1994-09

Creating a real-time three dimensional display for the Janus combat modeler.

Vaglia, James A.

Monterey, California: U.S. Naval Postgraduate School

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1994	Master's Thesis

4. TITLE AND SUBTITLE
Creating a Real-Time Three Dimensional Display for the Janus Combat Modeler (U)

5. FUNDING NUMBERS

6. AUTHOR(S)
Vaglia, James Arthur

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
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 United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Several training readiness deficiencies were noted during the mobilization of the National Guard roundout brigades in support of Desert Shield/ Storm. One of the areas was the brigade and battalion staff battlefield synchronization skills. National Guard Brigade armories are normally located throughout large geographic areas. Due to the cost and time required to bring all the units to a common training area, this training is conducted only once a year. The problem addressed in this research is to create a visualization tool capable of rendering the Janus combat modeler in a three dimensional environment using scripted files and real time data. The visualization tool must be networked via telephone modems to allow the brigade to conduct unit training while at their home station.

The approach taken was first to design a directory structure that places the terrain files in unique locations that are accessible by the terrain conversion programs and the 3D visualization program. The next step was to create programs to convert the Janus terrain files. Construction of a three dimensional environment capable of displaying real-time information from Janus followed. The last step was to produce an interface for the modem communications and to display that information in real time in the virtual environment.

The result is the creation of the Janus-3D Visualizer capable of accurately depicting a Janus scenario running locally or from a remote site. This tool provides commanders with a three dimensional perspective of the battlefield, emplacement of the weapon systems and engagements during a battle.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Virtual Environment, Three-Dimensional, Combat Modeling, Terrain Generation, Real-Time, JANUS, NPSNET, Networking	93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Approved for public release: distribution is unlimited

CREATING A REAL-TIME THREE DIMENSIONAL DISPLAY FOR THE JANUS COMBAT MODELER

James A. Vaglia
Captain, United States Army
B.S., Slippery Rock University of Pennsylvania, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Department of Computer Science

ABSTRACT

Several training readiness deficiencies were noted during the mobilization of the National Guard roundout brigades in support of Desert Shield/ Storm. One of the areas was the brigade and battalion staff battlefield synchronization skills. National Guard Brigade armories are normally located throughout large geographic areas. Due to the cost and time required to bring all the units to a common training area, this training is conducted only once a year. The problem addressed in this research is to create a visualization tool capable of rendering the Janus combat modeler in a three dimensional environment using scripted files and real time data. The visualization tool must be networked via telephone modems to allow the brigade to conduct unit training while at their home station.

The approach taken was first to design a directory structure that places the terrain files in unique locations that are accessible by the terrain conversion programs and the 3D visualization program. The next step was to create programs to convert the Janus terrain files. Construction of a three dimensional environment capable of displaying real-time information from Janus followed. The last step was to produce an interface for the modem communications and to display that information in real time in the virtual environment.

The result is the creation of the Janus-3D Visualizer capable of accurately depicting a Janus scenario running locally or from a remote site. This tool provides commanders with a three dimensional perspective of the battlefield, emplacement of the weapon systems and engagements during a battle.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

A. BACKGROUND

In 1992, Congress mandated the initiation of the Simulation in Training for Advanced Readiness (SIMITAR) program. The SIMITAR program is a part of the Army National Guard Combat Readiness Reform Initiative Act of 1992. The purpose of this program is to correct the training readiness deficiencies noted in the mobilization of the Army National Guard roundout/roundup brigades during operation Desert Shield/Storm. The intent of the program is three fold: to demonstrate the potential of using advanced technologies to train National Guard units, to learn how to incorporate a digital training environment into future training, and to "establish a model for a Battle Lab." The Advanced Research Projects Agency's (ARPA) mission, as the project manager, is to stimulate the research while focusing on dual use applications and improving cost and performance within the Department of Defense (DOD). [FUNK94][WEST94][ARMY93]

The SIMITAR project was designed to look at the areas noted as training deficiencies. Once the areas were identified, the next step was to look at a combination of existing and new technology to correct the training deficiencies. These new systems should be affordable, realistic, require minimal manpower to operate, and be distributed.

One problem area noted was brigade and battalion staff battlefield synchronization skills. It is this weakness that the Janus-3D Visualizer attempts to alleviate. Janus, itself, is an accepted Army combat modeler and can be used to train battalion and brigade staffs[JANU93c]. However, it is two dimensional and thus limits the realism with which commanders can view the battle. Also, it is not networked and would require the National Guard units to travel to a central location to conduct their training. The motivation for this research, therefore, is to produce a vehicle that will allow the National Guard units to remain at their home station and fight a Janus simulated battle while viewing that battle in three dimensions. This three dimensional view verses Janus' traditional two dimensional view provides a more realistic picture of the terrain. With this added realism, commanders can emplace and deploy their units to make the most of the cover and concealment more

readily seen in three dimensions. In addition, replaying the Janus scenario in three dimensions gives the commanders an additional, more effective, tool to use when conducting After Action Reviews (AARs). Finally, the wide area networking capability allows for more frequent training as scarce dollars will be saved on travel.

B. OBJECTIVE

The ultimate objective of this research was to produce a portable three-dimensional virtual environment integrated with Janus 3.17 (UNIX) and capable of communicating over a wide area network via modem and commercial telephone lines. The method used to accomplish this objective was first to design a directory structure that places the terrain files in unique locations that are accessible by the terrain conversion programs and the 3D visualization program. The next step was to create programs to convert the Janus terrain files. To validate the conversion, we used the replay program [WALT92]. Subsequent to the validation of the terrain conversion, the construction of a three dimensional environment capable of displaying real-time and scripted information from Janus was accomplished. Finally, an interface was produced for the modem communication software in order to display that information in real time in the virtual environment [UPSO94].

The success of this research is measured by the ability of the Janus-3D Visualizer to accurately depict a Janus 3.17 (UNIX) scenario running on an Hewlett Packard (HP) workstation and, at the same time, display the scenario on another Janus-3D Visualizer reading the same data from a modem.

C. ORGANIZATION OF THESIS

This thesis is organized into six chapters. This chapter provides the motivation and objectives of the work performed. Chapter II describes the Janus combat modeler, NPSNET II, and previous work related to this research. Chapter III provides a description of the Janus-3D Visualizer to include the conversion of terrain, communication requirements and functionality, and the three dimensional display. The Janus terrain file formats and conversion into files readable by the Visualizer is detailed in Chapter IV.

Chapter V gives a detailed description of the Janus-3D Visualizer screen layout and software architecture. Chapter VI consists of the conclusions and recommendations for future work. There are also two appendices. Appendix A is the user's guide, which provides instructions on how to install, set up, and use the 3D Visualizer. Finally, Appendix B details the Janus terrain file format.

II. OVERVIEW OF JANUS AND NPSNET II

A. THE JANUS COMBAT MODELER

The primary purpose of Janus is to train brigade and lower staffs in ground combat operations through the use of a computer based war-gaming simulation. The program allows the staffs to plan and conduct combat operations against an opposing force. Janus displays and adjudicates the engagements of the two forces from the commencement to the conclusion of the battle. Thus, Janus derives it's name from the two faced Roman god that guarded the gates of Rome and the patron of beginnings and endings. [FUNK94]

1. History

The original Janus simulation was fielded in 1978. The program was developed at the Lawrence Livermore National Laboratory (LLNL) to model nuclear effects. This version later became known as Janus (L). The program gained notoriety for its graphical interface. U. S. Army Training and Doctrine Command (TRADOC) acquired the prototype in 1980 through the Janus Acquisition and Development program. TRADOC Analysis Command, White Sands Missile Range (TRAC-WSMR) became the proponent agency for the army and modified Janus (L) to meet Army combat development needs. This model became known as Janus (T). [WEST94]

Both systems gained popularity with their users and several modified versions were spread throughout the user community. The Army realized the value of the system for use in both analysis and training and in 1989 began the Janus (ARMY) Program.The program was designed the incorporate the best of Janus(L) and Janus (T) into a standardized multipurpose system. The combination of Janus (L) and Janus (T) became known simply as Janus. [WEST94]

The original Janus was developed on the Digital VAX suite of computers with the graphics displayed on Tektronix Model 4225 graphics workstations. In 1992, TRAC ported the model to run using the UNIX operating system with graphics displayed on X-Window

workstations. For the purpose of this research the UNIX version of Janus 3.17 was used. [JANU93c]

2. Description

Janus is a monolithic two-sided ground combat simulation designed for conflict up to brigade versus division levels. However, normally smaller battles are fought. The model is monolithic in that it runs on one machine with several other systems displaying the opposing forces. The system is closed in that there are no outside influences. Two-sided refers to the fact that the operators sitting at the various terminals cannot see the disposition of the opposing forces until they come in contact with or are detected by their simulated forces. Janus is interactive, the controllers at each of the workstations monitors, reacts to, directs, and redirects all actions of the units under their control. The model focuses on the engagements of individual land based fighting systems. The engagements are stochastic, in that certain events occur according to the laws of probability. Janus provides the ability to replay troop movements, artillery fires, and individual engagements. This ability provides useful information in the conduct of after action reviews (AAR) and analysis of new weapons systems. [JANU93c]

The Janus battle is displayed on a digitized terrain map developed from digitized terrain elevation data (DTED) provided by the Defense Mapping Agency (DMA). The terrain format is similar to the military topological maps, and includes contour lines, roads, rivers, vegetation, and urban areas. These terrain features affect line of sight for detection and engagements and vehicle movement similarly to they was they would in the real world. Janus map symbology includes operational overlays, and military map unit symbols or icon depicting the actual weapon system. [FUNK94]

3. Hardware

For the purpose of this research, the Janus software runs on a hardware suite consisting of a Hewlett-Packard Model 715/50 host computer networked via ethernet to one or more Sun Classic workstations, and a Silicon Graphics Indigo2 Extreme. Each workstation has a 19-inch color monitor, a keyboard, and a three-button mouse. The Silicon

Graphics Indigo2 Extremes also have modems to establish connections with the Silicon Graphics Indigo2 Extremes at the other armories.

4. Software

The Janus model is composed of sixteen executable programs written in FORTRAN. The programs may be divided into four major groups: data base management; scenario creation/execution; scenario analysis, and utility programs. [JANU93d]

a. Data Base Management

There are five programs used to manage the map, weapon systems, and graphic symbols data bases. TRNFLTR allows the user to select a smaller section of the master terrain file and converts the data so that Janus can read the data. The user then uses the TED to select the contour intervals, levels of urban and vegetation densities, and to draw rivers and roads. The CSDATA program is used to manage the weapons systems characteristics and maintains the weather conditions, prior to being read into a scenario. Once in a scenario, PED used to modify some of the weapons systems data base The last data base management program is SYMBOLS. This program is used to modify the graphic symbols displayed on the digitized map. [JANU93d]

b. Scenario Creation / Execution

Six programs compose the scenario creation / execution portion of the model. FORCE is used to incorporate units into a scenario and to modify the force structure in existing scenarios. The program MERGE combines the two opposing forces and produces another scenario. INITSCEN initializes the battlefield environments and allows the user to delete preplanned events. VFYSCEN and GRFVFY will print out system characteristics and any errors they find in the data base of a scenario. The last program in this area is JANUS. This executable program is used to conduct initial planning and to execute the planned scenario. [JANU93d]

c. Scenario Analysis

POSTP and JAAWS are used to display kill sum reports, results of artillery fire, direct fire, and detection and engagement reports. These can be used in analyzing previously executed scenarios: the tactics used and the employments of new weapons. [JANU93d]

d. Utility

There are two utility programs. HELPEDIT is used to create or modify the janus help files for the user. The last executable program, FORMS, is used to create and modify the forms that the data is presented to the user. [JANU93d]

B. NPSNET II

NPSNET is a real-time, interactive, distributed, three-dimensional visual simulation system. It was developed by students at the Computer Science Department of the Naval Postgraduate School (NPS) in Monterey, California. NPSNET is designed to be a low-cost simulation system run on Silicon Graphics, Inc. IRIS workstations. [SGI90][OSBO91][ZYDA92]

NPSNET II is written in Kernighan & Ritchie C. Input and output devices that are supported include the keyboard, button/dialbox, mouse, Spaceball, and the screen. NPSNET uses a generic simulator to operate the vehicles on the battle field. This allows the user to become any ground, sea, or air vehicle in the simulated world that he wishes. The user can change to another vehicle by simply pressing a button rather than switching hardware. The system is networked via ethernet using protocol data units (PDUs) to communicate with other systems on the network. The user has the ability to change the time of day, turn on and off the textures, and to record or run a scripted files.[OSBO91]

Vehicles and objects in NPSNET are modeled using NPS Object File Format (NPSOFF). This file format was developed at NPS and is an ASCII formatted language which incorporates many Graphics Library (GL) function calls. This encapsulation enables the object to easily be transported between various programs and gives the user an abstract

manner to reference the objects. Since the objects are stored in ASCII format the user can easily view and edit the file.[SGI90]

The terrain in NPSNET is displayed using several levels of detail stored in quadtrees. This is explained in more detail later in this chapter under the previous work of CPT Randall Mackey.[MACK91]

Current work on NPSNET includes the incorporation of the individual soldier in the virtual environment, temporal effects, dynamic terrain and the inclusion of semi autonomous generated from Janus and MODSAF. The group is also working on creating a system to provide NPSNET with spatialized sound.

C. PREVIOUS WORK

1. NPSNET Terrain

The original version of NPSNET was limited to an eight by eight kilometer area of terrain due to the amount of memory needed to store the terrain data. CPT Mackey's Thesis research [MACK91]focused on developing paging algorithms to swap terrain data in and out of memory to allow the use of the fifty by fifty kilometer data available and a quadtree data structure to store and access the various resolutions of the terrain. The terrain used is based on elevation postings every 125 meters.

The fifty by fifty kilometer terrain was converted into one by one kilometer squares. The smaller files are easier to swap in and out of memory. Then to implement the terrain paging, a sixteen by sixteen kilometer area of terrain is loaded into memory centered on the location of the user. As the user moves around the world and approaches the end of the terrain stored in memory, the program swaps the terrain farthest from the user with new terrain in the users direction.

The quadtree data structure allows the storage of multiple levels of resolution of the terrain. It is based on a recursive subdivision of data. Figure 1 shows the structure used. The root of the tree contains the lowest level of detail and the leaves contain the highest level of detail. For the highest level of detail every 125 meter elevation posting is used to

form polygons. The second level of resolution is formed by taking the data used to draw four 125 by 125 meter areas and drawing two triangles to depict a 250 by 250 meter area. The next level requires the data used to draw sixteen 125 by 125 meter areas to draw two triangles displaying a 500 by 500 meter area. The lowest level of detail takes all the data points required to draw sixty-four 125 by 125 meter areas to draw two triangles depicting the one kilometer grid square.

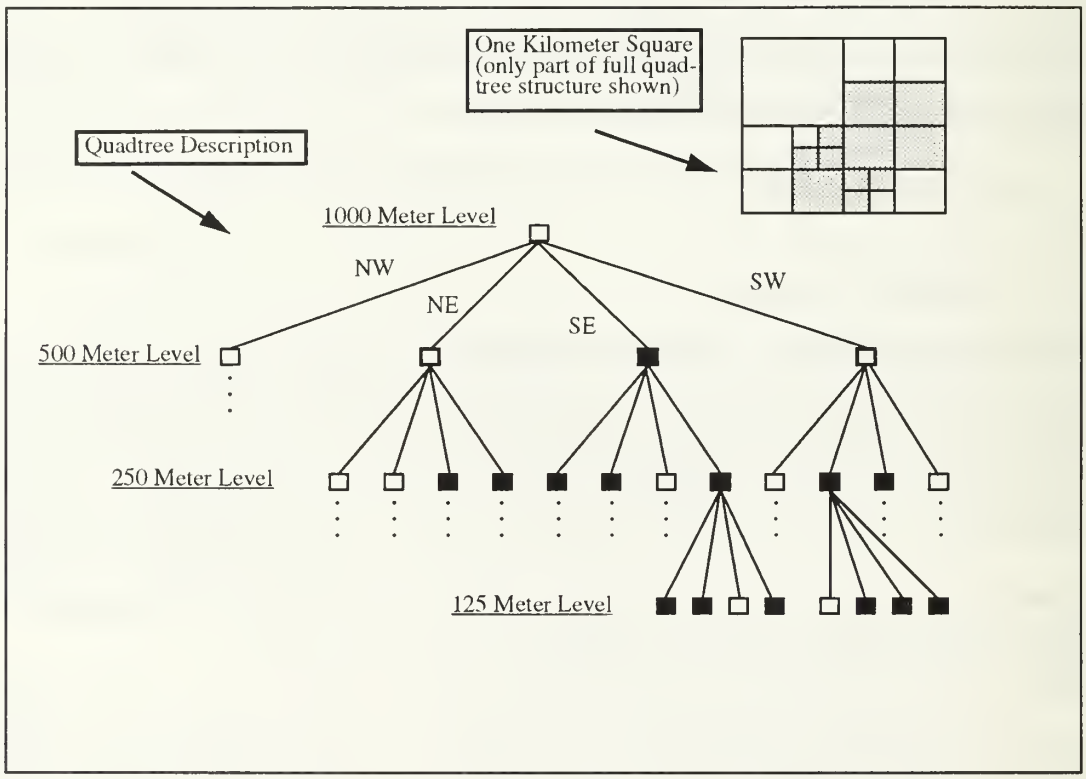

Figure 1. Multiple Resolution Quadtree

The next step is to render the terrain on the screen, this occurs in several steps. First, only the terrain in the field of view (FOV) is rendered. Next the levels of detail is determined, 0 to 1,500 meters highest level, 1,500 to 3,000 the second level, from 3,000 to 4,500 the third level with the remaining being rendered using the lowest level of detail. These cut off distances can be varied depending on the desired results. After the levels are determined fill polygons are drawn to close the gaps in the seams of the different levels of

the terrain. Lastly the objects such as, roads, rivers, urban areas, are only drawn on the terrain with the highest level of detail.

