls Hierarchy Chart, Module Purpose List, and a Module Call List.

The CCWF Structure Chart, Appendix A, is a treelike representation. Each node on the tree represents a module within the CCWF. Modules are broken down into sub-modules in top-down design fashion. Once all the modules of the CCWF are represented, execution will occur in a top-to-bottom, left-to-right sequence. Looping is also demonstrated in the structure chart via a ring-like arrow.

Figure 3.3 CCWF Self-documenting Code

```
/*************************************************************************
*               Author:         Rex Cobb
*               Written:        09 AUG 88
*               File Name:      process_DIAL0.c
*               Purpose:        process DIAL0 queue entry -- course change
*               Parameters:     structure view (type viewpoint), input course change,
                                        and lock on status
*               Returns:        updated structure view (myview in 3d.c)
*************************************************************************/

/*#define DPRINT 1 */  /* debug print statements */

process_DIAL0 (view,icrs,lock_on)

 viewpoint *view;       /* data structure 3D image is calculated relative to */
 short    icrs;         /* intended course */
 Boolean  lock_on;      /* true when look direction is locked onto course */
{
  short course_change=0;        /* change in course between dial 0 input */

#ifdef DPRINT
printf("process_DIAL0 %d\n");
printf("icrs = %d\n",icrs);
#endif DPRINT

 if (icrs < 0)
   {
     icrs = icrs + DEGREES_IN_COMPASS;
   }

 else if (icrs > DEGREES_IN_COMPASS)
   {
     icrs = icrs - DEGREES_IN_COMPASS;
   }
 course_change = icrs - (short)(view->crs);

 view->crs = (float)(icrs);

 if (lock_on)
    view->lookdir = view->lookdir + (float)(course_change);
}
```

Figure 3.4 CCWF Use of Manifest Constants
(excerpt from *constants.h*)

```
/* window id numbers */

#define BILLBOARDWIN 0
#define POSITWIN 1
#define DIALHELPWIN 2
#define TERRAINWIN 3
#define TWODTERRAINWIN 4
#define PERFORMWIN 5

 /* constants for 3d display */

#define LATITUDE 31                    /* subject cell */
#define LONGITUDE 131                  /* subject cell */
#define MINELEVATION 0                 /* lowest elevation in 2D map */
#define MAXELEVATION 1120              /* highest elevation in 2D map(meters)*/
#define DEGREES_IN_COMPASS 360
#define INITIAL_HEADING 4              /* degrees */
#define INITIAL_SPEED 0                /* knots */
#define INITIAL_ALTITUDE 35            /* yards */
#define INITIAL_LOOK_ANGLE 0           /* degrees */
#define INITIAL_LOOK_DISTANCE 0        /* yards */
#define MAX_NUMB_COURSE_TURNS 2
#define MAX_AIRCRAFT_SPEED 1000        /* knots */
#define MAX_SHIP_SPEED 35              /* knots */
#define MAX_ALTITUDE 5000              /* yards */
#define MAX_LOOK_ANGLE 150             /* degrees */
#define MIN_ALTITUDE -100
#define MIN_LOOK_ANGLE 0               /* degrees */
#define MAX_LOOK_DISTANCE 240          /* number of miles in cell*4 */
#define DIALTOYARDS 10                 /* conversion for dial input to yards */
#define WINDOWYARDS 100                /* multiplier used to properly update dial
                                          showing distance of look direction */

/* dial sensitivity constants, used to improve smoothness of dial input */

#define ALTSENS 6                      /* altitude DIAL2 */
#define COURSESENS 10                  /* course DIAL0 */
#define AIRCRAFT_SPEEDSENS 10          /* speed DIAL1 for aircraft */
#define SHIP_SPEEDSENS 20              /*   for ship */
#define LOOKANGSENS 20                 /* look angle DIAL5 */
```

The purpose of the Calls Hierarchy Chart, Appendix B, is to display calls to and by each function within the CCWF.

The List of CCWF Modules and their Purpose, Appendix C, describes every module in the CCWF and what function it performs.

The List of CCWF Modules and Who they are called by, Appendix D, describes every module in the CCWF and what module(s) it is called from.

All charts for the CCWF have been designed to improve the maintainability of the software system. Although charts do improve system maintainability, they are of only secondary importance since a programmer can only know for sure what the program does by reading the code itself. Program documentation in this format will significantly decrease the time required for future students to familiarize themselves with the CCWF system. The ease with which the software system can be learned will further promote the number of enhancements that can be implemented in a given amount of time.

4. User's Manual

During the delivery of a software system, an important piece of documentation for the ultimate user is the User's Manual, Appendix E. The CCWF User's Manual is written in the form of a stand-alone document which describes all the system features and how to make use of them. The user is assumed to have no prior knowledge of how to operate either the CCWF or a mouse-driven interface.

D. CONCLUSIONS

To ensure the maintainability of the CCWF, several important principles of software engineering have been followed.

These principles include spending sufficient time studying the project to determine its feasibility. Once the project is determined feasible, additional time must be spent performing an in-depth study. The in-depth study will result in a complete understanding of the problems needing to be solved. Prior to beginning system design, the user's requirements must be defined. If the ultimate user's needs aren't defined, an otherwise excellent system could be doomed to failure due to user rejection.

Once a complete system analysis is completed, the design process may begin. During design, emphasis is placed on a top-down structure with an effective use of modularity. Because of this design emphasis, maintainability is significantly increased due to the ease with which future enhancement teams will be able to familiarize themselves with the CCWF.

During analysis, design, and implementation of the enhancements to the CCWF, documentation plays an important part. Documentation takes on several forms in the CCWF. Primarily the code is intended to be self-documenting through the use of clean, well-structured modules. By using regular formatting and identifiers (variables, constants, module names) that have meaning, familiarization time is greatly reduced. Other documentation used to assist future enhancement teams includes comments and charts. Structure charts and hierarchical charts assist familiarization by showing the high-level view of what the different modules do and how they relate to each other.

Another form of documentation, designed for the ultimate user of the CCWF, is the User's Manual. This document is important since the long term success of the CCWF is completely dependent upon acceptance by the ultimate user. The User's Manual explains all the features of the CCWF and how to use them. Without this form of reference material, the probability of user rejection is increased.

Through proper analysis, design, and documentation of enhancements to the CCWF, system maintainability is significantly improved. Future enhancements will be implemented with much less difficulty and in a more time-efficient manner.

IV. ENHANCEMENTS TO THE USER INTERFACE

The primary objective of the command and control workstation of the future (CCWF) is to present multiple "chunks" of information to a user. This information must be presented in a fashion that can be interpreted and reacted to in a minimal amount of time. An effective method of displaying the information is through the use of multiple windows, one chunk of information for each window.

A. USE OF MULTIPLE WINDOWS

Whenever more than one chunk of information is being presented to a user, the chunks are to be divided up into separate entities. A rule of thumb followed during the design of the CCWF is to avoid presenting more than five to nine chunks to a user at any one time [Ref. 4: pg 62]. The CCWF uses five separate windows to provide the user with the ability to analyze the available sensor information as needed. Figure 4.1 shows the relational layout of the CCWF display. Each window displays a different type of information to the user.

1. Three-Dimensional Terrain Window

The Three-Dimensional Terrain Window displays a visualization of surrounding terrain calculated relative to the user's position. To generate this view, the CCWF accesses a preprocessed database of Digital Terrain Elevation Data (DTED) provided by the Defense Mapping Agency (DMA) [Ref. 1: pp 35-42].

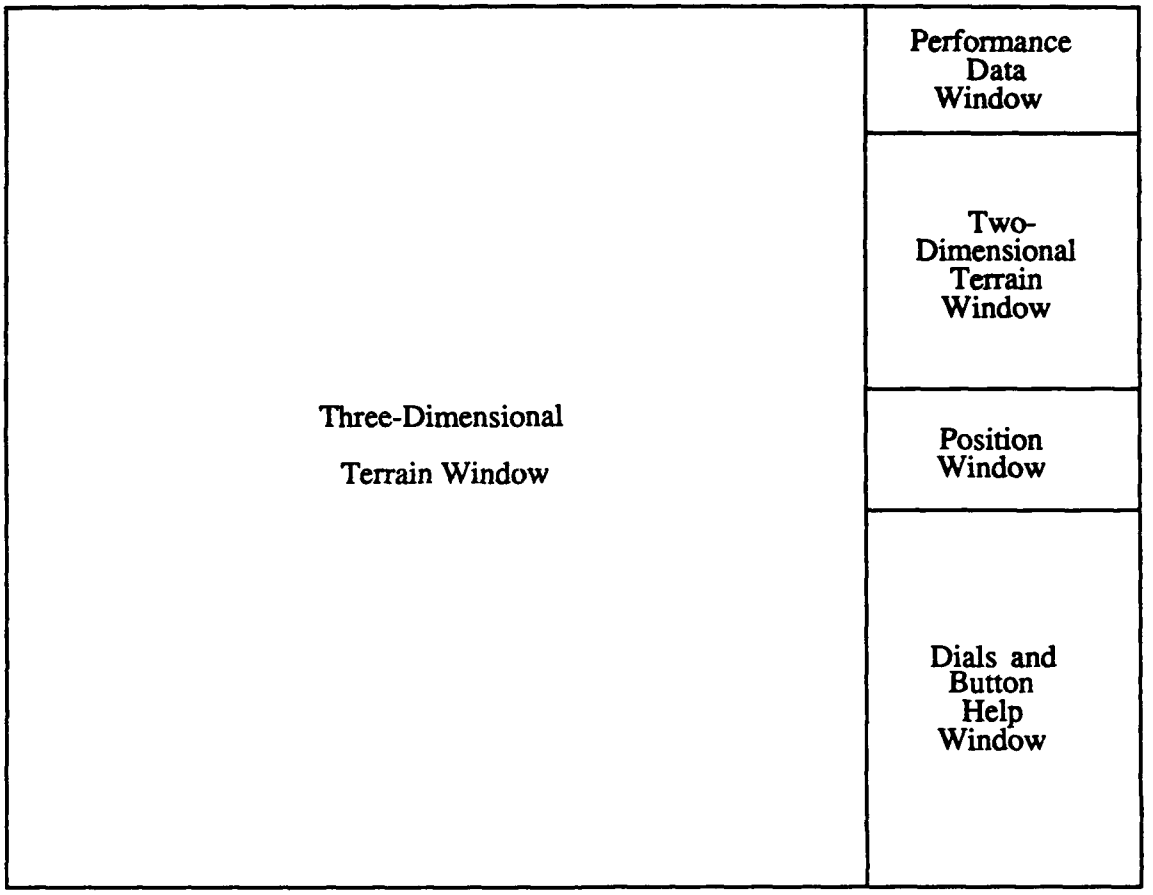


Figure 4.1 CCWF Window Layout

The DTED, used in the CCWF, describes a 60 mile square (cell), whose bottom left corner is located at 31° North Latitude, 131° East Longitude. The subject cell includes parts of one of Japan's southern islands and the Sea of Japan. This specific cell was selected due to its interesting combination of land, islands, and water.

The DTED consists of a series of data points occurring at approximately 100 yard intervals throughout the cell. This equates to a data array of 1,200 by 1,200, or 1.44 million data points for each cell. Since each data point is represented in the form of a 16 bit integer, the memory storage required for each cell is 2.88 megabytes.

a. Techniques Used to Draw the Image

Due to the large number of data points used to describe a single cell, the CCWF uses two independent techniques to speed up the generation of the three-dimensional image. First, the initial 1,200 by 1,200 array of data points is preprocessed. The preprocessing is performed prior to running the CCWF and results in a new, compressed, data structure of less than one megabyte in size [Ref. 1: pp 53-61]. This compressed data structure is more efficient and therefore allows the CCWF to generate three-dimensional images more quickly than when using the original array.

A second technique used is drawing the terrain in three different levels of resolution. Displaying the terrain in this fashion models the way the viewer actually sees. Objects that are close to the viewer are seen with a high level of detail. As objects get further away, the level of detail decreases accordingly. Objects that are far off in the distance are seen with little detail, if they are seen at all.

