12-1997

Requirements, Design and Prototype of a Virtual User Interface for the AFIT Virtual Spaceplane

John M. Lewis

Requirements, Design and Prototype of a Virtual User Interface for the AFIT Virtual Spaceplane

THESIS

John M. Lewis, Captain, USAF

AFIT/GM/ENP/97D-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GM/ENG/97D-02

Requirements, Design and Prototype of a Virtual User Interface for the AFIT Virtual Spaceplane

THESIS

John M. Lewis, Captain, USAF

AFIT/GM/ENP/97D-02

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

AFIT/GM/ENG/97D-02

Requirements, Design and Prototype of a Virtual User Interface for the AFIT Virtual Spaceplane

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Meteorology

John M. Lewis, B.S.

Captain, USAF

December 1997

AFIT/GM/ENG/97D-02

Requirements, Design and Prototype
of a Virtual User Interface for the
AFIT Virtual Spaceplane

John M. Lewis, B.S.

Captain, USAF

Approved:

[redacted] 26 Nov 97

LtCol. Martin Stytz, Phd., USAF — date
Chairman

[redacted] 26Nov97

Maj. Keith Shomper, Phd, USAF — date
Member

[redacted] 26Nov97

Maj. Sheila Banks, Phd., USAF — date
Member

ACKNOWLEDGMENTS

First, I would like to thank LtCol. Stytz for his guidance and imagination during the design of the virtual spaceplane. He always encouraged me, in his own unique way, to look at things from a different perspective. Next I would like to thank Maj. Banks for her guidance and help. Her knowledge in the area of Human-Computer Interaction provided an excellent repository to bounce ideas and offer suggestions. She also helped to keep LtCol. Stytz's imagination within acceptable limits. I would like to thank Maj. Shomper, the third member of my committee, for his support in providing me the fundamentals needed to work in a graphics environment. Many times I found myself reviewing matrix multiplication and coordinate systems remembering what he taught me.

Thanks Dan "the Man" Zambon for all the hard work you did trying to get an HMD. By the way, my wife is glad you enjoyed the Green Bay cup cakes. Also, many thanks to David Doaks. I just want you to know we all appreciate the work you did for us making sure we could get our work done on the weekends. It was also nice to have a smoking buddy that I could shoot the breeze with every now and then.

I would like to thank the many people who had no choice but to make the graphics lab their home during the past year and a half. Their conversations, unique personalities, and suffering provided a great release valve just knowing that I am not alone here.

I would like to thank the rest of the dynamic trio, Lt. Troy Johnson and Lt. Scott Rothermel, for their help and encouragement. Troy, the great debater, always made sure I had another perspective while making my design decisions. Scott, the great programmer, helped me to understand the true meaning of object oriented programming.

Finally, I would like to thank Shanna, my wife, for her love, support, and encouragement during this ordeal. I definitely would not have survived AFIT without her. For two people who spend a lot of time together, this assignment was the biggest test of our marriage and we both passed with flying colors. I know in my heart that this quest has made us stronger and I look forward to a long and wonderful life with her.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The United States Air Force is evaluating the feasibility of designing a military spaceplane capable of accomplishing military objectives from a low earth orbit and atmospheric flight regimes. Current efforts are involved in determining the scientific, operational, and budgetary constraints associated with this concept.

This thesis looks at the exploration of new interface techniques associated with the design of a virtual spaceplane and is a subset of the overall virtual spaceplane effort, which will assist researchers in determining the feasibility of a military spaceplane. Interface techniques are integrated into a virtual user interface that is designed to accommodate expected operations associated with atmospheric and low earth orbit military operations. We expect these operations to include satellite deployment and recovery, reconnaissance, and space station construction and resupply.

The focus of the virtual user interface design effort involves the application and integration of current interface design methodologies and virtual environment technologies to support the functionality of a virtual spaceplane.

CHAPTER ONE – INTRODUCTION

OVERVIEW

The Air Force Space Command (AFSPC) and the Air Force Material Command (AFMC) are evaluating the benefits of using military spaceplanes for a variety of low earth orbit and atmospheric missions. The AFSPC-AFMC Military Space Plane Integrated Concept Team (ICT), directed by AFSPC and AFMC commanders, is conducting this evaluation [DOD97a].

Military space operations are becoming increasingly critical to the Global Engagement capabilities of the Air Force. Achieving safe, reliable, affordable, and routine access to the space environment is becoming an important cornerstone to national security. Current military space launch capabilities, largely based upon upgraded ballistic missiles, can not support the expected increase in military space operations. "What is desired is to rebuild space access with new forms of transportation that embody the aircraft like characteristics of safety, reliability, operability, supportability, producability, testability, and affordability" [DOD97a] Many of the studies previously undertaken in the last forty years indicate that reusable military space planes are expected to achieve these objectives.

THESIS STATEMENT

This thesis will present the design, development, and prototype of a virtual user interface. My primary goal is to create a virtual user interface that will support the mission requirements of a virtual spaceplane capable of flight in both the space and

atmospheric regimes. We will create and evaluate the virtual user interface using as a basis the standards and principles of human-computer interaction. In addition, the virtual user interface will draw upon various interaction techniques associated with virtual environment technology.

SCOPE

Design and implementation of a virtual user interface is the primary goal of this research. We will apply the principles of information oriented, systematic graphic design [Marcus90a], and Discount Usability Engineering [Nielson93] to ensure the virtual user interface complies with industry standards and basic guidelines. We will create an architecture that allows rapid prototyping of the interface components. This architecture will also manage the flow of input and output between the user and the simulation. We will create a set of design specifications to serve as the foundation for interface components and virtual widgets to ensure consistency and standardized interaction techniques throughout the virtual environment. We will then combine, organize, and interconnect these components and virtual widgets to create the overall virtual user interface. We will conduct a heuristic evaluation to ensure the virtual user interface complies with industry standards and basic guidelines appropriate to the virtual spaceplane simulation.

It is not the intent of this research to duplicate existing space travel technology. This research will explore new methods of interaction between the user and the flight controls. It will also explore new methods for reducing the amount of information currently provided to space shuttle crews.

APPROACH/METHODOLOGY

The research to develop a virtual user interface can be decomposed into four parts: interface architecture, interface specifications, implementing the design process, and results.

INTERFACE ARCHITECTURE

The interface architecture is a subset of the overall virtual spaceplane system architecture. Primary concern is the management of input and output. In addition, the interface architecture will manage the flow of information between the virtual user interface and the virtual environment. The goal is to create an interface architecture that has access to all aspects of the virtual environment through a centralized location. This design limits the interconnection between the user interface and the rest of the simulation. Consequently, we can design and prototype interface components without adversely affecting other parts of the simulation.

INTERFACE SPECIFICATIONS

Interface specifications provide explicit rules to guide the development process and insure consistency among the various interface components. The goal here is to create standard interaction techniques that may be reused throughout the overall virtual user interface.

IMPLEMENTING THE DESIGN PROCESS

This area involves bringing the pieces created in accordance with the design specifications together in a logical and organized manner to create the overall virtual user interface. In addition, the design and testing of various prototype components is performed during implementation.

RESULTS

Finally, the results will verify the success of these three previous areas. The goal here is to ensure the architecture, design specifications, and implementation all came together to create a virtual user interface that conforms to industry standards and basic guidelines. These results are determined according to the accomplishment of various design requirements, the outcome of a heuristic evaluation, and the impact the interface has on a user's degree of presence.

THESIS OVERVIEW

The seven chapters and three appendices provide background and a description of the design, development, and prototype of a virtual user interface for the AFIT virtual spaceplane. Chapter Two, Background, introduces the three relevant areas associated with the design of a virtual user interface. These areas are information oriented, systematic graphic design, Discount Usability Engineering, and virtual environment technology. Chapter Three, Requirements, details the requirements levied against the design of the virtual user interface. Chapter Four, Architecture and Design, provides the interface architecture and its functionality in the management of user interaction. This

chapter also discusses the creation of the design specifications necessary to insure consistency among the interface components. Chapter Five, Implementation, presents the methods and justifications associated with the integration of the various components in creating a virtual user interface. Chapter Six, Results, determines the success of the virtual user interface based on accomplishment of the requirements, the outcome of a heuristic evaluation, and the effect the virtual user interface has on a user's degree of presence. Chapter Seven, Recommendations, suggests methods and techniques that can improve the existing interface capabilities of the virtual spaceplane.

CHAPTER TWO - BACKGROUND

INTRODUCTION

This chapter explores the areas of research associated with the development of a virtual user interface. Methods and technologies supporting this effort are information oriented, systematic graphic design, Discount Usability Engineering, and virtual reality technologies. The sections in this chapter provide a description of these areas as they apply to the development of the virtual user interface.

GRAPHIC DESIGN

A basic technique in achieving effective visual communication involves establishing explicit rules, specifications, and guidelines for the "visible language" that affects the user interface. Information oriented, systematic graphic design was the methodology used to create the virtual spaceplane's virtual user interface (VUI). This design methodology involves the use of organization, typography, color, shapes, textures, imagery, and sequencing to design a graphical user interface [Marcus90a]. The next few subsections discuss each of these design components.

ORGANIZATION

Organization determines how the interface presents information and involves developing requirements, formats, and design criteria for two and three-dimensional graphical layouts. Determining the best organization for the virtual spaceplane requires a

determination of the cockpit layout as well as maintaining consistency among the different categories of information.

TYPOGRAPHY

Typography is the selection of typeface and typesetting, which involves determining the best type of font to use under different circumstances. Font presentation is also an inherent part of typography. A key component in the determination of typography for the virtual user interface, is the display resolution of the head-mounted display. A font's functionality and readability are two factors that determine what tradeoffs to make in a virtual application.

COLOR

Color is probably the most important aspect in developing a user interface. "With respect to learning and comprehension, color is superior to black-and-white in terms of the viewer's processing time and emotional reactions, and there is a difference in a viewer's ability to interpret information" [Marcus90b]. There are some basic advantages of color within a virtual user interface:

- Color emphasizes important information
- Color identifies subsystems or substructures
- Color portrays natural objects in a realistic manner
- Color portrays time and progression
- Color reduces errors of interpretation
- Color adds coding dimensions
- Color increases comprehensibility
- Color increases believability and appeal

SHAPE

The shape of an object increases the amount of information presented to the user with a minimum amount of data. For example, the military uses silhouette shape to train individuals about an enemy's arsenal. Once the information about an object is learned, the shape serves as a visual link to that information in memory. This aspect takes advantage of a person's ability to comprehend images faster than numbers or text [Stokes90].

TEXTURES

Textures add more support to color and shapes by conveying a pseudo-realistic appearance to objects. This aspect provides additional support to the degree of presence by enhancing the overall appearance of the virtual user interface.

IMAGERY

Imagery involves using signs, icons, and symbols to provide clues about what something is or does. International street signs use symbols to convey messages about traffic or locations. Individuals can easily translate these symbolic messages regardless of their language. Like shapes, imagery takes advantage of a person's ability to comprehend images faster than numbers or text.

SEQUENCING

Sequencing is a natural and logical progression of events and information. Its primary purpose is to eliminate confusion by presenting information in a manner

consistent with the events taking place around the user. An example is to change the information on a display to present fuel status when the user requests it.

"The basic technique in achieving effective visual communication involves establishing explicit rules, specifications and guidelines for visible language that affects the user interface" [Marcus90a]. Our first step in information oriented, systematic design is creating the specifications necessary to begin building the virtual user interface. Appendix A, Design Specifications, presents specifications for various design aspects to the components of the virtual user interface. Chapter 4 (Architecture and Design) contains information regarding the development of the design specifications. Our second step involves combining the components in a logical manner to create the virtual user interface. Chapter 5 (Implementation and Results) describes this process.

USABILITY ENGINEERING & HEURISTIC EVALUATION

"The human-computer interface into the virtual environment is responsible for providing the user with means for communicating commands to the system, for gathering the user's instruction to the system and presenting them to the system, and for presenting the system's response" [Stytz96b]. Usability Engineering is a design approach for developing and testing the functionality of a user interface. It encompasses the aspects of learnability, efficiency, memorability, error prevention, and user satisfaction [Neilson93]. Because of the lack of users to test the interface, the design evaluation efforts concentrate on learnability and memorability.

LEARNABILITY AND MEMORABILITY

Learnability addresses how soon a user can productively begin working on a system. We can aid learnability in a variety of ways, e.g. online help and documentation, training, or an explanatory interface where the objects themselves provide information on how to use them. Memorability is the ability of users to return to the system after a period of non-use and quickly produce results. This aspect goes hand in hand with learnability. If a system is easy to learn, chances are it is easy to remember.

HEURISTIC EVALUATION

Heuristic evaluation is one of four techniques used in discount usability engineering, a simple methodology for applying the concepts of usability engineering [Nielson90]. Lack of test users during the design process is the primary reason for using heuristic evaluation instead of the entire discount usability engineering method.

Heuristic evaluation, as part of the discount usability engineering method, is a list of ten guidelines used to evaluate a user interface (Table 1). Eight of these guidelines, the exceptions being preventing errors and error messages, are applied to the heuristic evaluation of the virtual spaceplane interface. Preventing errors is an inherent part of designing a virtual user interface. Normally, when a virtual simulation encounters an error it tends to be catastrophic or very noticeable, which minimizes the need to discuss error messages.

Table 1: Heuristic Evaluation Guidelines

Guidelines	Description
Simple and Natural Dialogue	All information should appear in a natural and logical order. Dialogues should not contain irrelevant or rarely needed information.
Speak the User's Language	Dialogue should express itself clearly in words, phrases, and concepts familiar to the user. Interface interaction should not present information inconsistent with the user's knowledge domain.
Minimize the User's Memory Load	Instructions should be visible or easily retrievable. Users should not have to remember information from one part of the dialogue to another.
Consistency	Information, buttons, text, etc. should work in harmony with each other. Users should not have to wonder whether different words, situations, or actions mean the same thing.
Feedback	Keep users informed about what is going on, through appropriated feedback within a reasonable time. Never let the user wonder what is going on.
Clearly Marked Exits	Users should be able to leave an unwanted state with ease. If a user makes a wrong decision, provide an escape route.
Shortcuts	Provide a means to speed up the interaction for the experienced user. Shortcuts enhance the productiveness and allow experienced users to excel.
Good Error Messages	Expressed in plain language, precisely indicate the problem, and constructively suggest a solution. Users should never have to wonder what went wrong and why.
Prevent Errors	Develop a design that prevents problems from occurring in the first place. If an error does occur, it should not be the fault of the interface.
Help and Documentation	Help and documentation should be easy to search, focus on the user's task, list concrete steps to carry out, and not be too large. Users should not need help when it comes to using help and documentation.

VIRTUAL ENVIRONMENT TECHNOLOGY

"Virtual environment technology is the human experience of using displays, sensors, and effectors to perceive and interact with a synthetic environment and its actors as if they were real" [Stytz96b]. Therefore, "the main characteristic of the virtual reality paradigm is the creation of the *illusion of immersion* of the user in a computer-generated environment" [Bryson93]. The degree to which a virtual user interface induces the "illusion of immersion" varies depending of the type of environment modeled, the form

of human computer interaction (HCI), and the level of user presence within the environment [Stytz96b]. Figure 1 shows how each of these components interact with one another to approximate the degree of illusion a user may experience

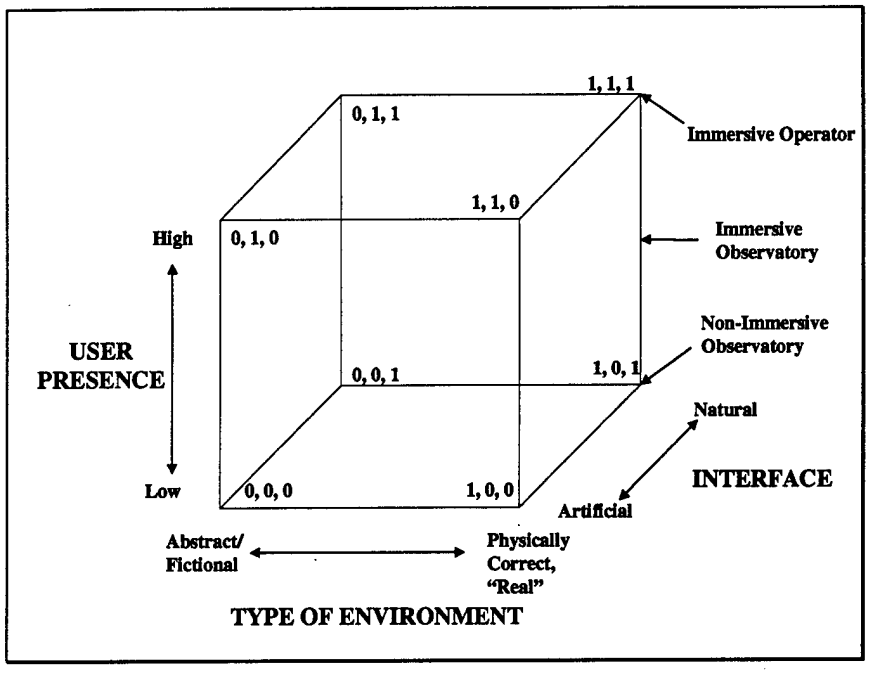

Figure 1: Dimensions of Virtual Environment Technology [Stytz96b]

Point (0,0,0) represents the degree of immersion realized by early command line interfaces and point (1,1,1) is the highest illusion possible. Figure 1 is technology independent; that is, point (1,1,1) represents a virtual environment with the highest presence, physical correctness, and natural interface possible given available technology [Stytz96b].

TYPE OF ENVIRONMENT

Virtual environment modeling depends on the presentation of the virtual surroundings and the behavior of the virtual entities within it. Environment modeling ranges from the abstract/fictional to the physically correct/real. A simple cube on a flat surface with no

"realistic" features can represent an abstract/fictional virtual environment. On the opposite end of the spectrum, a virtual representation of a planetary surface created from scientific data may be physically correct/real. The better the environment, with regards to its appearance and lighting, the greater the chances the user will feel immersed within it.

Object behavior provides additional support to the environment by enhancing visual cues. Users expect to see objects in the virtual environment behave as they do in the real world. Research indicates that "an image is required to provide us with relatively few cues before we take the image seriously as that of a real object with a location in three dimensional space" [Bryson93]. By observing objects behave as they would in the real world and even being able to interact with them supports the "illusion of immersion" [Bryson93].

INTERFACE

"Human-computer interaction is the aspect most apparent to the human participant in a virtual environment and to a large extent determines its acceptance as a portrayal of reality" [Stytz96b]. A virtual environment user interface provides a means for direct interaction with the virtual environment. This range of interaction varies from an artificial interface to a natural/real-time interface. A two-dimensional flight simulator with a keyboard and no sound is an example of an artificial interface. A three-dimensional flight simulator with a head-mounted display (HMD), stereo surround sound, and a haptic feedback joystick is an example of a natural interface. Naturalness of the

interface and lag time are two factors that have a significant effect on where an interface lies on this range.

A natural user interface allows a "user to interact with the environment in ways modeled on real-world interaction" [Bryson93]. For example, while driving a virtual car, the virtual user interface should mimic the appearance and functionality of an actual car interior. Adding a head-mounted display can enhance the perception of the virtual car interior and enhance the illusion of sitting inside a car. "The three dimensional interface and immersive feel gives the user control over the virtual world as a participant from within the environment, rather than a viewer from outside" [Bryson93].