2. NPSNET/Janus Scripted Files

CPT Richard Smith's work involved reading a Janus scripted scenario into NPSNET to produce more realistic motion paths and better emplacement of weapon systems. Figure 2 shows the crux of his work. The approach he used was to generate an NPSNET script file by using Janus scenario files. The binary files used are SYSTEMXXX.DAT, DPLOYXXX.DAT, FORCXXX.DAT, and JSCRNXXX.DAT. (The XXX represents the digits of the scenario number.) These four files contain the weapon system's characteristics, the force composition, unit types, and emplacement and movement data. With the NPSNET file built, the user could manipulate the vehicles. These changes in the NPSNET scenario can be recorded and converted back into DPLOYXXX.DAT. The new scenario could then be run in Janus giving the vehicles better emplacement and more realistic movements.[SGI90]

3. NPSNET: Janus3D

The previous integration of NPSNET and Janus was accomplished by CPT Pat Warren and CPT John Walter, Figure 3. Their work involved creating three dimensional terrain from Janus terrain files. This process requires running the terrain data through fourteen conversion steps. This produces three types of files readable by NPSNET. Several of the files require modification prior to translating a different terrain and recompling the programs with the new variables. Once the terrain conversion is accomplished the terrain files need to be move to other directories. These new directory paths need to be added NPSNET paths and then recompile the simulator to be able to display the new terrain.[WALT92]

They were also able to create NPSNET scripted files from the Janus post processes files. To visualize and verify their work prior the incorporating it into NPSNET they constructed a two dimensional tool able to read the terrain and scripted files.

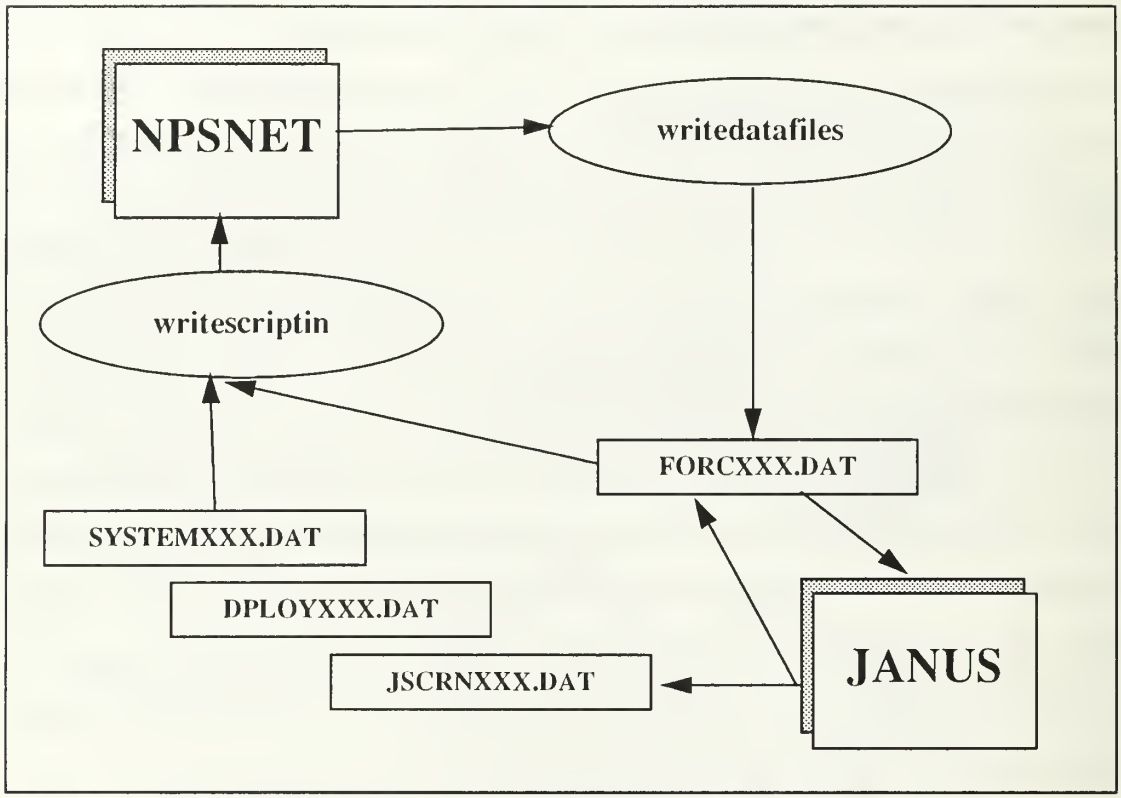

Figure 2. Creating Realistic Motion Paths

The culmination of CPT Warren and CPT Walter's work was displaying the real time vehicle movements and engagements in NPSNET. To accomplish this they made used of existing Janus calls to remote display terminals and wrote algorithms the calculate the speed and direction of the vehicles.

4. Summary

Janus has been in the Army since 1978 and continues to be improved. The model's display is two dimensional, the next step logically is to create a three dimensional display. NPSNET is a viable vehicle to use in creating the new display. NPSNET was developed at the Naval Postgraduate School as a low cost three dimensional virtual simulator. One of the goals of the SIMITAR project is to keep the costs minimal. This integration was accomplished by CPT Warren and CPT Walter. Their previous work in converting Janus

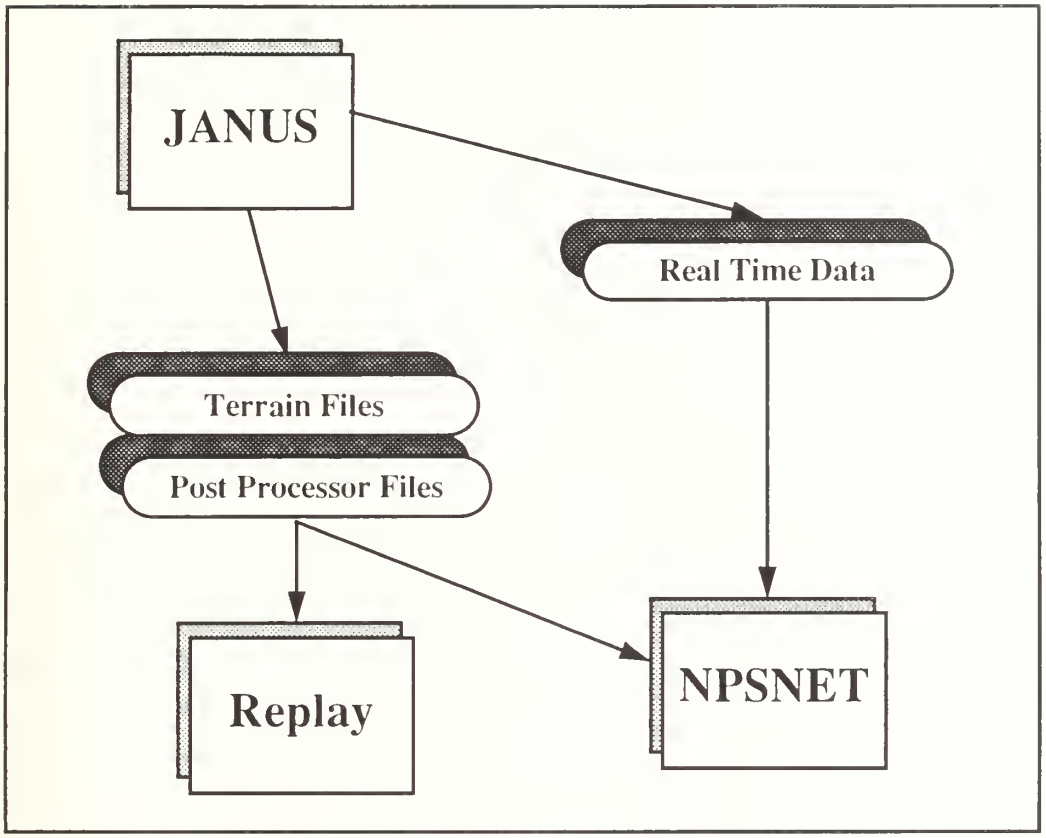

Figure 3. NPSNET/JANUS3D

terrain required hard coding the file paths and recompiling the code prior to executing the conversion programs. The data files and terrain paths for NPSNET were also hard coded with the program having to be recompiled prior to execution. This single use of the code requires several copies of the executable to run the various scenarios.

CPT Smith succeed in reading in preprocessor files into NPSNET, modifying the deployment files, creating more realistic movement paths, then running the modified files in Janus. His work did not address the issues involved in displaying the modified scenario in NPSNET while the user made changes as the scenario executed.

The desirable program is one that is more robust. The focus of this research is on developing a program that does not require recompling the code for every scenario or the hard coding of the terrain and data file paths.

III. JANUS3-D VISUALIZER OVERVIEW

A. INTRODUCTION

The Janus-3D Visualizer provides a three dimensional perspective of the Janus combat modeler. The user can fly through the battlefield to watch the battle progress from any vantage point, or select individual vehicles and view the battle from their vantage point. The program is written in ANSI C and was designed to run on Silicon Graphics Incorporated (SGI) graphics workstations operating IRIX 5.2. For this research, the Janus-3D Visualizer was implemented on a SGI Indigo 2 Extreme graphics workstation.

The program consists of three major components: the terrain conversion programs, communication software, and the virtual environment display. Figure 4 shows the directory structure. The programs that make up the visualizer are located in the visualizer directory with the supporting programs and files in the eight subdirectories. The headers directory contains header files for the programs in the visualizer directory and the global variables for the visualizer. The libs, rdobj3, and imagesupport directories contain the programs needed to manipulate the three dimensional objects (models). The three dimensional models are in the models directory. The datafiles directory contains files used to initalize the information panel, 2D icons and 3D models. The programs used to network the visualizer are located in the src directory. Finally, the terrain conversion programs and the terrain files are stored in the terrain directory

B. TERRAIN CONVERSION OVERVIEW

Janus produces a two dimensional terrain map displaying rivers, roads, urban areas, and vegetation with elevation portrayed using contour lines. This information is stored in the TERAINxxx.DAT file with the xxx denoting the three digit terrain number. We had to convert this information into formats that are readable by the Janus-3D Visualizer. The conversion processes requires seventeen steps to produce three types of files. Refer to Appendix A for more information on the conversion process.

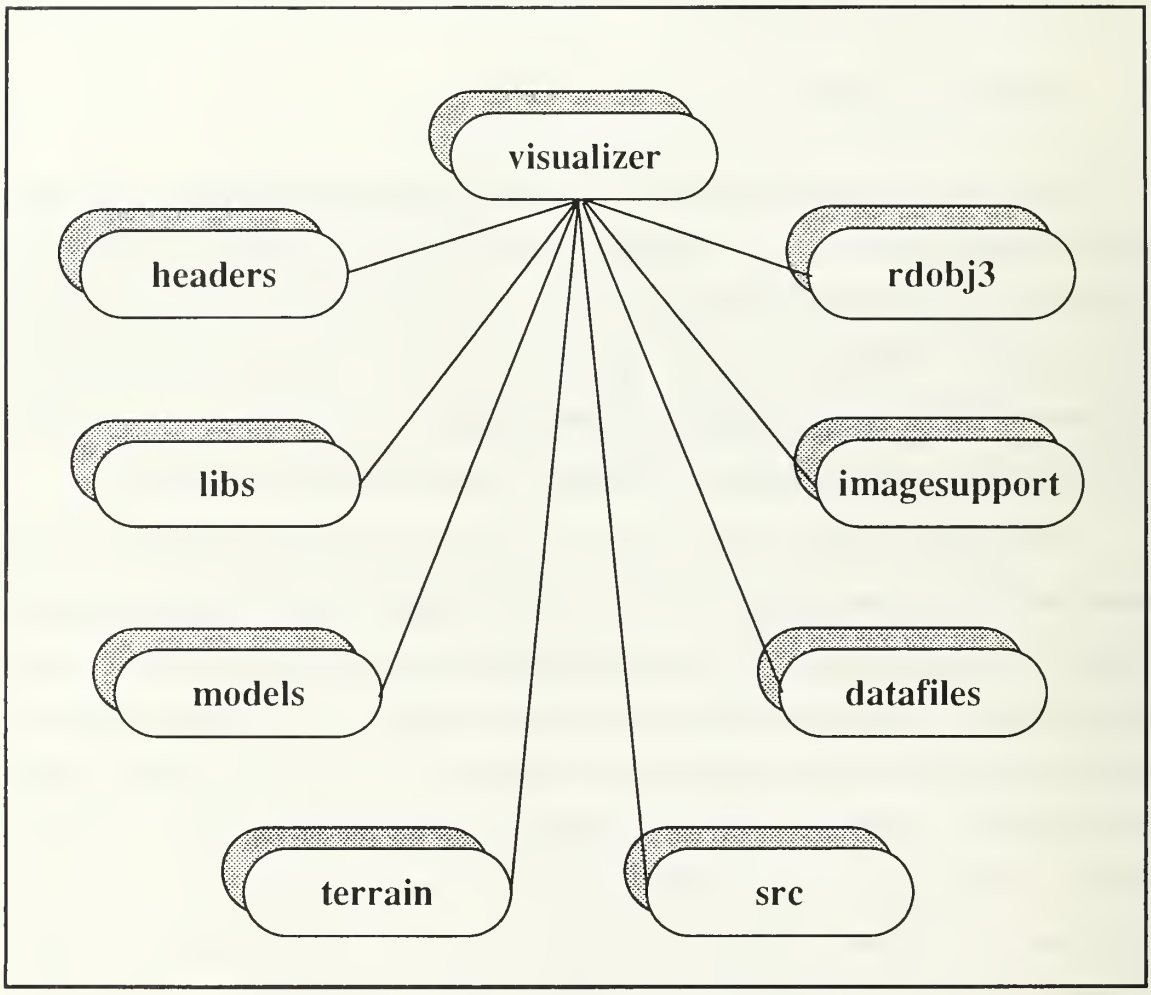

Figure 4. Janus-3D Visualizer Directory Structure

The gray scaling of the elevation posts is used to create the two dimensional display. The gray scaling gives the user a sense of elevation without the use of contour lines. Black denotes the low lying terrain and white denotes higher elevation.

The three dimensional terrain is created by using one of the two remaining file formats. The mesh is constructed by using the GL graphics function to draw t-meshes. Each point for the mesh terrain is stored in a structure that contains the elevation, color, and normals. [SGI90]

The second format is the most complicated and displays the most detail. For this, the terrain is divided into 1 Km grid square files. Each file is then stored in a quad tree

structure containing four levels of detail. These files are then swapped in and out of memory according to the users location. A detailed description of the terrain conversion and file structures is in Chapter IV.

C. VISUALIZER OVERVIEW

The purpose of the visualizer is to display a Janus scenario in real-time or from a scripted file of a previous scenario. The Visualizer's screen layout was designed to mimic the Janus combat modeler's screen. The transition from one system to the other is less confusing with the information being displayed in similar locations. As such, the Visualizer is composed of three windows: a two dimensional map window, a control window, and the large viewing window. This window displays a three dimensional view as opposed to the two dimensional map displayed in Janus. The user can interact with the Visualizer through the use of a keyboard and a mouse. Figures 5 and 6 show the two screens.

The Visualizer's two dimensional map is in the lower right hand side of the screen. This gray scale map indicates the locations of the Janus units. The two dimensional icons are drawn on the map in their respective locations in the Janus battlefield. The circle in the center of the screen is the user's location. The single line extending from the center of the circle is the velocity vector and indicates the user's speed and direction. The triangle is the field of view for the user and represents the area drawn in the three dimensional window. The blue lines are drawn every kilometer horizontally and vertically. The lines are based on the upper left hand corner of the map window and not the UTM coordinate system. The map scale can be varied by using the scale buttons in the control panel. The scaling of the map allows the user to pick an individual icon more easily. The user can select a vehicle by placing the courser on the two icon and clicking the left mouse button. This will move the user to that location and the view presented in the large window will be the view the selected vehicle is seeing. The user's view point will move with the tethered vehicle. To untether click the middle mouse button and the user can move around the virtual word.