Each level of resolution is determined relative to the distance of the terrain from the viewer. In the foreground, nearest the viewer, polygons are drawn with vertices at 100 yard intervals. This is the highest resolution possible since every available data point is being used. The terrain located in the image's middle-ground uses data points at 1,200 yard intervals to form polygons. The background is drawn with vertices at 12,000 yard intervals. Generating the three-dimensional terrain using three levels of resolution reduces the number of polygons being drawn from greater than 100,000 to approximately 10,000. This factor of ten decrease in the number of polygons being drawn significantly increases the speed with which the three-dimensional image is generated [Ref. 1: pp 49-52].

b. Coloring Terrain by Elevation

To provide a more realistic image to the viewer, the terrain is color-coded relative to elevation. To do this, during initialization of the CCWF, *calc_breakpoints()*, generates an array of 16 breakpoints evenly distributed between the cell's minimum and maximum elevation. Prior to the drawing of a terrain polygon, *make_polly()*, compares the y coordinate (elevation) of one vertex of the polygon to the array of breakpoints. From this comparison the current color is established and used in the following polygon fill operation. The resulting image uses green shades in lower elevations, yellow shades at medium elevations, and shades of red for higher elevations.

2. Dial and Button Help Window

The Dial and Button Help Window, (see Figure 4.2), performs a dual function. First, this window provides the user with a description of the dial and button box functions. Second, current platform status information is displayed.

a. Help Function

The Dial and Button Help Window is a form of "on-line help" provided to the user. On-line help will increase the ease with which the user is able to familiarize himself with the CCWF system. Due to this ease of familiarization, the probability of long-term user acceptance is increased.

Help is provided in the form of functional descriptions below a drawing of each dial. The descriptions identify what effect manipulation of each dial will have on the user's platform. Also displayed is the unit of measure for each dial's function.

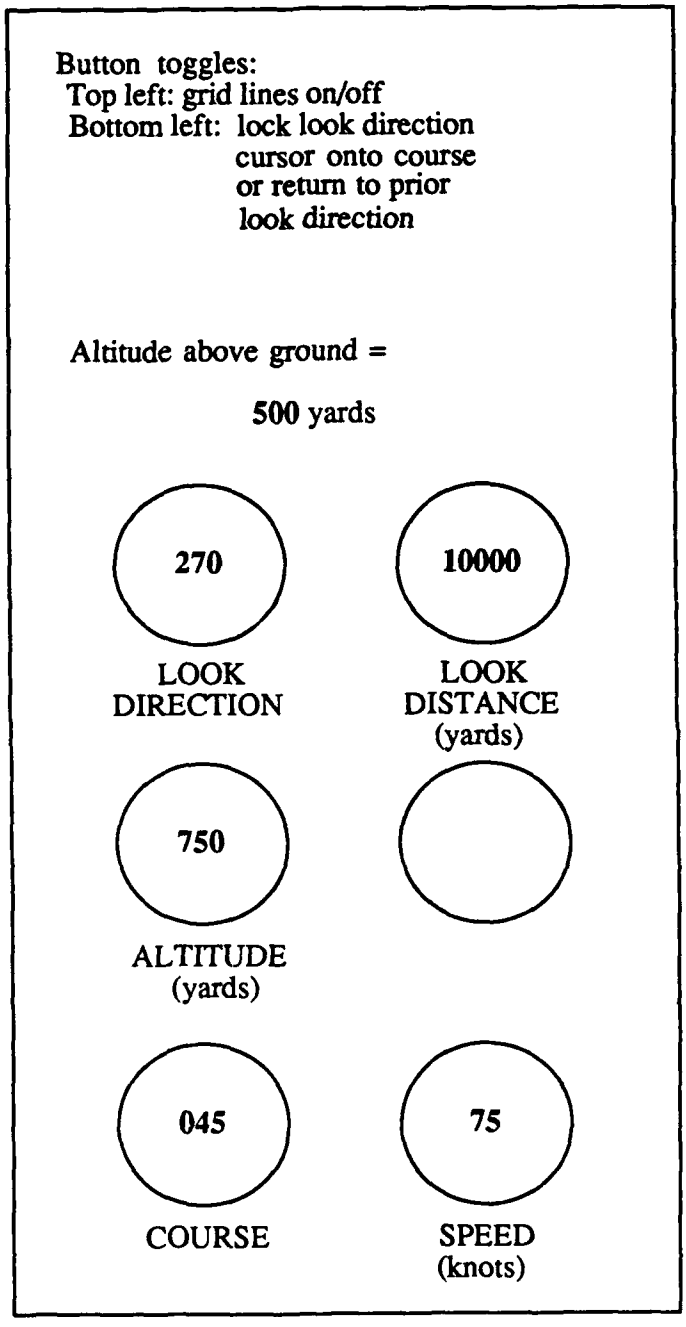


Figure 4.2 Dial and Button Help Window

Help is also provided for the use of the Button box. Each active button's location is described along with the effect each has on the displays.

b. Platform Status Function

The second role of the Dial and Button Help Window is to provide continuously updated status information. The status information, relative to the viewer's current platform, includes: course, speed, altitude, look direction, look distance, and altitude above ground.

The platform's course is displayed in degrees from true north. Speed is in knots (nautical miles per hour) and is limited to less than or equal to 35 for a ship. The maximum speed allowed for an aircraft is 1,000 knots. Altitude, actually the height of the viewer's eye, is displayed in yards above sea level. The altitude for a ship is set to a constant ten, since the bridge of many ships is approximately 30 feet above the water. The maximum altitude for an aircraft is 5,000 yards. Look direction is displayed in degrees from true North. Look direction is independent of course. This is demonstrated by a viewer who is on a platform traveling due North but looking at an object off the starboard beam. The viewer's course is 000 $^{\circ}$ and his look direction is 090°. The three-dimensional terrain image is generated from the look direction. Look distance is displayed in yards from the viewer's current platform. Turning the look distance dial changes the length of the black line in the Two-Dimensional Terrain Window. By manipulating both the look direction and look distance dials, the user can determine the range and bearing to an object. To do this, place the end of the black line on the object of interest and read the look direction and look distance displays for range and bearing data. Altitude above ground, in yards, is also displayed in the Dial and Button Help Window. This value is calculated by

subtracting the elevation at the point directly below the user's platform from the altitude of the platform. If this value is less than or equal to zero, the platform will crash. To recover from a crash, the user must increase platform altitude until the elevation above ground is greater than zero.

The dual role design of the Dial and Button Help Window provides an efficient, understandable method to display both help and status information. In addition, due to the co-location of data with a graphical display, immediate feedback is provided to the user for every dial manipulation.

3. Position Window

The Position Window, (see Figure 4.3), displays the current position data for the user's platform. Latitude and longitude are provided in units of degrees, minutes, and seconds. Whenever the user's platform speed is greater than zero, the latitude and longitude are updated by *update_posit()*. The user can plot these position readings on a navigational chart. After a given amount of time, the actual track of the user's platform will be formed. Another, more efficient, method of locating the user's platform in relation to the surrounding area, is by observing the CCWF's Two-Dimensional Terrain Window.

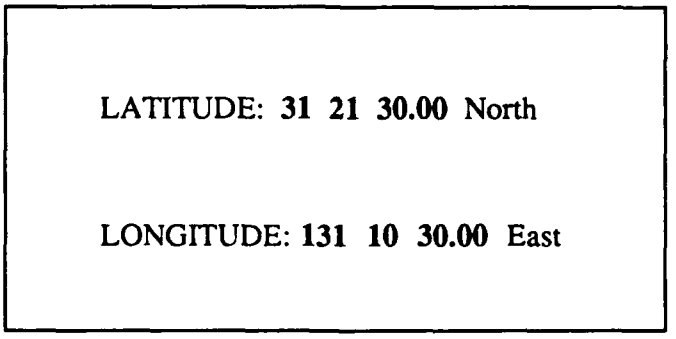


Figure 4.3 Position Window

4. Two-Dimensional Terrain Window

a. Piloting Features

The Two-Dimensional Terrain Window displays the entire cell (60 mile square) to the user (see Figure 4.4). Within the window, the user's current position is indicated by a small red circle. As the user's platform moves, the red circle moves accordingly. This CCWF feature removes the need for constantly plotting latitude and longitude on a navigational chart (piloting). With this time-intensive requirement removed, more time can be spent analyzing other sensor data.

Figure 4.4 Two-Dimensional Terrain Window

Another piloting feature provided in the Two-Dimensional Terrain Window is a line of dead reckoning (DR). The magnitude of the red line (DR) in the Two-Dimensional Terrain Window is controlled by the platform's speed. The length of the DR line represents how far the platform will travel in six minutes. The direction of the DR line is equal to the platform's course.

b. Use as a Radar Repeater

By manipulating the look direction and look distance dials, the black line in the Two-Dimensional Terrain Window models the behavior of a bearing and range cursor on a radar repeater. This feature can be used to determine the bearing and range to another platform. It can also be used to determine bearings and ranges to a series of terrain points in order to generate a navigational fix.

The two-dimensional terrain is color-coded relative to elevation. Because of this feature, a user is better capable of selecting prominent geographical features to determine bearings and ranges. This capability is particularly important when performing coastal navigation in a region with a low, featureless coastline. This type of coastline would not provide any points to navigate by using radar alone. The radar operator would have to use geographical features that are further inland. Positive identification, on a navigational chart, of these inland points would be difficult. Therefore, the bearings and ranges taken would be of little navigational value.

Using the CCWF Two-Dimensional Terrain Window, the operator is able to positively identify inland points on a chart. This identification is made possible since the terrain is color-coded relative to elevation. By determining the point's

elevation from its color, the user can compare the elevation to the chart and make a positive identification of the subject point. Once identified, the bearing and range to the point is determined and an approximate position is determined.

c. Implementation

The two-dimensional terrain is drawn as the CCWF is being initialized. Prior to drawing the terrain image, *read_terrain_data()* reads the DTED into a 1,200 by 1,200 array. To draw the terrain, *draw_2D_terrain()*, processes the data array by drawing a small rectangle color-coded relative to elevation. Using every data point in the array (1.44 million points), generating an image takes approximately 11 seconds. By processing every other point (720,000 points), only six seconds is needed to draw the image. This almost 50 percent reduction in processing results in a more time-efficient algorithm with minimal reduction in resolution.

To improve the user interface, the two-dimensional terrain image is drawn in both buffers simultaneously. By doing this, the user actually sees the image form. Providing something to look at increases the friendliness of the CCWF. This is true because the user will be less likely to get impatient while waiting for a response.

The course DR (red) and look direction (black) lines are drawn in the Two-Dimensional Terrain Window using overlay planes. The use of two overlay planes allows the CCWF to draw images of up to three colors on top of an already existing image [Ref. 5: pg 11-1]. The use of overlay planes by *update_2D_position()*, allows the CCWF to make changes within the Two-Dimensional Terrain Window without continuously redrawing the image.

5. Performance Data Window

The Performance Data Window displays the number of polygons being drawn in the three-dimensional terrain image. This value is calculated in *draw_cell()* by summing the number of calls to *make_polly()* and *draw_skirt()*. Each call represents the drawing of one polygon. The final sum is updated for each frame and displayed as seen in Figure 4.5.

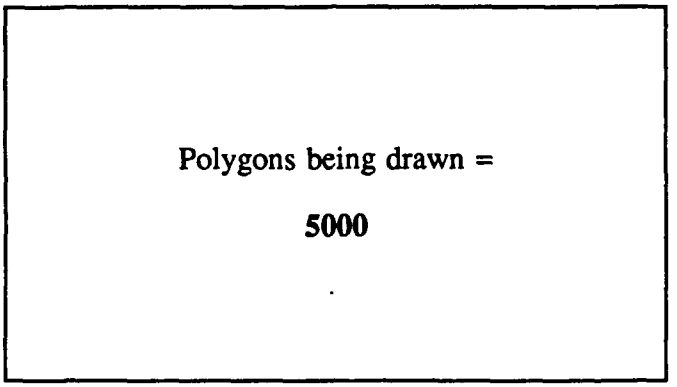


Figure 4.5 Performance Data Window

B. USE OF POP-UP MENUS

When designing a system to have a friendly user interface, minimizing the number of actions the user needs to remember is important. The CCWF accomplishes this objective through the use of a series of pop-up menus. On the IRIS 4D-70GT graphics workstation, there are two types of pop-up menus. The IRIS window manager, Max, provides a system pop-up menu. The IRIS GL graphics library also provides the system developers the capability of defining system specific menus. Both types of IRIS menus are used by the CCWF.