There are two sources of lag time that apply significantly to the virtual user interface: input device and frame rate induced. Input device lag is the time between when the user initiates an action and when the environment displays a response to that action. A delay of more than one tenth of a second can destroy the "illusion of immersion." "Human sensitivity to the time lag between an action and the displayed result makes minimizing lag a priority in increasing virtual presence" [Bryson93]. Frame rate induced lag is the time between successive updates of the display. A frame rate of at least 15 frames per second is a good rule of thumb to maintain the "illusion of immersion."

USER PRESENCE

User presence defines the user's visual perception of being in the virtual environment. User presence varies from low to high depending on the display and haptic hardware used and image quality of the virtual environment. For example, interacting with a virtual environment using a keyboard and monitor visually induces a very low sense of presence.

A user interacting with a virtual environment using an HMD, a dataglove, and a haptic feedback device usually induces a very high sense of presence. Numerous devices enhance user presence within a virtual environment, the ones that apply to the virtual spaceplane are tracking devices and head mounted displays (HMD).

Tracking Device

An electromagnetic (EM) tracking device tracks head, hand, or body movements. It serves as a link between the user and the virtual environment. Tracking devices use an EM field to determine the position and orientation of a remote receiver. In EM tracking a stationary transmitter generates a low frequency magnetic field vectors and the receiver, mounted on the user, intercepts these field vectors. The system determines the orientation and position of the receiver relative to the transmitter. Our simulation takes this orientation and position and converts them into orientation and position with respect to the viewpoint within the virtual environment. Figure 2 is an example of what a transmitter and receiver may look like.

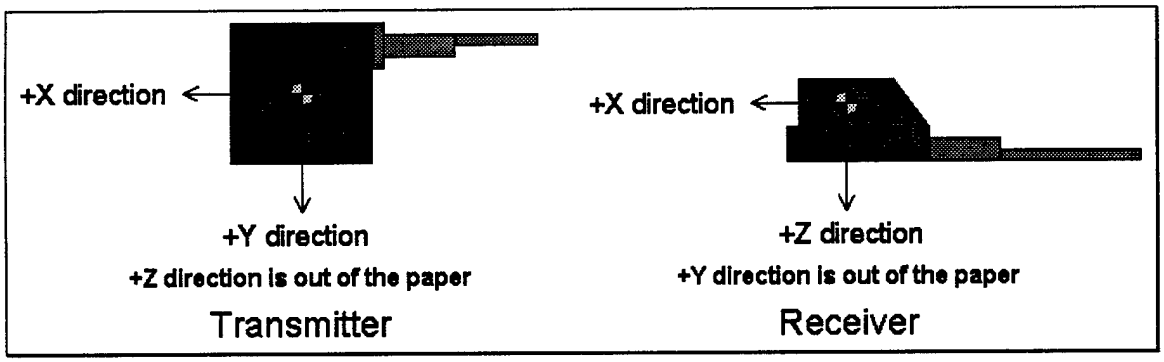

Figure 2: Electromagnetic Transmitter and Receiver

Head Mounted Display

A head-mounted display (Figure 3) provides visualization in graphic applications. Its primary purpose is to provide a sense of immersion within the virtual environment.

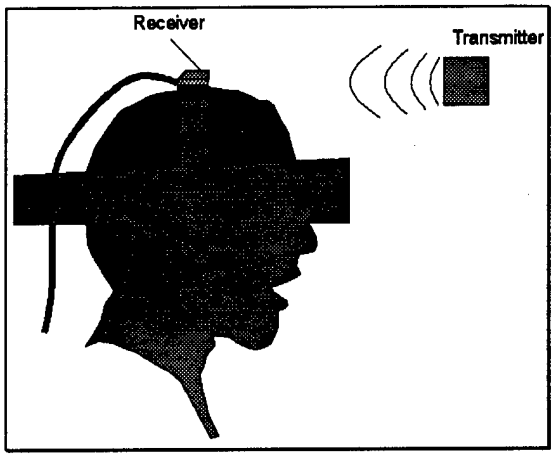

Figure 3: Example of a head mounted display with tracking device

A head-mounted display consists of a set of cathode ray tube (CRT) displays or liquid crystal displays (LCD) mounted to some type of headgear [HEAR97]. There are two displays, one for each eye, which provide a visual representation of the virtual environment. A tracking device receiver is on top of the headgear and oriented relative to the viewing coordinates within the virtual environment. As the user moves their head, the tracking device updates the viewing position and orientation within the virtual environment. For example, as the user moves their head to the left, the viewing orientation in the virtual environment moves to the left. Consequently, the user receives a new viewing position that appears as if they are actually looking around from inside the virtual environment. Resolution and field of view of a head mounted display helps to enhance the "illusion of immersion" by providing peripheral vision in addition to the view in front.

CONCLUSION

Information oriented, systematic graphic design, heuristic evaluation, and virtual reality are three major areas associated with designing a virtual user interface. Graphic design provides the necessary tools for presenting information visually. Heuristic evaluation helps to avoid common problems associated with the interaction between the user and the interface. Virtual reality is the domain in which the user interface resides. Understanding the concepts of virtual reality techniques is necessary for connecting the user and the interface together in a virtual environment. Design of the virtual user interface uses concepts and ideas from each of these areas to create a modifiable spaceplane cockpit.

CHAPTER THREE – REQUIREMENTS

INTRODUCTION

Requirements for the virtual spaceplane are split into five categories: simulated capabilities of a military spaceplane, missions, user interface, virtual environment, and miscellaneous. Because of the cooperation of two other students in developing the virtual spaceplane, this thesis addresses only the user interface requirements. Readers should refer to the companion theses by Lt. Troy Johnson and Lt. Scott Rothermel for information concerning the remaining requirements [Johnson97] [Rothermel97].

SIMULATE CAPABILITIES OF A MILITARY SPACEPLANE

Our virtual spaceplane must simulate the operation and characteristics of a military spaceplane (Table 2). We did not levy any accuracy requirements because there is no military spaceplane to measure accuracy against.

Table 2: Capability Requirements

ID	Description
	Flight Characteristics
1.1	Simulate runway maneuvers.
1.2	Simulate flight through the atmosphere.
1.3	Simulate maneuvering in space.
1.4	Simulate transition to and from space.
	Manual Operations
1.5	Provide capability to manually operate in the atmosphere.
1.6	Provide capability to manually operate in space.
	Automatic Operations
1.7	Provide capability to automatically takeoff.
1.8	Provide capability to automatically fly predetermined route.
1.9	Provide capability to automatically enter orbit.
1.10	Provide capability to automatically modify orbital parameters.
1.11	Provide capability to automatically re-enter the atmosphere.
1.12	Provide capability to automatically land.

MISSIONS

A military spaceplane will conduct a variety of missions to achieve military objectives by exploiting space. In our initial prototype, the virtual spaceplane supports a small portion of the military spaceplane's mission capabilities (Table 3).

Table 3: Mission Requirements

ID	Description
	Missions
2.1	Support rendezvous with orbiting objects.
2.2	Support deployment of a satellite.

USER INTERFACE REQUIREMENTS

Our primary goal of the interface requirements is to explore unconventional interaction and display methods for controlling a military spaceplane and reducing the amount of information presented to the user in current space flight operations. This thesis will focus on the evolutionary prototyping and heuristic evaluation of interface components. Table 4 lists the user interface requirements.

Requirement 3.1 For the initial version of the virtual spaceplane, development concentrated on the functionality and design of the user interface. Using a mouse provides a familiar means of interaction, because almost everyone already knows how to use one. This familiarity can aid the learning process for new users before they try to operate the virtual spaceplane using more elaborate techniques.

Requirement 3.2 The virtual spaceplane is designed to be an immersive virtual environment. Using a head-mounted display provides natural and instinctive interaction between the user and the interface. This natural form of interaction enhances the degree of illusion mentioned in Chapter Two and aids in the learning process. Allowing the user

to instinctively change the viewing orientation decreases their cognitive load by removing one more thing for them to learn; i.e. how to change their view.

Table 4: User Interface Requirements

ID	Description
	Interaction Methods
3.1	Interface interaction shall be available via a three-button mouse.
3.2	Viewing shall support a head-mounted display with head tracking.
3.3	Keyboard interaction will provide auxiliary functionality.
	Configurable Cockpit
3.4	Users shall be able to display information selectively and interactively.
3.5	Information displays shall be positioned interactively by the user.
	Display Information
3.6	Display atmospheric orientation, direction, and status information of the VSP.
3.7	Display orbit entry and re-entry state information of the VSP.
3.8	Display space orientation, direction, and status information of the VSP.
3.9	Display consumable status (propellants, life-support, etc.).
3.10	Display potential target's orientation, direction, and status information.
3.11	Accomplish assistance in locating and acquiring potential targets.
3.12	Accomplish assistance with system management and diagnostics.
3.13	Investigate a hypertext paradigm for information display.
3.14	Minimize obstruction of the user's view.
	Flight Controls
3.15	Do not use a throttle and stick to control the VSP.
3.16	Allow users to change the orientation and direction of the VSP in the atmosphere.
3.17	Allow users to change the orientation and direction of the VSP in space.
3.18	Accomplish assistance in changing the state of the VSP.

*Note: Lt. Troy Johnson will address requirements 3.9 and 3.13 [Johnson97].

Requirement 3.3 Auxiliary functionality utilizing the keyboard allows designers to test and evaluate the functionality of interface components rapidly. Attaching a component's functionality to the keyboard was much faster than placing it in the virtual environment because of the amount of coding involved. Evaluating the functionality with the keyboard means only placing it in the virtual environment once.

Requirement 3.4 Current space shuttle technology displays information regardless of whether the pilot needs it or not. Consequently, astronauts become overwhelmed by the amount of information they have to sort out just to get what they need. Allowing the user

to selectively display information reduces cognitive overload and presents only the required information necessary to complete the mission.

Requirement 3.5 Due to the overwhelming amount of information available in the space shuttle and the limiting space of its cockpit, shuttle information and controls are positioned in some rather awkward places. Allowing placement of information and controls anywhere within the virtual spaceplane provides the user with the flexibility to structure priority items in any manner they see fit. Therefore, needed information can be placed in front of the user while less important information can be placed aside or removed.

Requirements 3.6, 3.7, 3.8 Because the virtual spaceplane will simulate atmospheric and space flight, as well as the transition in between, information concerning the orientation, direction, and status of the aircraft must be presented. Displays will only present information appropriate to the current flight domain. Providing space flight data when the virtual spaceplane is flying in the atmosphere increases the amount of unnecessary information, which is something we are trying to avoid. In addition, the amount of consumables on board will also be displayed. This information is important because it provides the user with clues about the aircraft's condition and whether there are sufficient supplies to complete missions.

Requirements 3.10, 3.11 As with any military operation, a user needs to know the status and condition of potential targets (friendly and enemy) as well as how to locate them. Potential target displays should provide the type of target, orientation and direction, and its alliance (friend or foe). Information will also provide assistance in locating particular targets of interest.

Requirement 3.12 Because of the expected complexity of a military spaceplane, a single pilot cannot maintain complete awareness of the conditions of every onboard system. Consequently, the virtual spaceplane will provide assistance with system management and diagnostics to prevent information overload of the user.

Requirement 3.14 Because we do not have to construct a cockpit that is similar to a real-world counterpart, minimizing obstruction of the user's view takes full advantage of virtual environment technology. This requirement provides the user with an unrestricted view of the virtual environment, reduces the possibility of obstructing other objects from view, and makes landing operations easier to accomplish.

Requirement 3.15 One aspect of our research is to develop and test new paradigms for flight control. Creating a flight control interface without a conventional throttle and stick requirement allows for new interaction techniques to be applied in both atmospheric and space flight operations.

Requirements 3.16, 3.17 Although the ultimate goal of the military spaceplane is autonomous flight control, there is still a need to provide manual control functionality; e.g., evasive maneuvers or emergency landing. Manual flight control provides the user with the capabilities to change the direction, orientation, and speed of the virtual spaceplane.

VIRTUAL ENVIRONMENT

An effective virtual environment must encourage the user to make a mental transition from participating in a computer simulation to the perception that the user is in the virtual

environment. Achieving this transition requires a virtual environment that looks and feels like what the user expects. Environmental requirements are listed in Table 5.

Table 5: Virtual Environment Requirements

ID	Description
	Environment
4.1	Present convincing terrain surrounding Edwards AFB.
4.2	Present convincing portrayals of the Earth, Sun, Moon, and stars.
4.3	Simulate multiple constellations of Earth orbiting satellites and space station.
4.4	Simulate transition between day/night and atmosphere/space.

MISCELLANEOUS

The remaining requirements that do correspond to the previous categories are listed in (Table 6). These requirements deal primarily with Distributed Interactive Simulation (DIS) transmission protocols and hardware performance.

Table 6: Miscellaneous Requirements

ID	Description
	Miscellaneous
5.1	Accept state information of remote entities via DIS protocols.
5.2	Transmit state information of the VSP via DIS protocols.
5.3	Achieve a mean frame rate of 15 frames per second on a four processor, 250 MHz, R4400, SGI Onyx with Reality Engine^2 graphics equipped with 16 Mb of hardware texture memory.

CONCLUSION

A primary goal in this thesis effort is to address the user interface requirements listed in Table 4 using the fundamental concepts and methodologies mentioned in the previous chapter. Our first step in achieving this goal is the creation of a flexible architecture that supports rapid design of the interaction between the user and the interface without impacting the rest of the simulation.

CHAPTER FOUR – ARCHITECTURE & DESIGN

INTRODUCTION

This chapter presents the architecture of the virtual user interface and the design specifications for creating the interface components. The interface architecture specifically focuses on exploiting various Performer routines and class methods to handle interaction input and output. An in-depth review of the overall virtual spaceplane architecture is discussed by Rothermel [Rothermel97]. For specific information about Performer routines, please refer to the *IRIS Performer Programmer's Guide* [SGI95].

Design specifications describe the various components of the virtual user interface. In this section, we address the overall interface layout, standardized color-coding, and the designed functionality of various virtual widgets.

ARCHITECTURE

Figure 4 is a Rumbaugh [Rumbaugh91] diagram of the interface architecture, which is an extract from the simulation architecture. Lines between classes imply an association between them, i.e. sharing of information. For example, PropModel gets the time from SimClock. By default, associations are two-way. Arrows are not standard Rumbaugh syntax; however, arrow use here is to show one way associations. Diamonds show aggregation or composition. For example, the Sim class is composed of a Renderer, Selection Manager, SimObject, and Input Modifiers.

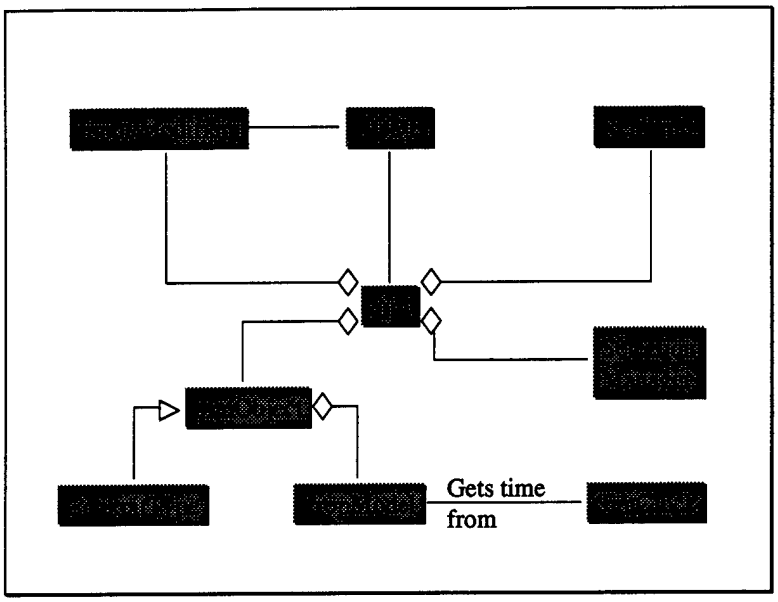

Figure 4: Cockpit Architecture

SIM

The Sim class has the responsibility for translating user interaction and running the simulation. The Sim class contains the Input Modifiers, a Selection Manager, a Renderer, and representations for all the entities in the virtual environment. This implementation ensures a centralized location; i.e., the Sim class, has access to every part of the simulation.

COMMON OBJECT DATABASE

The Common Object DataBase (CODB) is a virtual environment architecture that uses containerization and a central runtime data repository. Containers are data structures for moving large amounts of structured data between system components, for the management and control of inter-component communication as well as for passing data [Stytz96b]. The CODB accomplishes this task with member functions designed to read

and write to a designated container. For example, an Input Modifier places information into a container with a ***BeginWrite ()*** command and the Sim class accesses the same container with a ***BeginRead ()*** command.

INPUT MODIFIERS

Input Modifiers consist of the classes IO_Mouse, IO_Keyboard, and IO_Bird. IO_Mouse is a class linking the Performer mouse event handling routines to the CODB. IO_Keyboard is a class linking the Performer keyboard event handling routines to the CODB. IO_Bird is a class providing functionality for the Ascension Flock of Birds™ tracking device.

RENDERER

This class has the responsibilities of setting up the graphics hardware, maintaining the Performer scene graph, and computing the rendering. With the exception of requiring SimObjects to create and update their local geometry, the Renderer encapsulates all rendering functions.

SELECTION MANGER

The Selection Manager class manages all entities identified for selection. Each selectable entity is given an index value. When the user clicks on a selectable entity with the mouse, the Selection manager provides the entity's index value to the Sim class.

SIMOBJECT

The SimObject class is a base class that provides a foundation to develop all simulation entities and methods to manipulate and manage them. A SimObject's primary purpose is managing an entity's local geometry. Each SimObject also has access to a propagation model, which describes their movement in the virtual environment. They also have a camera offset which the Renderer uses when positioning the viewpoint to a SimObject. By positioning the viewpoint, we can "teleport" the user from one SimObject to another. This functionality causes the viewpoint to appear attached to a SimObject.

COCKPIT

A Cockpit is an instance of a SimObject. It is always associated with the viewpoint so whenever the viewpoint moves the Cockpit moves along with it. Therefore, it has the ability to attach to other SimObjects. This functionality uses the camera offset mentioned above to give the Cockpit the flexibility to move through the virtual environment without restricting it to a single propagation model.

All user interactions with the virtual environment process through the Sim class that then delegates the necessary changes to the appropriate components of the simulation. Cockpit receives the necessary data from Sim to update its geometry and the information it presents to the user. Sim redirects information, sent to it by Cockpit, to other components of the simulation. By restricting the flow of information to and from the Cockpit, we are able to design and prototype the virtual user interface without disrupting other simulation components.

PROPMODEL

PropModel is a base class for managing propagation models used by SimObjects. It contains methods for manipulating and querying the derived subclasses. PropModels are created and reused to describe different dynamic models throughout the simulation (Table 7). Dynamic models are those that take into account the physics associated with different modes of travel, e.g. space and atmospheric flight. Other models handle the celestial mechanics for planetary motion. There is also a model that directs the motion of remote distributed interactive simulation (DIS) entities with the virtual spaceplane environment. For a detailed description of these models, refer to Johnson [Johnson97].