The control panel is positioned above the two dimensional map. This panel contains buttons and user information. The information displayed includes the current terrain loaded

in the Visualizer and which script file is running, if any. The current location of the user is displayed and, if tethered, the selected vehicle's data is displayed. By using the buttons the user can call another location, run a script file, pause the script, or terminate the logging of scripts. Users can also turn the display of vegetation and urban areas on and off, list the keyboard and mouse functions with the help button and exit the program.

The last and largest window is the three dimensional display. The user can view the battle in three dimensions from a selected vehicle or from any desired vantage point around the battlefield using the stealth mode.

Figure 5. Janus (3.17) Screen

Figure 6. Janus-3D Visualizer Screen

D. COMMUNICATIONS OVERVIEW

The communications architecture for the Janus-3D Visualizer consists of two networks. The first is the ethernet local area network (LAN), which connects the Visualizer to the Janus Combat Modeler. The second is the wide area network (WAN). On this network, a Janus-3D Visualizer at one location can connect to another Visualizer at a remote location via commercial telephone lines. The communications networks are illustrated in Figure 7. [UPSO94]

The LAN is composed of the Hewlett Packard workstation running the Janus scenario, five to twenty Sun Classic workstations used for the controller/player work stations, and the SGI Indigo 2 Extreme where the Janus-3D Visualizer resides. The HP broadcasts unit movement, direct fire, indirect fire and detonation PDUs across the ethernet to the SGI. The Visualizer reads the PDUs off of the network, stores them in a script file and displays the information on the screen. This allows the user to view the Janus scenario running in real-time in the three dimensional environment.

The WAN consists of only the SGI workstations. The user at one location, brigade or battalion, can call another Janus-3D Visualizer at another location and establish a connection over the telephone line. A SGI workstation can only connect to one other SGI workstation at a time using the WAN. Once the connection is established, the unit that was called transmits the same types of PDUs it extracts from its LAN to the calling unit's SGI workstation. This network allows the user to see the battle's progress from one of its sister or parent unit's perspective as well as their own view point.

For more detail on the software communications architecture and modifications made to the Janus code refer to *Design and Implementation of a Software Communication Architecture for the JANUS-3D Visualizer* written by CPT Upson.[UPSO94]

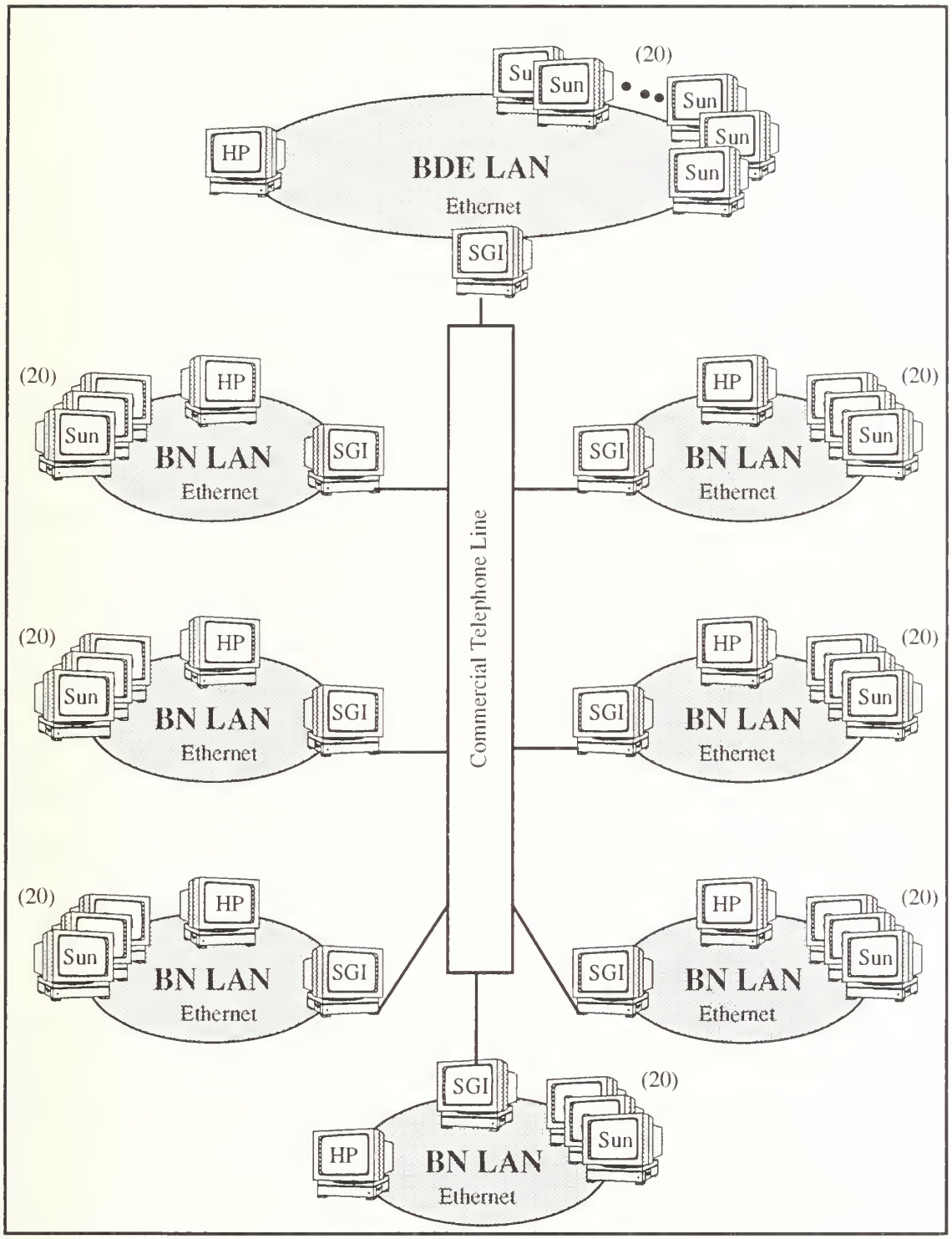

Figure 7. Janus-3D Visualizer Communications Network

IV. TERRAIN

A. JANUS TERRAIN FILES

In order to be able to convert the Janus terrain files, we must first understand how they are used and how they are stored. There are three types of terrain files in Janus; MASTERxxx.DAT, TERAINxxx.DAT, and TSCRNxxx.DAT. [JANU93d]

1. MASTERxxx.DAT

The MASTERxxx.DAT file is created by TRAC from DTED data. It contains the elevation points, river and road networks, vegetation, and urban data for large tracts of land. This file is too large to be displayed in Janus. TRNFLTR is used to create smaller pieces of terrain. To do this, the user inputs the lower left hand X and Y UTM coordinates and the dimensions of the terrain desired. Then the program constructs the file TERAINxxx.DAT.

2. TERAINxxx.DAT

This file is created by TRNFLTR. It contains the elevation points, river and road networks, vegetation, and urban data for a small area. TERAINxxx.DAT is used by Janus when creating a scenario. The file can be modified in TED to add new terrain features, or change existing features.

3. TSCRNxxx.DAT

TSCRNxxx.DAT is created by TED, which reads in the TERAINxxx.DAT file. The user can then modify the river and road networks, the vegetation and urban densities and the interval distance for the contour lines. This file contains bridge, city, road, river, and tree information for the same area contained in TERAINxxx.DAT.

For the purpose of this research, the TERAINxxx.DAT file is manipulated since it is used to display the map for the Janus scenarios. With the information stored in TERAINxxx.DAT, we can create three dimensional terrain, road and river networks. We can also build urban and tree canopies.

B. TERAINxxx.DAT File Format

TERAINxxx.DAT is stored as a binary file in two blocks. The first, and smaller, of the blocks contains the header information. This record is made up of 48,120 bytes. The second block contains the elevation and point data. The size of this block varies with the size of the map and the spacing of the elevation postings. The first and last four bytes of the two blocks contain the number of bytes in the record and can be discarded when retrieving the information. For a complete description of TERAINxxx.DAT, see Appendix B.

The specific information in the header block, in order, is:

1. The lower left X and Y Universal Transverse Mercator (UTM) grid coordinates.

2. The width and height of the map in kilometers.

3. The number of grids wide and height of the map.

4. Movement factors for wheeled vehicles, tracked vehicles, and personnel on foot, first in urban areas and then in areas of vegetation.

5. Height factors for vegetation and urban areas.

6. Probability of lines of sight for vegetation and urban areas.

The last section of the header block is the river and road arrays. They consist of two three dimensional arrays. The first and second fields are the X and Y UTM coordinates of the river or road. The last field contains a zero if the coordinate is the end point of a river or road, a one if it is a continuation of the previous node, and a negative one if it is the last coordinate in the array.

The second block contains the elevation and grid information which includes: the elevation, concealment, density, trafficability, terrain roughness, and flags to indicate rivers, obstacles, fire, chemicals, smoke, and high explosives. Figure 8 shows the breakdown of the word into bits and what information they represent.

.

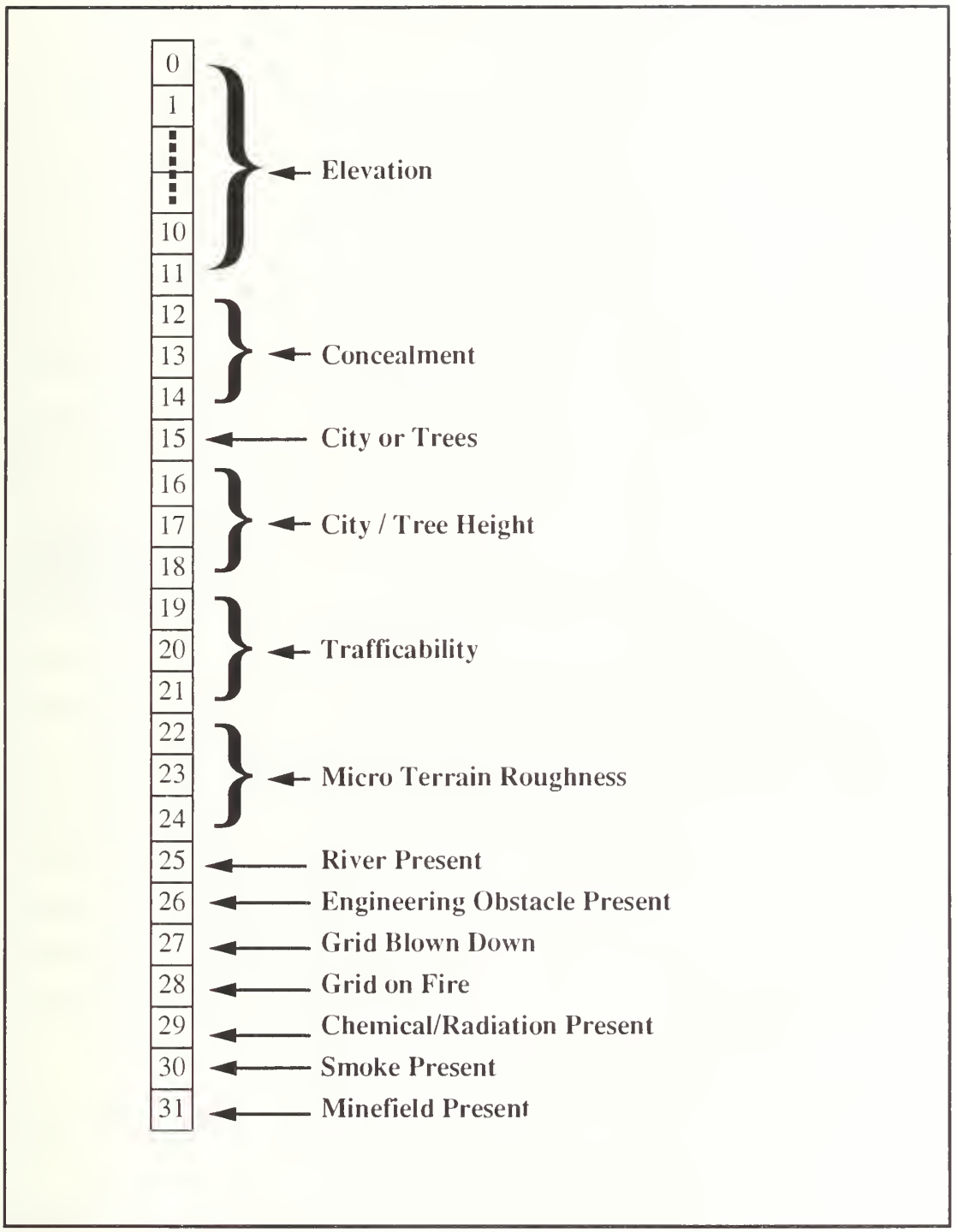

Figure 8. Elevation and Grid Data Word

C. VISUALIZER TERRAIN FILE FORMATS

The terrain displayed in the Visualizer is based on previous work conducted at the Naval Postgraduate School. Three types of terrain are used in NPSNET. The first gray scale elevation, is used for the two dimensional map. Three dimensional map can be shown in either mesh format or polygonal format stored in a quadtree structure in order to allow for several levels of detail. The terrain file paths were hard coded into the program. In order to change the terrain the user had to change the paths and recompile the program. One of the goals of this research was to produce a robust system that could display any terrain the user wanted. To accomplish this, we had to come up with a directory structure that would place the converted files into a terrain unique location in order for them to be displayed in the Visualizer based on a command line option.[MACK91][WALT92]

The terrain directory structure we designed centralized all of the terrain related files and programs into the terrain directory as shown in Figure 9. The janusfiles directory contains the TERAINxxx.DAT files to be converted. All the terrain conversion programs are located in the makeground directory. The coverfiles, elevfiles, roadrivfiles, textcoverfiles, textobjectfiles, and textquadfiles directories store the temporary files needed while converting the terrain into formats readable by the Visualizer. Once the conversion has been completed and it is verified that the new terrain can be displayed, these files can be removed. The remaining directories contain the converted terrain. They are named according to the name of the terrain files they store. The subdirectories in these directories contain the terrain files, vegetation coverage files, script files and the terrain parameters.

D. TERRAIN CONVERSION

1. Janus Terrain File Breakdown

The first step in converting the Janus terrain file is to break it down into usable sections and create the directory architecture to store the terrain. The program readtrrn.c performs this task. It takes as inputs the three digit Janus terrain number and the name of the new directory to be created and produces the terrain specific files directory as pictured

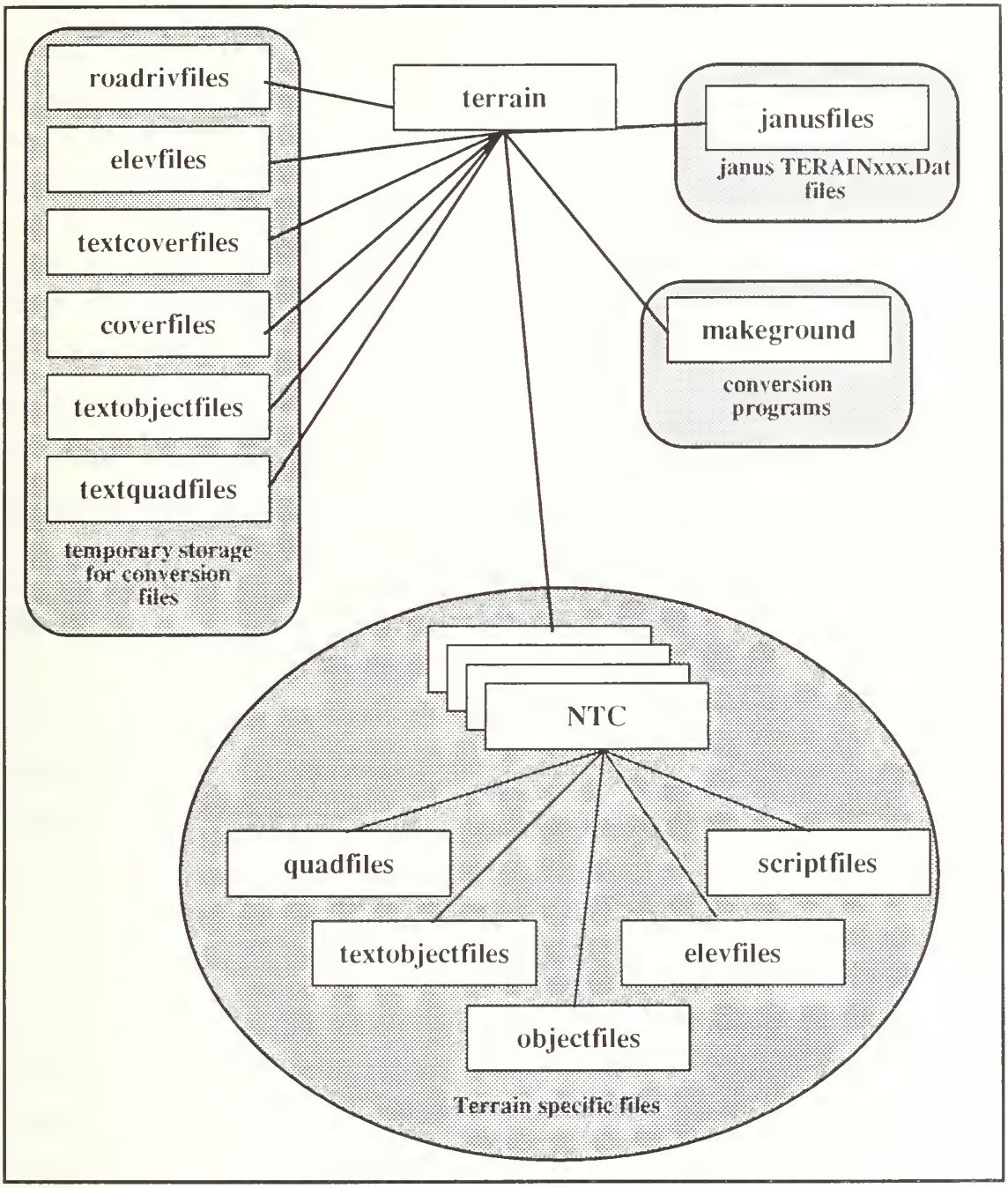

Figure 9. Terrain Directory Structure

in Figure 9. The program splits the data into four files: globals.dat, xxx.riv, xxx.road, and xxx.ele. Globals.dat contains the map information to include the lower left coordinates, size of the map, number of elevation points, multiplication factors for urban and vegetation

areas, and the spacing of the points. This file is read by all the subsequent conversion programs to initialize the global variables. The xxx.riv and xxx.road files contain the coordinates of the map's rivers and roads. The Elevation and Grid Data Word, Figure 8, is dissected and stored in this file.