1. Max System Menu

The pop-up menu provided by the IRIS window manager, Max (see Figure 4.6) is accessed by locating the cursor on any window title bar and pressing the right mouse button (menu button).

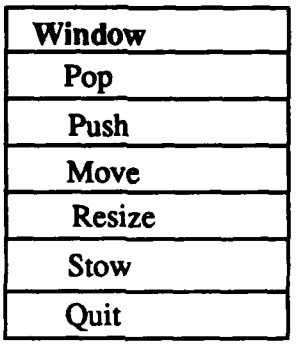


Figure 4.6 Max System Menu

The primary use of the Max system menu in the CCWF is to provide the user with the capability of moving and/or resizing the Two-Dimensional Terrain Window. By enlarging the window, the user can perform more detailed operations within the Two-Dimensional Terrain Window. By moving the window to a more centralized screen location and enlarging it, the user is better able to concentrate on the two-dimensional view (see Figure 4.7). This capability is most important when land is beyond the viewer's range of visibility.

A secondary use for the Max system menu is allowing the user to exit the CCWF system. This capability is also provided by the CCWF system menu.

2. CCWF System Menu

The second type of pop-up menus used by the CCWF are the menus designed by the CCWF system developers. The main menu (see Figure 4.8) is

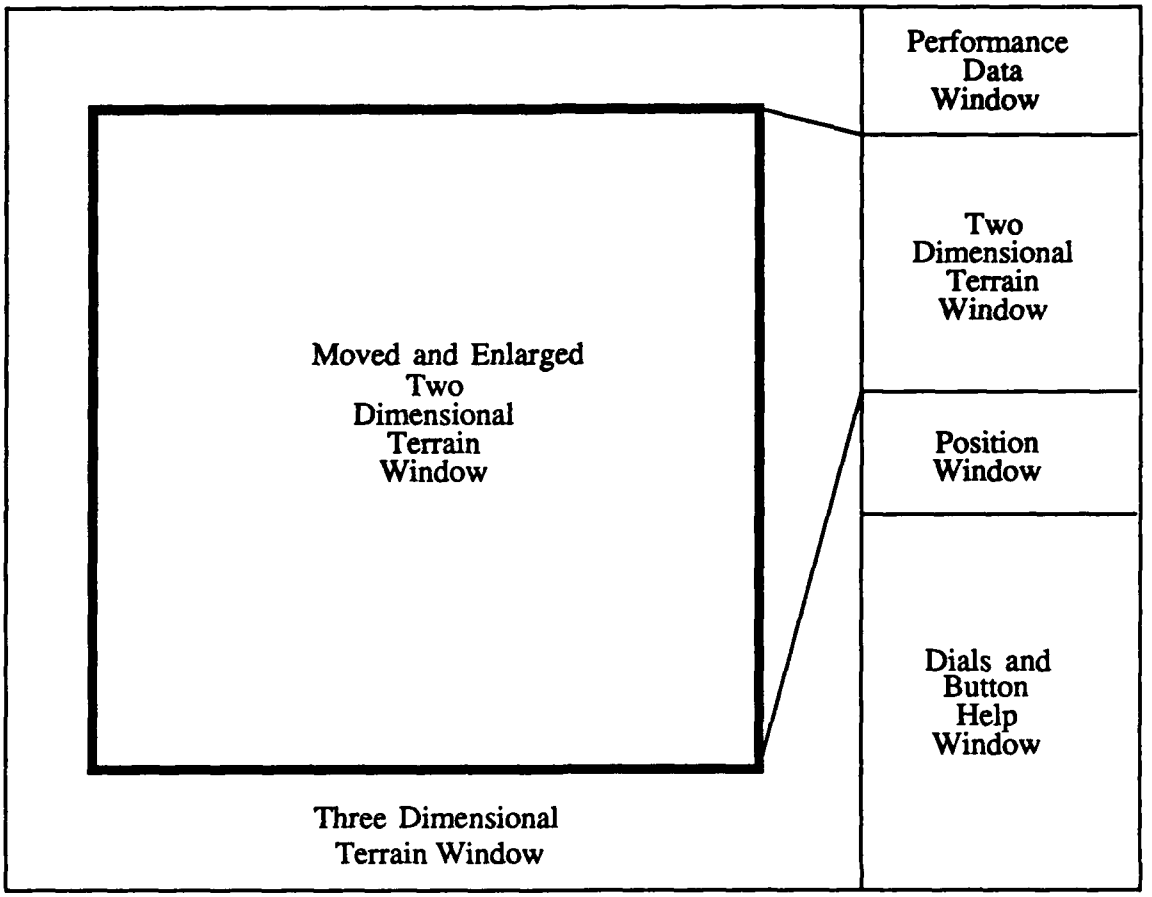


Figure 4.7 Moved and Enlarged 2-D Window

accessed by locating the cursor in the interior of any window and pressing the menu button. The menus are defined in *DefineMenu()*. In order to support the objective of providing a friendly user interface, the CCWF system menus limit the number of alternatives provided to the user. Sanders and McCormick [Ref. 6: pg 62] state that, "If more than seven alternatives are displayed on the screen, a person may tend to forget the first alternative before the last ones are read." To stay within the guideline of no more than seven alternatives, the CCWF uses a series of sub-menus.

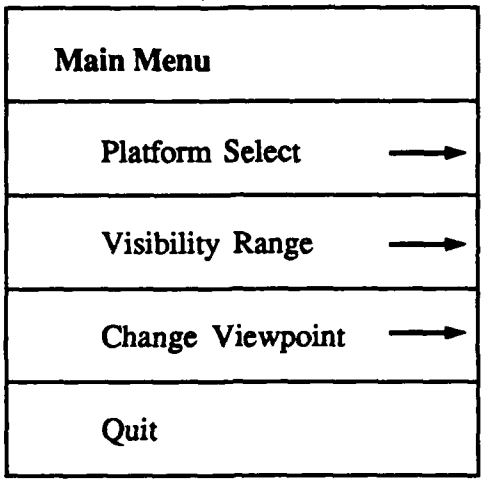


Figure 4.8 Main Menu

3. Use of Sub-Menus

These sub-menus are accessed by rolling the mouse cursor off the side of any main menu selection containing an arrow. Using this method establishes a hierarchical menu structure which makes sense to the user. The user decides what type of function in the main menu is desired and by rolling the mouse to the side of that selection is provided with specific actions that can be executed. In the CCWF, there are three sub-menus: platform select, visibility range, and change viewpoint.

a. Platform Select

Selection of this sub-menu allows the user to choose which type of platform is desired (see Figure 4.9). By selecting "Ship," the three-dimensional image generated by the CCWF is at a constant altitude of 30 feet. Maximum speed is limited to 35 knots and the user does not have the ability to change altitude. By selecting "Aircraft," which is the default mode, the user has the ability to increase speed to 1,000 knots and altitude to 5,000 yards.

Choose platform
Ship
Aircraft

Figure 4.9
Platform Select
Sub-Menu

b. Visibility Range

Selection of this sub-menu allows the user to set different levels of visibility (see Figure 4.10). The available choices are: Unlimited (default mode), 15 miles, ten miles, five miles, and three miles. By selecting the lesser values, inclement weather can be simulated by decreasing the amount of terrain the viewer is capable of seeing. A positive side effect of decreasing the visibility, is speeding up the rate at which the CCWF is capable of generating the three-dimensional image.

Select Visibility
Unlimited
15 miles
10 miles
5 miles
3 miles

Figure 4.10 Visibility Sub-Menu

c. Change Viewpoint

The capability to move from one viewpoint to another is provided by this sub-menu. When "Select new viewpoint with left mouse" is selected, the cursor

immediately goes to the Two-Dimensional Terrain Window. At this point, the user chooses a new viewpoint within the window by placing the cursor on the desired location. Pressing and releasing (clicking) the left mouse button at this time causes the CCWF to redraw the three-dimensional image as if the viewer were now located at the new viewpoint.

In 2D window:
Select new viewpoint with left mouse
Return to original view with middle mouse

Figure 4.11 View Change Sub-Menu

In the Two-Dimensional Terrain Window, the original viewpoint is marked by a small yellow circle which continues to move in accordance with the original course and speed. The new viewpoint is indicated by a small red circle. The course DR (red) and look direction (black) lines are also provided at the new viewpoint. The user may manipulate the dials (course, speed, altitude, look direction, and look distance) to change the images as desired. The user may change viewpoints as many times as desired using the Main and Change Viewpoint menus. When ready to return to the original viewpoint, marked by the yellow circle, the user clicks the middle mouse button.

This ability to change viewpoints can be used in several different ways to further improve the effectiveness of the CCWF. These uses include: performance of over-the-horizon targeting, looking around bad weather by changing viewpoint, and viewing from the perspective of another platform.

C. USE OF DIALS AND BUTTONS

The IRIS button and dial boxes function as the primary user interface. The active buttons function as toggle switches. The dials allow the user to drive the viewer's platform around the cell using dial manipulations that make sense.

1. Button Toggles

Two buttons have been activated for use as toggle switches in the CCWF. The top left button turns the ocean's grid lines on and off. These grid lines are used to provide a sense of motion to the viewer who is out of sight of land [Ref. 1: pg 48].

The other toggle switch, bottom left button, affects the Two and Three-Dimensional Terrain Windows. By pressing this button, the look direction cursor (black line) in the Two-Dimensional Terrain Window is locked onto the course DR (red line). The three-dimensional image reflects this change immediately. The purpose of this feature is to allow the user, who is looking in a direction away from the platform's course, to quickly view the image directly ahead of the platform. The look direction will remain locked onto the course direction until the toggle switch is pressed again. When pressed a second time, the viewer's image will return to the original look direction.

2. Dial Location vs. Function

The assignment of a dial's function is made considering the frequency with which the user will generate that type of input (see Figure 4.2). Most frequent user input, while driving the platform, will be course and speed changes. Because of this

frequency, course and speed dials are located at the bottom of the dial box. This location will provide the greatest ease of operation for the user. When the viewer's platform is an aircraft, altitude is frequently changed and therefore, the altitude dial is just above the course dial.

The top two dials, in Figure 4.2, are look direction and look distance. These dials are used either to view a specific object of interest or, when used together, determine the bearing and range to an object of interest. Because of their complementary function, an effective spatial relationship is needed. To provide this relationship, which results in the best possible user interface for this situation, the dials are located next to each other. This is a common sense arrangement which eases the user's familiarization process.

D. CONCLUSIONS

The most important part of any system could very well be the user interface. The CCWF is designed to provide an interface which is as friendly to the user as possible. This friendly user interface is accomplished in several different ways.

First, through the use of multiple windows, five different kinds of information are provided to the user simultaneously. The wealth of information available from the CCWF assists the user in more effectively performing his job. In addition to reporting a large quantity of information, the CCWF presents this information in a manner which is easily interpreted. This is demonstrated by the CCWF color-coding the terrain by elevation in both the Two and Three-Dimensional Terrain Windows.

Second, the CCWF uses a form of on-line help to effectively reduce the amount of information the user must remember. On-line help is provided in both the Dial and

Button Help Window and through the use of pop-up menus. The information displayed in the Dial and Button Help Window acts as a reminder to the user of the functionality of all active dials and buttons. Pop-up menus, when displayed using the menu button, remind the user of features available and provide an easy method of selecting a desired feature.