Table 7: Propagation Models

Model Type	Description
FreeFlight	Ignores all physical laws of motion. Used mostly for debugging and stationary entities.
TaxiProp	Terrain following model. Allow the virtual spaceplane to travel down a runway.
Sun, Earth, Moon	Drives the motion of the planets and sun
AeroProp	A six-degree of freedom aerodynamic model obtained from Wright Labs Flight Simulation Branch.
AstroProp	An orbital mechanics model supplied by the Air Force Academy's Astro Physics Department.
OrbitEntryProp	Supplied by the Air Systems Center's Experimental Research Division (ASC/XR) to transition the virtual spaceplane to and from space.
DISEntityProp	Represents the motion of remote distributed interactive simulation (DIS) entities.

SIMCLOCK

Our SimClock class controls the flow of time throughout the virtual spaceplane simulation. Using the ***SetTimeFactor ()*** method we can affect how a propagation model calculates an entity's next location within the virtual environment. Increasing the time

factor of the simulation speeds up all propagation models much like pressing the fast forward button on a VCR. We can also slow and pause the simulation.

DESIGN SPECIFICATIONS

Before constructing the virtual user interface, the design specifications were developed. These specifications provide explicit rules to guide the development process and insure consistency among the various interface components [Marcus90a]. Appendix A (Design Specifications) lists the different components of the virtual user interface and the specifications of each. In the rest of this section, we discuss the development and justification for these specifications.

COCKPIT LAYOUT

Our cockpit layout is a geometric structure surrounding the users' viewpoint within the virtual environment. It provides a reference for building the interface and helps prevent disorientation by indicating directional vectors with respect to the viewpoint. As a reference, the placement of various interface components becomes much easier because everything is consistently the same distance away from the user. Figure 5 shows the first version of the cockpit layout.

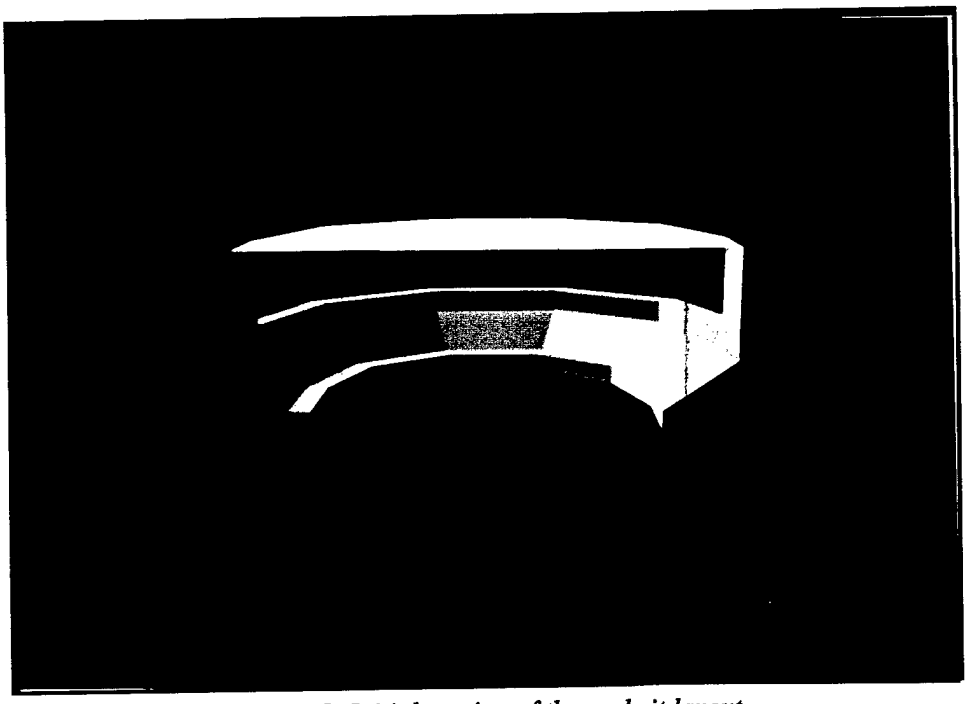

Figure 5: Initial version of the cockpit layout

This design accomplishes the above objectives, but restricts our view of the virtual environment. There is no overhead view and the opaque consoles restrict the forward view. Our second version of the cockpit layout (Figure 6) reduces the viewing restrictions because the roof and side panels are gone. This layout allows a complete 360-degree view around and above the cockpit. Unfortunately, the opaque consoles still restrict too much of the forward view. This restricted view makes manually landing the virtual spaceplane extremely difficult and also hides potential targets from view while in orbit. Consequently, the user would have to reorient the virtual spaceplane to ensure a selected target is not hidden behind the cockpit structure.

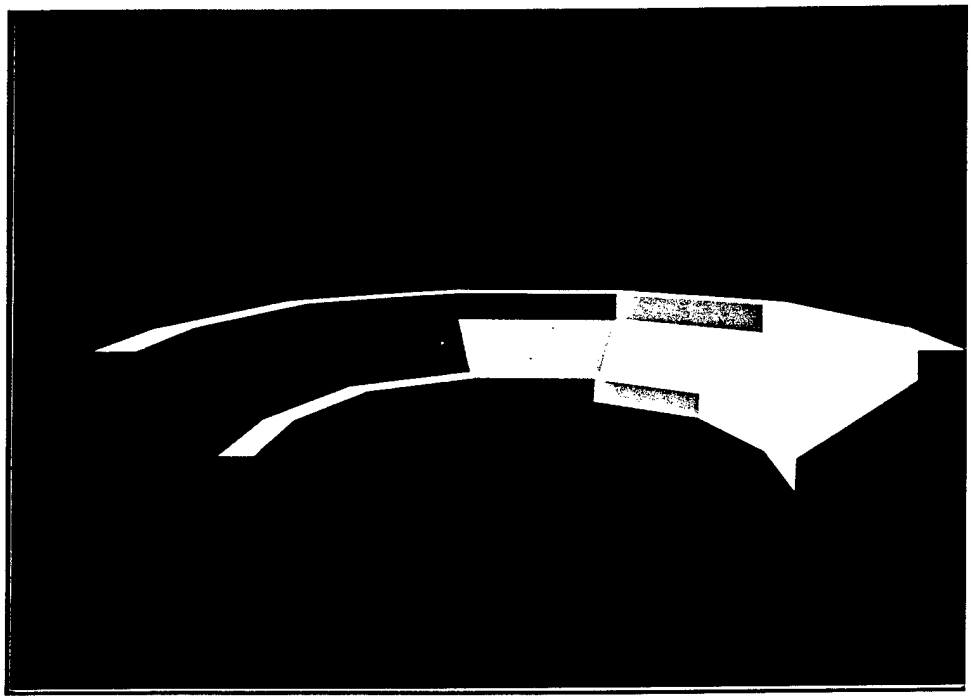

Figure 6: Second version of the cockpit layout

Our final version of the cockpit layout (Figure 7) still meets our objectives (to prevent user disorientation and provide a sense of forward direction) with a less obstructive layout. With the transparent consoles, we now have a completely unrestricted view of the virtual environment.

Unfortunately, even with an unrestricted view of the virtual environment, we were not able to fully interact with it. Our transparent structure implies that "if you can see it you can select it." Because the structure lies in the part of the scene tree identified for selection, the only thing we select is the structure. We did not want to alter the functionality of the Selection Manager, because this would prevent us from selecting anything in the Cockpit. Instead, we explicitly identify the cockpit structure as non-selectable by skipping it in the scene tree traversal. As a result, we can select objects through the cockpit structure.

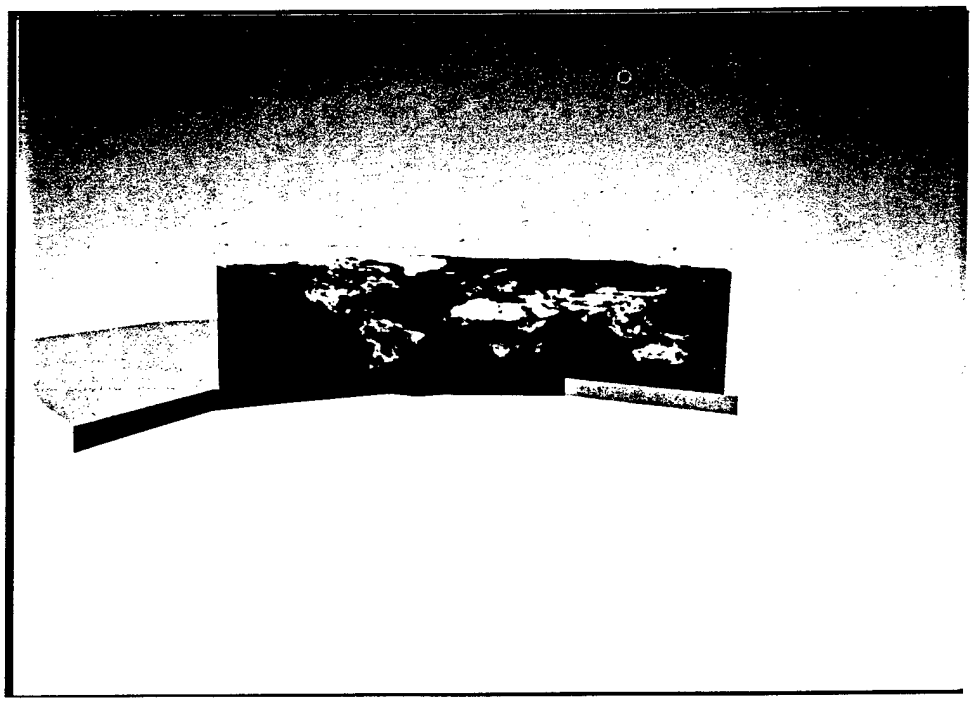

Figure 7: Final version of the cockpit layout

STANDARDIZED COLOR CODES

Research shows users respond more rapidly to color coding than to shape coding or to alphanumeric coding [Stokes90]. Our virtual user interface has two types of color-coding (single purpose and environmental). Single purpose color coding uses population based stereotypes with representative meanings. Population based stereotypes are social perceptions about what something means. For example, people who drive cars know that red means stop, yellow means caution, and green means go. For most pilots in the military, red means danger, yellow means caution and green means all clear. For the purpose of this thesis, our population-based stereotypes will imply the military application. Environmental color coding uses environmental colors to reflect the reality of the real world; i.e., light blue for sky.

Because the virtual spaceplane simulation is a military application, we must have a color scheme that addresses force identification. Table 8 lists the single purpose color-codes in current military applications and Table 9 lists those in the virtual spaceplane.

Table 8: Standard Force Identification Color Codes

Color	Representation
Red	Enemy
Yellow	Unknown
Blue	Friendly

Table 9: Virtual Spaceplane Force Identification Color Codes

Color	Representation
Red	Enemy
Yellow	Unknown
Green	Friendly

Green identifies friendly forces because it is difficult to distinguish a blue object in space. Green is also the only other color in the spectrum we could attribute to a friendly force due to military based stereotyping. In addition to force identification, the virtual spaceplane also uses a single purpose color code for system status (Table 10).

Table 10: Virtual Spaceplane System Status Codes

Color	Representation
Red	Warning
Yellow	Caution
Green	All Clear

Color coding the status of a system provides immediate information about the system's state. This color coding method also yields greater response accuracy when the time to absorb display information is brief [Stokes90].

Remaining subsections address the different color schemes associated with the various interface components.

TYPOGRAPHY

Another essential element in the development of the cockpit is typography. Typography includes the characteristics of individual elements (typefaces and typestyles) and their groupings (typesetting techniques) [Marcus90b]. Our virtual user interface uses Helvetica and Times Elfin with sizes ranging from 24 to 48 pitch.

Button text, constellation names, status indicators, and flight indicators use Helvetica because it is more legible at small sizes than other fonts. Dynamic text on panels and the virtual heads up display use Times Elfin. We use Times Elfin for dynamically changing text because it is the only dynamic font available in the Performer graphics suite [SGI95]. A dynamic font is a three-dimensional object in the virtual environment that can have its values changed at runtime. Following are some basic guidelines [Marcus90b] we consider when using typography in the virtual user interface.

- Generally, use no more than three typefaces in a maximum of three point sizes each.
- Text should be set flush left and numbers should be set flush right.
- For fixed width fonts, justified lines of text can slow reading speed by twelve percent.
- Avoid capital lines of text, which can slow reading speed by twelve percent.

BUTTONS

Buttons are the primary means of interaction between the user and the virtual environment user interface. Each button is related to a specific functionality or application that affects not only the button's state, but entities in the virtual environment as well. The three types of buttons are radio, push-to-activate, and toggle, which provide a wide range of interaction flexibility. Radio types allow selection of one button out of a group of related buttons; e.g., buttons on an old car radio. At any given time, only one button in the group is active. Push-to-activate types are active as long as the user

continues to press the button; e.g., pressing the mike button on a CB radio. Toggle types function similar to a light switch (push in to activate, pop out to inactivate).

There are three possible states for every button: active, inactive, and disabled. When a button is active (on) so is the function it controls. When a button is inactive (off) the function it controls is inactive. When a button is disabled (not available for selection) the application it represents is not required.

Figure 8 contains an example of an early button design. Green buttons with black text represent active buttons. Blue buttons with white text represent inactive buttons. Grey buttons with black text represent disabled buttons. Button geometry consists of a flat two-dimensional polygon with a colored texture, which gives the button a three-dimensional look.

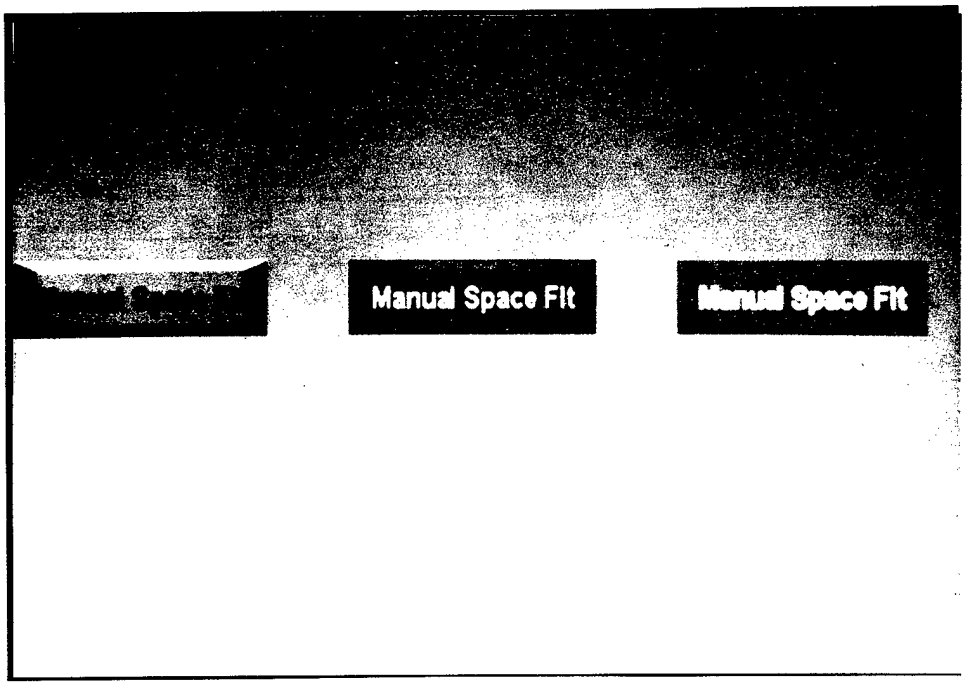

Figure 8: Early Button Design

This particular button design did not provide an appropriate three-dimensional appearance. Color is the only thing that changes when turning a button from off to on.

This indication does not provide enough feedback to the user. Its color is also too prominent in the cockpit, which is distracting when trying to direct attention to other areas in the virtual environment.

Figure 9 is the current button design. Light gray buttons with black text represent active buttons. Medium gray buttons with black text indicate inactive buttons. This color scheme provides excellent contrast due to the gray background and black text in the foreground [Stokes90]. Dark gray buttons with light gray text represent disabled buttons. We use a subdued color scheme because the disabled button functions as a placeholder and does not attract the user's attention.

This button design emulates the buttons in a windows desktop environment. We chose this particular design because it presents a clear three-dimensional representation of a button. A button actually appears to be "popped out" when it is inactive and "pushed in" when it is active. In addition, without test users to perform usability studies, we fell that anyone already familiar with the personal computers will immediately recognize it as a button.

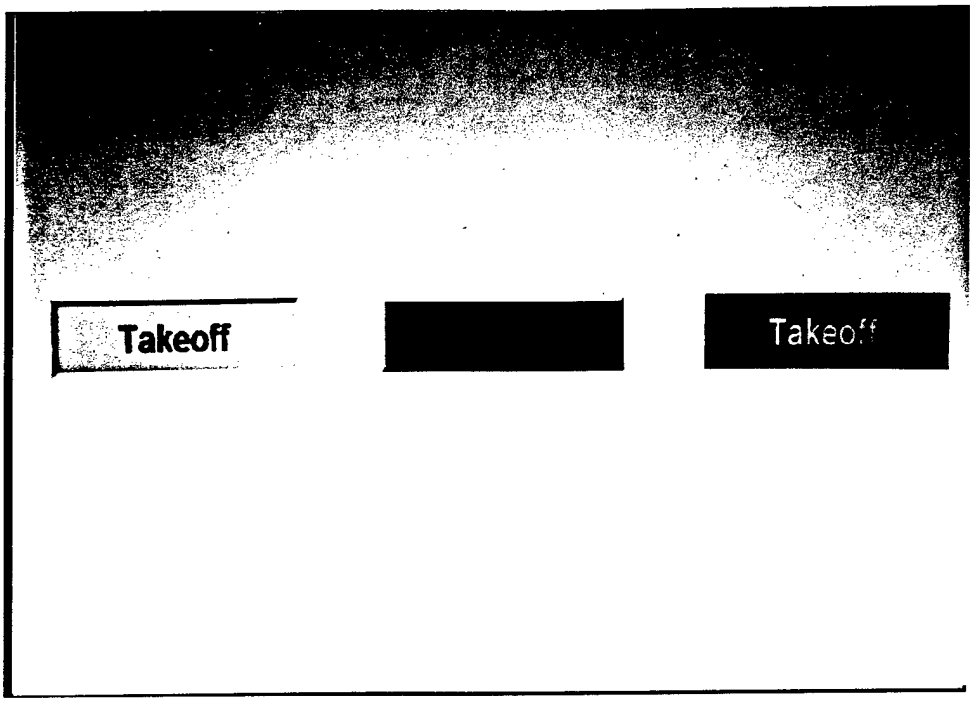

Figure 9: Current Button Design

CONSOLE PANELS

Console panels behave similar to a window in a PC desktop environment. Users can move them around, and minimize them in the same manner. A panel's primary purpose is to contain all the controls and information, e.g. buttons, text and displays, relevant to a particular aspect of a virtual space plane mission. For example, there is a target panel for target acquisition, a navigation panel for navigation, etc. Figure 10 represents an early version of a console panel.