2. Janus to Visualizer Coordinate Conversion

The Visualizer and Janus use different local coordinate systems to display the terrain. Figure 10 visually depicts these differences. Janus stores the elevations in a sequential file with the first point representing the lower left hand corner of the map. The remaining points are stored from left to right and from the bottom to the top of the map. The last point in the sequential file is the elevation of the upper right hand corner of the map. This sequential storage allows the points to be referenced by a two dimensional array with (0,0) as is the index to the first point. Janus uses the X axis to traverse the map horizontally and Y to traverse the map vertically. The distance between the points is determined by the "GAP" which is stored in the header file. This also gives us the number of points in each horizontal row and vertical column of the array. Janus then uses Equation 1 and Equation 2 to translate the UTM coordinates of the map location for the terrain and vehicles to a local coordinate system based on the array indexes.

$$X_{JanusLocal} = \left(\frac{X_{JanusLocation} - X_{UTMLowerLeft}}{GAP} \right) \qquad \text{Eq 1}$$

$$Y_{JanusLocal} = \left(\frac{Y_{UTMLowerLeft} - Y_{JanusLocation}}{GAP} \right) \qquad \text{Eq 2}$$

The Visualizer uses the upper left hand corner of the map for its index to the array and traverses the array from left to right and from top to bottom. The Visualizer also uses the Z index to traverse the map from top to bottom. The Y axis is used to indicate elevation. To read the map elevation points into the Visualizer we use Equation 3.

$$Z_{VisualizerLocal} = MapHeight - Y_{JanusLocal} \qquad \text{Eq 3}$$

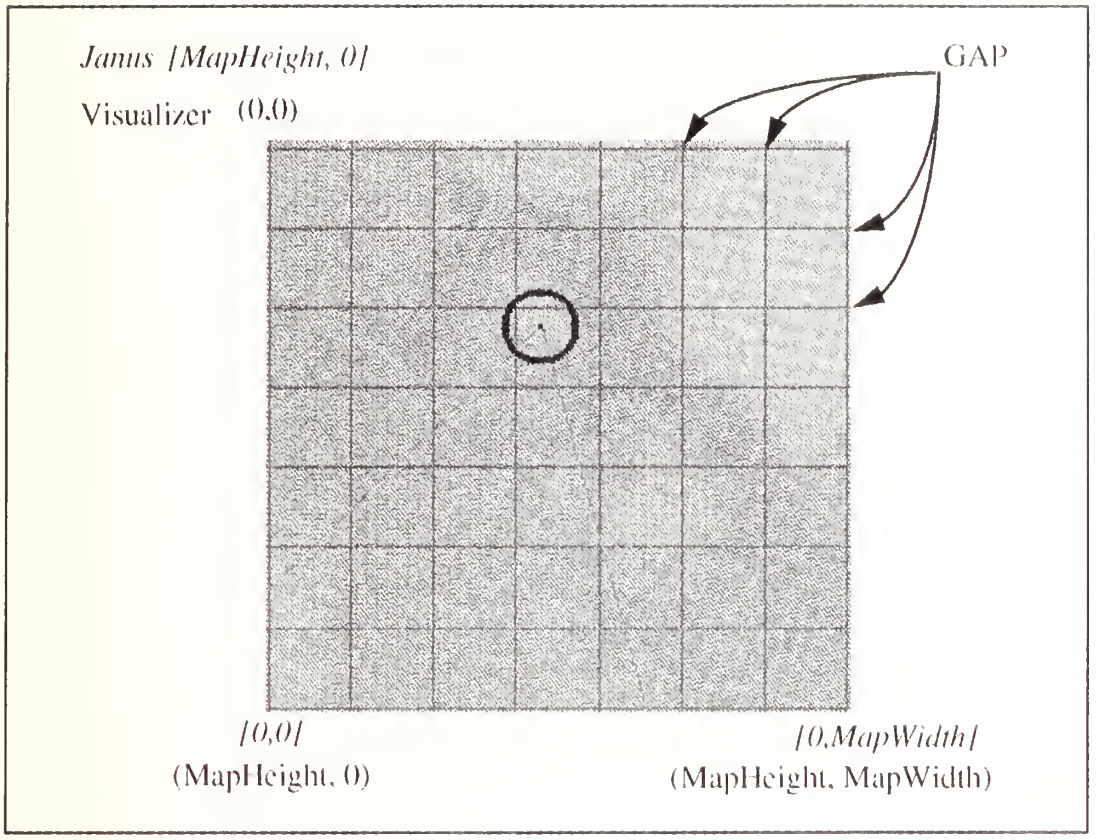

Figure 10. Visualizer and Janus Coordinate Systems

3. Two Dimensional Map Generation

To create the two dimensional map, we first write the terrain elevation points from the file xxx.ele to the file elev.bin.dat. The points are stored in this file using the Visulaizer's coordinate system, Figure 10. They are stored as binary to save on memory. When we display the map, we use a gray-scale. To create the various shades of grey, we use Equation 4 [PRAT90]. This gives us an accurate representation of the elevations with the shade of grey getting lighter as the elevation increases.

$$COLOR = \left(ELEVATION_{MIN} + 255 \times \left(\frac{ELEVATION - ELEVATION_{MIN}}{ELEVATION_{MAX} - ELEVATION_{MIN}} \right) \right) \qquad \text{Eq 4}$$

4. Generating Mesh Terrain

The mesh terrain is simple to store and display. However we cannot display the roads and rivers on the terrain without significant tearing [MACK91]. As mentioned in Chapter III, we use the GL t-mesh function to build the mesh [SGI90]. Each point in the array is used to create one of the vertices in the mesh triangle. The algorithm traverses the points from left to right and from the bottom to the top of the map filling in the mesh triangles. When the last coordinate of the row is reached, the algorithm moves to the next row and starts the process over. Figure 11 is a graphical representation of this process.

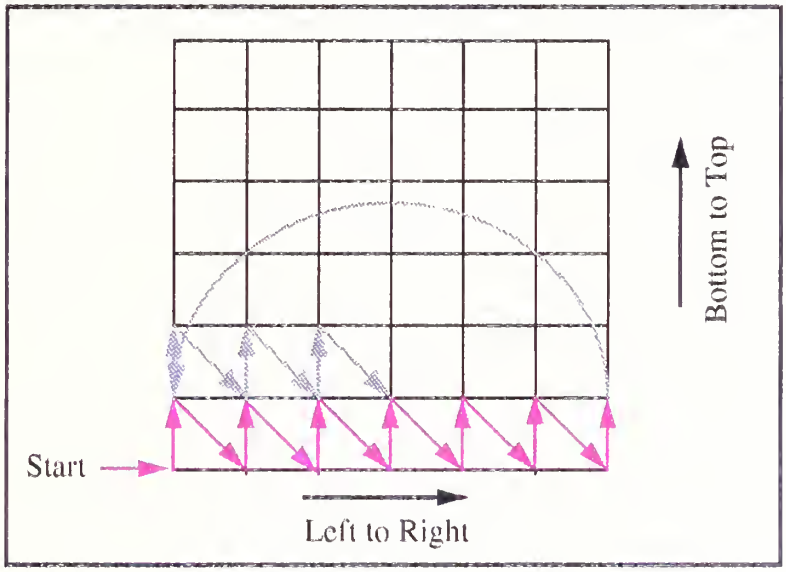

Figure 11. Tri-Mesh Traversal Pattern

5. Generation of Polygonized Terrain

The polygonized terrain is drawn by using two triangles to represent a grid node. The data for each grid node is stored in a quadtree giving us four levels of resolution. The resolution of the terrain to be drawn is based on the distance from the viewer. The highest resolution being drawn close to the viewer and the lowest resolution being drawn farthest from the viewer. This reduces the number of polygons required to render a scene and increases the frame rate. The storage of the points is based on the length of the side of the triangle. The lengths of the sides are shown in Table 1.

Level of Detail	Length of Side
HIGH	GAP
HIGH-MEDIUM	GAP * 2
LOW-MEDIUM	GAP * 4
LOW	GAP * 8

Table 1. Resolutions and Side Lengths

To draw the terrain using the lowest level of detail we need eight GAP lengths (nine elevation points) to define a side of the triangle, therefor each quadtree will consist of eight-one elevation points (nine by nine). When we convert the terrain using readtrrn.c, if Equation 5 is not satisfied for the terrain length and width we create the extra points giving them the elevation of the lower left corner of the map and store them in the quadtree.

$$INTEGER = \left(\frac{MapHightorWidth}{GAP \times 8} \right) \qquad \text{Eq 5}$$

The quadtrees are built using the program genblockcov.c while conv_blockcov_2bin.c stores them as binary values. The grid nodes are stored in individual files coverXXXXZZZZ.bin.dat. The Xs and Zs are used by the Visualizer to index the quadtree when rendering that piece of the terrain to the screen.

The last step is to add the rivers and roads. As discussed previously, Janus stores them as a collection of points. We first create polygons with a width of 10 meters from these points, using the program makeroads.c. The rivers and roads are then stored in the appropriate quadtree file. They will only be rendered when the highest level of detail for that piece of terrain is drawn. To see the rivers and roads on the three dimensional terrain requires the terrain to be drawn twice with the river or road decaled on top of the terrain. This requires extra CPU time and slows down the frame rate. [MACK91]

6. Generation of City and Tree Canopies

The information required to draw the city and tree canopies is located in the file xxx.ele. To draw a canopy we use the t-mesh function again based on the elevation points.

This allows us to use two triangles to draw the canopy for a grid node. The height of the canopy is determined by the height factor specified by each elevation point.[WALT92]

If a river or a road is located in the grid node, then a simple canopy can not be drawn since it would hide the feature and would not look realistic. In this case, the trees or houses are placed on the grid randomly. The number is determined by the height factor assigned to the grid. Once they are placed, they are checked to see if they intersect a river or road. If they do, they are moved to another location. These verification algorithms were written by CPT Warren and CPT Walter. [WALT92]

V. VISUAL DISPLAY

In this chapter, we discuss the design of the Visualizer's display screens and the functioning of the buttons and keyboard. We describe the work performed to initialize the Janus-3D visualizer and how we display the Janus scenario.

A. SCREEN LAYOUT

As discussed in Chapter III the Janus-3D Visualizer uses three windows to render the Janus scenario. The design of the windows was to allow smooth transition from the Janus program running on the HP 715/50 to the Janus-3D Visualizer running on the SGI Indigo 2 Extreme. The user can determine what vehicle or area they wish to see in three dimensions while looking at the Janus scenario on the HP and quickly look at that vehicle/ area on the Visualizer's two dimensional map. With simple mouse and keyboard strokes it will be displayed on the three dimensional screen.

1. Three Dimensional Window

The three dimensional window is the largest of the three windows. This window displays a fourty-five degree field of view (FOV) from the user's location. The user can see the layout of the terrain in three dimensions to include the vegetation and urban areas. The vehicles, rivers and roads in the user's FOV are also displayed on the window.

2. Two Dimensional Map Window

The map is rendered on the screen using grey scaling described in Chapter IV. The horizontal and vertical lines are drawn to indicate one kilometer grid squares. Upon initialization of the Visualizer the entire map data base is displayed in the window. Due to the number of two dimensional icons that can be drawn and the proximity of the icons, we have given the user the ability to change the scale of the map. This feature is similar to the zoom function in Janus. The user can display the entire map, a quarter of the map, one eight of the map, and one-twentyfifth of the map. In selecting the latter three scales the area of

the display is centered on the users location. As the user moves around the battlefield, the map drawn will shift to keep the users location in the middle of the screen.

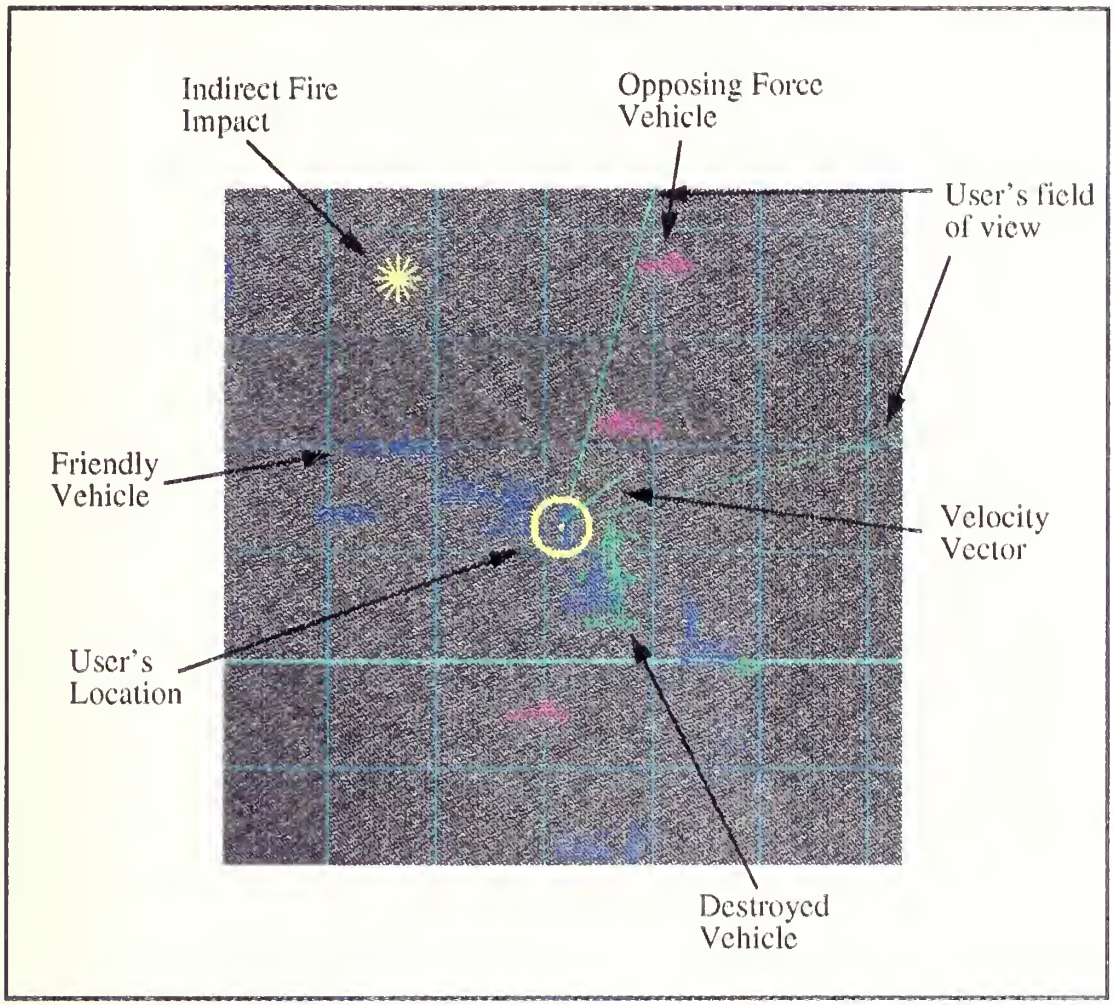

Figure 12. Janus-3D Visualizer Two Dimensional Map

Figure 12 is and example of what the user sees in the two dimensional map window. The user is indicated by a yellow circle. The green triangle is the user's FOV, this is the area displayed in the three dimensional window. The green line extending from the center of the circle is the velocity vector and indicates the direction and speed of the user. The two dimensional icons on the map are rendered blue for friendly forces, red for opposing forces, and green for destroyed vehicles. The two dimensional icon can be pictures similar to the

weapon system they represent or numeric. The numbers range from one to six hundred in red and blue. The numbers are based on the index number from the Janus scenario. The last information displayed on this map is the location of the indirect fire impacts. This is done by flashing a yellow star on the coordinates of the impact.