Finally, by locating controls used for input (dials and buttons) in a logical, easy to remember position, user familiarization time will be minimized. Because of this minimal user familiarization time, the probability of user rejection will be reduced, and long term acceptance of the CCWF will be probable.

V. CONCLUSIONS

A visualization tool to be used by today's naval officer, to assist in assessing multiple sensor inputs, is provided by the Command and Control Workstation of the Future (CCWF). The CCWF prototype, currently being developed using low cost graphics workstations, demonstrates the need for a tool which presents numerous chunks of information in a manageable form. This study focuses on enhancing the CCWF through improved system maintainability and a friendlier user interface.

A. SYSTEM MAINTAINABILITY

Future enhancement teams must spend adequate time understanding the current CCWF system. This preliminary study is to be completed prior to beginning the design of any new features. Familiarization time required by future enhancement teams will be greatly reduced due to the efforts of this study. Once familiarization is achieved, feasibility of enhancing the CCWF is determined. Once determined feasible, the problem statement and user requirements are updated.

While designing solutions to the problems and requirements, a structured approach is best for future system maintainability. Through the use of a top-down design methodology, CCWF enhancements are implemented.

In addition to top-down design, modularity plays a significant role in system maintainability. Modularization simplifies the CCWF by breaking up the program into small pieces. Each piece, module, is tasked with doing only one thing.

To assist the enhancement team during system familiarization and enhancement design, comprehensive documentation is provided. Documentation of the CCWF is accomplished primarily through the use of self-documenting code. Additional documentation is provided through the use of charts and a User's Manual. Documentation is to be constantly updated as new enhancements are implemented.

Due to the dynamic nature of the CCWF, proper and complete analysis and design must be accomplished. If future enhancement teams do not invest adequate time in design and analysis the feasibility of further enhancements will be reduced.

B. USER INTERFACE

During the analysis phase of system development, the user's requirements are updated. This updated requirements list provides the enhancement team with a series of features to be implemented to improve the user interface.

The primary enhancement provided by this study is to improve the presentation method of multiple chunks of information to the user. The CCWF's use of multiple windows is an ideal way to integrate numerous displays of data onto one monitor.

Coloring terrain by elevation in both the two and three-dimensional displays increases the realism of the image. In addition to an increase in realism, data displayed is easier to interpret due to the coding of elevation data by color.

Another method used by the CCWF to improve the user interface is on-line help. This on-line help is available in two fashions. First, the Dial and Button Help Window provides both platform status information and functional explanations for the user input controls (dials and buttons). The second method the CCWF uses to provide on-line help is through the use of pop-up menus.

Pop-up menus increase the ease with which the user becomes proficient with the CCWF. The user has less to learn because possible actions are listed in menus. To execute an action, the user selects the desired action through the menu. Thus, the effective use of menus decreases the number of procedures the user is required to learn.

Development of a user interface that is as friendly as possible is a top priority for the CCWF. The importance of this goal, a friendly user interface, is not to be underestimated at any time during system development. Not emphasizing the importance of the user interface can result in long-term project failure due to user rejection.

C. FUTURE WORK

Follow-on work in enhancing the CCWF will include integration of a lighting model. More realistic three-dimensional images can be generated using Gouraud shading. On the Silicon Graphics, Inc. IRIS 4D/70GT, Gouraud shading can be implemented "free" to the CCWF.

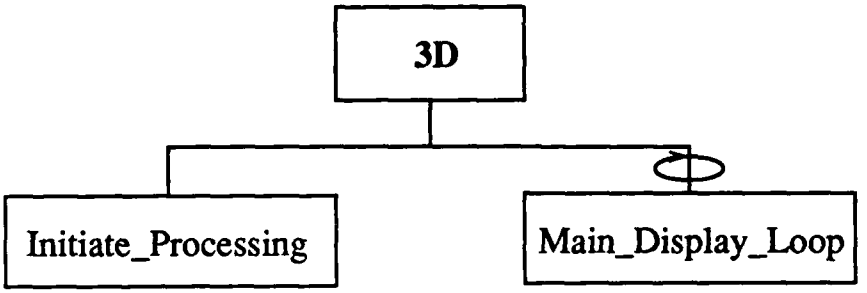
The CCWF should be able to generate the three-dimensional image without the use of a preprocessed data file. Making use of the 1200 by 1200 DMA data array, that already exists in the CCWF, will remove the need for data duplication.

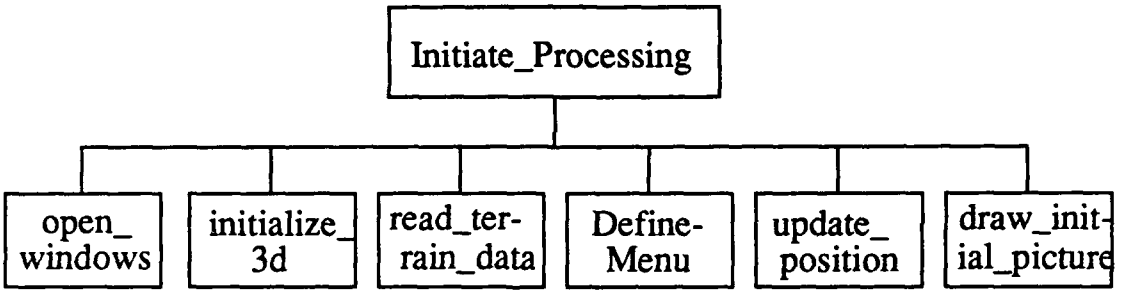
An underwater image needs to be included within the Three-Dimensional Terrain Window. The capability of selecting a submarine as a platform could actuate an entirely separate set of drawing routines. The visualization provided would include platforms that already exist from the surface image. Integration of the surface and sub-surface views will add an entirely new dimension to the CCWF.

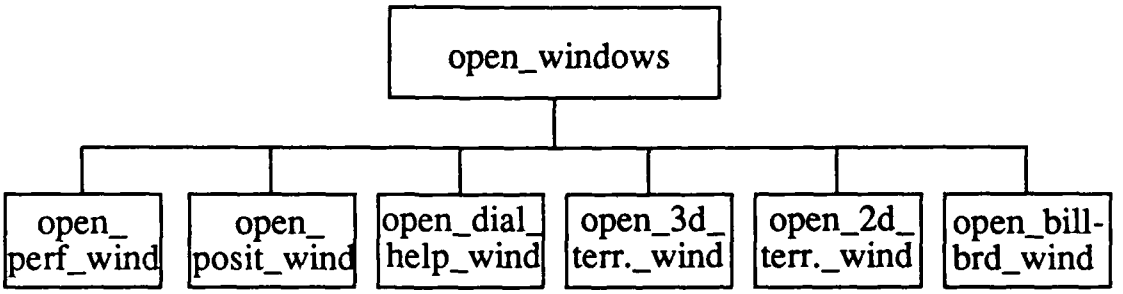
Other enhancements include: an NTDS type display, ability to change viewpoint to a platform selected from the NTDS display, integration of cultural features available from DMA data files, and the ability to display tactical information about selected contacts [Ref. 1: pp 74-75].

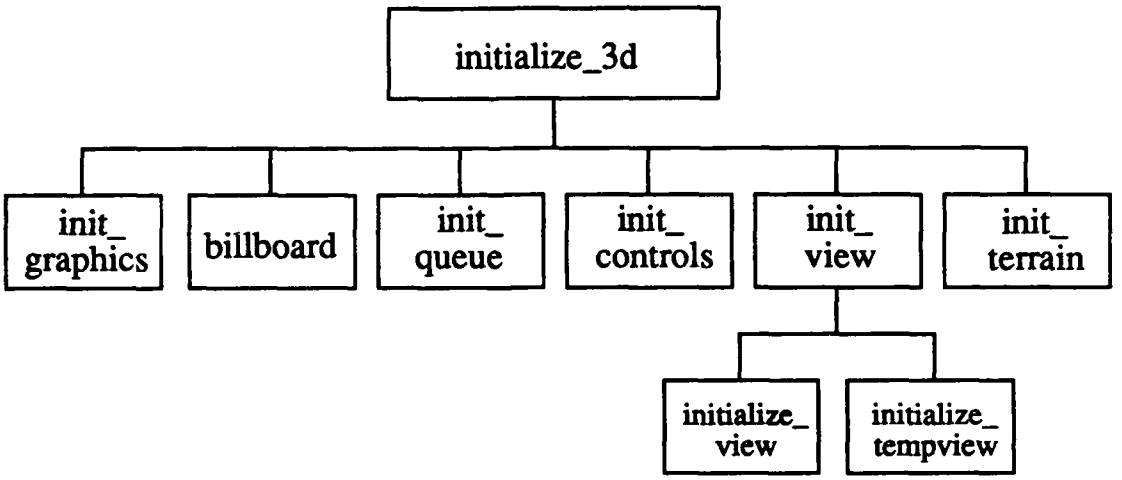
The list of potential features needed by the CCWF is constantly growing. To ensure that future enhancements are feasible, design and implementation teams must always strive for a high level of system maintainability. In addition, the future implementation of all enhancements needs to provide a friendly user interface to ensure user acceptance and long-term CCWF success.