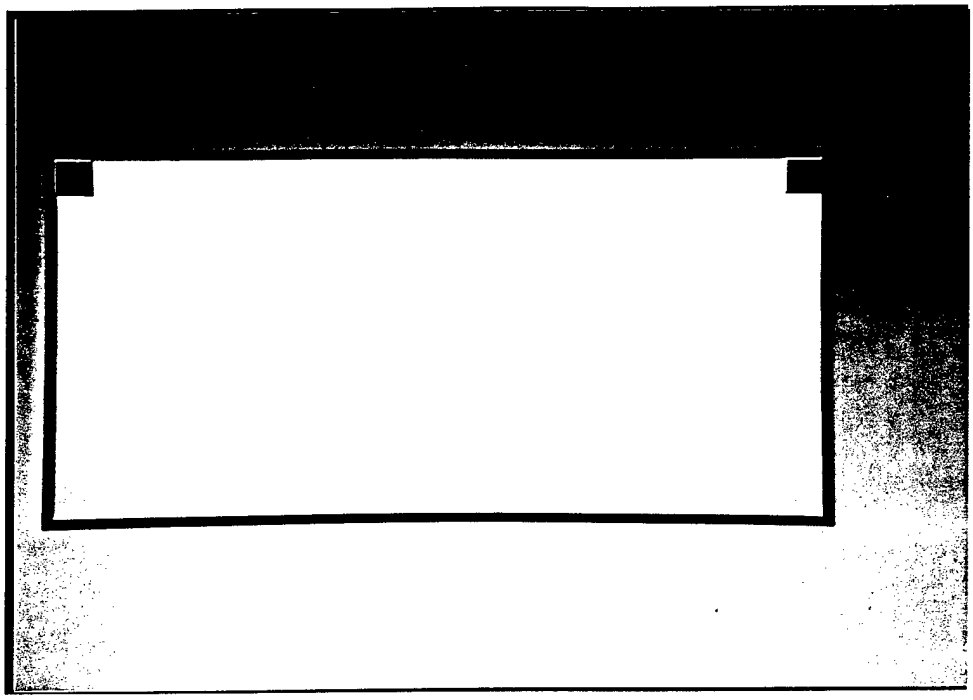

Figure 10: Early Panel Design

The early designs had a move button in the upper right hand corner of the panel. By pressing this button, the user can reposition the panel anywhere within the confines of the cockpit. A medium gray base on the panel provided excellent contrast between buttons and panel background, making these items easy to see. However, there are a few problems with this particular design. First, it is easy to block the move button with the flight control system. If one releases this button behind the control grid, minimizing the flight control system is the only way to regain access to it. Second, panels do not behave exactly as a window because there is no functionality for minimizing or maximizing. Finally, we have the same problem with the opaque panels restricting the view to the virtual environment as the early cockpit layout design. Our panel restricts visibility and with a few of these up in the cockpit our view gets severely limited.

Our current console panel design (Figure 11) has a much longer move button, making it harder to accidentally hide it behind other cockpit geometry. A minimize button is in

the upper right hand corner of the panel. There is only one on each panel because of the association with a PC window environment. In addition, we also allow the user to select objects through the panels, just as we do for the cockpit structure.

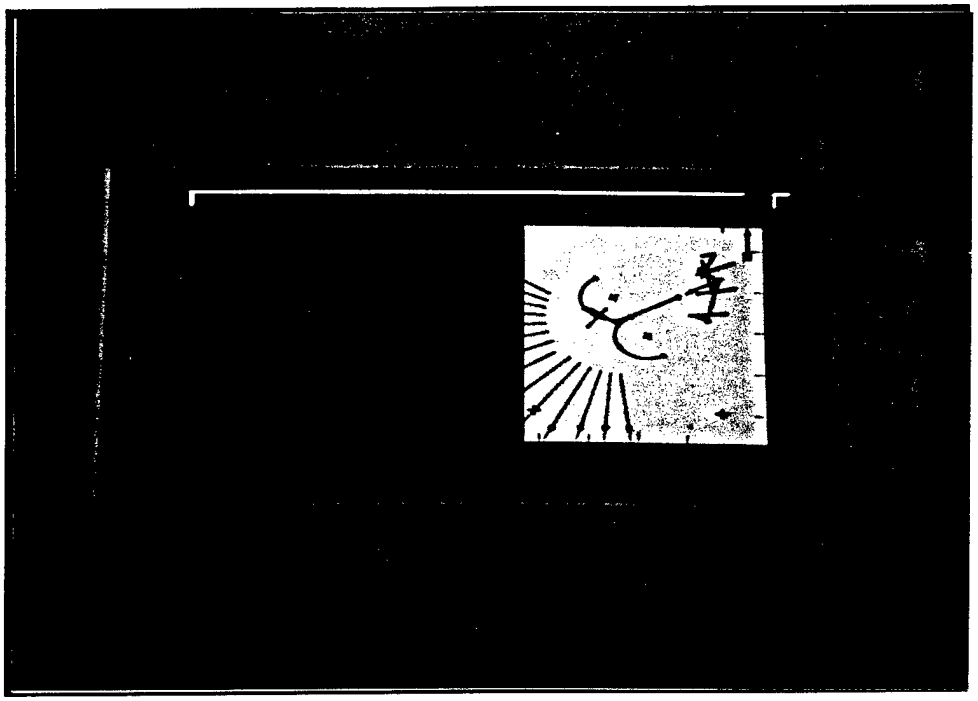

Figure 11: Current Panel Design

Currently, there are nine console panels inside the virtual spaceplane cockpit. Table 11 lists each of these panels along with a brief description. A more detailed description of each panel is in Chapter Five, Implementation.

Table 11: Console Panels

Panel Name	Description
Control Panel	Contains a miniaturized version of the flight controls.
Aero-Flight Panel	Organizes and provides information applicable to atmospheric flight operations.
Space-Flight Panel	Organizes and provides information applicable to space flight operations.
Navigation Panel	Provides information about the location and direction of the virtual spaceplane with respect to the surface of the earth It also presents information about the location and type of various ground-based entities.
Mission Panel	Organizes and provides information about the virtual spaceplane's payload. It also provides controls necessary to perform missions, e.g. deploying a satellite.

Target Panel	Organizes and provides information and controls applicable to target selection, target identification, target viewing, target manipulation, and target interception.
Trajectory Panel	Presents information about dynamic atmospheric pressure, kinetic and potential energy, and temperature associated with orbit entry and re-entry.
Virtual Environment Panel	Provides the capability to change various aspects of the virtual environment, e.g. the rate of time, amount of ambient, etc.
Engineering Panel	Provides diagnostic and time critical information concerning the virtual spaceplane's system and consumables.

DISPLAYS

"Displays provide a structured and deliberate presentation of information about the state of a man-machine interface [Stokes90]." Each has their own look and feel dependent upon their purpose and functionality. All displays in the virtual user interface fall into one of three categories: qualitative, quantitative, or hybrid [Stokes90]. Qualitative displays provide overall situation awareness according to condition (good/bad) or kind (friend/foe) with a single indicator. Quantitative displays provide information about the measurable value of some variable in amounts or portions. Hybrid displays provide information in both a qualitative and quantitative format.

We also categorize displays in the virtual user interface as interactive or non-interactive. Interactive displays allow a user to modify the information it presents. Non-interactive displays only present information.

Currently, there are six displays on various panels inside the virtual spaceplane cockpit. Table 12 lists each of these displays along with a brief description. Chapter 5, Implementation, will discuss the details about each display in the virtual user interface.

Table 12: Displays

Panel Name	Description
Attitude Direction Indicator	Presents information about the virtual spaceplane's pitch, roll, altitude, velocity, and heading.
Orbit Display	Presents information about the orbit and location of the virtual spaceplane and current target.
Map Display	Presents the location and direction of the virtual spaceplane with respect to the surface of the earth. It also shows the location of various ground-based entities.
Target Display	Presents a three-dimensional model of the current target.
Trajectory Display	A plot based on altitude versus velocity presenting information about dynamic atmospheric pressure, kinetic and potential energy, and temperature associated with orbit entry and re-entry.
HTML Browser	Provides diagnostic and time critical information concerning the virtual spaceplane's system and consumables with the functionality of a web browser.

STATUS INDICATORS

Status indicators are a hybrid display providing situation awareness and the measurable value of some variable. Their main purpose is to provide an indication (warning, caution, and all clear) about the state of the virtual spaceplane. We use the system status color-coding to provide the visual indication.

Our first design, using dynamic text, changed the color of the text whenever its representative variable meets a condition for warning, caution, or all clear. This design achieves the purpose of a status indicator, but does not provide enough user interaction, because it does not allow the user to selectively display the information. In addition, this design adds visual clutter to the virtual user interface and restricts our view of the virtual environment.

A second design used sliders, which change length based on the current value of the associated variable. When the associated variable meets certain conditions, the slider changes to the appropriate color. Its length gives an indication of the variable's value. If the user wants to know the variable's value, the value must be interpreted from the scale. This design was rejected due to the excessive amount of time it takes to interpret the value. In addition, the slider restricts the range of possible values.

As shown in Figure 12, our current design consists of a 1.0 x 3.0-cm polygon with text identifying the variable the status indicator is representing. To the right of the indicator is the numeric value of the variable, which displays either manually or automatically. Depending on the current situation, the user may not want the status indicator to display the numeric value. We provide the capability to turn the value on or off by clicking on the status indicator or using the console buttons. This user interaction derives its functionality on the smart-light-switch method [Cooperstock97], which is an interaction technique that turns on lights automatically using motion sensors or manually using a switch.

Figure 12: Current Status Indicator Design

Figure 13 presents a logic diagram of this algorithm as it applies to status indicators. Each box represents the state of the indicator. Directed lines between the boxes represent the action that takes place to change the indicator's state. Selecting the status indicator manually changes its appearance by displaying numerical values. Its appearance also changes automatically by displaying numerical values and changing colors when the indicator's variable is either in or out of tolerance.

Incorporating this algorithm into the status indicators provides some automation in the virtual user interface. This automation creates a dynamic interface with capabilities that respond to changes in the virtual environment, which in turn should increase presence. A quote from Steve Bryson, Computer Sciences Corporation, explains this claim best [Bryson93].

"The behavior of things in the virtual world can also enhance the degree of presence. Based on anecdotal evidence, we will take an object more seriously as a real object if that object has some kind of non-trivial behavior. Behavior makes the object more interesting and engages the user. In particular, if the object reacts to the user in interesting ways then that object will be more compelling, enriching the presence of the virtual environment."

Steve Bryson,1993

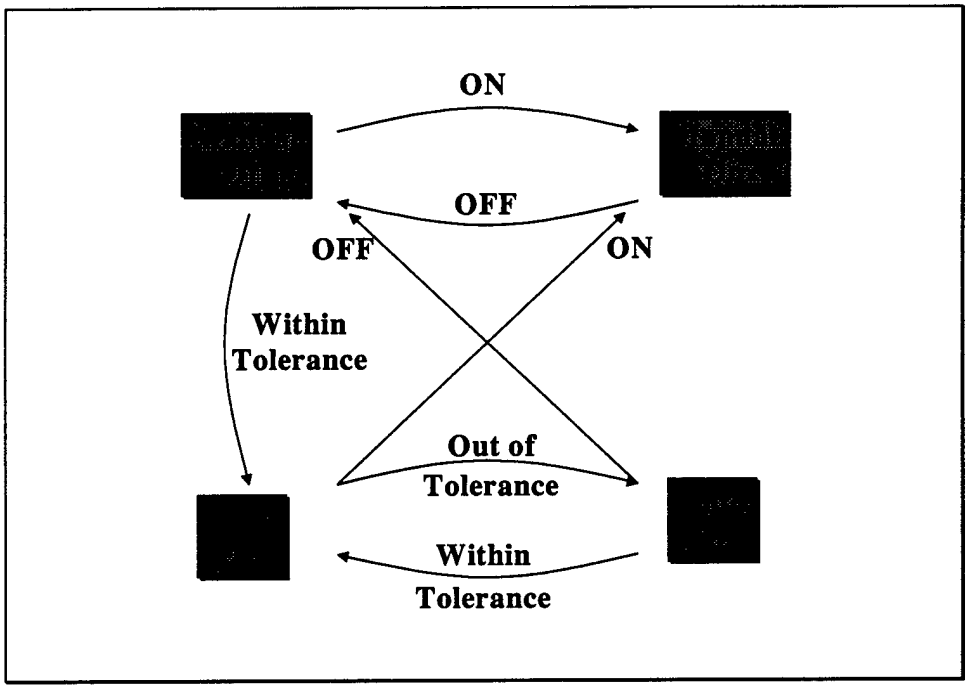

Figure 13: Logic Diagram for Status Indicators

LOCATORS

Locators provide a visual representation about an entity's position, type, and orientation in the virtual environment. Two types of locators, space and ground locators, use shape coding and force identification color-coding to indicate an entity's type and alliance. A user can request additional information about a locator's associated entity by selecting it with the mouse. After selecting a locator, our virtual user interface presents additional information about that locator's entity.

Space locators (Figure 14) identify space borne entities. They use shape coding to identify the type of entity it represents. Each space locator is a symbolic geometric representation of its entity, approximately fifty times larger. This size difference makes it easier to locate and identify distant entities in the virtual environment.

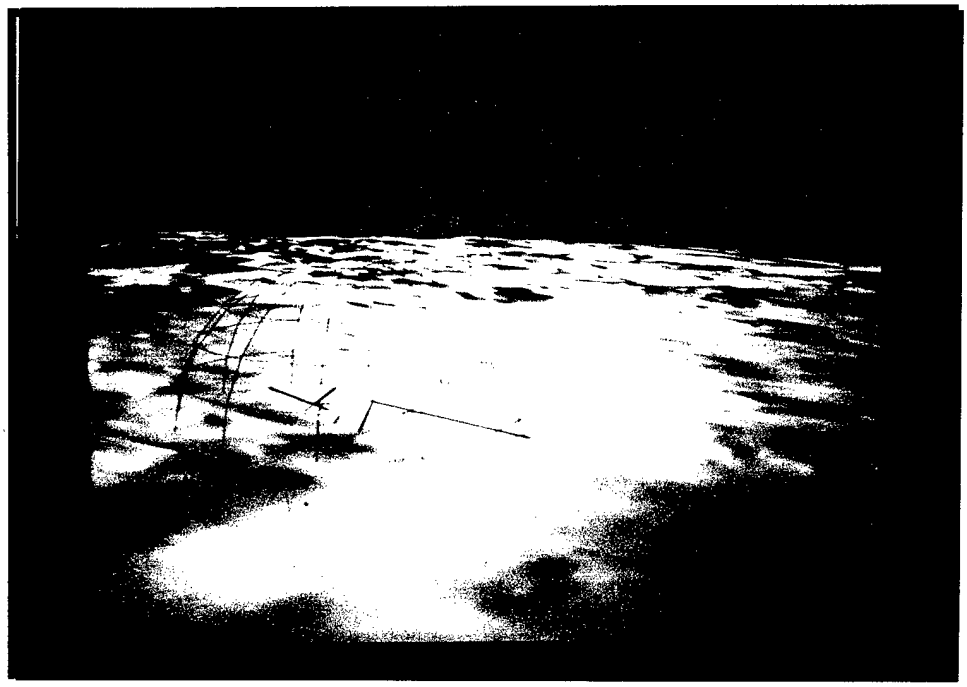

Figure 14: Space and Ground Locators

Ground locators (Figure 14) identify ground-based entities. Due to the diversity and number of expected ground entities, we did not create separate locators for each type of ground entity. Instead, ground locators are a wire frame dome centered over the ground entity's position.

TRAILS

Trails provide a visual "history" about an entity's previous location. Trails also indicate direction because users can predict by extrapolation where an entity is going. Four types of trails in virtual spaceplane are past, future, orbit, and transition.

Past trails (Figure 15) start at the entity's position, either in the environment or on a display, and follow behind it as it moves. They use the force identification color code to ensure consistency with an entity's locator.

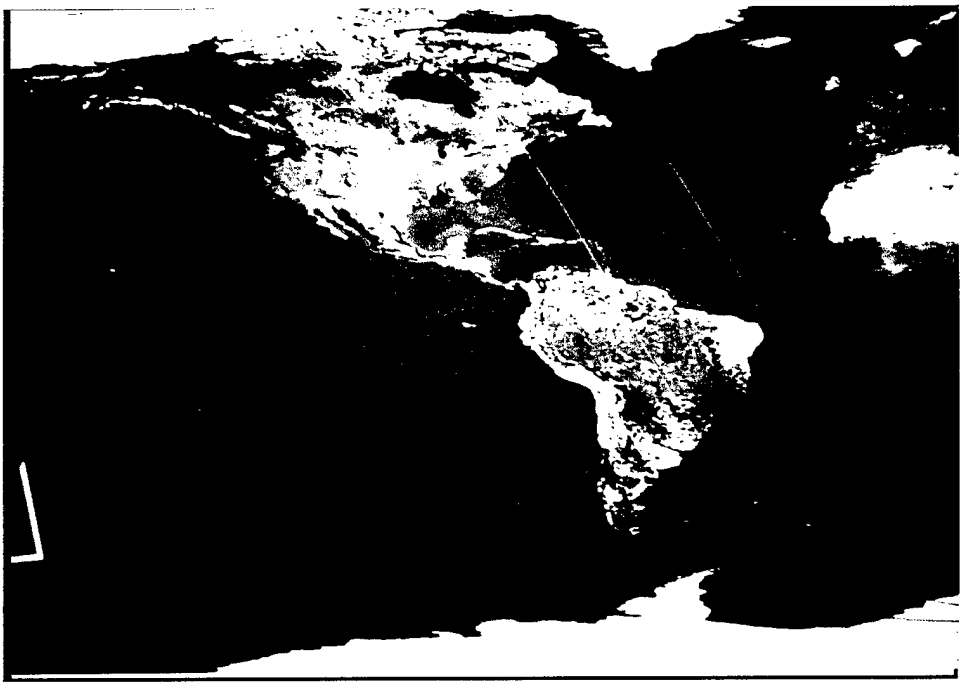

Figure 15: Past and Future Trails

Future trails (Figure 15) start at an entity's position and extend out front indicating where the entity is going. Putting them on entities in the environment would not provide any additional information that past trails are not already providing and would only increase visual clutter. Therefore, future trails are only used on the map display. Future trails are color coded blue to maintain consistency with our design specifications discussed in Appendix A.

Orbit trails (Figure 16) are used on the orbit display; i.e., a three-dimensional globe. They provide a visual representation of a space entity's orbit trajectory with respect to the earth. A system status color code identifies if the virtual spaceplane's orbit is safe (green) or if it will intersect the earth (red). Target orbits are white because we do not

expect them to intersect the earth and it helps to distinguish then from the virtual spaceplane's orbit. From the cockpit, orbit trails in the virtual environment do not provide enough information since the earth blocks much of the orbit from view. By putting orbit trails in a display, the user has a "god's eye" view of an entity's entire orbit.