3. Control Panel

The control Panel is pictured in Figure 13. This window provides the user visually the status of the program configuration and a graphical interface to change the status. We used GL calls to draw the buttons. The default/off color is light blue; when selected the color is changed to green. To select a button the user places the cursor in the box and presses the left mouse button. The Visualizer determines the X and Y pixel locations. If it is with in the bounds of a button, the program executes the indicated command. The buttons are used to call other units, a connection can only be established if there is another Janus-3D Visualizer running at that location. The buttons are also used to change the two dimensional map scale, read script files, display objects, and exit the program. The text information displayed in the panel is the terrain data base displayed, the script file running and the position and vehicle information of the vehicle selected.

B. KEYBOARD

The keys to control the user's motions in the three dimensional environment are located on the right side of the keyboard, Figure 14. These keys were selected to keep the keys in one location. This location allowed us make three groups: Vehicle movement, view direction, and elevation. We used the most logical set key for each of the groups. The arrow keys are used to change the direction and speed of the stealth vehicle. The end key stops the vehicle. The elevation is controlled by the page up and page down keys. The user's view direction is changed by using the pad arrow keys to look left, right, up, or down. The pad 5 key resets the view offset to the stealth vehicle or tethered vehicles direction of travel.

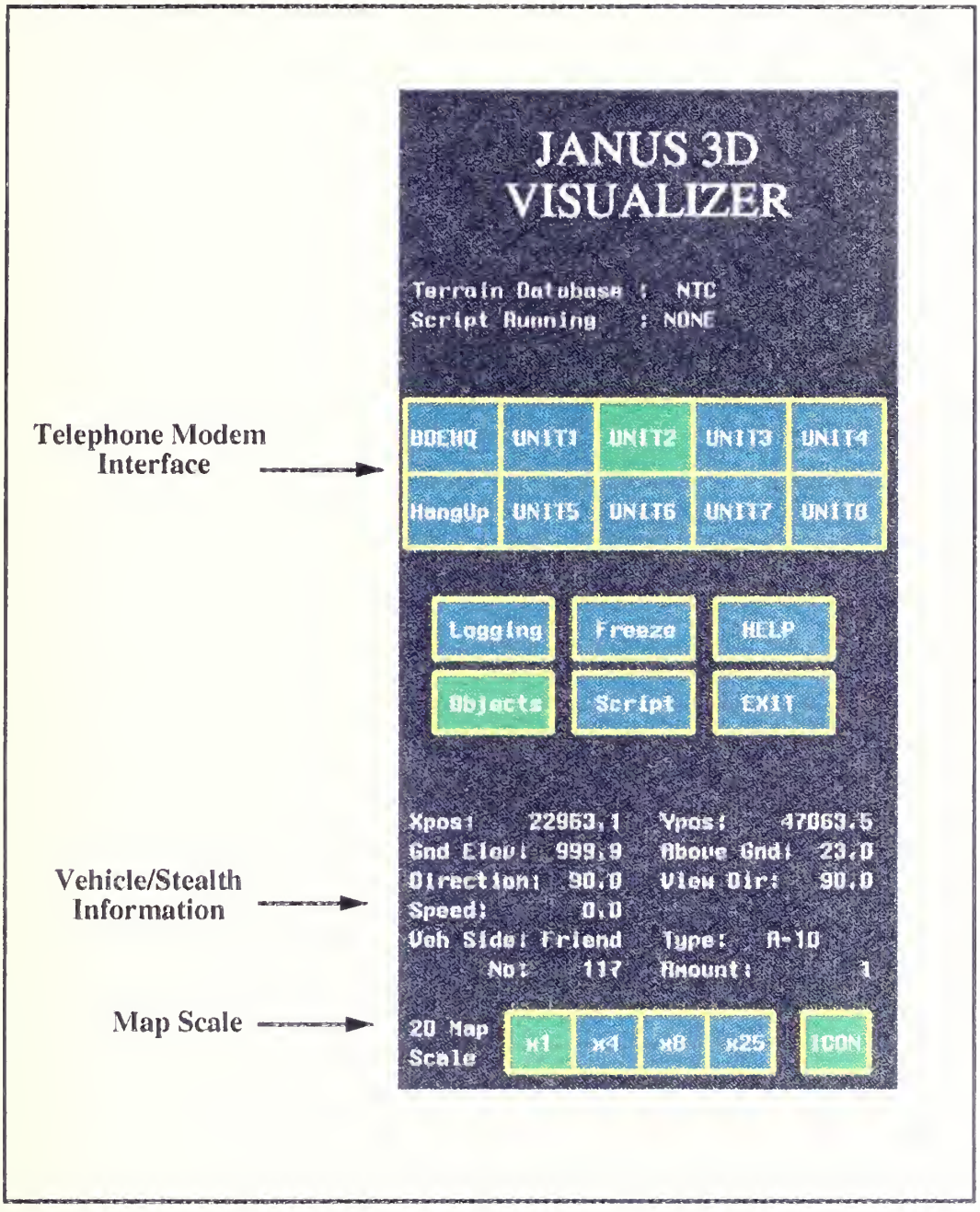

Figure 13. Janus-3D Visualizer Control Panel

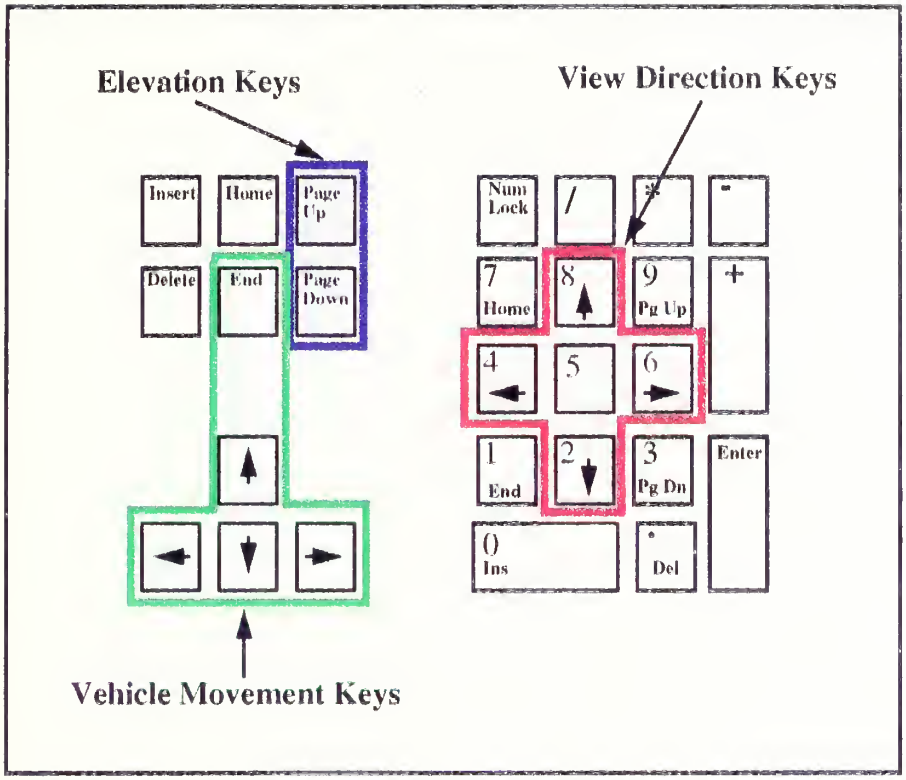

Figure 14. Keyboard Command Keys

C. INITALIZATION OF THE VISUALIZER

In order for the Visualizer to display a Janus scenario we must read in the terrain, setup the vehicles, and initalize the ethernet and modem communication. To start the Visualizer, the user enters the name of the terrain file to be displayed. The program uses that name to build the paths to the terrain files, object files, script files, and the global variables associated with that particular terrain. We used the directory structure created by the terrain conversion process to access these files. By building the paths in this way we have eliminated the need for hard coding the scenario specific information. Once the terrain paths are created we can load the terrain, vegetation and urban files into the system.

We then look at displaying the correct two dimensional and three dimensional model for the Janus entities in the scenario. To accomplish this we use the system type table

extracted from the National Guard master data base. This database contains the system names, graphics symbol number, and vehicle parameters. The vehicle parameters include the range, speed, weapon type, and detectability. This database is managed by the system administrator. The Janus combat modeler uses this database to determine system capabilities and the graphic symbol for every entity in each scenario. From this database, we extracted the system name and graphic symbol from this database to build our Janus to Visualizer vehicle database, janusveh.dat. Table 2 shows an extract from janusveh.dat. Refer to Appendix C for the complete file. Janusveh.dat consists of six fields. The first determines the side: one is friendly vehicles, two is opposing vehicles. The second field is the sequence number. Field three is the Janus name of the vehicle. This name is displayed on the control panel when the vehicle is selected. The fourth field is the Janus graphic symbol number. The fifth and sixth fields are the indices into the visualizer's vehtypearrays which indicate the three dimensional model and two dimensional icon that must be drawn. Upon initialization, janusveh.dat is read into two arrays; friendyvehtypearray for the friendly vehicles and enemyvehtypearray for the hostile vehicles. The janusveh.dat file has to be updated whenever the system type table is modified. Otherwise, the vehicles in the Janus scenario will not be portrayed correctly in the Visualizer.

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Model
1	1	AH64/HEL	1	33	26
1	95	AVENGER	16	97	10
2	131	T-72/MIS	76	8	51

Table 2. Janusveh.dat (sample portion)

The last step in the initialization of the visualizer is to set up the network and modem communications. A function call is made that sets up a shared arena and establishes a connection with the local ethernet. Within this function, a subprocess is spawned to read the PDUs sent by the Janus combat modeler. Next, the modem is configured and another subprocess is spawned that listens for a connection from a remote caller. Both of these processes run concurrently with the main Visualizer process. [UPSO94]

D. VISUALIZER MAIN APPLICATION LOOP

The Visualizer's main application loop continuously executes until it is terminated by the user. Figure 15 shows the flow of execution through the loop. This is how we display the Janus scenario in the Visualizer.

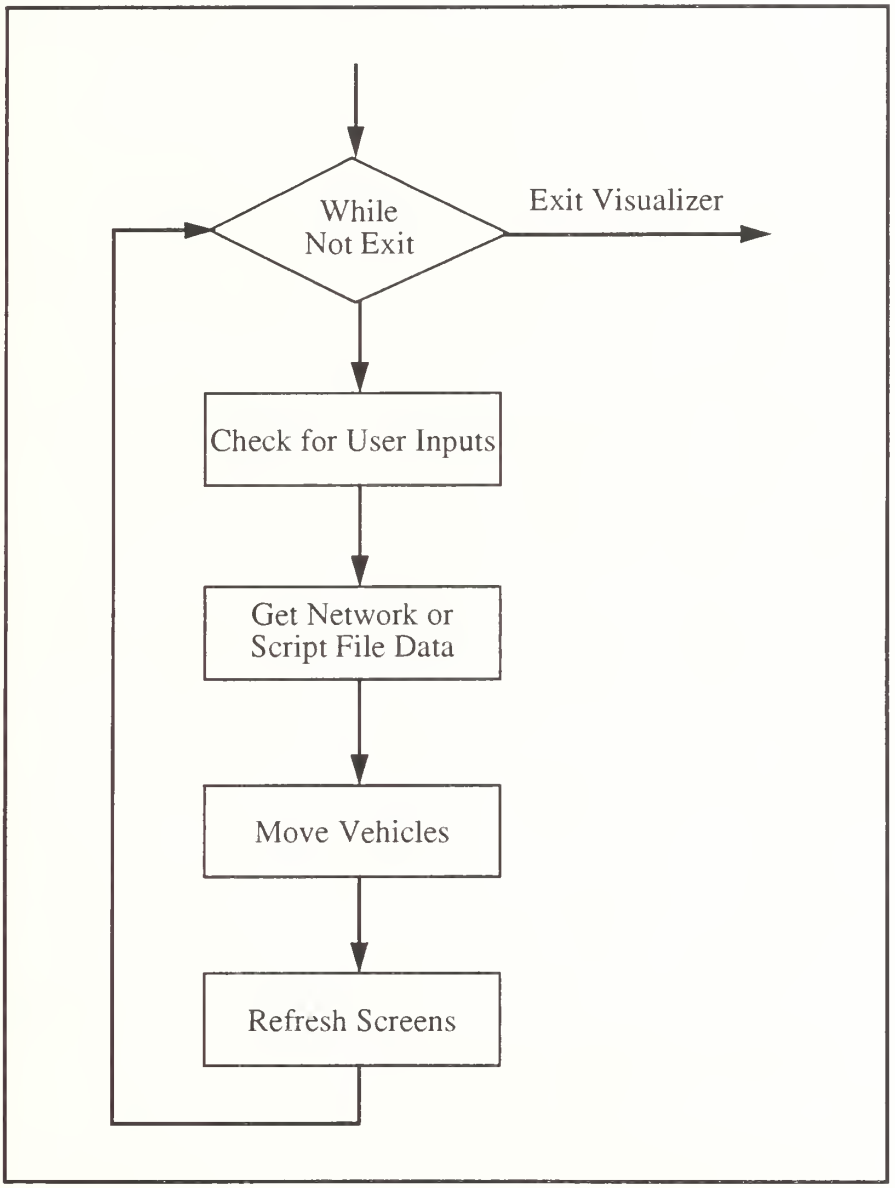

Figure 15. Visualizer Main Application Loop

1. Check for User Inputs

After the initialization is complete the Visualizer checks the event queue to see if the user has input a key or mouse command. These commands can be; from the keyboard to move the user's vehicle or change the view direction, from the two dimensional map, to switch vehicles, or from the control panel to execute one of the button commands. If there is something in the queue the program will process that event then continue down the loop.

2. Check Network for PDUs or Read Scripted File

The Visualizer can receive input from one of the following sources; ethernet, modem connection, or scripted file. Limiting the input to one source eliminates the possibility of displaying a vehicle in several locations because the sources may be running the scenario at different times or may be executing different scenarios.

The format for the PDUs and script file events are the same. There are four type of events that we execute: movement, direct fire, indirect fire, and detonation.

a. Movement

The movement message gives us the Janus vehicle number, direction of travel, view direction, speed, the janus graphics symbol, the number of weapon systems this entity represents, and the local Janus X and Y map location. We take this information and store it in the Visualizer's vehicle array. The Janus vehicle number is the index into the array. If the number is greater than six hundred; the vehicle is hostile. We use the Janus graphics symbol number to index the enemyvehtypearray. This gives us the Janus name, three dimensional model, and two dimensional icon. Janus vehicle numbers less than six hundred we use the friendlyvehtyparray to obtain this information. We load this information and the remaining information into the vehicle array.

b. Direct Fire

The direct fire message gives us the Janus vehicle number or the firer and the target. We use these numbers to index the Visualizer vehicle array to get the X and Z

locations of both vehicles. Using the arctan function we can determine the direction to turn the firers weapon. We also put this information into the shotarray so that is will drawn in the three dimensional window.

c. Indirect Fire

The indirect fire gives us the firer and the Janus X and Y location that they are firing toward. With this information we can determine the direction to turn the firer's weapon and show a muzzle flash for that system in the three dimensional terrain.

d. Detonation

The detnotation message gives us the firer and the Janus X and Y location where the round hit the ground. We show the information on the two dimensional map with a star and in the three dimensional terrain with an explosion.

3. Move the Vehicles

To move the vehicles in the Visualizer, we use two functions: movethejeep for the user's vehicle and movetheundrivenveh for the Janus vehicles. Both functions use first order dead reckoning to determine the vehicle's new position. This is done by calculating the change in time since we last updated the vehicle's position. Next, using the vehicle's direction of travel, speed, and current position, we calculate the new position. Since we use the user's vehicle position to determine what is displayed in the three dimensional window, in movethejeep we also construct the viewing point. This allows the three dimensional display to move with the user.

4. Update the Display

The last step in the main loop is to redraw the screens. We determine the terrain to draw in the three dimensional window based on the user's location and viewing point. Next we determine which vehicles are dead. This is done by looking at the amount of entities the Janus vehicle represents, if that amount is zero and we have not already destroyed that vehicle, we kill it. We then place all the vehicles in the user's field of view on the terrain.

With this done we now draw the indirect fire, detonation and direct fire information. As mentioned earlier for the direct fire we show a muzzle flash from the firer and for the detonation we display and explosion at the detonation's location. To display a direct fire we draw a line from the firer to the target vehicle. If the firer is a friendly vehicle we color the line blue, the line is red if the firer is hostile. This completes the three dimensional window. We then update the information for the control panel and draw that window. For the two dimensional map we determine the scale and area of the map to draw, then draw it and place the two dimensional icons on the map. With all the windows updated we start the main application loop over again.