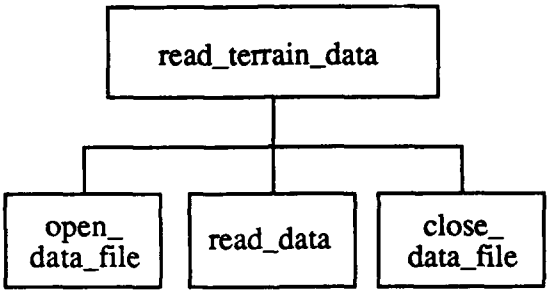
APPENDIX A - STRUCTURE CHART

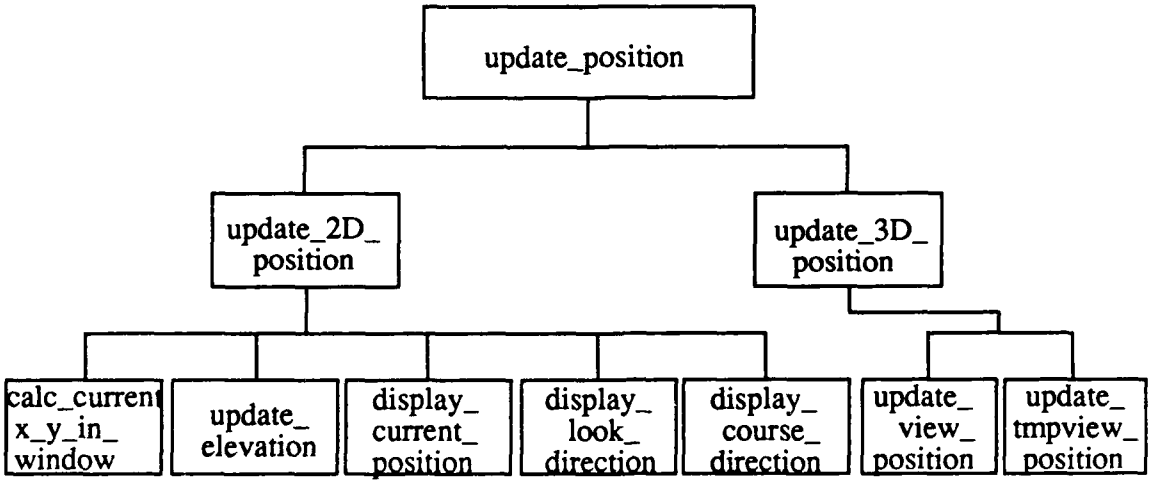


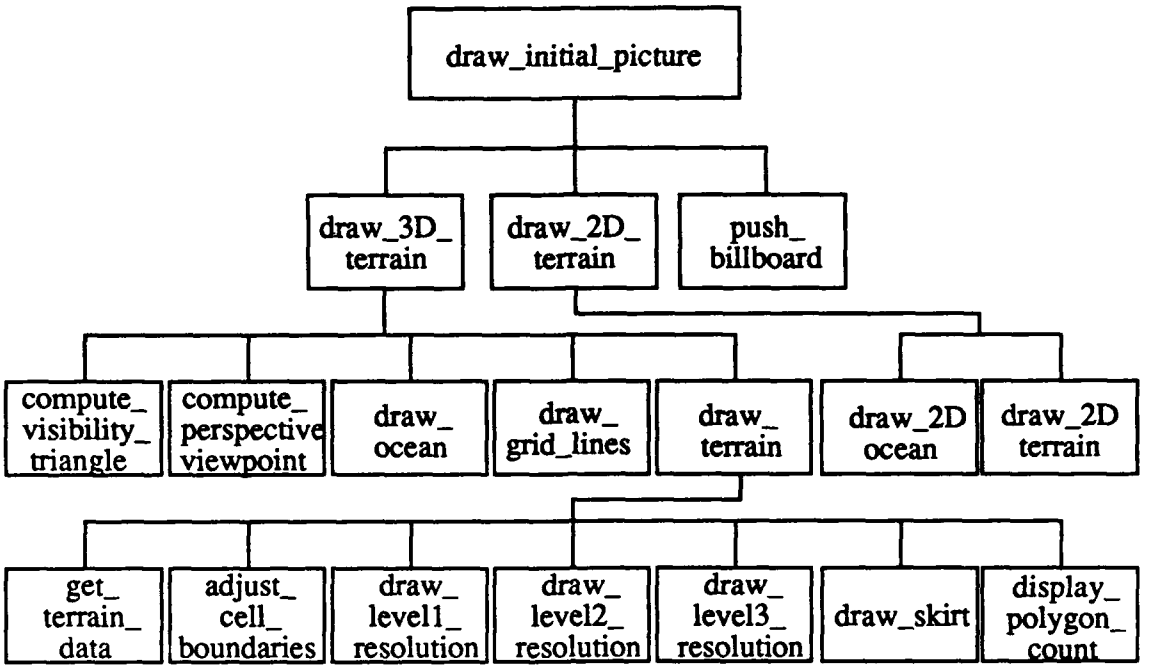


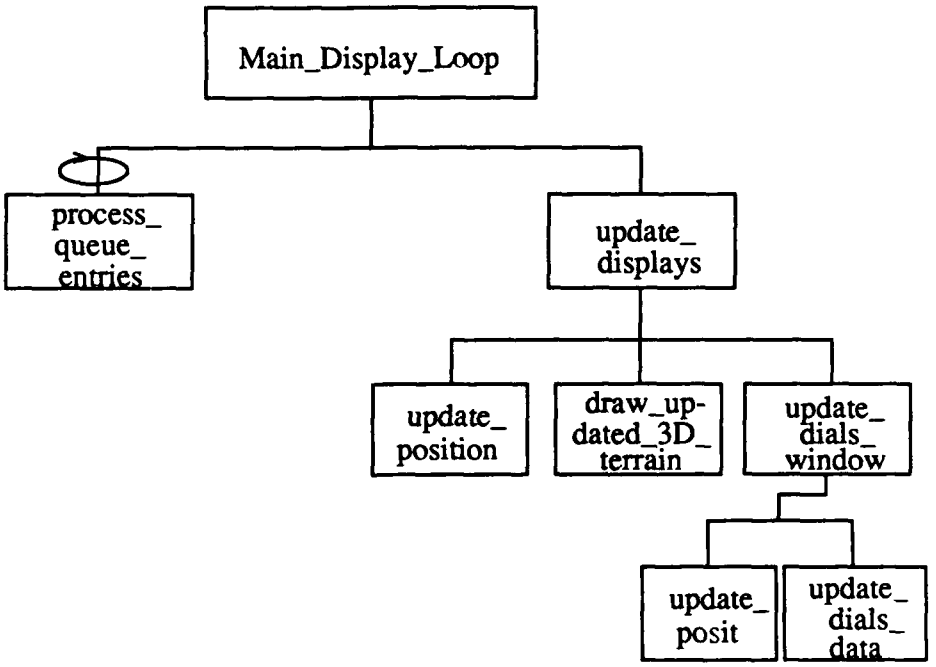


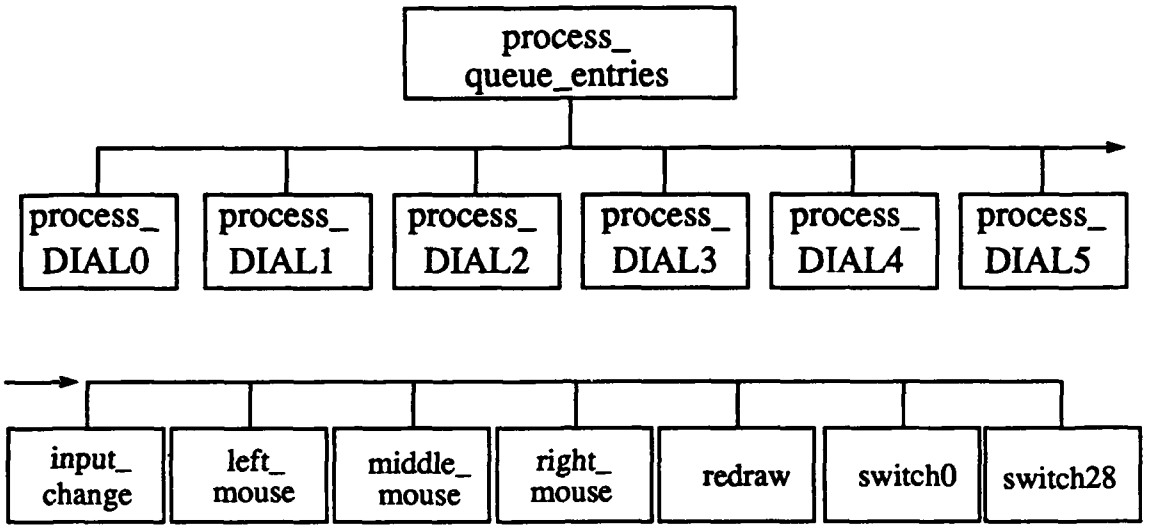

initialize_3d
init_ graphics
billboard
init_ queue
init_ controls
init_ view
init_ terrain
initialize_ view
initialize_ tempview


read_terrain_data
open_ data_file
read_data
close_ data_file

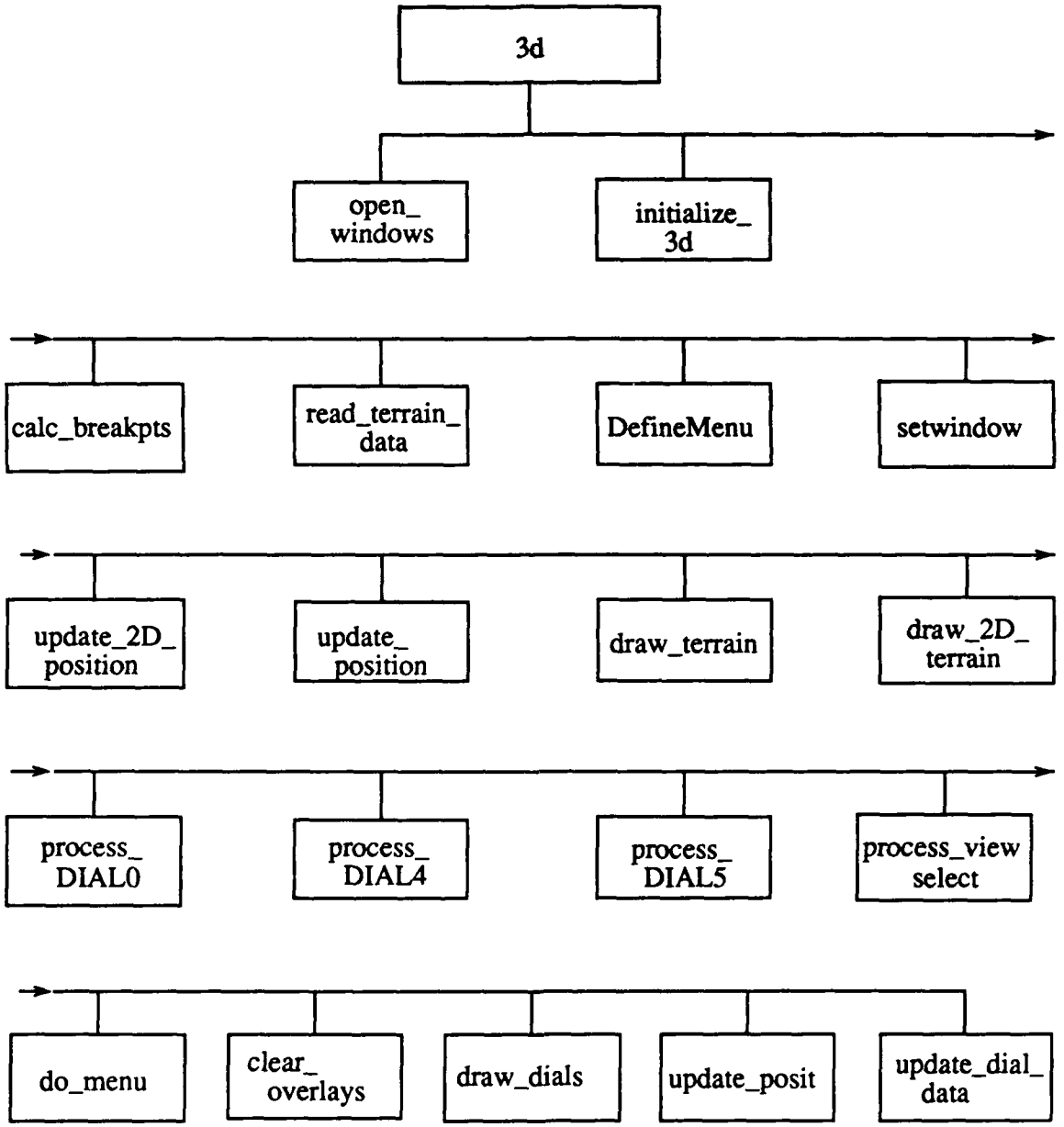

update_position
update_2D_ position
update_3D_ position
calc_current x_y_in_ window
update_ elevation
display_ current_ position
display_ look_ direction
display_ course_ direction
update_ view_ position
update_ tmpview_ position

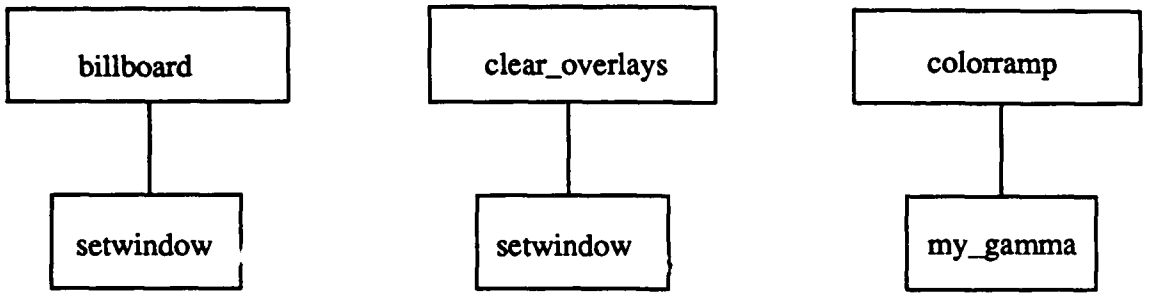

draw_initial_picture
draw_3D_ terrain
draw_2D_ terrain
push_ billboard
compute_ visibility_ triangle
compute_ perspective viewpoint
draw_ ocean
draw_ grid_lines
draw_ terrain
draw_2D ocean
draw_2D terrain
get_ terrain_ data
adjust_ cell_ boundaries
draw_ level1_ resolution
draw_ level2_ resolution
draw_ level3_ resolution
draw_skirt
display_ polygon_ count