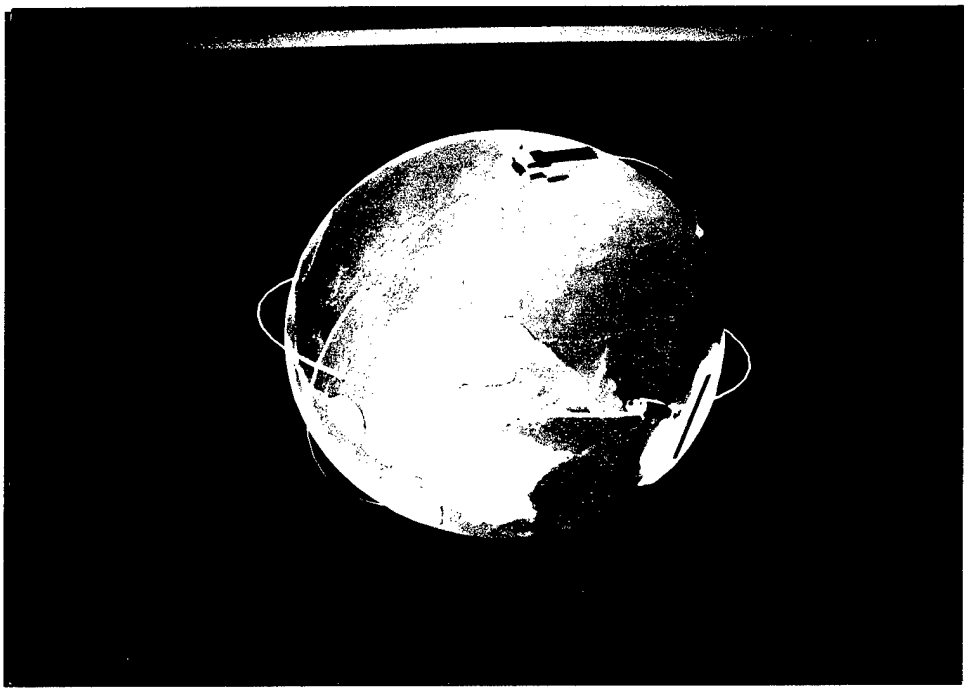

Figure 16: Orbit Trails

The single transition trail (Figure 17) within the interface is on the three-dimensional globe display and serves to provide information about the future orbit trajectory of the virtual spaceplane as it transitions from one orbit to another. We use blue because it indicates the virtual spaceplane's future trajectory, much like the future trails.

Figure 17: Transition Trail

POINTERS

A pointer is a virtual indicator for the direction to take to find a particular object. A pointer's primary purpose is to reduce user disorientation. Two types of pointers in the virtual user interface are earth and target. An earth pointer (Figure 18) provides the shortest re-orientation direction to align the virtual spaceplane with the earth. A target pointer (Figure 18) provides the shortest re-orientation direction to align the virtual spaceplane with the current target.

Pointer geometry consists of a flat two-dimensional 1.0 x 1.0-cm polygon with a texture symbolically identifying the type of pointer. An earth pointer has a pictorial representation of the earth and a target pointer looks like the cross hairs of a riflescope. Choosing the earth symbol for the earth pointer is obvious. Choosing the cross hairs for the target pointer is not. After re-evaluating who our possible users may be; i.e., military

pilots, the cross hairs seem to be a good choice because this symbol historically represents shooting at a target.

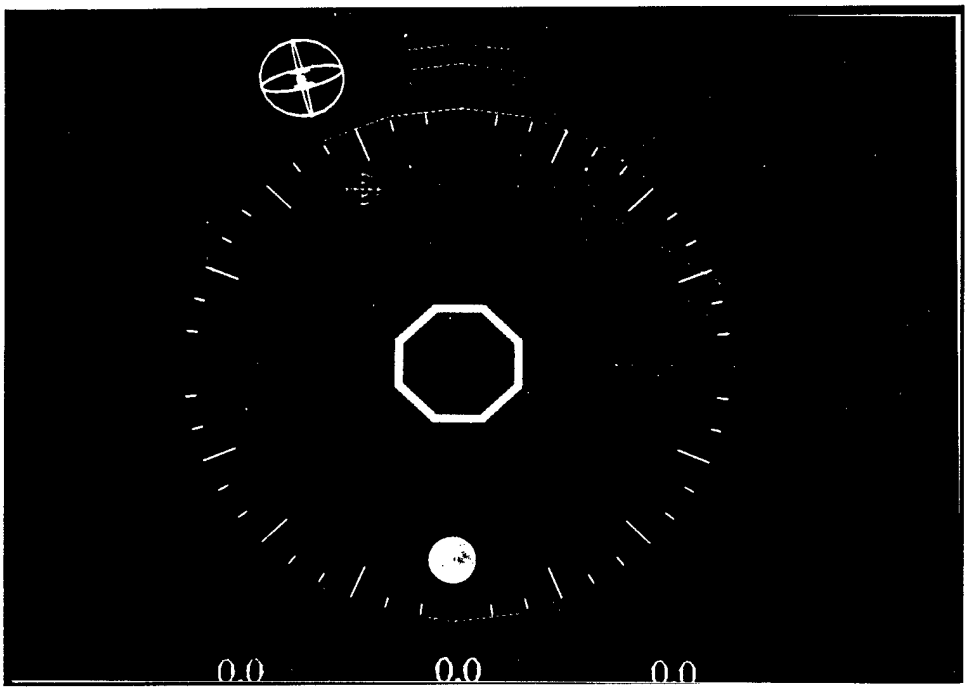

Figure 18: Earth and Target Pointer

SLIDERS

Sliders display a value between some minimum and maximum values. For example, a slider extending half way up a scale from zero to a hundred represent a value of fifty. There are two types of sliders: time and control. Time sliders (Figure 19) show the total amount of time something takes to complete. Each segment of a time slider represents different sections of the total time. For example, time until initial burn and time until final burn.

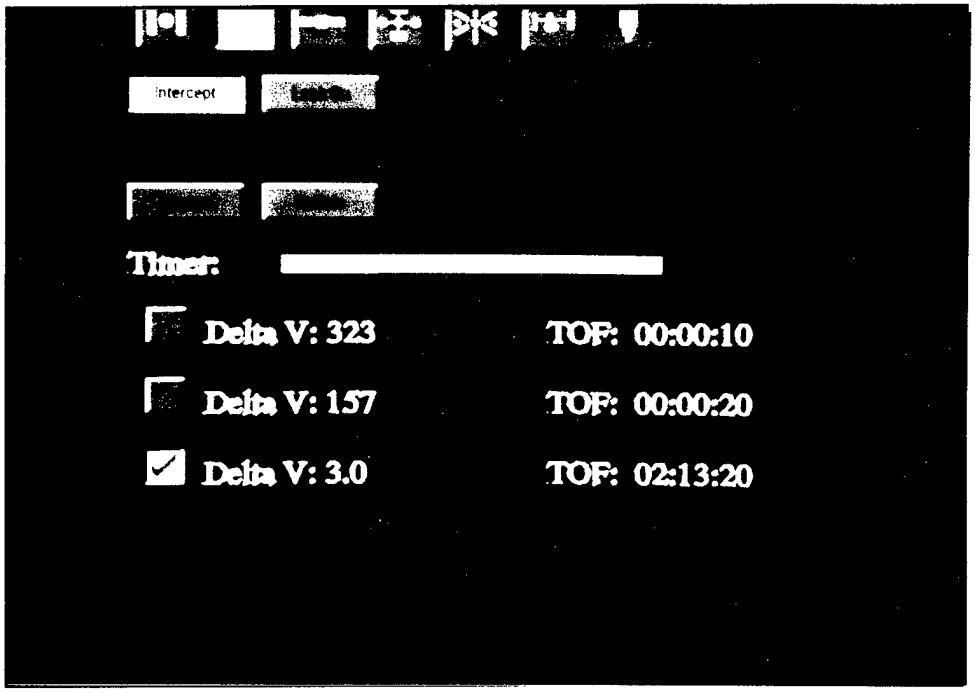

Figure 19: Time Slider

Control sliders (Figure 20) are similar to scroll bars found on all window applications. Control sliders are used to select a value between a maximum and minimum value. To change the current value of the control slider, the user clicks on the bar with the left mouse button and drags the cursor along the length of the slider. The length of the bar changes to represent the value of the control slider. For example, our throttle slider allows the user to change the throttle percentage of the virtual spaceplane's engines. Dragging the bar from zero to fifty means the user has increased the throttle percentage to fifty percent.

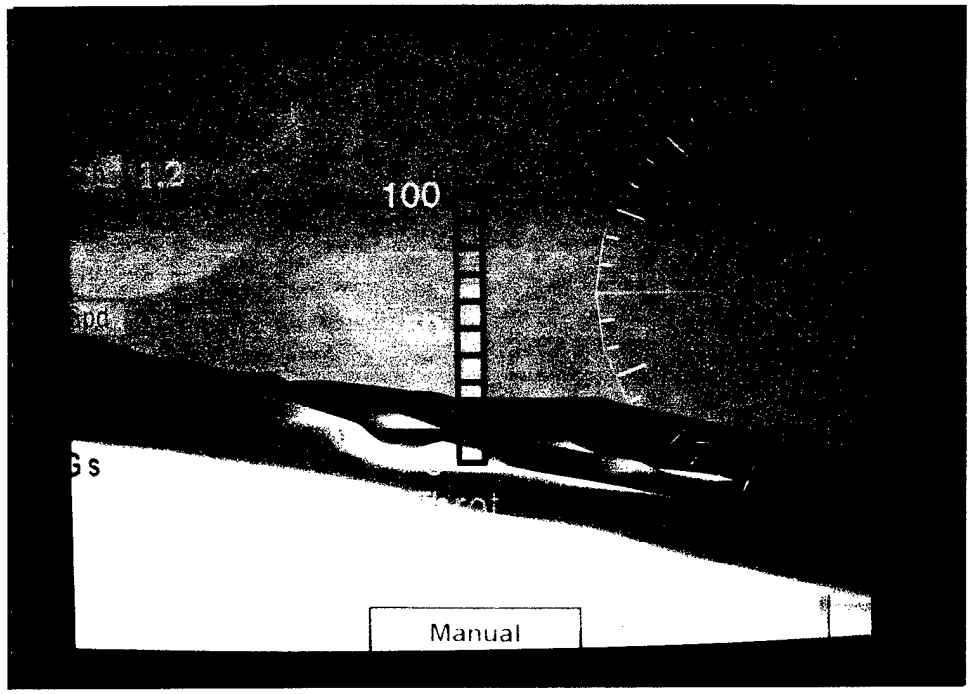

Figure 20: Control Slider

TARGET RETICLES

Target reticles indicate the currently selected target, be it a space or ground based entity. Initially, we thought the shape of an entity's locator provided enough information. Problems arose when the number of entities increased to a point where it became difficult to determine exactly which entity was selected, particularly if that entity was in close proximity to other entities of the same type. Target reticles overcome this problem by surrounding the selected entity with a wire frame outline causing it to stand out from the background.

White is the color code for a target reticle because it identifies any target regardless of their alliance. Using white allows greater flexibility in the target reticle's functionality because the shape and color of the entity's locator already provides alliance information.

In addition, this neutral color provides good contrast with the dark background in the virtual environment, which makes it easy to see selected target.

A target reticle is 100 km in diameter and its shape is that of a wire frame sphere. This design does not increase the visual clutter in the interface. Figure 21 shows a target reticle identifying a selected target from a group of similar targets.

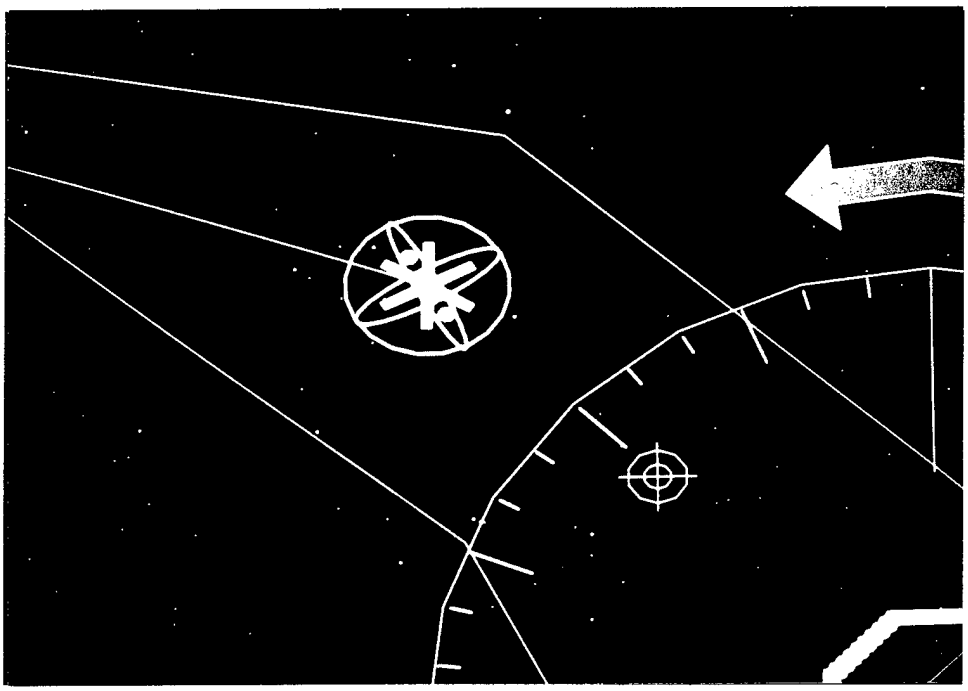

Figure 21: Target Reticle

ANIMATED MIMICS

An animated mimic is a display device that presents the direction and rate of flow for a fluid or motion. Perhaps the single most important goal in display design is the mapping between the underlying domain and the visual representation of that domain provided by a graphic display. For animated functional mimic displays (Figure 22) this translates into the relationship between the physical rate of flow that exists in the

underlying domain and the subjective impression of flow that results from viewing the display [Bennett95].

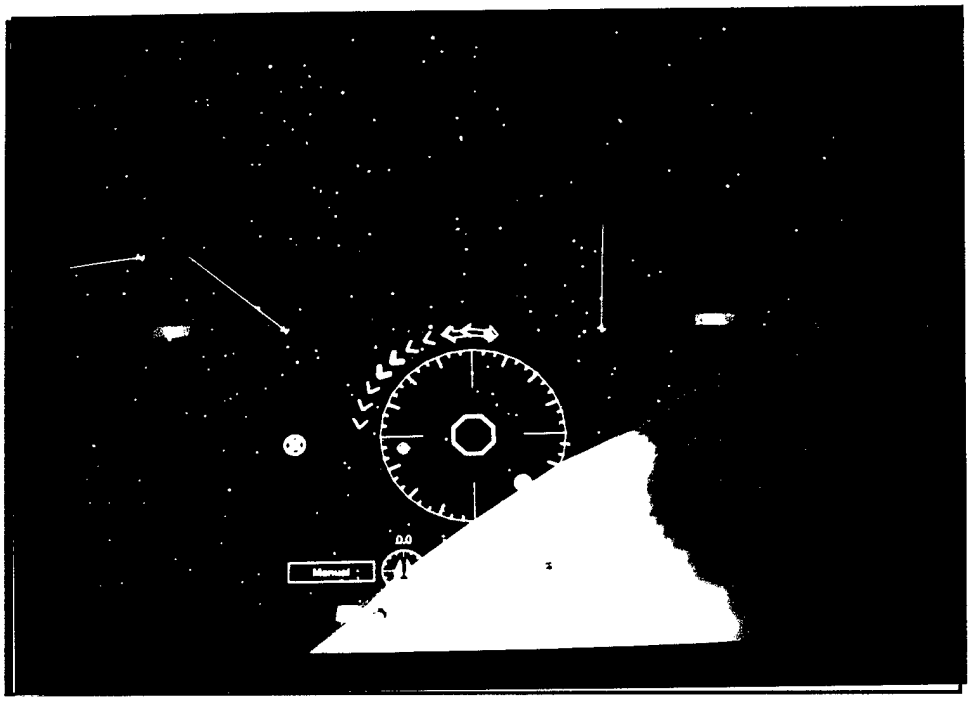

Figure 22: Animated Mimic

Our animated mimic consists of a sequence of arrows in the direction of flow (in our case direction of roll). Arrows (angled contours) provide a visual cue that reinforces the direction of motion [Bennett95]. We create the waveform, the animation part of the mimic, by changing the pixel size from one to three and the color of each arrow sequentially in the direction of roll. White is the color of the waveform and it provides enough contrast to distinguish it from the green arrows. Velocity is indicated by the number of degrees (visual angle) a waveform moves during a one-second interval (degrees/second), which provides a visual cue about the rate of roll. Although there are no specific values for the rate of roll, we find that users readily correlate the movement of the waveform with the rate of roll.

CONSTELLATIONS

Constellations are a grouping of bright stars appearing on the celestial sphere and named after religious or mythological figures, animals, or objects. For the virtual spaceplane, constellations provide an aesthetic view for anyone interested in astronomy and serve as a backup navigation system (celestial navigation). Figure 23 shows some of the constellations in the virtual environment.

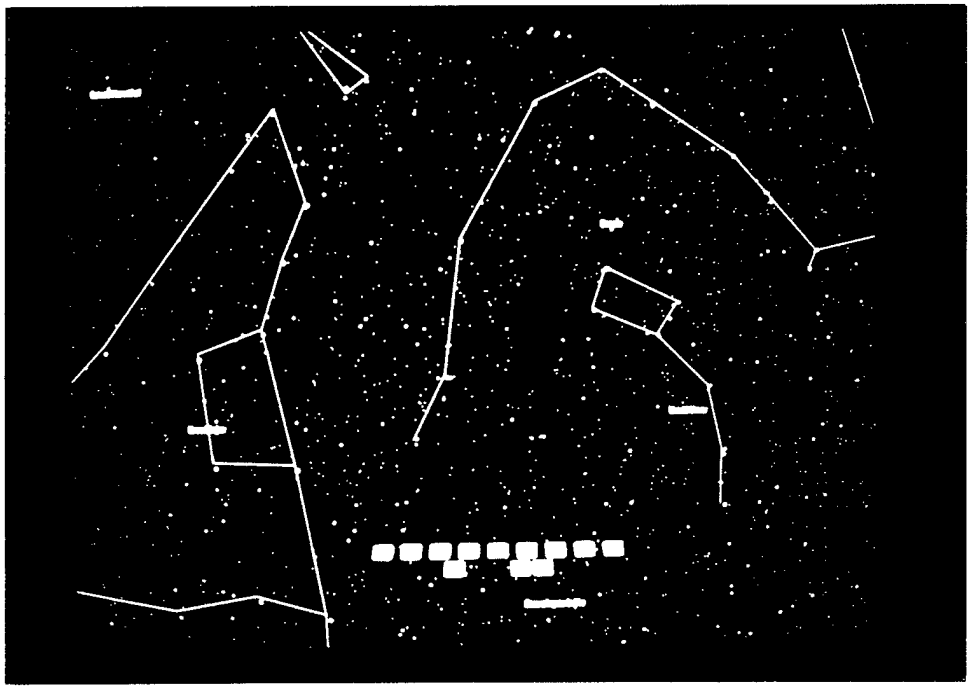

Figure 23: Constellations

We use blue as the color code for constellations for two reasons. First, they are used for celestial navigation, which implies direction. Second, blue appears subdued against the black background. Because of the large number of constellations in the simulation, we do not want them to be distracting. The blue lines and text against the black background does not prevent the user from seeing constellations. This color scheme also

does not distract the user's attention when directed to something else in the virtual environment.