VI. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

A. CONCLUSION

The objective of this research was to create a three dimensional tool to give a commander another view of the battlefield for use in training their staff and subordinate commanders. The tool had to be networked to allow the respective units to remain at their home station and conduct the training. The previous work in this area showed that it was possible to create a three dimensional display for Janus. However, the system was not robust and required several copies of the executable code to run the various scenarios.

Through this research we were able to create a directory structure and programs to convert the Janus terrain so that the files were placed in unique locations. This removed the necessity of hard coding the file paths allowing the terrain conversion programs and the virtual environment to operate independently. The Janus terrain file can be converted at one unit. The files and other Janus files can be distributed to all the units in the Brigade. After placing the terrain files in the appropriate location in the terrain directory, the brigade could run a Janus scenario using a common terrain. We were also able to design a way to display the Janus scenarios without having to convert the preprocessor files before executing them using the Janus-3D Visualizer to view them in three dimensions. Here again we eliminated the requirement for hard coding the file paths. We were also able to produce a user friendly interface to the modem communications and to the program itself. As a result of this research the Janus-3D Visualizer is currently being fielded in the Army National Guard.

B. TOPICS FOR FUTURE RESEARCH

Several topics for further study can be derived from this investigation. All of them are related to either improving the visual display or to add more functionality to the Janus-3D Visualizer. These modifications will further enhance the realism of the simulation.

The Janus combat modeler allows the user to display operations overlays. The addition of this feature to the three dimensional environment will allow the user to see the boundaries and control measures and use this to help plan offensive and defensive

operations. The commander and staff can fly around the terrain and determine the choke points for movement and the dead space for weapon systems prior to deploying their forces.

The addition of temporal effects would add to the realism of the Visualizer's display. The commander could see the battle as it progressed during the night or during a snow storm. All battles are not fought during the day with good visibility as currently portrayed by the program.

Currently the user is only an observer, the next step is to allow the user to interact with Janus, thus becoming a player. The insights gained in weapon's deployment by viewing the battle in three dimensions could quickly be changed in the Visualizer. The effects of the change could be seen in Janus and used by analysis and war-fighters to test new weapons systems and tactics.

JANUS 3D VISUALIZER

(USER's HANDBOOK)

Naval Postgraduate School

Advisor: Dr. David Pratt

Written by: Chris Upson and Jim Vaglia

Directory Structure and Program Files

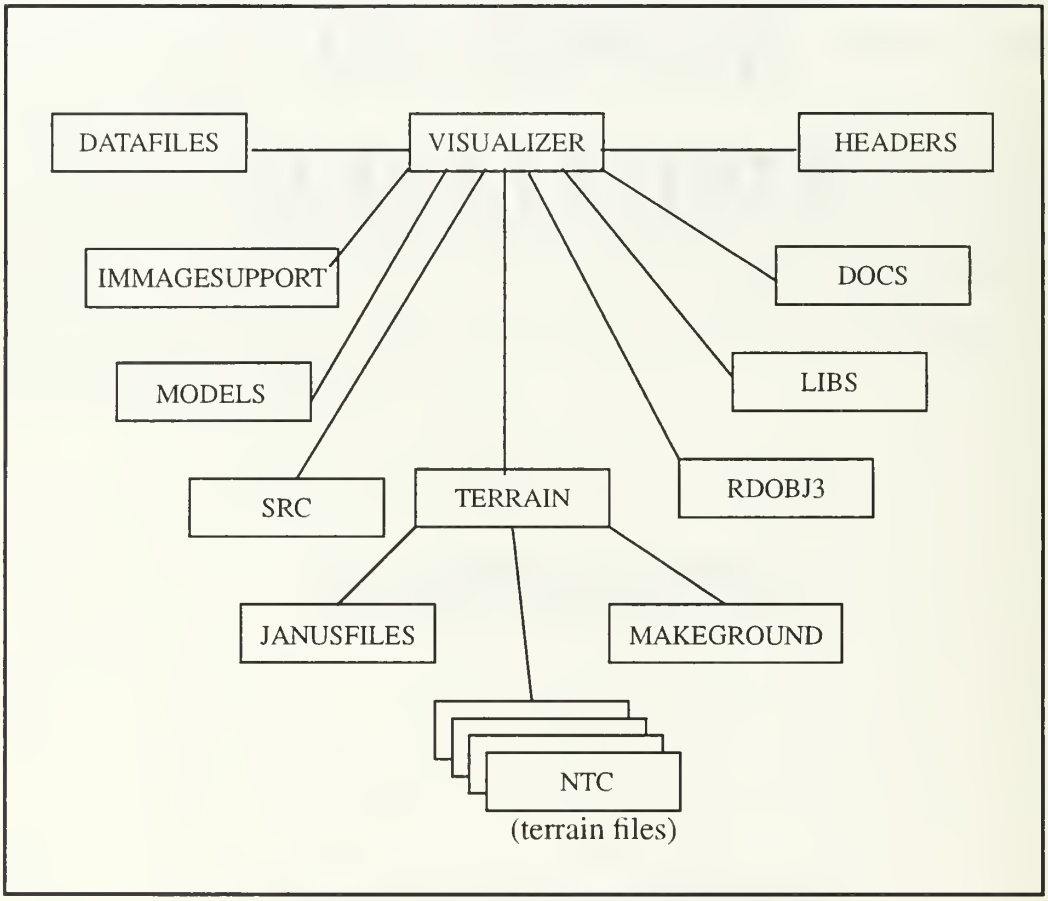

Figure A-1. Directory Structure

The program is broken down into the following directories and files.

Figure A-1 shows the directory structure of the Janus-3D Visualizer.

visualizer The main program files are located here.

datafiles This directory contains the input files.

headers The header files for all the programs are located in this directory.

imagesupport	This directory contains the imaging programs.
libs	The header files for the manipulation of 'off' objects.
models	The three dimensional models are located in this directory.
rdobj3	The files to manipulate the 'off' objects are in this directory.
src	The communication programs are in this directory.
terrain	The terrain files, script, and terrain conversion programs are located in this directory.
docs	Contains Frame Maker postscript and text versions of the user's handbook and briefing slides.

The following programs are located in the visualizer directory.

acds.font	This contains the fonts used to produce the icons for the vehicles on the two dimensional map.
checkintersect.c	This file contains the routines for terrain intersections.
cover.c	This file contains the routines to open terrain files.
dogsncats.c	This file contains the functions used to switch from one vehicle to another and define the cursor.
drawcover3d.c	This file contains the routines to draw the 3d terrain.
drawobjs.c	This file contains the routines to draw the 3d objects.
fontdef.c	This file is used to manipulate the fonts.
infopannel.c	This file contains the display routines for the information panel
janus3d	This is the executable code for the visualizer.
jeep.c	This is the main program. It creates the display window and ties all of the other programs together.
jeepmot.c	This contains the procedures to move the vehicles.

map.c	This draws the two dimensional map and the icons that are displayed on it.
menus.c	This file contains the information that is displayed when the popup menus are selected.
modem.c	This file contains the routines for modem communications.
network.c	This file contains the routines to read the communication packets.
readfiles.c	This file contains the procedures to read in the terrain and vehicle data files.
readobjs.c	This file contains the routines to read the objects.
util.c	This file contains functions to fill in the polygons and place vehicles on the ground.
viewbounds.c	This file contains the procedures to display the three dimensional objects and terrain on the screen.

Installation Procedures

To install the Janus-3d Visualizer, load the file **visualizer.tar.Z** in the directory you want to install the program. Once loaded, uncompress the file by typing **uncompress visualizer.tar.Z [enter]**. When this is finished, untar the files. This will separate the program into the individual files and subdirectories. The command to do this is **tar -xvof visualizer.tar[enter]**. You are now ready to run the program in the default mode.

Communications Setup

1. Ethernet Network Setup

The interface with the local area ethernet network is set up and maintained by the network library in the **visualizer/src** directory. The **net_open** call just prior to the beginning of the main application loop in **jeep.c** establishes this interface. Ensure the value of **BCAST_INTERF** as defined in **headers/jeep.h** is correct for your network. You can check for your system's network interfaces by using the "netstat -rn" command. Also check

the port definitions for the send and receive ports that are listed as **UDP_SEND_PORT** and **UDP_RECV_PORT**. If you change any definitions in **jeep.h**, ensure you recompile both the **src** and **visualizer** directories.

2. Modem Setup

These modem setup procedures apply to the US Robotics Sportster 9600 modem that was used in the design of the Janus-3D Visualizer prototype. If you have a different model modem, please consult its user's guide where appropriate to ensure these procedures will result in proper setup.

All modem processes, to include opening, transferring data and closing, are contained in the file **modem.c** in the **visualizer** directory. Its header file, **modem.h** is in the **headers** directory. The modem interface is established with the **modem_open** call immediately following **net_open** in the Visualizer initialization process.

Once you have connected your modem to the system, determine what serial port it is connected to. Then check this with the **MODEM_PORT** definition in **modem.h**. Our default is port "2". If yours is different, change the **MODEM_PORT** definition and recompile the **visualizer** directory. Also, even though default settings are loaded into the modem upon its initialization within the Visualizer, check the dip switches on the back of the modem to ensure that 1, 2, 5 and 6 are up and 3, 4, 7 and 8 are down. Pages B-4 and B-5 in the user's guide contain more detailed informations on the dip switch settings. Finally, once the Visualizer has completed its initialization process and is ready to run, the Auto Answer (AA), Data Terminal Ready (DTR) and Clear to Send (CS) lights on the front panel should be illuminated.

Terrain Conversion

To convert the Janus TERRAINxxx.DAT into files that are readable by the Visualizer follow these instructions. There are seventeen steps in the process. These steps need to be executed in the order presented. The conversion process takes awhile, suggest you run the

programs running in the back ground. To run a program in the background type & after the command and prior the pressing enter.

The janus TERAINxxx.DAT needs to be placed in the janusfiles directory. You need to ensure that the temporary storage directories in the terrain directory are empty, some of the conversion files append to existing files, this will cause erroneous data to be stored in the files. Then execute the following steps to convert the terrain:

1. readtrrn <terrain #><terrain name>

This program reads the TERAINxxx.DAT located in the janusfiles directory.First the program uses the terrain name to create the root directory for the header files, terrain files and script files needed in the conversion process and the Janus-3D visualizer. The subdirectories created in the terrain directory are: elevfiles, objectfiles, quadfiles, scriptfiles and textobjectfiles. Readtrrn creates five files. The files globals.dat and janus.text are placed in the terrain directory. Globals.dat contains the map parameters and is used by the other conversion programs and Janus-3D Visualizer to initiate the global variables. Janus.text contains the same information but with the text names of the variables. The other three files are placed in the janusfiles directory. The file xxx.ele contains the map elevation and grid information. The remaining files; xxx.riv and xxx.road, contain the coordinates of the rivers and roads respectively.

2. gen_binary_elev <terrain #><terrain name>

The program reads in globals.dat and xxx.ele. Next the program creates the file elev.bin.dat which contains only elevation data and places them in the sub-terrain directory elevfiles.

3. make_tri_mesh <terrain #><terrain name>

The program reads in globals.dat and the elev.bin.dat file that was created in step three. Elev.mesh.bin is created containing the terrain mesh information and is placed in the same directory as elev.bin.dat.

4. conv_elev2block_bin <terrain #><terrain name>

Conv_elev2block_bin reads in globals.dat and elev.bin.dat and creates one kilometer grid square files. For a 50 Km by 50 Km map the program creates 2500 files and stores them in the elevfiles directory.

5. janus2nps <terrain #><terrain name>

Janus2nps reads globals.dat and xxx.ele files. The program then creates and places the file cover.dat in directory textcoverfiles. Cover.dat contains the elevation, normal, and colors of the points.

6. reverseroads <terrain #><terrain name>

Janus reads the map information from the lower left hand corner. NPSNET bases the location of objects on the upper left hand corner. This program modifies the coordinates of xxx.riv and xxx.road so they can be read into NPSNET terrain. The location of the files is in the roadrivfiles.

7. makeroadfile <terrain #><terrain name>

Makeroadfile reads the globals.dat, xxx.road and xxx.riv files. The program then creates the file roads.dat. This file contains the information and points needed to draw the rivers and roads as polygons.

8. makeroads <terrain #><terrain name>

Makeroads reads globals.dat and xxx.ele files. The program then creates and places the file roadcover.dat in directory roadrivfiles. Roadcover.dat contains the elevation, normal, and colors of the points.

9. makenewtrees <terrain #><terrain name>

This program extracts the density, city or tree, road, and river information from the xxx.ele file. The files treecover.dat and citycover.dat are created and stored in the directory textobjectfiles.

10. maketrees <terrain #><terrain name>

Maketrees reads in the treecorver and citycover files, compares them with xxx.ele to insure that the trees and cities are not on the roads. Then the program creates seven city and seven tree files to store the modified information.

11. genblockcov <terrain #><terrain name>

This program creates one kilometer by one kilometer grid square text files containing polygon descriptions. These files are stored in the directory textcoverfiles.

12. conv_blockcov2bin <terrain #><terrain name>

Conv_blockcov2bin converts the text files created by genblockcov and converts then to binary format. The new files are stored in the coverfiles directory.

13. genquadcov <terrain #><terrain name>

This program reads in the files created by conv_block2bin and places then into a quadtree structure. This files are then stored in the textquadfiles directory.

14. conv_quadcov2bin <terrain #><terrain name>

The textquadfiles are converted into binary format by this program and then are stored in the quadfiles directory located in the terrain specific directory.

15. genblockobj <terrain #><terrain name>

Genblockobj creates the tree and city canopies for the terrain. This files are in text form and placed in the textobject directory.

16. conv_block_obj_to_bin <terrain #><terrain name>

The textobjectfiles created by the program genblockobj are converted into binary format by this program and then are stored in the objectfiles directory located in the terrain specific directory.

How to Use the 3D Visualizer

Getting Started

Prior to running the program for the first time you need to change the file **units.dat** located in the **datafiles** directory. This file contains the names and telephone numbers of the units that can be called via the modem. There can be a maximum of nine units and phone numbers in the file. The names can be a maximum of six characters or letters on a line by itself. The telephone number associated with that unit should be on the following line. The telephone number can consist of a maximum of twenty numbers. e.g.:

199INF

17032212935

To execute the program, you need to be in the **visualizer** directory on a Silicon Graphics machine. At your unix prompt, type **janus3d NTC [enter].** (The terrain name can be substituted by any of the terrains you have in the terrain directory).The initial screen will be displayed with the credits. Note that at the lower center of the screen, information will be displayed as the different data files are read into the program. Once the program is finished loading, the working screen will appear. (See Figure A-2) With the main screen up, start Janus(A) 3.17 running. By starting the visualizer running first, when Janus initializes it's screens all the initial positions of the janus units will be transferred and

displayed. Prior to beginning a scenario, remove all script files from the terrain directory. Otherwise, the new files will be appended to the old files.

There are three main sections to the display: the 3D view, 2D map and the vehicle information panel. The largest area is the 3D view. At initalization you are in the stealth mode. Through the use of the keyboard you can move freely throughout the battlefield. This area will display the world from your reference point. The other option is to be tethered to a vehicle. In this case, the 3D view will be from the vehicle's position.

The blue rectangle is the information panel. The panel contains the buttons to call other units, change the two dimensional map display, read scripted files, and stealth / janus vehicle information. This gives the user a numerical reference to where you are on or above the battlefield, the directions of travel and view, speed, and vehicle orientation, ID number and type. Direction is based on 0 degrees equates to North.

The lower right hand corner is the 2D map. The vehicles are iconized and color coded to make identification easy. The location of the icon is it's location on the battlefield. The line originating from each of the icons is the direction of travel with the length signifying the speed. (Longer lines indicate higher speeds). The yellow circle is your current location, while the green 'V' is the area of the map shown on the 3D display; the field of vision.

Moving in the Visualizer

There are two modes: tethered and stealth. While in the tethered mode, the user can change the viewing direction to the left or right and up or down. The direction and speed of the vehicle are determined by Janus(A) 3.17 running on the Hewlett Packard or a script of a previously run battle scenario. The stealth mode allows the user to move freely throughout the battlefield with all movements determined by the user through the keyboard.