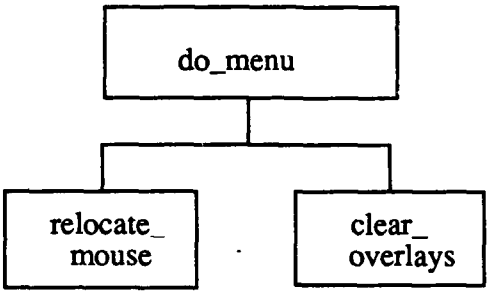

Main_Display_Loop
process_ queue_ entries
update_ displays
update_ position
draw_up- dated_3D_ terrain
update_ dials_ window
update_ posit
update_ dials_ data

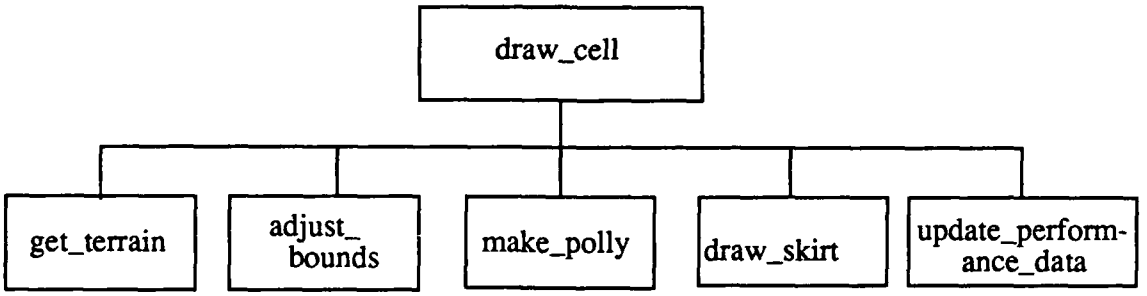

process_
queue_entries
process_
DIAL0
process_
DIAL1
process_
DIAL2
process_
DIAL3
process_
DIAL4
process_
DIAL5
input_
change
left_
mouse
middle_
mouse
right_
mouse
redraw
switch0
switch28

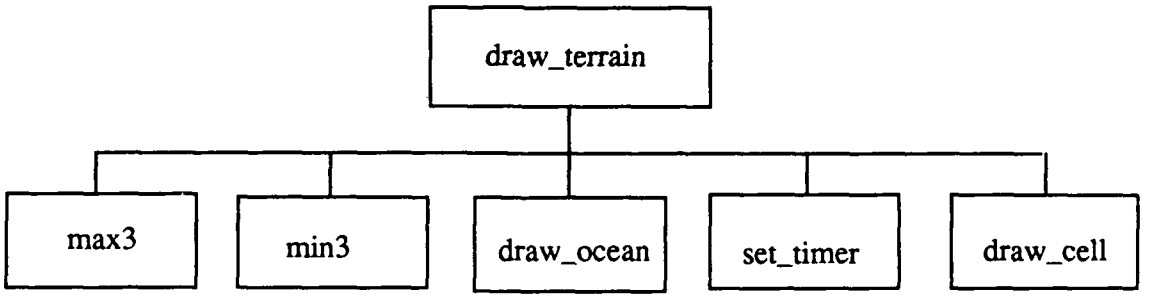
APPENDIX B - CALLS HIERARCHY CHART

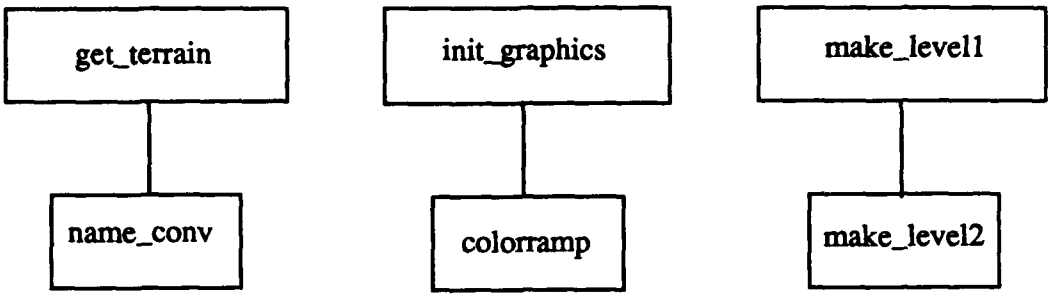


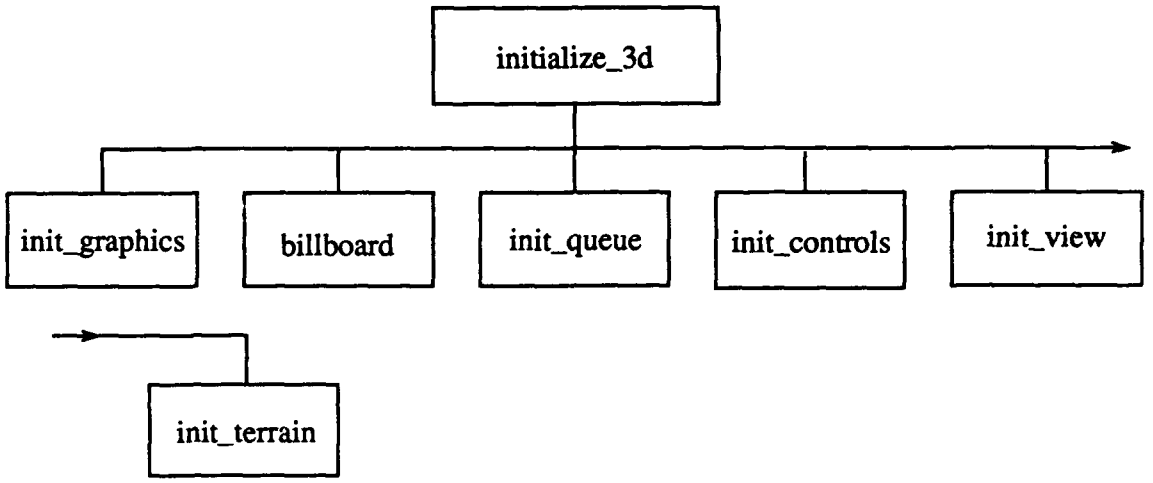

billboard
setwindow
clear_overlays
setwindow
colorramp
my_gamma

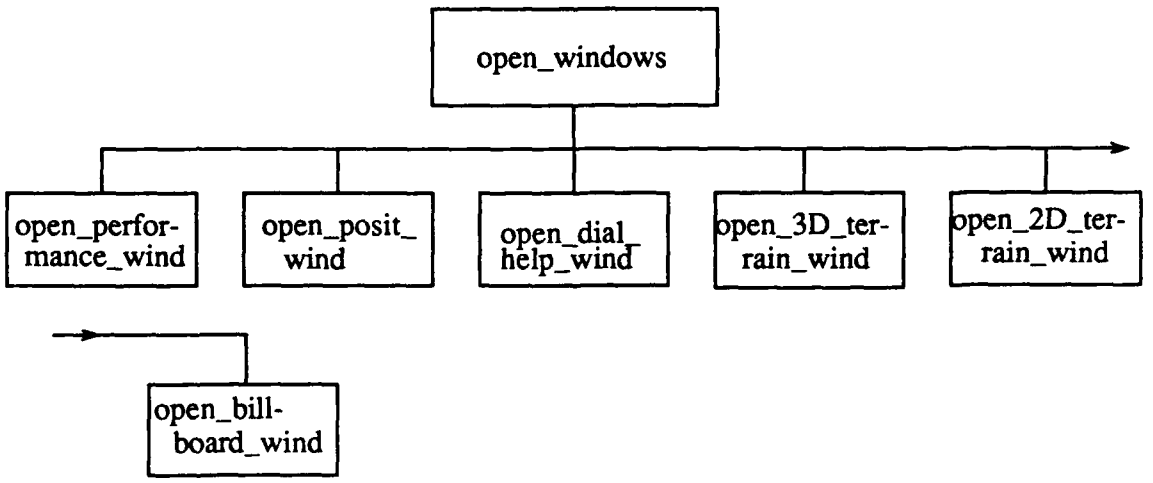

do_menu
relocate_ mouse
clear_ overlays

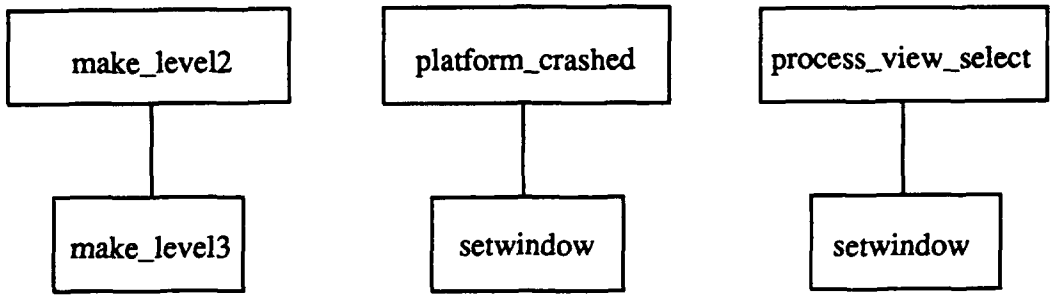

draw_cell
get_terrain
adjust_ bounds
make_polly
draw_skirt
update_perform- ance_data

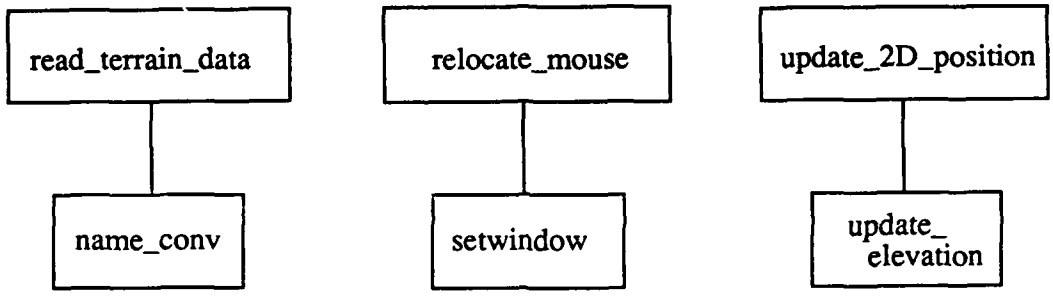

draw_terrain
max3
min3
draw_ocean
set_timer
draw_cell

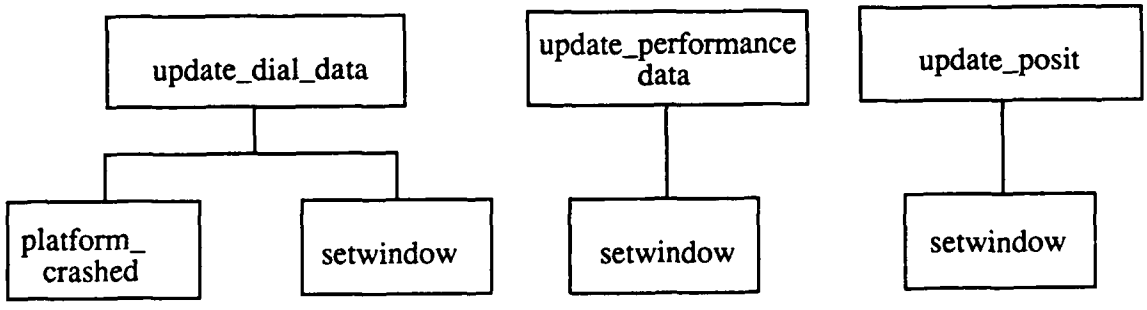

get_terrain
init_graphics
make_level1
name_conv
colorramp
make_level2


initialize_3d
init_graphics
billboard
init_queue
init_controls
init_view
init_terrain


open_windows
open_perfor-
mance_wind
open_posit_
wind
open_dial_
help_wind
open_3D_ter-
rain_wind
open_2D_ter-
rain_wind
open_bill-
board_wind


make_level2
make_level3
platform_crashed
setwindow
process_view_select
setwindow


read_terrain_data
name_conv
relocate_mouse
setwindow
update_2D_position
update_
elevation


update_dial_data
platform_
crashed
setwindow
update_performance
data
setwindow
update_posit
setwindow