The names of the constellation have a unique feature different from any other text in the virtual user interface. No matter what orientation the viewpoint, they always appear right side up to the user. This functionality, which exists in the Performer graphics suite [SGI95], ensures that we never get to see upside down constellation names regardless of our orientation.

WAYPOINTS

Waypoints are intermediate points along a route of flight. Our autopilot system uses waypoints to guide the virtual spaceplane along a predetermined flight route. Figure 24 shows what our waypoints look like.

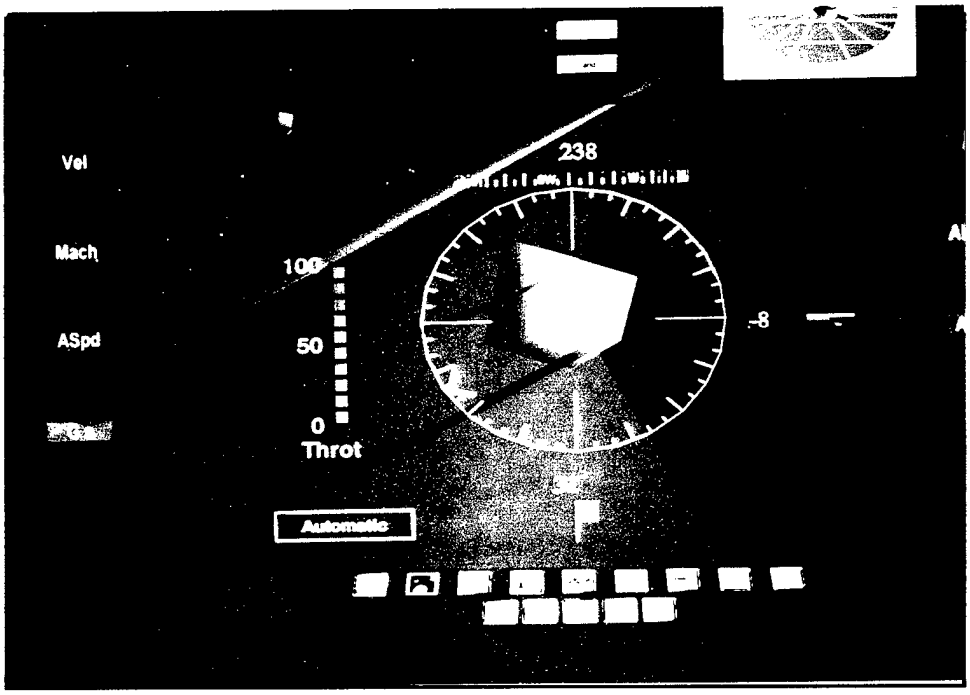

Figure 24: Waypoints

A waypoint is symbolized with an animated diamond structure positioned at a latitude, longitude, and mean sea level altitude. We chose a diamond structure because many of the current waypoint navigation systems use a similar structure. Table 13 lists a waypoint's attributes and a description of each.

Table 13: Waypoint Attributes

Attribute	Description
30% Opaque	Subdued against the background environment
Dark Blue	Color code for indicating direction
Highly Specular	Emulates a beacon effect
Rotating	Animation creates the illusion of an actual object
Lat/Long/Alt Positioning	Easy to position in the virtual environment. Autopilot system uses the same position values

Waypoints are not for visual flight navigation. The primary reason for using them is to prevent surprising the user when the autopilot system executes a maneuver. As the virtual spaceplane approaches a waypoint, the user can expect a change in direction and orientation towards the next waypoint.

CONCLUSION

The first half of this chapter presented the architecture of the virtual user interface and how it exploits various Performer routines and class methods to handle interaction input and output. In the second half, design specifications were presented that describe the various components of the virtual user interface. We addresses the overall interface layout, standardized color-coding, and the design and functionality of various virtual widgets. In Chapter Five, Implementation, we discuss the implementation of these design specifications into the virtual user interface.

CHAPTER FIVE - IMPLEMENTATION

INTRODUCTION

With the architecture and specifications in place, all that is left is putting the user interface together in a manner that ensures compliance with the heuristic guidelines of Discount Usability Engineering. This chapter provides a description of the design process used in the development of the overall virtual spaceplane. In addition, we present a summary of the interaction data flow, which describes how the architecture manages user interaction. Finally, we present the steps and decisions made concerning the implementation of the virtual environment user interface.

IMPLEMENTATION

DESIGN PROCESS

The virtual user interface was implemented and evaluated using a rapid prototyping process. Our most important requirement in the design process for the virtual environment user interface is the ability to make rapid and drastic changes in its design and functionality without interfering with other components of the simulation. This ability is due in large part to the architecture developed by Rothermel [Rothermel97]. The second most important component of designing the virtual user interface is the synch and stabilize design approach used by all three members of the design team. Synch and stabilize is a design approach where we continually synchronize what we do as individuals and periodically stabilize the program in increments as the project proceeds,

rather than once at the end of the project [Cusumano97]. Each member begins with a copy of the program for designing components associated with their respective areas in the research. By implementing changes into our own copy of the program we can quickly debug and prototype the components of the simulation. When each of us reached a point in the design process where we feel our prototype is complete and approve of its performance we would integrate all three versions into a single demonstration copy. We would then continue on our part of the program design using the new copy of the demonstration. This process insures we have a stable program to work with after each integration and once the project is complete.

INTERACTION DATA FLOW

Figure 25 is a data flow diagram, which shows the route information travels between the various classes. Users initiate commands or actions through interaction with the virtual user interface. This interaction occurs with a three-button mouse, a keyboard, or an electromagnetic tracking device for detecting head motion. Arrows indicate the flow of information between the different classes. Output is a visual presentation to the user via state changes in the cockpit or in the virtual environment.

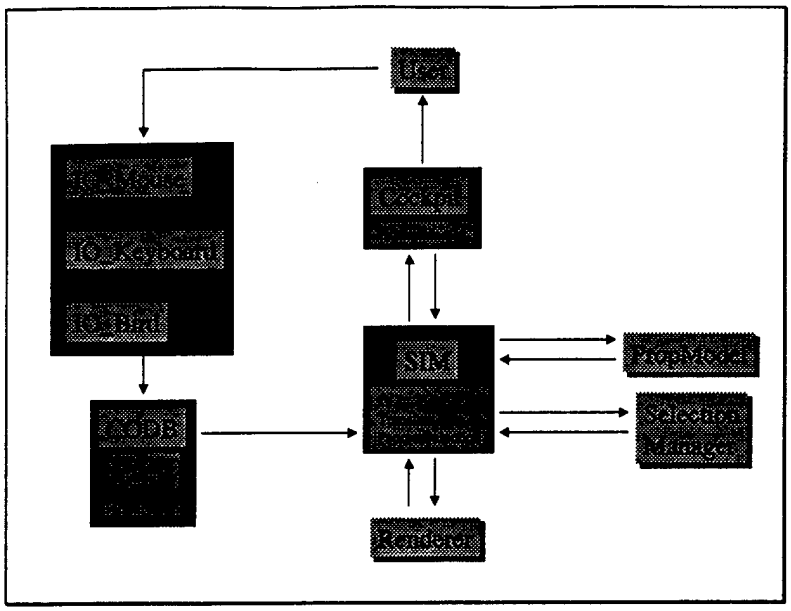

Figure 25: Cockpit Data Flow Diagram

Common Object DataBase

With respect to the virtual user interface, the CODB handles inter-component communications between input devices (mouse, keyboard, and tracker) and the application. Using the CODB to manage input devices decouples them from the main simulation loop so the application can retrieve input information from the CODB and not wait on the input device.

IO_Mouse

The IO_Mouse class links the Performer mouse event handling routines to the CODB. ***Poll ()*** is the primary method within IO_Mouse, which the Sim class calls every frame to update the CODB mouse container (Figure 26). ***Poll ()*** first checks to see if the mouse cursor is in the window and then processes any mouse events with a call to ***processMouseInput ()***. ***processMouseInput ()*** opens the CODB mouse container and

updates its variables. If the user does not initiate a mouse click, then we only update the mouse cursor position. If the user does initiate a mouse click, then the variable *button* gets the token value associated with the pressed button.

```
struct mouse_struct
{
     float     x, y,;                  // x, y relative to the lower left window corner
     int       xpos, ypos;             // x, y relative to the window center
     int       buttons;                // value of pressed button (LEFT, CENTER, RIGHT)
     boolean   in_window;              // TRUE if cursor is in window, FALSE otherwise
     float     winx, winy;             // window size
     float     normalx, normaly;       // normalized x, y wrt the lower left window corner
     pfuMouse  *lastmouse;
};
```

Figure 26: CODB Mouse Container

Table 14 identifies the functionality assigned to each mouse button. Clicking on objects is performed with the left mouse button because there are many existing applications that already have this type of interaction; e.g., word processors. Consequently, this form of interaction will be easy to learn and remember. Changing viewpoint orientation is done with the middle mouse button. The right mouse button is for adjusting the viewpoint's field of view.

Table 14:Button Assignments

Mouse Button	Functionality
Left	Selection
Middle	Changing Viewpoint Orientation
Right	Field of View Adjustments

IO_Keyboard

IO_Keyboard is a class linking the Performer keyboard event handling routines to the CODB. ***Poll ()*** is the primary function in IO_Keyboard, which the Sim class calls every frame. ***Poll ()*** checks to see if there is a keyboard event and then processes it with a call to ***processKeybdInput ()***. If there is no keyboard event, the IO_Keyboard opens the

CODB keyboard container (Figure 27) and sets its variables to zero. If there is a keyboard event, IO_Keyboard assigns values to the variables in the keyboard container based on the token value of the pressed key.

```
struct keyboard_struct
{
     short     value;          // token value of the pressed key
     long      device_num;     //
};
```

Figure 27: CODB Keyboard Container

IO_Bird

The IO_Bird class provides functionality for the Ascension Flock of Birds™ tracking device. This class is different from the other two Input Modifiers because the process is forked off and runs in the background separate from the rest of the simulation. This allows IO_Bird to update the CODB bird container (Figure 28) faster than the simulation frame rate thereby ensuring the container always has the latest information. ***OpenBird ()*** is the initialization method, which the Sim class calls during the startup initialization. It creates the CODB bird container, initializes the forked process, and returns the process identification number. Sim then calls the ***Calibrate ()*** method and initializes the position and orientation of the Ascension receiver relative to its transmitter. ***FlyBird (Bird *mybird)*** takes an argument of type Bird, which the Sim class calls after initialization. After the initial call, its process forks off to run in the background. It then goes into a loop and updates the CODB bird container with the position and orientation of the receiver with respect to the transmitter, which is the raw data. ***GetRawPosAndOri ()*** reads the raw position and orientation of the receiver from the CODB bird container. ***GetOrientation ()*** and ***GetPosition ()*** use the data retrieved by ***GetRawPosAndOri ()*** and

converts it to the viewpoint position and orientation relative to the virtual environment. Sim calls these two functions once per frame and passes their return values to the Renderer class.

```
typedef struct
{
     pfVec3    xyz;    // viewpoint position
     pfVec3    hpr;    // viewpoint orientation
} bird_struct;
```

Figure 28:CODB Bird Container

Renderer.

For the purposes of user interaction data flow, the Renderer manipulates the viewpoint orientation and field-of-view whenever the user presses the middle and right mouse buttons respectively. Changing the viewpoint orientation simulates head motion within the cockpit. Changing the field of view provides a zoom capability.

Selection Manager

Our Selection Manger automates the use of Performer's ***pfChannel::pick*** function by returning the index value assigned to a selected object. ***Process_pick (const float mx, const float my)*** is the main function in this class. It first makes a call to the pfChannel function ***pick (int mode, float px, float py, float radius, pfHit *picklist)***, which is for screen to world-space ray intersections on a pfChannel's scene. Intersections will only occur with parts of the database that are within the viewing frustum, and enabled for picking intersections. ***pfChannel::pick*** returns the number of successful intersections with the channel scene according to *mode*. *Mode*, a bitwise OR of tokens, specifies the behavior of the scene tree traversal and the type of information returned from the picking

process. Variables *px* and *py* identify a two-dimensional point in normalized screen coordinates in the range 0.0 to 1.0 (with the lower left corner being (0.0, 0.0)), that corresponds to the channel location used for picking. This two-dimensional point is for creating a ray from the viewer eyepoint through the near clipping plane to intersect with the channel scene. *Radius* is the radius of the picking region in normalized channel coordinates used for the picking of lines. We currently have no need to pick lines, therefore this value is zero. Upon return from the function, *picklist* (a user supplied pointer) gets the address of an array of pointers to the selected objects. If the return value is greater than zero, the ray segment intersected is a valid structure in the scene and the variables of the selection structure (Figure 29) get the new variables given in *picklist.* If the return value is less than or equal to zero, there was no geometry selected. Sim uses the function ***GetIndex (...)*** to retrieve the selected object's index value and hitpoint. A hitpoint is the point of intersection on the selected geometry.

```
typedef struct
{
     pfNode  *picked_node;     //identifies the node intersected by the ray
     pfPath  *picked_path;     //identifies the scene path of the intersected node
     char    *picked_pathname; //a character array containing the name of the path string
     pfVec3   picked_hitpoint; //3-D pt of intersection wrt the node's local coord system
     short    item_hit;        //state variable (1 for a hit, 0 for a miss)
} picks;
```

Figure 29: Selection Structure

Sim

Everything that happens in the simulation, except initialization and destruction, resides with the Sim class method ***GO ()*** (Figure 30). This method contains the main simulation loop, which is the heart of any virtual environment.

```
void Sim::GO ()
{
     .
     .
     .
     while (notDone)
     {
          //Signal Input Modifiers to update their respective CODB containers
          my_keyboard->poll ();
          my_mouse->poll ();

          if (useBIRD)
              my_renderer->SetHeadPosOri(my_bird->GetPosition (),
                   my->GetOrientation ());
          .
          .
          //Translate input values into commands
          TranslateInputs ();
          .
          .
          Update () //Updates all entities in the virtual environment
          .
          .
          my_renderer->FinalizeSceneGraph ();
          .
          .
          pfSync ();
          pfFrame ();
          .
          .
     } // End of while loop
} // End of Sim::GO
```

Figure 30: Sim Class **GO ()** *Method*

Every repetition of this loop begins with the input devices updating their appropriate CODB containers as mentioned previously. Once the Input Modifiers update the CODB containers with the latest user input, the next method called is ***TranslateInputs ().*** ***TranslateInputs ()*** is a Sim method that manages input and output between the input devices, the virtual user interface, and the simulation components. Next, all the entities in the virtual environment update themselves in order to modify their geometry or propagate through the virtual environment. Renderer then rebuilds the scene graph with a call to ***FinalizeSceneGraph ()*** before allowing the Performer routines ***pfSync ()*** and ***pfFrame ()*** to render the current frame.

Just as ***GO ()*** is the heart of the virtual simulation, ***TranslateInputs ()*** is the heart of the virtual user interface. Appendix B, TranslateInputs Method, contains an abbreviated

version of ***TranslateInputs ().*** ***TranslateInputs ()*** first opens the CODB keyboard container and then uses the container variable value to invoke the necessary callback functions. Callback functions are grouped according to a specific index value and when the parameter *value* equals an index value, ***TranslateInputs ()*** invokes those particular callback functions. The CODB keyboard container closes after completion of the last callback function. Next, ***TranslateInputs ()*** opens the CODB mouse container. Because the function uses all three mouse buttons, it must determine which button the user has pressed.

The first check is on the left mouse button. Then ***TranslateInputs ()*** asks the Selection Manager to determine the index value and hitpoint of a selected object (if any). This involves passing the normalized x and y window coordinates to the Selection Manager. If we have a valid index, meaning the user selected an object in the scene, we invoke a call to ***ProcessSelection (index, hitpoint).*** This method uses the object's index value or hitpoint to invoke methods for changing the state of the cockpit or to change other parts of the virtual environment.

If the user presses the middle mouse button anywhere in the window, the Renderer re-orients the viewpoint based on the mouse inputs. The user changes the orientation of the viewpoint by dragging the mouse cursor in the direction they want to look. This functionality allows the design team to create a user interface that accounts for the user's head movement.

If the user presses the right mouse button, the Renderer class updates the field of view. Changing the field of view provides a zoom capability, which is quite useful for viewing objects far away in the virtual environment. Pressing the right mouse button and

dragging the mouse forward decreases the field of view. Dragging the mouse backwards increases the field of view. Once we complete the third button check, we close the CODB mouse container.

IMPLEMENTATING THE VIRTUAL USER INTERFACE DESIGN

Aero-Flight Control System

The aero-flight control system is a group of interface components directing the operations and information associated with atmospheric flight operations. It allows the user to manually change the direction and speed of the virtual spaceplane during atmospheric flight. In addition, it also functions as a heads up display to provide information to the user while allowing them to keep their eyes on the virtual environment outside the cockpit.

Figure 31 shows what the virtual spaceplane's aero-flight control system looks like. Its position is approximately 40 cm in front of the user at eye level. The color scheme is green, except for the status indicators, because it stands out exceptionally well during day and night simulations. We divide the information on the aero-flight control system into eight categories based on functionality and type of information displayed.

- 8 status indicators
- A control interface for manually changing the direction and orientation
- A control slider for manually adjusting the throttle
- Pitch and roll indicators
- A compass dial
- Instrument Landing System(ILS)
- Flight indicator

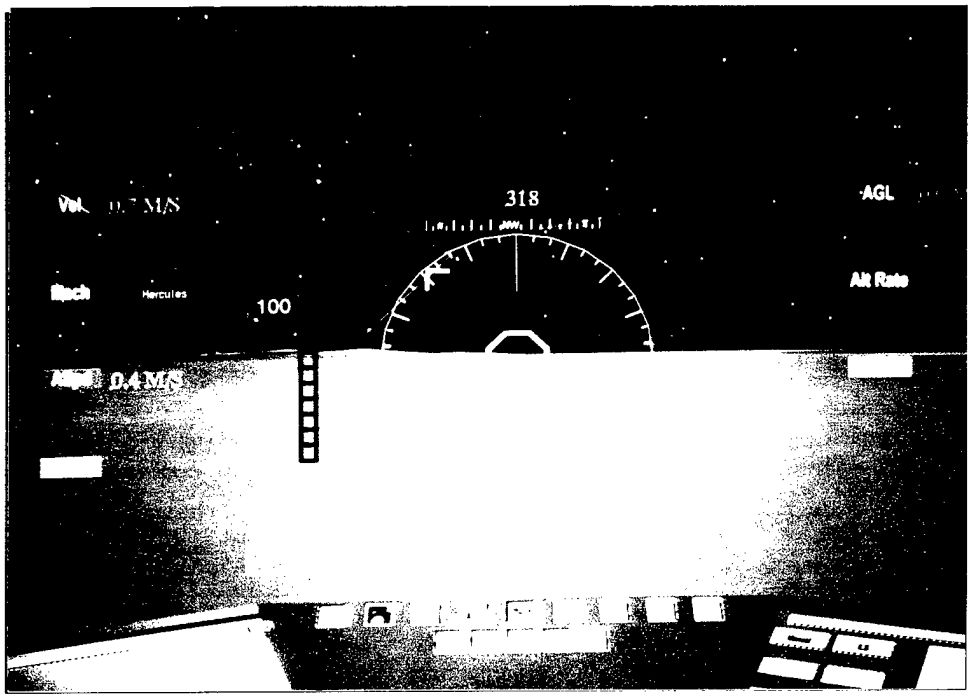

Figure 31: Aero-Flight Control System

Status Indicators

Status indicators are displayed on both sides of the aero-flight control system and split into two categories. The four status indicators on the left display the virtual spaceplane's flight characteristics, which are aircraft velocity, Mach, airspeed, and G forces. On the right are the remaining four indicators displaying the virtual spaceplane's vertical profile (altitude above ground level (AGL), altitude mean sea level (MSL), altitude rate, and altitude acceleration). By organizing the indicators in this manner, the user can quickly learn the positions of all eight status indicators by remembering the two categories (flight characteristics and vertical profile) [Nielson93][Marcus90a]. Both the AGL and MSL status indicators share the same position on the aero-flight control system. When the virtual spaceplane is below 3,000 feet AGL, we show the AGL status indicator. Above this level, the aero-flight control system displays the MSL status indicator. This design emulates the functionality of the radar altimeter currently in use on the space

shuttle [NASA94]. Some of the status indicators have minimum and maximum state values to determine when their respective variable becomes out of tolerance (Table 15). These values are an estimate based on our visual perception while using the system.