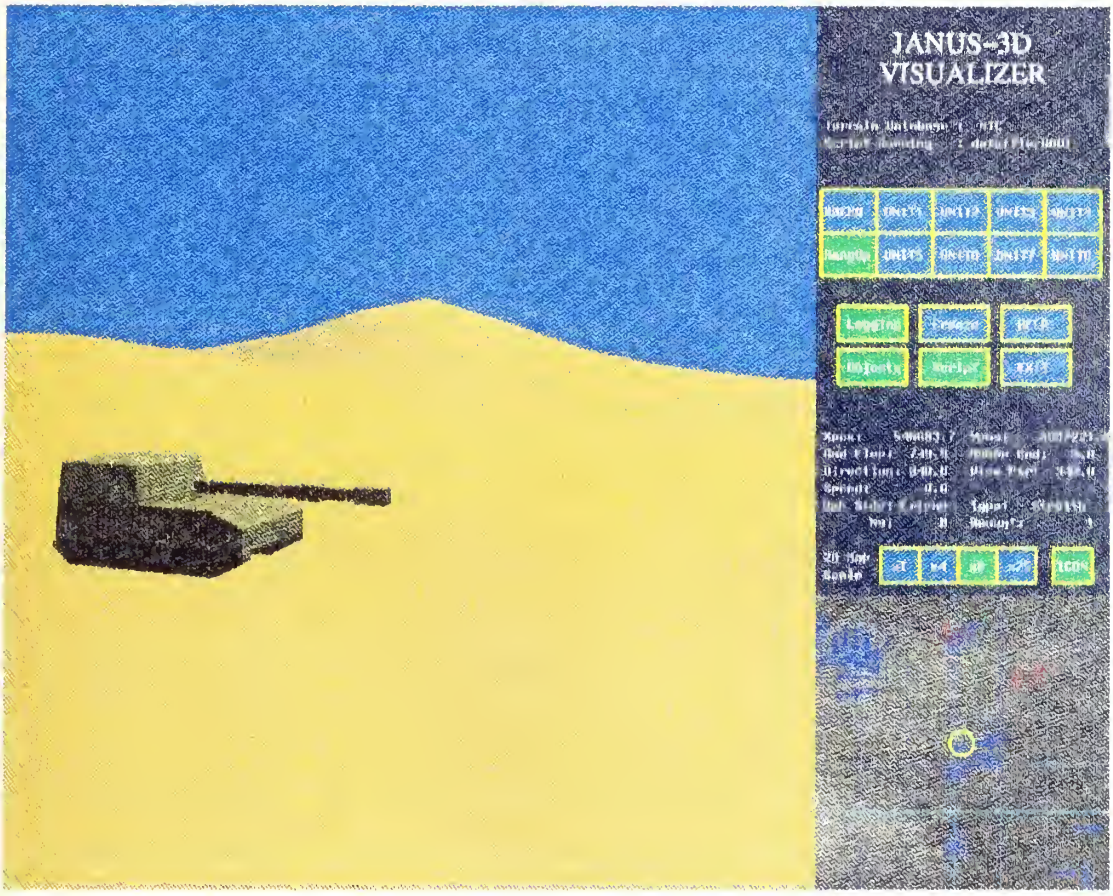

Figure A-2. Main Screen

Using the mouse

Left Mouse Button: Select a vehicle to tether on - Place the mouse cursor on a vehicle's icon or number in the 2D map area and click the left mouse button. The 3D screen will display what the selected vehicle can see in it's current direction of travel.

Middle Mouse Button: Untether- To untether from a vehicle, click the middle mouse button and the user will be in the stealth mode with the same view as from the vehicle that was deselected.

Right Mouse Button: Select Menu - Press the right mouse button while anywhere on the screen and the popup menu will appear. From this menu, you can take a picture of the screen.The image will be stored in the visualizer directory as snapshot#. The # will increase for each image stored during a session.

Using the keyboard

Tethered mode.

The **pad left arrow key** will move the field of view to the left.
The **pad right arrow key** will move the field of view to the right.
The **pad up arrow key** will allow the user to look up.
The **pad down arrow key** will allow the user to look down.
The **pad 5 key** will reset the view to the direction of travel of the tethered vehicle.

Stealth mode.

The **left arrow key** will change the direction of travel to the left.
The **right arrow key** will change the direction of travel to the right.
The **up arrow key** will increase the speed of the stealth vehicle.
The **down arrow key** will decrease the speed of the stealth vehicle.
The **end key** will stop the stealth vehicle.
The **pad left arrow key** will move the field of view to the left.
The **pad right arrow key** will move the field of view to the right.
The **pad up arrow key** will allow the user to look up.
The **pad down arrow key** will allow the user to look down.
The **pad 5 key** will reset the view to the direction of travel of the stealth vehicle.
The **page up key** will increase the elevation of the stealth vehicle.
The **page down key** will decrease the elevation of the stealth vehicle.

Information panel (Figure A-3)

The information panel contains the buttons to interact with the program and displays pertinent information about what is currently occurring in the program. The two lines under the title let the user know what terrain was loaded and, if they are running, a script and which script is running. The other non-interactive section of the panel is the vehicle information. The user is given the x and y grid coordinates of the vehicle, the ground elevation and elevation above ground of the stealth or tethered vehicle above the ground, the direction of travel, direction that the vehicle is looking and the speed of the vehicle are displayed. The last information displayed is the side (Friend or Hostile), the

Janus vehicle number, the Janus name from the master list located in datafiles/janusveh.dat (Type), and the number of systems that the icon and 3d model represent (Amount).

Calling another unit.

To call a unit place the mouse cursor in the box containing the name of the unit you wish to call, then press the left mouse button. The box will turn green and the program will try to establish a connection with that unit. Once a connection is established, current PDUs will immediately be transferred to the calling unit from the remote unit.To terminate the connection, place the mouse cursor in the Hang Up box, press the left mouse button. The Hung Up button will turn green and a box will appear asking if you really want to hang up. Select the OK button to terminate the connection or the No button to hide the box. If the OK button is selected, the box will automatically disappear once the connection has been terminated.

Logging

The logging button default is on (green). This will cause the program to create and store script files in the terrain directory. The program will not create a script file for the script files you are running or for the information displayed from another unit. If you do not want to create script files, move the cursor into the logging box and press the left mouse button. The box will turn blue to indicate that the script files are not being stored.

Freeze

The Freeze button only works when running a script file. This will stop the scripted vehicles from moving while allowing the stealth vehicle to travel around the frozen battlefield. To freeze the scripted file, move the mouse cursor into the Freeze box and press the left mouse button. The box will turn green indicating the script is frozen. To unfreeze, repeat the above procedure and the button will turn blue.

Help

When selected, the help button will display the keyboard inputs to move around the battlefield. To select the help menu, move the mouse cursor into the help box and press

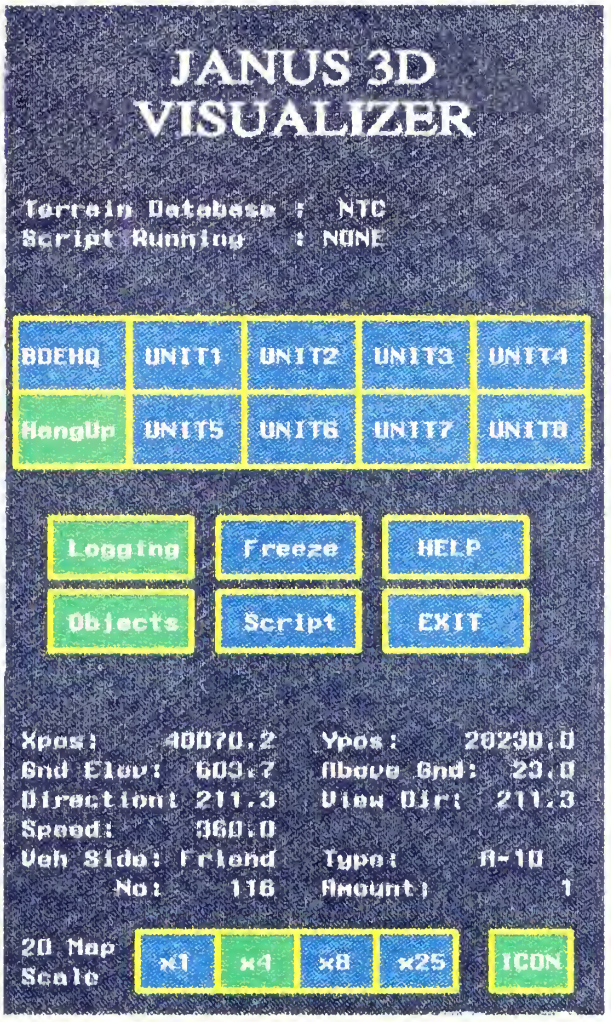

Figure A-3. Control Panel

the left mouse button. This will cause a large box containing the help information to appear in the information panel. When you are done looking at the help information, select the OK button and the box will disappear.

Objects

The Objects button default is on (green). With the objects selected, the trees and urban areas will be displayed. To remove the trees and urban areas move the mouse courser

into the objects box and press the left mouse button. To redisplay the objects, repeat the above procedure and the box will turn green.

Script

To run a script, move the mouse cursor into the script box and press the left mouse button. A box containing a maximum of twenty script files will appear. Move the mouse cursor to the script file you want to run and press the left mouse button. The box will disappear and the name of the script file will appear at the top of the information panel. Each of the script files are twenty minutes long. Before each session, remove the old script file. This will keep the number of files to a minimum.

Exit

To exit the visualizer, move the mouse cursor into the Exit box and press the left mouse button. An exit box will appear and ask if you want to exit the program. Select the appropriate button to terminate the program.

Map Scale

The 2d map is initially set to x1. This displays the entire terrain file. x4,x8, x25 displays one fourth, one-eight, and one-twentyfifth of the map respectively (centered on your location). To change the map scale move the mouse courser into the box containing the desired scale and press the left mouse button.

Icon/Num

This changes the 2d icon display. To change from the default icon setting to the Janus number move the mouse cursor into the icon/numeric box and press the left mouse button. By repeating this process you can toggle between number and icon in the 2dmap.

2D Display

The map gives the user a gray scale elevation representation of the terrain. Black is the low ground and white is the high ground. The grid squares on the map represent one kilometer grid squares, Figure A-4.

The vehicles can be depicted as numerics or as icons. The numbers range from one to six hundred for both forces. The icons / numerics for the friendly forces are blue, the enemy forces are red, and the dead vehicles are green. The user's location is indicated by a yellow circle. The green triangle extending from the circle is the field of view that is displayed in the 3D window.

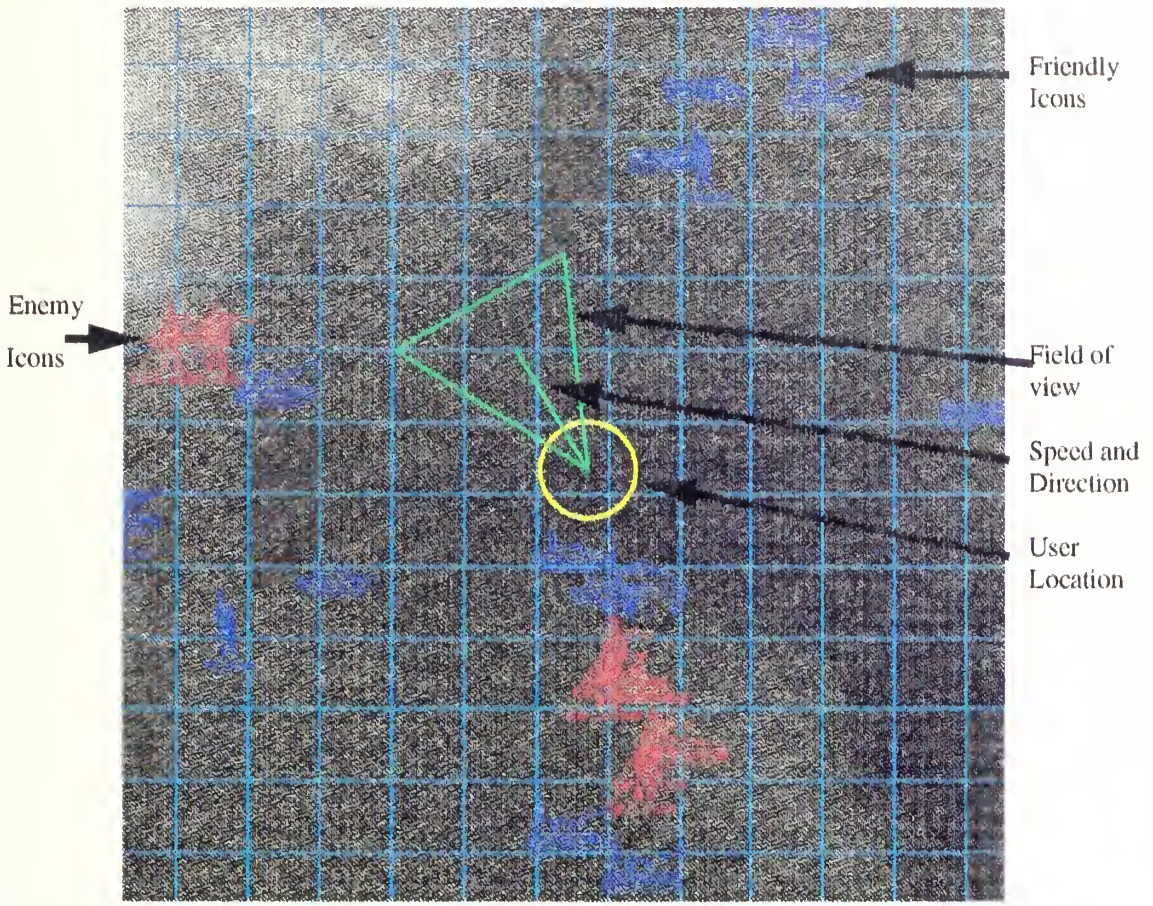

Figure A-4. 2D Map Display

APPENDIX B: JANUS TERAINxxx.DAT FILE FORMAT

This appendix details the file format of the Janus file terrainxxx.dat file. We used this file to create the two and three dimensional terrains in the Janus-3D Visualizer. The file is stored in two blocks. The first block contains the header information. The second contains the elevation posts. The information listed is compiled form the file GLOBTRRN.FOR.

HEADER BLOCK

SIZE	TYPE	VARIABLE	Meaning
four bytes	N/A	N/A	size of block
four bytes	float	XLL	Lower Left X UTM
four bytes	float	YLL	Lower Left Y UTM
four bytes	float	XWIDE	# Km Wide
four bytes	float	YTALL	# Km Tall
four bytes	int	IDIMX	# cells in X direction
four bytes	int	IDIMY	# cells in Y direction
one byte	int	KURMB1,1	wheeled
one byte	int	KURMB1,2	trafficability
one byte	int	KURMB1,3	levels
one byte	int	KURMB1,4	1 thru 7
one byte	int	KURMB1,5	
one byte	int	KURMB1,6	
one byte	int	KURMB1,4	
one byte	int	KURMB2,1	tracked
one byte	int	KURMB2,2	trafficability
one byte	int	KURMB2,3	levels
one byte	int	KURMB2,4	1 thru 7

Table 3. TERRAINxxx.DAT File Format: Header Block

SIZE	TYPE	VARIABLE	Meaning
one byte	int	KURMB2,5	
one byte	int	KURMB2,6	
one byte	int	KURMB2,7	
one byte	int	KURMB3,1	Footed
one byte	int	KURMB3,2	trafficability
one byte	int	KURMB3,3	levels
one byte	int	KURMB3,4	1 thru 7
one byte	int	KURMB3,5	
one byte	int	KURMB3,6	
one byte	int	KURMB3,7	
one byte	int	KVEGM1,1	Wheeled
one byte	int	KVEGM1,2	trafficability
one byte	int	KVEGM1,3	levels
one byte	int	KVEGM1,4	1 thru 7
one byte	int	KVEGM1,5	
one byte	int	KVEGM1,6	
one byte	int	KVEGM1,7	
one byte	int	KVEGM2,1	Tracked
one byte	int	KVEGM2,2	trafficability
one byte	int	KVEGM2,3	levels
one byte	int	KVEGM2,4	1 thru 7
one byte	int	KVEGM2,5	
one byte	int	KVEGM2,6	
one byte	int	KVEGM2,7	
one byte	int	KVEGM3,1	Footed
one byte	int	KVEGM3,2	trafficability
one byte	int	KVEGM3,3	levels
one byte	int	KVEGM3,4	1 thru 7

Table 3. TERRAINxxx.DAT File Format: Header Block

SIZE	TYPE	VARIABLE	Meaning
one byte	int	KVEGM3,5	
one byte	int	KVEGM3,6	
one byte	int	KVEGM3,7	
ten bytes	N/A	N/A	Empty bytes
one byte	int	KHGTS1,1	Vegetation
one byte	int	KHGTS1,2	Heights
one byte	int	KHGTS1,3	levels
one byte	int	KHGTS1,4	1 thru 7
one byte	int	KHGTS1,5	
one byte	int	KHGTS1,6	
one byte	int	KHGTS1,7	
one byte	int	KHGTS2,1	Urban
one byte	int	KHGTS2,2	Heights
one byte	int	KHGTS2,3	levels
one byte	int	KHGTS2,4	1 thru 7
one byte	int	KHGTS2,5	
one byte	int	KHGTS2,6	
one byte	int	KHGTS2,4	
one byte	int	KPLOS1,1	Probability
one byte	int	KPLOS1,2	LOS
one byte	int	KPLOS1,3	Vegetation
one byte	int	KPLOS1,4	
one byte	int	KPLOS1,5	
one byte	int	KPLOS1,6	
one byte	int	KPLOS1,7	
one byte	int	KPLOS2,1	Probability
one byte	int	KPLOS2,2	LOS
one byte	int	KPLOS2,3	Urban

Table 3. TERRAINxxx.DAT File Format: Header Block

SIZE	TYPE	VARIABLE	Meaning
one byte	int	KPLOS2,4	
one byte	int	KPLOS2,5	
one byte	int	KPLOS2,6	
one byte	int	KPLOS2,7	
sixteen bytes	N/A	N/A	empty bytes.