APPENDIX C - CCWF MODULES AND THEIR PURPOSE

Source Code Files

3d -- main program driver contains main event loop

adjust_bounds -- updates cell boundaries prior to drawing 3D terrain view

billboard -- draws title screen while data structures are being loaded

calc_breakpoints -- calculates 16 altitude levels, used when choosing color for 2D and 3D views

clear_overlays -- clears overlay planes in all windows

colorramp -- establishes user defined color map

DefineMenu (mainmenu.c)-- defines popup menus

do_menu -- processes different popup menu selections

draw_2D_terrain -- draws two dimensional terrain map in 2D window

draw_cell -- draws 3D image given current location and heading

draw_dials -- draws dials and specifies their uses in dial help window

draw_ocean -- draws blue ocean plane

draw_skirt -- draws underlying polygon to conceal differences in resolution boundaries

draw_terrain -- calculates data and calls drawing routines to draw 3D image

get_terrain -- read in terrain data used to draw 3D image

init_controls -- initializes dials using setvaluator() and noise()

init_graphics -- configures IRIS for operation in z-buffering mode; calls procedure colorramp to define color map

init_queue -- selects devices to be queued using qdevice()

init_terrain -- establishes terrain data structure

init_view -- initializes variable view, initial viewpoint when program started

initialize_3d -- displays billboard and calls modules required to set up 3d

make_level1 -- draws level one polygons in 3D image

make_level2 --draws level two polygons in 3D image

make_level3 --draws level three polygons in 3D image

make_polly (draw_poly.c)-- draws polygons in 3D cell colored relative to altitude

maxmin -- max4,min4: calculates maximum/minimum of four float values; max3,min3: calculates maximum/minimum of three float values; returns answer in integer form

my_gamma (gamma.c)-- puts a gamma corrected color ramp into the color map

name_conv -- converts input latitude and longitude to file name

open_2D_terrain_wind -- opens window used to draw 2D terrain map

open_3D_terrain_wind -- opens window used to draw 3D terrain map

open_billboard_wind -- opens window used for displaying opening billboard

open_dial_help_wind -- opens window used for displaying dial box

open_performance_wind -- opens window used for displaying polygon count data

open_posit_wind -- opens window used for displaying position data

open_windows -- calls modules required to open all windows

platform_crashed -- creates crash in 3D window when platform's altitude is less than the current elevation

process_DIAL0 -- processes Dial 0 queue entry: course change

process_DIAL4 -- processes Dial 4 queue entry: look direction

process_DIAL5 -- processes Dial 5 queue entry: look distance

read_terrain_data -- reads in terrain data used to draw 2D land image

relocate_mouse -- moves mouse location to within 2D window, executed when user selects "Change viewpoint" from popup menu

set_timer (timer.c)-- activates system clock, used for performance checks

setwindow -- sets window and current window variable to gidc

update_2D_position -- updates position mark, course dead-reckoning vector, and look-direction vector in 2D window

update_dial_data -- inserts current course, speed, altitude, look-distance, look-direction, and look-angle into dials display

update_elevation -- updates global elevation variable with current location's ground elevation

update_performance_data -- updates polygon count display in performance data window

update_posit -- updates latitude and longitude data in posit window

Header files

constants -- definition of manifest constants

filenames -- path to CMA data, needed to establish data structure before 2D and 3D images are drawn

terrain -- definition of manifest constants used mainly in drawing routines

typedef -- defines enumerated types: viewpoint, octant, ptr1, ptr2, ptr3

APPENDIX D - CCWF MODULES AND WHO THEY ARE CALLED BY

Source Code Files

3d -- main driver routine

adjust_bounds -- draw_cell

billboard -- initialize_3d

calc_breakpoints -- 3d

clear_overlays -- 3d, do_menu

colorramp -- init_graphics

DefineMenu (mainmenu.c) -- 3d

do_menu -- 3d

draw_2D_terrain -- 3d

draw_cell -- draw_terrain

draw_dials -- 3d

draw_ocean -- draw_terrain

draw_skirt -- draw_cell

draw_terrain -- 3d

get_terrain -- draw_cell

init_controls -- initialize_3d

init_graphics -- initialize_3d

init_queue -- initialize_3d

init_terrain -- initialize_3d

init_view -- initialize_3d

initialize_3d -- 3d

make_level1 -- get_terrain

make_level2 -- make_level1

make_level3 --make_level2

make_polly (draw_poly.c) -- draw_cell

max3, min3 (maxmin.c) -- draw_terrain

my_gamma (gamma.c)-- colorramp

name_conv -- get_terrain, read_terrain_data

open_2D_terrain_wind -- open_windows

open_3D_terrain_wind -- open_windows

open_billboard_wind -- open_windows

open_dial_help_wind -- open_windows

open_performance_wind -- open_windows

open_posit_wind -- open_windows

open_windows -- 3d

platform_crashed -- update_dial_data

process_DIAL0 -- 3d

process_DIAL4 -- 3d

process_DIAL5 -- 3d

read_terrain_data -- 3d

relocate_mouse -- do_menu

set_timer (timer.c) -- draw_terrain

setwindow -- 3d, billboard, clear_overlays, platform_crashed, process_view_select, relocate_mouse, update_dial_data, update_posit

update_dial_data -- 3d

update_2D_position -- 3d

update_performance_data -- draw_cell

update_posit -- 3d

Header files

constants -- 3d, billboard, calc_breakpoints, clear_overlays, colorramp, do_menu, draw_2D_terrain, draw_dials, draw_ocean, draw_skirt, my_gamma, get_terrain, init_controls, init_graphics, init_view, open_2D_terrain_wind, open_3D_terrain_wind, open_billboard_wind, open_dial_help_wind, open_performance_wind,open_posit_wind, platform_crashed, process_DIAL0, process_DIAL3, process_DIAL4, read_terrain_data, relocate_mouse, setwindow, update_2D_position, update_dial_data, update_elevation, update_performance_data, update_posit, update_position

filenames -- get_terrain, read_terrain_data

terrain -- adjust_bounds, draw_cell, draw_ocean, draw_poly, draw_terrain, process_view_select

typedef -- 3d, do_menu, draw_2D_terrain, get_terrain, init_terrain, init_view, DefineMenu, make_level1, make_level2, make_level3, process_DIAL0, process_DIAL4, read_terrain_data, relocate_mouse, update_2D_position, update_dial_data, update_posit, update_position

APPENDIX E - CCWF USER'S MANUAL

The Command and Control Workstation of the Future's (CCWF) primary feature is to generate a three-dimensional visualization of a 60 nautical mile square area (cell). Using the Silicon Graphics, Inc. IRIS 4D-70GT graphics workstation the user is able to drive around the cell at a near real-time rate of animation. This visualization is displayed in the Three-Dimensional Terrain Window on the IRIS monitor (see Figure E.1).

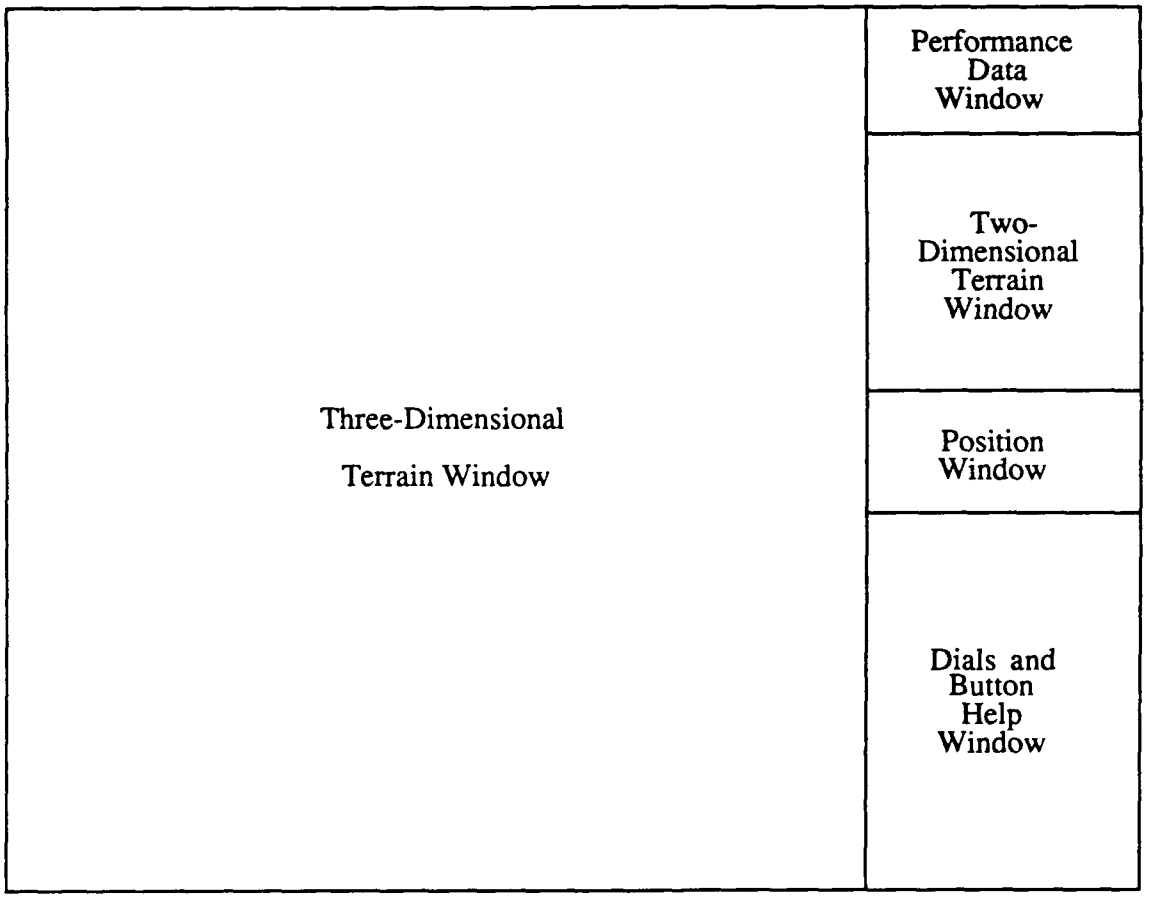


Figure E.1 CCWF Window Layout

1. Navigating

In order to navigate (drive around) the three-dimensional image, the user manipulates the dials of the IRIS dial box. Figure E.2 is a drawing of the dial box which has a functional label below each dial. For example, to alter course, turn the bottom left dial either left or right dependent upon the desired action. The dials affect the platform (ship or aircraft) currently being viewed from.

2. Use of Dials

The dials' functions are as follows.

a. Course

True direction platform is traveling. The unit of measure is degrees. Course is represented by a red line in the Two-Dimensional Terrain Window (see Figure E.1). The length of the red line is equal to the distance the viewer's platform travels in six minutes at current speed.

b. Speed

The rate at which the platform is moving. The unit of measure is nautical miles per hour (knots). Maximum speed for a ship is 35 knots. Maximum speed for an aircraft is 1000 knots.

c. Altitude

The vertical distance between the platform and sea level. The unit of measure is yards. Altitude for a ship is a constant ten yards. Maximum altitude for an aircraft is 5000 yards.