Table 15: Maximum and Minimum Tolerance levels

Status Indicator	Minimum Tolerance	Maximum Tolerance
Velocity	200	N/A
Mach	0	50
Airspeed	100	N/A
G Force	-1	8
Above Ground Level	100	N/A
Mean Sea Level	100	N/A
Altitude Rate	-100	N/A
Altitude Acceleration	-100	N/A

Control Interface

In the center of the aero-flight control system is the control interface. It consists of a grid (transparent circle), a position indicator (octagon), and a heading indicator (arrow). This design simulates the handling characteristics of a conventional "stick." Pressing the left mouse button on the grid and holding it down activates the maneuvering controls. As the user moves the mouse, the position indicator moves, simulating the movement of a stick. Rolling the virtual spaceplane to the right and left is the same as flying a conventional aircraft, but pitching up and down is opposite. To pitch the virtual spaceplane up, the user drags the position indicator to the top of the grid. Pitching down requires a downward motion.

Around the inner edge of the control grid is a heading indicator. As the virtual spaceplane changes direction, the heading indicator shows which direction the virtual spaceplane is heading by changing its position along the inner edge of the control grid. For example, if the heading is 350 degrees, the heading indicator is positioned 10 degrees to the left of the control grid centerline. As the virtual spaceplane changes heading to

true north, the heading indicator moves closer to the control grid centerline. When it is lines up with the centerline, the virtual spaceplane is heading due north. Our heading indicator uses the control grid like the face of a compass. Major lines on the control grid represent (clockwise from the top) north, east, south, and west. We use an arrow as the heading indicator because it symbolically represents direction.

The aero-flight control system is directly in front of the cockpit. This position provides a visual indication on the orientation of the virtual space plane. Consequently, the user will always know what direction the virtual spaceplane is facing. Our original idea was to have the aero-flight control system remain fixed in front of the user's viewpoint. As the user moved their head, the aero-flight control system moved with the viewpoint. This implementation created disorientation and made it difficult to find the front of the virtual spaceplane.

Control Slider

To the left of the control grid is the throttle control slider. Its range is from 0 to 100 percent, which is consistent with current aircraft functionality. Originally, the throttle control slider was horizontal and above the control grid. As the user increases throttle to the engines, the slider moved to the right. Perceptually, this creates a visual disconnect with current methods of throttle functionality. Pilots in a conventional aircraft push a throttle either forward or up to increase engine thrust. To maintain consistency the throttle slider is now vertical and to the left of the control grid. We chose the left side of the control grid because the throttle in military aircraft is to the left of the pilot.

Pitch/Roll Indicators

Below and to the right of the control grid is the roll and pitch indicator respectively. Both indicators have a slider and a numeric value. As we roll the virtual spaceplane, the roll indicator's slider increases in the direction of the roll. Its maximum value, in degrees, increases positively up to 180 degrees as we roll right and decreases negatively to minus 180 degrees as we roll left. As we pitch the virtual spaceplane, the pitch indicator's slider increases in the direction of pitch. Its maximum value, in degrees, increases positively up to 90 degrees as we pitch up and decreases negatively to minus 90 degrees as we pitch down. Both indicators use the system status color code to indicate when the virtual spaceplane is beyond recommended pitch and roll values. Table 16 lists these values.

Table 16: System Status Constraints for the Pitch and Roll Indicators

Color Code	Pitch Range	Roll Range
All Clear	0 to ±45	0 to ±45
Caution	±45 to ±75	±45 to ±90
Warning	±75 to ±90	±90 to ±180

Compass Dial

Above the control grid is a compass dial, which presents the true heading of the virtual spaceplane. The current heading is in the center of the dial directed towards the user's viewpoint. Letters on the dial show the direction relative to north, south, east, and west directions. Above the dial is the numeric value of the heading, in case the user needs to know the exact heading.

Instrument Landing System

An instrument landing system provides glide slope and bearing information to users trying to manually land the virtual spaceplane. The instrument landing system consists of

a glide slope line and a bearing line on the control grid (Figure 32), and a landing corridor in the virtual environment (Figure 33). A glide slope line is a horizontal line on the control grid. It indicates whether the virtual spaceplane is above or below the glide slope. The glide slope for the virtual spaceplane extends from the runway landing point at a five-degree angle up to the top of the atmosphere; i.e., 30 kilometers. When the glide slope line centers on the control grid the virtual spaceplane is directly on the glide slope. If the glide slope line is above (below) the center of the control grid, the virtual spaceplane is below (above) the glide slope. A bearing line is a vertical line on the control grid indicating whether the virtual spaceplane is to the left or right of the runway centerline. When the bearing line is in the center of the control grid, the virtual space plane is directly on runway centerline. If the bearing line is to the left (right) of the center of the control grid, the virtual spaceplane is to the right (left) of the runway center.

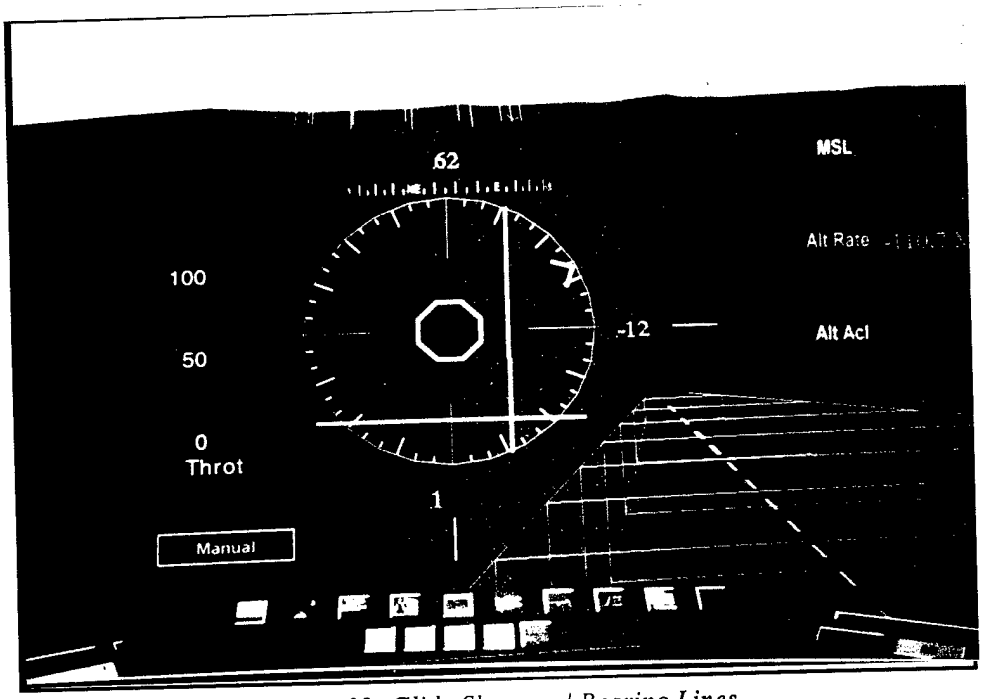

Figure 32: Glide Slope and Bearing Lines

A landing corridor (Figure 33) is a virtual tunnel positioned at the approach end of the runway and extends along the glide slope out to a distance of seven kilometers. Its purpose is to provide the user with a three-dimensional flight path to the runway. During landing operations, the user flies the virtual spaceplane into the landing corridor and follows the centerline. We use transparent single purpose color coding for warning, caution, and all clear. When the virtual spaceplane is completely inside the corridor, all the sides are green. When the virtual spaceplane intersects a side, that particular side turns yellow. When the virtual spaceplane is outside the corridor, the side it is closest to turns red. Its transparent design prevents obstructing the view of the virtual environment. An animated centerline down the middle of the landing corridor simulates the landing lights. To turn the instrument landing system on or off, the user simple presses the "ILS" button located on the aero-flight panel.

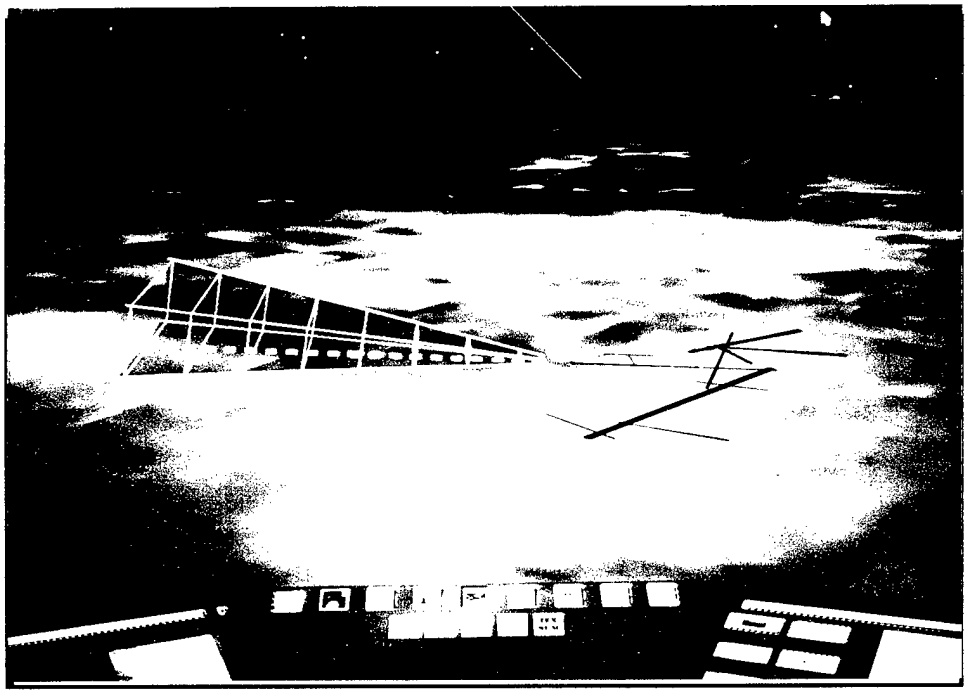

Figure 33: Landing Corridor

Flight Indicator

Below and to the left of the control grid is the flight indicator. It provides a visual cue to the user indicating which flight mode the virtual spaceplane is currently in, manual or automatic. When the virtual spaceplane is in manual flight, the flight indicator shows "Manual" and the user has complete control over direction and speed. When the autopilot is on, the control position indicator disappears, the selection process disengages the throttle, the flight indicator shows "Automatic," the user has no control over direction and speed. For more information about the functionality of the autopilot, refer to Johnson [Johnson97].

Space-Flight Control System

The space-flight control system is a group of interface components directing the operations and information associated with space flight operations and allows the user to manually change the orientation of the virtual spaceplane. In addition, the space-flight control system also functions as a heads up display to provide information to the user while allowing them to keep their eyes on the virtual environment outside the cockpit.

Figure 34 presents the virtual spaceplane's space-flight control system. Its position and color scheme is the same as the aero-flight control system. We classify the information on the space-flight control system into five categories based on functionality.

- 2 Status indicators
- A control interface for manually adjusting the heading and pitch
- A roll change interface for manually adjusting the roll
- 2 pointers
- Pitch, roll, and heading indicators
- Flight indicator

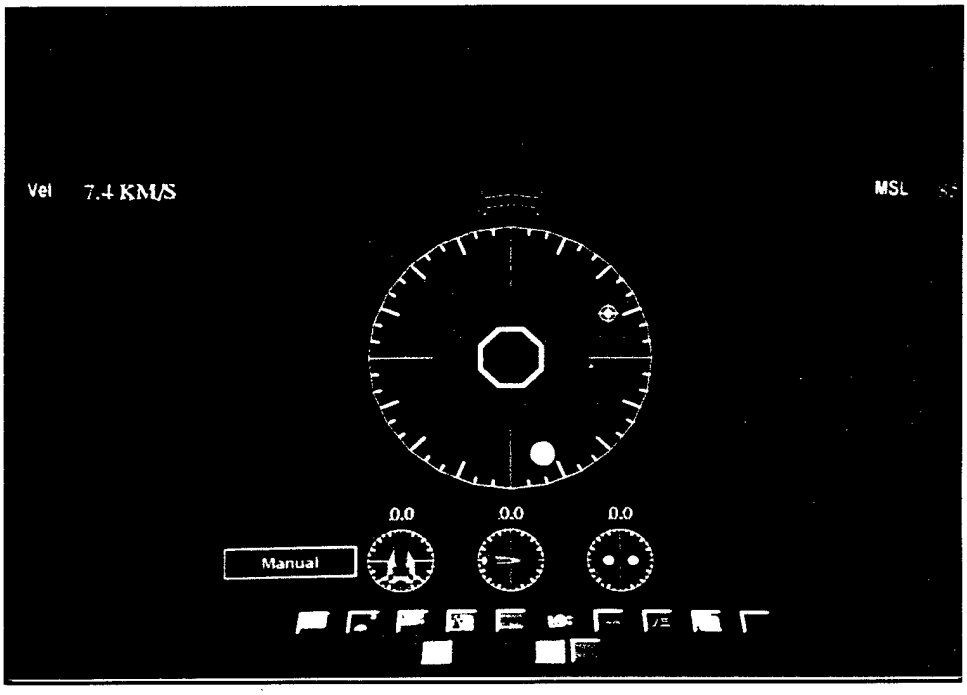

Figure 34: Space-Flight Control System

Status Indicators

There are two status indicators, one on each side of the control system. On the left is the velocity status indicator and on the right is the altitude mean sea level status indicator. While developing the virtual spaceplane, we determined that only these two were important enough to be on this interface. Their placement within the system is similar to the aero-flight control system (flight characteristics on the left and vertical profile on the right).

Control Interface

In the center of the space-flight control system is the control interface. It looks exactly like the one in the aero-flight control system because we wanted to maintain a consistent look and feel. Controlling pitch interaction with the flight controls is accomplished similarly to the aero-flight control system. However, because of the intricacies of space flight, we implement heading changes in this interface instead of roll

changes. To operate the heading and pitch control, the user clicks and drags the position indicator towards the new orientation. When the user releases the mouse button, the cockpit stops its movement, the control position indicator returns to the center of the control grid, and the virtual spaceplane maneuvers into the new orientation.

Roll Interface

In addition to the heading and pitch changes, we allow the user to change roll with a separate interface control. Heading and pitch changes can orient the virtual spaceplane in any direction so there is no need to incorporate roll with the control grid interface. However, we give the user the ability to change roll manually for whatever reason; i.e., reconnaissance, solar event shielding, etc. Pressing the left mouse button down on the double arrow above the control grid and holding it down activates the roll functionality. After the initial click, an animated mimic appears indicating the direction and rate of role. Dragging the mouse to the right causes the cockpit to roll right and the mimic animates to the right also. As the rate of roll increases (decreases), the animation in the mimic increases (decreases). When the user releases the mouse button, the cockpit stops its roll, and the virtual spaceplane maneuvers to its the new orientation. Similar operations occur for rolling left.

Reality Disconnect

In order to maintain a sense of physical realism and enhance interface functionality we developed a principle of movement called "reality disconnect." In a "reality disconnect" movement, the user orients the cockpit with the space-flight control system in the direction they want the virtual spaceplane to face. The virtual spaceplane does not begin the orientation change until after the user completes orienting the cockpit.

Controlling the orientation of the virtual spaceplane in this manner conserves fuel and eliminates the need for a mission control. Currently, orientation changes in the space shuttle are calculated at mission control and sent to the onboard crew. This information is then input into the control system, which fires the necessary thrusters [NASA94].

Pointers

There are two pointers, a target pointer and an earth pointer, on the inside edge of the control grid. Figure 34 shows the target pointer pointing to the currently selected target and the earth pointer pointing to the center of the earth. Their main purpose is to prevent disorientation by indicating the shortest way to orient the virtual spaceplane to view either the earth or the current target.

Pitch, Roll, and Heading Indicators

Because we use the reality disconnect concept, we need to provide information to the user showing the difference in the cockpit's heading, pitch, and roll from the virtual spaceplane's orientation. This difference is displayed using three orientation indicators below the control grid (Figure 35). All three indicators consist of a circle with radial tick marks representing degrees, a dial, and a numeric value. Each dial is a symbolic representation of the virtual spaceplane from a different perspective. In Figure 35, the dial on the left is for heading changes, the dial in the middle is for pitch changes, and the dial on the right is for roll changes. The numeric value provides an exact measure of orientation change. This design provides better visual communication to the user, incorporating both numerics and graphics, than a gauge or a numeric value alone.