Table 3. TERRAINxxx.DAT File Format: Header Block

The next section contains the river information. The rivers are composed of three four byte float arrays. The first array contains the X coordinate, the second the Y coordinate. The last array contains a negative one, zero, or one. The space reserved for them in the file is 12000 bytes. This is the total derived from 3 arrays 1000 entries and each entry is four bytes. This is the number of bytes you need to read and/or skip before reaching the road array.

The road array is similar to the rivers with the exception that the arrays are 3000 entries long. This therefor is 27000 bytes long. After reading the roads we reach the end of the block. The last four bytes should have the same value as the first four bytes. This is a check to ensure that you are reading the correct number of bytes.

ELEVATION POST BLOCK

The next four bytes contain the number of bytes that we will be reading in the elevation post block. We get this number by adding one to IMIDX and one to IMIDY then multiplying the two numbers. Each of the words are four bytes long. To access the information stored in the words we use the masks in the following table.

Bit location in Word	Data Contained	Number of Bits	Mask
0 -11	Elevation	12	00000FFF
12-14	Concealment	3	00007000
15	City or Tree	1	00008000
16-18	City/Tree Height	3	
19-21	Trafficability	3	00380000
22-24	Micro Terrain Rough-ness	3	01C00000
25	River Present	1	02000000

Table 4. TERRAINXXX.DAT File Elevation Word Format

Bit location in Word	Data Contained	Number of Bits	Mask
26	Engineering Obs Present	1	04000000
27	Grid Blown Down	1	08000000
28	Grid on Fire	1	10000000
29	Chemical/Radiation	1	20000000
30	Smoke Present	1	40000000
31	High Explosive	1	80000000

Table 4. TERRAINXXX.DAT File Elevation Word Format

APPENDIX C: JANUSVEH.DAT

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	1	AH64/HEL	1	33	26
1	2	AH64/LB	62	33	26
1	3	LH/HELLF	59	33	75
1	4	LH/LB	59	33	75
1	5	OH58D	17	29	27
1	6	NONE	0	0	0
1	7	NONE	0	0	0
1	8	105H/TOW	43	4	1
1	9	155H/SP	3	4	0
1	10	155H/TOW	60	4	1
1	11	155_HIP	3	4	1
1	12	AFAS	45	4	4
1	13	NONE	0	0	0
1	14	NONE	0	0	0
1	15	60MM_MRT	27	1	103
1	16	81MM_MRT	27	1	103
1	17	120_MRT	2	1	103
1	18	4.2"_MRT	2	1	103
1	19	81MMIMRT	27	1	103
1	20	NONE	0	0	0
1	21	NONE	0	0	0
1	22	NONE	0	0	0
1	23	8"HOW/SP	0	4	1
1	24	MLRS	5	13	11
1	25	NONE	0	0	0
1	26	NONE	0	0	0
1	27	M548	100	0	10
1	28	TRK/AMMO	30	79	106
1	29	HEMT/AMO	31	79	106

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	30	TRK-8X8	69	0	10
1	31	SEMI-TRL	55	0	10
1	32	NONE	0	0	0
1	33	NONE	0	0	0
1	34	NONE	0	0	0
1	35	NONE	0	0	0
1	36	NONE	0	0	0
1	27	NONE	0	0	0
1	28	NONE	0	0	0
1	39	NONE	0	0	0
1	40	NONE	0	0	0
1	41	NONE	0	0	0
1	42	NONE	0	0	0
1	43	NONE	0	0	0
1	44	NONE	0	0	0
1	45	NONE	0	0	0
1	46	NONE	0	0	0
1	47	NONE	0	0	0
1	48	WRECK	59	33	26
1	49	NONE	0	0	0
1	50	ASP/IF	31	79	106
1	51	M1	6	9	1
1	52	M1A1	6	9	1
1	53	M1A2	6	9	1
1	54	M1A1_HY	6	9	1
1	55	AGS/105	39	9	1
1	56	NONE	0	0	0
1	57	NONE	0	0	0
1	58	FIFI	8	4	2
1	59	M2A2	8	4	2
1	60	M3A2	29	4	2

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	61	M3A1	29	4	2
1	62	NONE	0	9	0
1	63	NONE	0	0	0
1	64	M113/GLD	57	1	3
1	65	M113/TOW	46	1	3
1	66	M113/ENG	7	1	3
1	67	M113/50	7	1	3
1	68	M113/M19	7	1	3
1	69	M113/AMB	45	1	3
1	70	NONE	0	0	0
1	71	NONE	0	0	0
1	72	HMMWV	64	97	10
1	73	HMMV/50	38	97	3
1	74	HMMV/ TOW	33	97	10
1	75	HMMV/M19	44	97	10
1	76	HMMV/M16	64	97	10
1	77	MOTOCYCL	52	10	10
1	78	LAV/TOW	50	4	2
1	79	LAV/25	49	4	2
1	80	HMMV/ AMB	62	97	10
1	81	NONE	0	0	0
1	82	M9_ACE	9	9	1
1	83	AVLB	19	21	1
1	84	CEV	20	9	1
1	85	M88	37	4	1
1	86	NONE	0	0	0
1	87	NONE	0	0	0
1	88	NONE	0	0	0
1	89	NONE	0	0	0

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	90	NONE	0	0	0
1	91	NONE	0	0	0
1	92	NLOS/LGT	53	97	10
1	93	NLOS/HVY	65	1	10
1	94	PIVAD	42	97	10
1	95	AVENGER	16	97	10
1	96	HAWK	61	105	11
1	97	ADATS	55	97	107
1	98	CHAPERAL	40	22	11
1	99	STIG_GNR	25	25	12
1	100	NONE	0	0	0
1	101	NONE	0	0	0
1	102	NONE	0	0	0
1	103	NONE	0	0	0
1	103	NONE	0	0	0
1	105	RIFLEMAN	21	27	12
1	106	MK19_GNR	63	27	12
1	107	SAW_GNR	22	27	12
1	108	M60_GNR	24	27	12
1	109	M203_GNR	36	27	12
1	110	TOW_TEAM	47	27	12
1	111	AAWS-M	23	27	12
1	112	CASUALTY	41	27	12
1	113	NONE	0	0	0
1	113	NONE	0	0	0
1	115	AH-1S	14	31	27
1	116	OH-58C	32	29	27
1	117	RPV	28	44	0
1	118	A-10	14	93	25
1	119	NONE	0	0	0
1	120	FOGM_MIS	13	0	0

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	121	F4	41	52	30
1	122	UH-60	26	31	28
1	123	CH-47C	53	31	28
1	124	NONE	0	0	0
1	125	NONE	0	0	0
1	126	NONE	0	0	0
1	127	SMK_GEN	10	1	3
1	128	M577	11	18	1
1	129	ITV	34	1	3
1	136	LOSAT	54	1	2
1	131	NONE	0	0	0
1	132	NONE	0	0	0
1	134	ASP/AIR	56	36	106
1	134	ASP/DF	31	36	106
1	135	NONE	0	0	0
1	136	NONE	0	0	0
1	137	NONE	0	0	0
1	138	NONE	0	0	0
1	139	TRK/CARG	30	36	105
1	140	TRK/AMMO	30	36	105
1	141	TRK/FUEL	54	36	105
1	142	GAM_GOAT	31	0	0
1	143	HEMT/POL	48	36	105
1	134	HEMT/CAR	31	36	105
1	145	TRK/WATR	58	36	105
1	146	NONE	0	0	0
1	147	MNT_YARD	113	0	0
1	148	AID_STN	39	0	0
1	149	HOLE	51	0	0
1	150	POW_CMPD	35	0	0
1	151	NONE	0	0	0

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
1	152	WRECKER	65	0	0
2	1	60/MORT	118	65	103
2	2	82/MORT	118	65	103
2	3	VASELIK	119	66	103
2	4	120/MORT	119	65	103
2	5	NONE	0	0	0
2	6	NONE	0	0	0
2	7	76MM/GUN	109	66	0
2	8	122HW/SP	97	66	0
2	9	152HW/SP	106	66	0
2	10	152G/TOW	109	66	0
2	11	120GN/SP	97	66	0
2	12	210HW/SP	120	100	0
2	13	NONE	0	0	0
2	14	NONE	0	0	0
2	15	122_MRL	98	58	11
2	16	180_MRL	95	56	11
2	17	220_MRL	95	56	11
2	18	300_MRL	95	58	11
2	19	NONE	0	0	0
2	29	NONE	0	0	0
2	21	NONE	0	0	0
2	22	NONE	0	0	0
2	24	NONE	0	0	0
2	24	NONE	0	0	0
2	25	NONE	0	0	0
2	26	NONE	0	0	0
2	27	NONE	0	0	0
2	26	NONE	0	0	0
2	29	NONE	0	0	0
2	30	NONE	0	0	0

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
2	31	NONE	0	0	0
2	32	HIND-F	71	33	0
2	33	HAVOC	77	33	75
2	34	HOKUM	87	33	75
2	35	HIP	96	99	75
2	36	NONE	0	0	0
2	37	NONE	0	0	0
2	38	NONE	0	0	0
2	39	NONE	0	0	0
2	40	NONE	0	0	0
2	41	NONE	0	0	0
2	42	NONE	0	0	0
2	43	NONE	0	0	0
2	44	NONE	0	0	0
2	45	NONE	0	0	0
2	46	NONE	0	0	0
2	47	NONE	0	0	0
2	48	NONE	0	0	0
2	49	NONE	0	0	0
2	50	ASP/IF	79	36	105
2	51	NONE	0	0	0
2	52	NONE	0	0	0
2	53	BMP1/AT3	111	45	52
2	54	BMP-2	105	45	52
2	55	FBMP-2	121	45	52
2	56	BMP2/FLR	105	45	52
2	57	NONE	0	0	0
2	58	NONE	0	0	0
2	59	BRDM-2	101	117	61
2	60	FBRDM-2	102	117	61
2	61	FBRDM/AT	117	117	61

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
2	62	NONE	0	0	0
2	63	NONE	0	0	0
2	64	BTR-60	99	46	61
2	65	BTR70/80	112	46	61
2	66	FBTR	123	67	61
2	67	FBRM	124	46	61
2	68	NONE	0	0	0
2	69	ACRV	94	46	0
2	70	MTLB	78	96	0
2	71	MTLB/AT	91	96	52
2	72	SCORPION	116	118	60
2	73	CASCAVEL	116	118	60
2	74	MOTRCYCL	66	10	0
2	75	NONE	0	0	0
2	76	ZSU_23-4	73	57	53
2	77	256	115	60	53
2	78	SA-8	103	47	61
2	79	FBMP/AD	122	45	52
2	80	SA-4	89	95	61
2	81	SA-13	75	17	52
2	82	SA18_GNR	67	31	12
2	83	SA-12	90	21	61
2	84	SA18_GNR	67	31	12
2	85	SA-15	84	95	53
2	86	SA-17	88	65	53
2	87	100MM/AD	125	100	0
2	88	ZSU_23-2	83	57	53
2	89	NONE	0	0	0
2	90	NONE	0	0	0
2	91	RIFLEMAN	66	27	12
2	92	AUTORIFL	108	27	12

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
2	93	AT4_GNR	107	27	12
2	94	RPG16_GN	68	27	12
2	95	AGS17_GN	104	27	12
2	96	AAWS_GNR	107	27	12
2	97	SNIPER	66	27	12
2	98	AT5_GNR	107	27	12
2	99	RPG7_GNR	68	27	12
2	100	RPK_GNR	108	27	12
2	101	NONE	0	0	0
2	102	NONE	0	0	[illegible]
2	103	NONE	0	0	0
2	104	NONE	0	0	[illegible]
2	105	NONE	0	0	[illegible]
2	106	NONE	0	0	[illegible]
2	107	NONE	0	[illegible]	0
2	108	NONE	0	0	0
2	109	TRK/CARG	92	36	105
2	110	TRUK/POL	93	36	105
2	111	TRK/AMMO	79	36	105
2	112	NONE	0	0	0
2	113	NONE	0	0	0
2	113	NONE	0	0	0
2	115	NONE	0	0	0
2	116	ASP/DF	79	36	105
2	117	ASP/AIR	82	83	105
2	118	NONE	0	[illegible]	0
2	119	NONE	0	[illegible]	0
2	120	AVLB	80	21	0
2	121	NONE	0	0	0
2	122	NONE	0	0	0
2	123	NONE	0	0	0

Table 5. Janusveh.dat

Side	Sequence	Janus Name	Janus Symbol	2D Icons	3D Models
2	124	FLOGGER	70	26	76
2	125	FROGFOOT	114	26	76
2	126	RPV	72	44	0
2	127	NONE	0	0	0
2	128	NONE	0	0	0
2	129	NONE	0	0	0
2	130	T-62A	73	8	60
2	131	T-72/MIS	76	8	51
2	132	T72+/MIS	76	8	51
2	133	T72/NMIS	76	8	51
2	134	FST-I	110	8	51
2	135	FST-II	110	8	51
2	136	FST-III	110	8	51
2	137	125-SPAT	85	100	0

Table 5. Janusveh.dat

LIST OF REFERENCES

[ARMY93] Department of the Army, *Army Focus 1993*, US Army Publication and Printing Command, The Pentagon, Washington, DC, September 1993

[FUNK94] Funk, Steven, *Information Paper ARPA/ARNG Advanced Technology Demonstration #2 Project SIMITAR*, Ft. Leavenworth, KS

[JANU93] Department of Army, *The Janus 3.X/UNIX Model User's Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[JANU93a] Department of Army, *The Janus 3.X/UNIX Model Data Manager's Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[JANU93b] Department of Army, *The Janus 3.X/UNIX Model Computer System Operator's Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[JANU93c] Department of Army, *The Janus 3.X/UNIX Model System Design Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[JANU93d] Department of Army, *The Janus 3.X/UNIX Model Software Design Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[JANU93e] Department of Army, *The Janus 3.X/UNIX Model Software Programer's Manual*, Headquarters TRADOC Analysis Center, ATRC-ZD, Ft. Leavenworth, KS, May 1993

[MACK91] Macky, Randall L., *NPSNET: Hierarchical Data Structures for Real-Time Three_Dimensional Visual Simulations*, Master's Thesis, Naval Postgraduate School, Monterey,CA, September 1991

[OSBO91] Osborne, William D., *NPSNET: An Accurate Low-Cost Technique for Real-Time Display of Transient Events: Vehicle Collisions, Explosions and Terrain Modifications*, Master's Thesis, Naval Postgraduate School, Monterey,CA, September 1991

[PRAT90] Pratt David R., *"showelev.c"*, computer program1990

[PRAT93] Pratt David R., *A Software Architecture for the Construction and Management of Real-Time Virtual Worlds*, Dissertation, Naval Postgraduate School, Monterey,CA, June 1993

[SGI90] Silicon Graphics Inc., *Graphics Library Reference Manual C Edition Version 4.0,* Silicon Graphics Inc., Mountain View,CA, April 1990

[SMIT93] Smith, Richard Samuel, *NPSNET: Scripting of the Three-Dimensional Interactive Systems for use in the JANUS Combat Simulation,* Naval Postgraduate School, Monterey,CA, September 1993

[UPSO94] Upson, Christopher S., *Design and Implementation of a Software Communication Architecture for the JANUS-3D Visualizer,* Naval Postgraduate School, Monterey,CA, September 1994

[WALT92] Walter, Jon C., and Warren, Patrick T., *NPSNET: Master's Thesis in Computer Science, JANUS-3D Providing Three-Dimensional Displays for a Traditional Combat Model*, Naval Postgraduate School, Monterey,CA, September 1992

[WEST94] West, Togo D. Jr., and Sullivan, Gordon R., *A Statement on the Posture of the United States Army Fiscal Year 1995*, Office of the Chief of Staff, United States Army, Congressional Activities Division (DACS-CAD), Washington DC, Feburary 1994

[ZYDA92] Zyda, Michael J, Pratt, David R, Monahan, Gregory, and Wilson, Kalin P., *NPSNET: Constructing a 3D Virtual World*, Symposium on 3D Graphics, '92 Proceedings, April 1992, pp. 147-156

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Dudley Knox Library Code 052 Naval Postgraduate School Monterey, CA 93943	2
3.	Chairman, Code CS/Lt Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
4.	Professor D. R. Pratt, Code CS/Pr Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	5
5.	Professor G.M. Lundy, Code CS/Ln Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	1
6.	MAJ Tom Allen ARPA-SIMITAR Fort Leavenworth, KS 66027	1
7.	Mr. Don Bennett Cubic Applic Inc. P. O. Box 13548 Fort Carson, CO 80913	1
8.	Mrs. Meg Champion LTSI Box 1825 Richmond Hill, GA 31324	1

9. Mr. Jeffrey K. Skilling 1
BDM Federal Inc.
P. O. Box 908
Fort Knox, KY 40121

10. Director 2
U.S. Army Research Laboratory
ATTN: AMSRL-CI (CPT Vaglia)
APG, MD 21005-5067