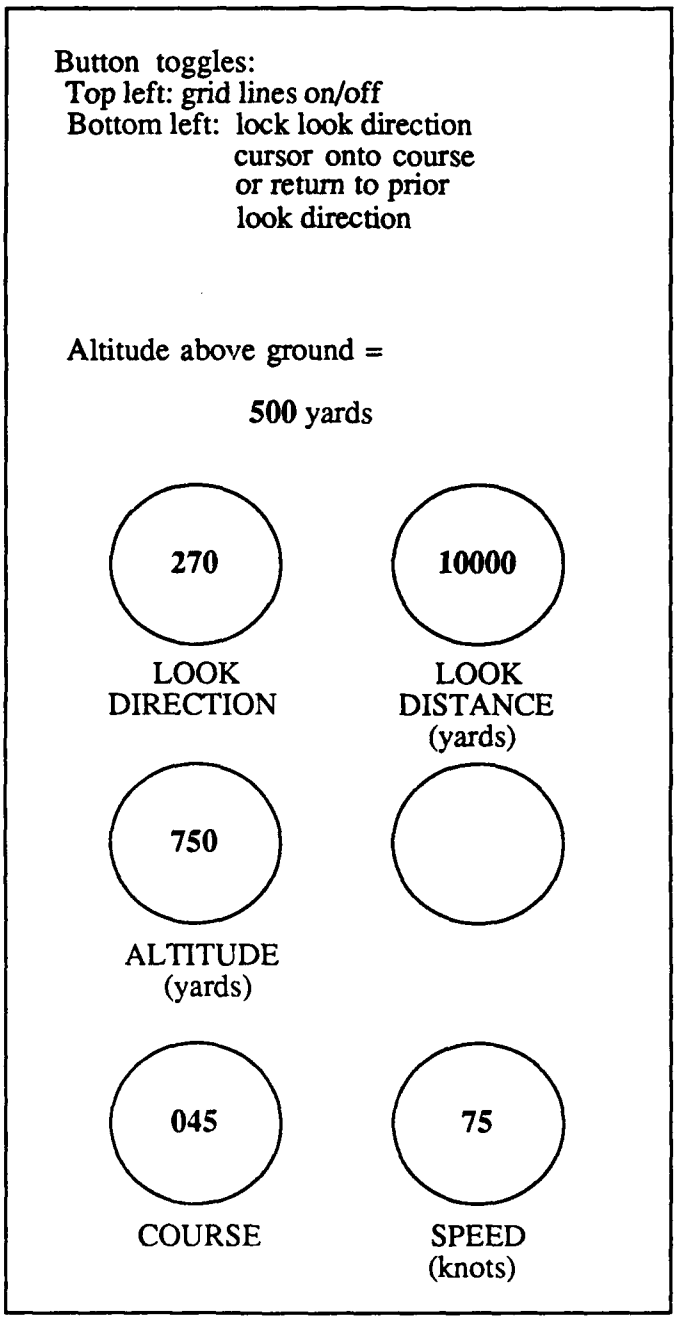


Figure E.2 Dial and Button Help Window

d. Look Direction

The true direction the viewer is looking. Look direction is independent of course. For example, if the ship is traveling due North and the viewer is looking at an object off his port beam, then Course will be 000 and Look Direction is 270. The unit of measure is degrees. Look Direction is represented by a black line in the Two-Dimensional Terrain Window.

e. Look Distance

The length of the black line in the Two-Dimensional Terrain Window. By manipulating both Look Direction and Look Distance the user can determine the bearing and range to an object of interest in the 2-D window.

The IRIS dial box is displayed in the Dial and Button Help Window (see Figure E.1). In addition to each dial's functional label, the current value is displayed inside the dial. Also displayed in this window is the current altitude above ground. Altitude above ground is the vertical distance, in yards, between the user's platform and the surface of the earth directly below. If this distance becomes less than or equal to zero, then the platform will crash. To recover from a crash, increase altitude until altitude above ground is greater than zero.

3. Use of Buttons

There are two buttons on the IRIS button box that affect the CCWF. By pressing the top left button, the grid lines are turned on or off. By pressing the bottom left button, the look direction is locked onto the course. When this lock is in effect, the Look Direction dial is disabled. To re-enable the Look Direction dial and return to previous Look Direction press the bottom left button again.

4. Position Display

Above the Dial and Button Help Window, is the Position Window. The latitude and longitude are displayed for the platform's current position (see Figure E.3).

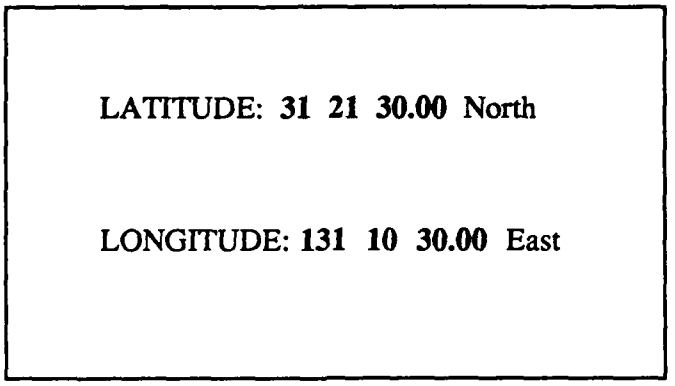


Figure E.3 Position Window

5. Two-Dimensional Display

Above the Position Window, is the Two-Dimensional Terrain Window. This image represents the entire cell. The land region is color-coded relative to elevation. Blue represents water. Dark green shades represent lower elevations. Yellow shades are middle elevations. Red shades are higher elevations. Also shown in this window is a small red circle indicating the platform's current position. Notice that as the platform moves, the small red circle moves accordingly. This feature allows the viewer to know his relative location within the cell at all times.

The 2-D window is the only window which may be moved and/or resized. To do this, position the mouse cursor (red arrow) on the top title bar of the 2-D Terrain Window. Press and hold down the right mouse button (menu button). A menu similar to Figure E.4 is displayed. While still holding the menu button down move the

mouse cursor (move mouse) to the selection (Move or Resize) desired. Releasing the menu button causes selection of the desired menu item.

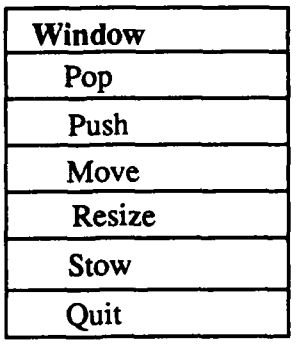


Figure E.4 Max System Menu

a. Moving 2-D Window

If MOVE is selected, as the mouse is moved, a red outline of the 2-D window will move around the screen. When the outline is in the desired location, press and release the menu button. The 2-D Terrain Window will now be redrawn in the new location.

b. Resizing 2-D Window

If RESIZE is selected, as the mouse is moved, a small red box will move around the screen. When the upper left-hand corner of the box is in the desired location, press and hold down the left mouse button, now drag the lower right-hand corner of the box to the desired window size by moving the mouse. When desired size is reached, release the mouse button.

6. Performance Data Display

Above the Two-Dimensional Terrain Window, is the Performance Data Window (see Figure E.5). The value displayed in this window is a count of the number of polygons being drawn to create the three-dimensional image. The polygons

being drawn are of two types: single triangles when drawing coastal terrain or double triangles, forming a square, when drawing inland terrain.

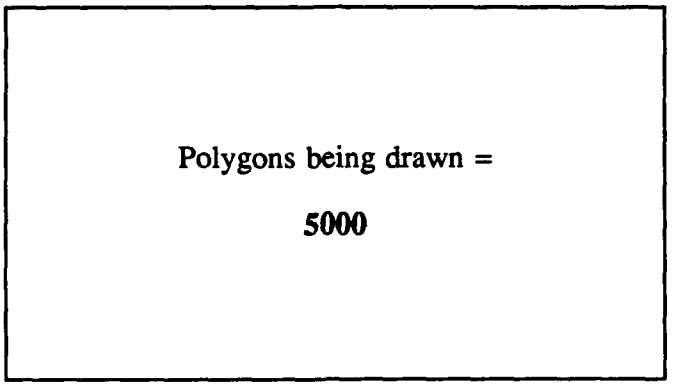


Figure E.5 Performance Data Window

7. CCWF Menu Features

To access the CCWF menu system, move the cursor to the interior of any window. Press and hold down the menu button. A menu similar to Figure E.6 appears. The arrows indicate that each item, except "Quit," has a sub-menu associated with it. To select a sub-menu, slide the mouse either left or right and the sub-menu

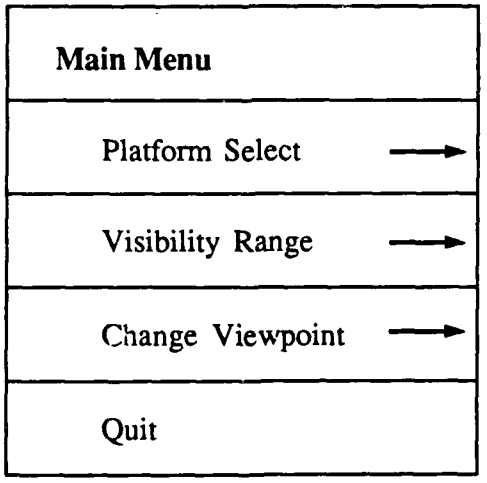


Figure E.6 Main Menu

will appear. The features available through sub-menus are as follows.

a. Platform Select

Allows selection of a platform from a sub-menu similar to Figure E.7. If "Ship" is selected (by releasing menu button when desired item is highlighted), altitude is set to ten yards and speed is limited to less than or equal to 35 knots. Selection of "Aircraft" (default) allows for maximum speed of 1000 knots and maximum altitude of 5000 yards.

Choose platform
Ship
Aircraft

Figure E.7 Platform Select Sub-Menu

b. Visibility Range

Provides a choice of visibilities to select from (see Figure E.8). "Unlimited" visibility is the default.

Select Visibility
Unlimited
15 miles
10 miles
5 miles
3 miles

Figure E.8 Visibility Sub-Menu

c. Change Viewpoint

Allows the user to change the position from which he is currently looking at the three-dimensional image. By selecting the "Select new viewpoint with left mouse" option (see Figure E.9), the mouse cursor is automatically relocated to the Two-Dimensional Terrain Window.

In 2D window:
Select new viewpoint with left mouse
Return to original view with middle mouse

Figure E.9 View Change Sub-Menu

At this point, position the mouse cursor within the 2-D window where the new viewpoint is desired. Once positioned, press and release the left mouse button. Notice, the 3-D image is now being drawn from the new position. Also notice, the old position is marked by a small yellow circle. The yellow circle will continue to move in accordance with the course and speed which was in effect when the viewpoint was changed. Viewpoint may be changed as many times as desired by following the above procedure. When ready to return to the original viewpoint, marked by the yellow circle, press and release the middle mouse button.

d. Quit

Selection of this menu option causes an exit to the program and return to the operating system prompt.

8. Conclusion

Through the use of a logical arrangement of controls and menu driven feature selection, a new user will be able to master the CCWF with only a short period of familiarization.

LIST OF REFERENCES

1. Harris, Frank E., Preliminary Work on the Command and Control Workstation of the Future, Master's Thesis, Naval Postgraduate School, Monterey, California, August 1988.

2. Whitten, J.L., Bentley, L.D., and Ho, T.I., Systems Analysis and Design Methods, Times Mirror, 1986.

3. Kernighan, B.W. and Plauger, P.J., The Elements of Programming Style, 2d ed., McGraw-Hill Book Co., 1978.

4. Bentley, Jon L., Writing Efficient Programs, Prentice-Hall, 1982.

5. IRIS GT Graphics Library User's Guide, Version 1.0, Silicon Graphics, Inc., Mountain View, California, 1988.

6. Sanders, M.S. and McCormick, E.J., Human Factors in Engineering and Design, 6th ed., McGraw-Hill Book Co., 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Center for Naval Analyses 4401 Ford Avenue Alexandria, Virginia 22302-0268	1
4.	Director of Research Administration, Code 012 Naval Postgraduate School Monterey, California 93943	1
5.	Dr. Michael J. Zyda Naval Postgraduate School Code 52, Dept. of Computer Science Monterey, California 93943-5100	2
6.	Mr. Bill West HQ, USACDEC Attention: ATEC-D Fort Ord, California 93941	1
7.	John Maynard Naval Ocean Systems Center Code 402 San Diego, California 92152	1
8.	Duane Gomez Naval Ocean Systems Center Code 433 San Diego, California 92152	1

9. James R. Louder
Naval Underwater Systems Center
Combat Control Systems Department
Building 1171/1
Newport, Rhode Island 02841

1

10. LT Rex Cobb
c/o Ray Hunt
665 Pinehurst Circle, NE
Palm Bay, Florida 32905

2