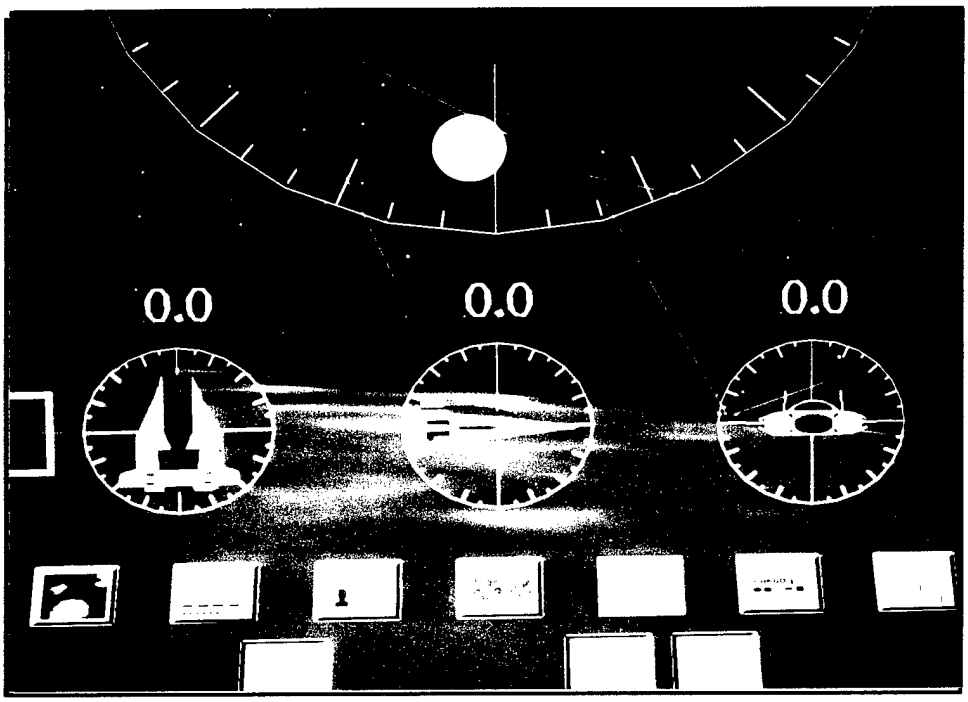

Figure 35: Pitch, Roll, and Heading Indicators

Flight Indicator

Below and to the left of the control grid is the flight indicator. It provides a visual cue to the user indicating which flight mode the virtual spaceplane is currently in, manual or automatic. When the virtual spaceplane is in manual flight, the flight indicator shows "Manual" and the user has complete control over orientation. When the user initiates an intercept or lockon maneuver, the autopilot takeover flight control, the control position indicator disappears and the flight indicator shows "Automatic." For more information about the functionality of the autopilot, refer to Johnson [Johnson97].

Control Panel

A control panel (Figure 36) is a console panel containing a miniature version of either the aero-flight or space-flight control systems. To activate the control panel, the user minimizes the current flight control system using the appropriate toolbar button (see

"Toolbars" section). The current flight control system is then removed from the front of the cockpit and placed on a console panel where it can be positioned anywhere within the confines of the cockpit.

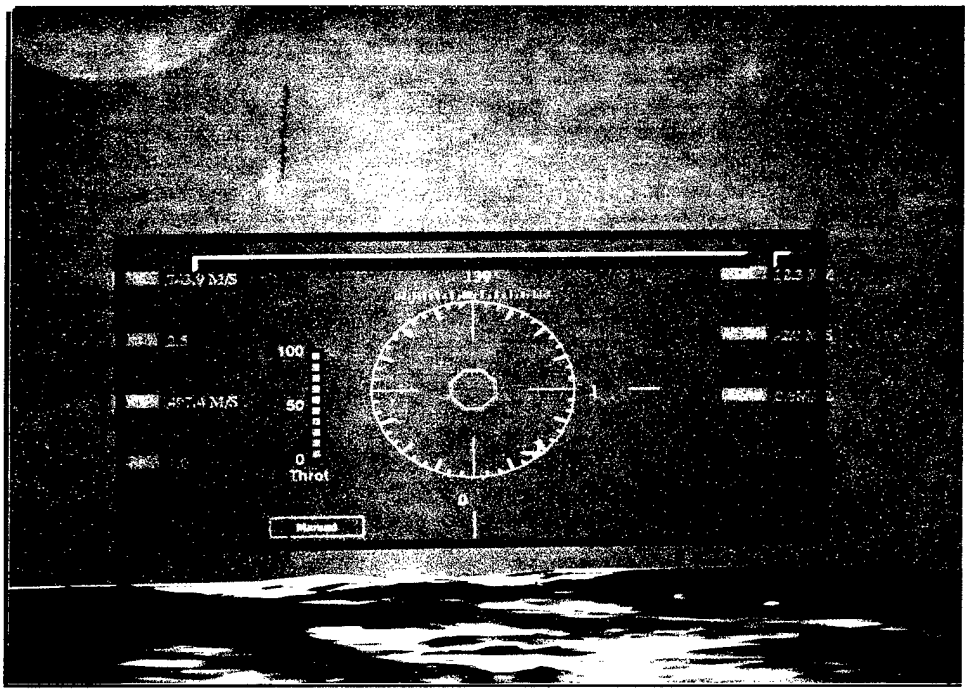

Figure 36: Control Panel

The primary reason for implementing this functionality is to declutter the view to the environment during periods when there is no requirement to manually control the virtual spaceplane. Orientation changes during a standard space flight rarely occur. In between maneuvers, providing flight information or control on a continuous basis may not be necessary.

Aero-Flight Panel

An aero-flight panel (Figure 37) organizes and provides information applicable to atmospheric flight operations. On the left of the panel are buttons, and on the right are displays.

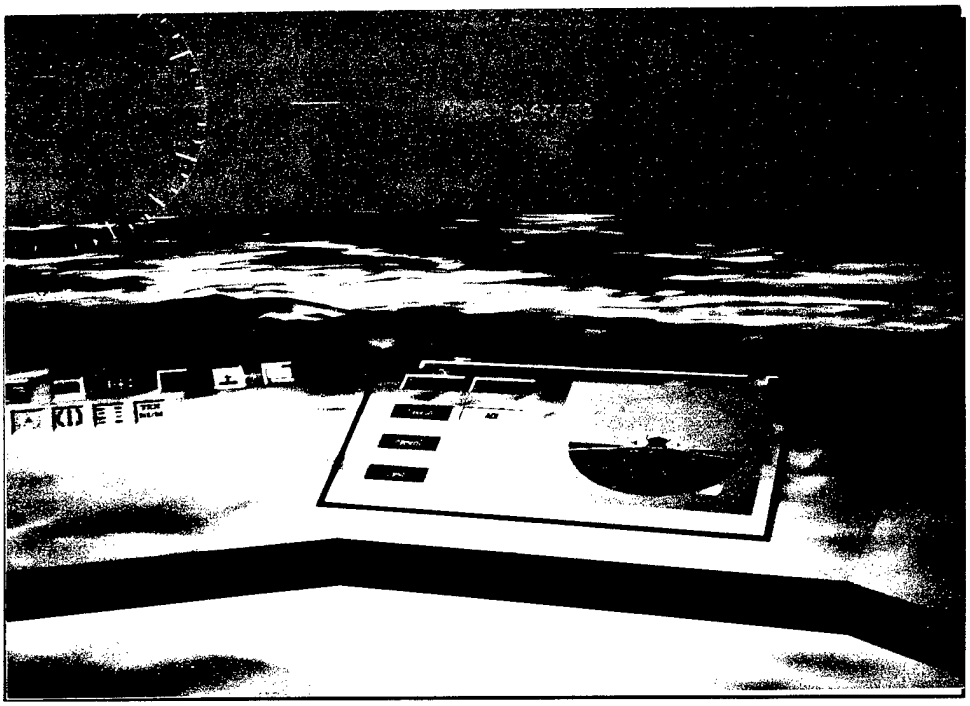

Figure 37: Aero-Flight Panel

Aero-Flight Panel Display

Currently, there is only one display on the aero-flight panel, an altitude-direction indicator (ADI). Wright Labs Advanced Cockpit Branch (WL/FIGD) is developing this particular ADI for use in their glass cockpit research. Our ADI is a hybrid display type because it presents conditional information (warning, caution, and all clear) as well as values (speed, altitude, pitch, roll, and heading). The ADI consists of a heading indicator, a horizontal sky-ground divider, velocity lines, altitude lines, and an aircraft symbol.

Using the display as a 360-degree compass face, the heading indicator positions itself on the display according to the aircraft heading (north is up, south is down, east is right, west is left).

We use an environmental color code for the horizontal sky-ground divider, light blue for sky and brown for ground. This part of the display functions in an inside-out frame of reference (relative motion on the display is what the user would view outside the cockpit)

[Stokes90]. As the aircraft pitches up, the dividing line moves down below the aircraft symbol (which remains stationary) and vice versa.

On the "ground" are velocity lines, which are horizontal lines that scroll down the display providing a sense of motion. As the virtual spaceplane flies faster, the velocity lines scroll faster. Starting from a single point and spreading out towards the user is a set of perspective lines. These perspective lines give the "ground" on the ADI a three-dimensional appearance. Perspective lines, together with the velocity lines, interact with the aircraft symbol to create the illusion that the symbol is flying over a horizontal surface.

Extending from the aircraft symbol are two altitude lines. These lines use the system status color code to warn the user of dangerous altitudes. When the aircraft is on the runway, the lines are horizontal. When the aircraft reaches its maximum altitude (just before its transition into orbit), the lines converge to form a single vertical line.

In the center of the display is the aircraft symbol. This part of the display functions in an outside-in frame of reference (the background is still while the aircraft moves) providing information about the aircraft's roll. We use a photo-realistic image of the rear of the virtual spaceplane to represent the aircraft symbol. The image, coupled with the three-dimensional effect of the ground, creates the illusion that the ADI is actually displaying the virtual spaceplane in flight.

Aero-Flight Panel Buttons

We divide the buttons on the left side of the panel into two sections. On the left is a column of buttons associated with the autopilot system and on the right are buttons for the ILS and ADI. "Autopilot" is a toggle button, which activates or inactivates the

autopilot system. While "Autopilot" is inactive, the remaining three buttons ("Takeoff," "Pattern," and "Land") are disabled (Figure 37), which indicates there are no autopilot options available to the user. When the user activates "Autopilot," we enable the remaining three buttons (Figure 38), which indicates that the options are now available but none chosen. Because these are radio type buttons, a user can activate only one of these autopilot systems. "Land" automatically lands the virtual spaceplane on the runway. "Takeoff" automatically launches the VSP down the runway and up to the orbit entry point. "Pattern" automatically flies the VSP in an orbit pattern around the launch point. In the right column are the "ILS" and "ADI" toggle buttons that allow the user to activate the instrument landing system and the altitude direction indicator. Activating the "ILS" button activates the ILS, which displays the bearing and glide slope lines on the aero-flight control system as well as the landing corridor. A user can activate the altitude direction indicator by toggling the "ADI" button.

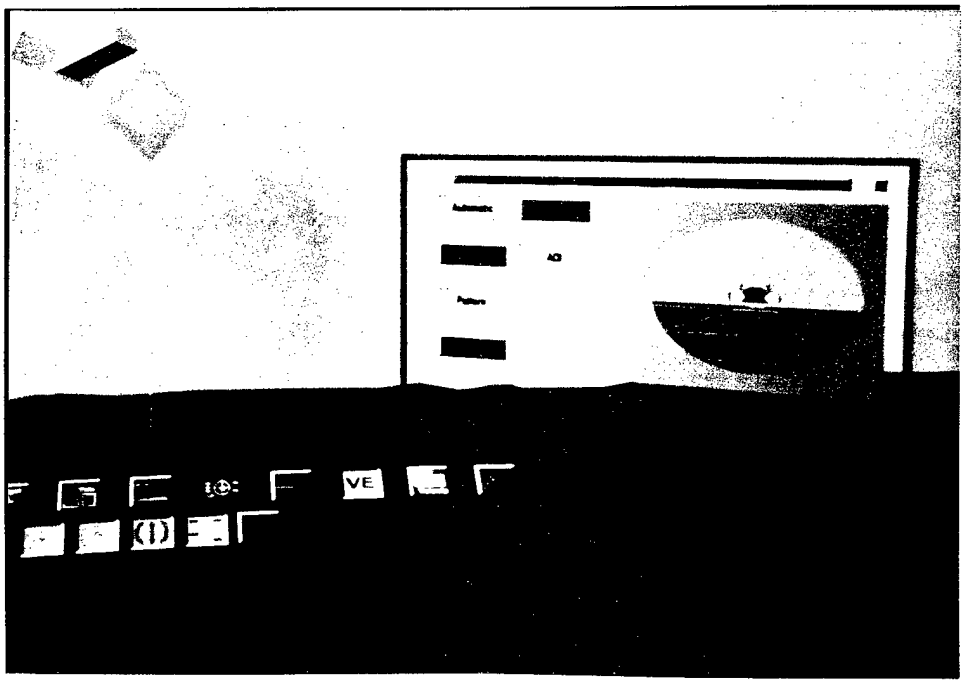

Figure 38: Autopilot Active

Space-Flight Panel

The space-flight panel (Figure 39) organizes and provides information applicable to space flight operations. Its purpose is to present at a glance basic orbital data concerning the virtual spaceplane, the currently selected target, and orbit information for intercepting a target. There is a three-dimensional orbit display in the center of the panel and three buttons on the right, which control the way this display presents information.

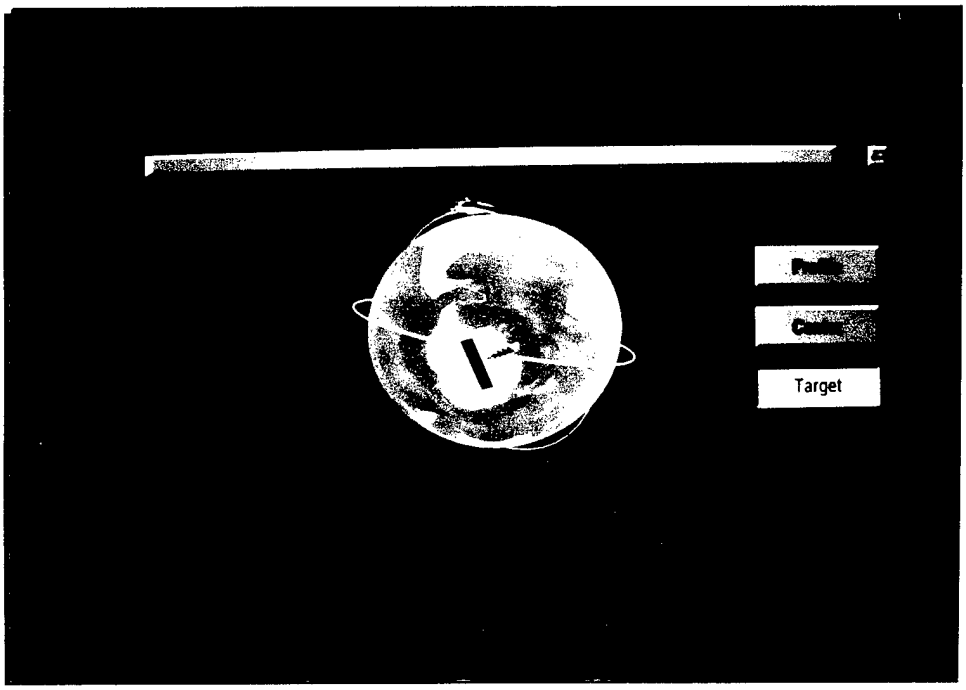

Figure 39: Space-Flight Panel

Space-Flight Panel Display

The orbit display is a hybrid display type, which presents conditional information (warning, caution, and all clear) about the virtual spaceplane's orbit and graphical representations of values (orbit altitude, position, direction, and orientation). Geometry for the orbit display consists of a three-dimensional sphere representing the earth; i.e., a

globe, display locators and orbital paths for the virtual spaceplane and current target, and a transition orbit.

Our three-dimensional globe functions as a map and a trackball. Using a map texture on the sphere provides a reference for determining the location and orbit trajectories for the virtual spaceplane and the current target. This display also provides a perspective on the current target location with respect to the virtual spaceplane's location. Including trackball functionality into the orbit display gives the user the ability to re-orient the display so they can view the entities and their orbits from a variety of perspectives. To re-orient the display, the user clicks on the globe and drags the mouse cursor in the direction they want it to roll.

A display locator is a miniature replica of the entity's geometry, not a symbolic representation. By using a replica and making its orientation and movement correspond to the actual motion in the virtual environment, we create the illusion that the user has a "gods-eye" view of earth. We calculate locator positions on the orbit display using the entities actual position in the virtual environment. AstroPropModel calculates the orbit trails by time stepping the entity's future location through one orbit cycle. These future locations become the coordinates for connecting the lines segments representing the orbit. There is also a transition orbit, which shows the route of flight the virtual spaceplane must take to intercept the currently selected target. This transition orbit does not appear on the display until the user selects a delta V option from the target panel. To change the transition orbit, the user selects a new delta V option.

Space-Flight Panel Buttons

There are three buttons on the right of the panel that allow the user to manipulate the orbit display ("Center," "Profile," and "Target"). "Center" button is a push to activate button that orients the display so the virtual spaceplane locator is facing the user. "Profile" button is a push to activate button that orients the display so the virtual spaceplane's orbit is parallel to the panel base. "Target" button is a toggle button that turns the current target's orbit trail and locator on or off.

Navigation Panel

A navigation panel (Figure 40) provides information about the location and direction of the virtual spaceplane with respect to the surface of the earth. The navigation panel also presents information about the location and type of various ground-based entities. Since it is physically impossible to view all the ground targets because of the earth, this panel provides the capability to view all the targets and see where the virtual spaceplane's position is with respect to the ground and the targets. A map display is located in the center of the panel with three buttons on the left.

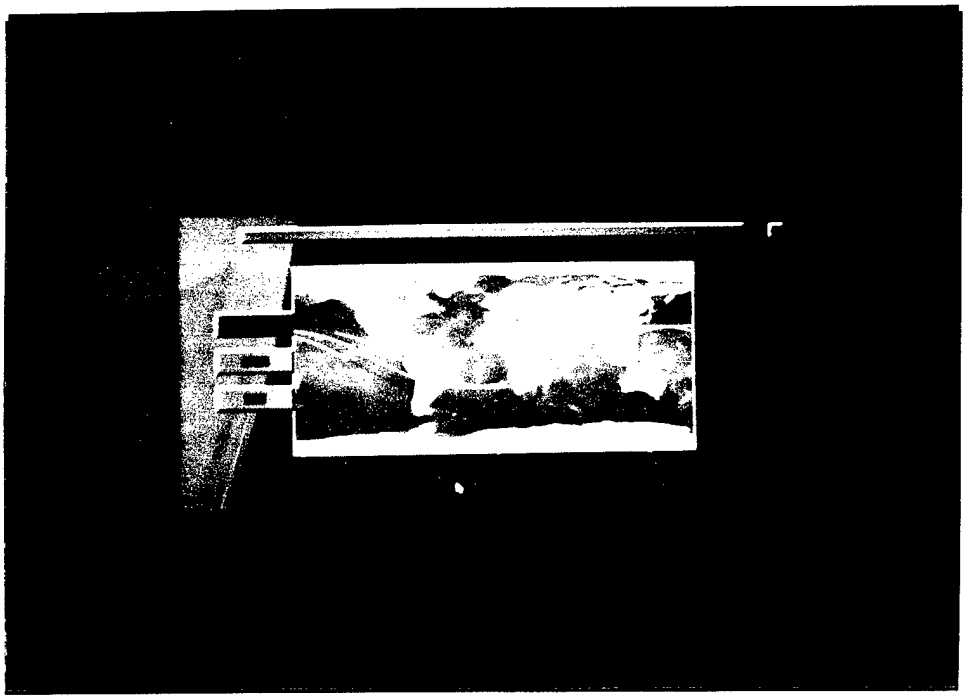

Figure 40: Navigation Panel

Navigation Panel Display

Our map display is a quantitative display type because it provides a graphical representation of an amount (location and direction), which consists of an Edwards AFB map, a Southern California map, a Mercator world map, past and future trails of the virtual spaceplane, ground locators, and waypoints. The location of the virtual spaceplane in the virtual environment determines which map representation the display shows. When the virtual spaceplane is within the Edwards AFB terminal area and below 16,000 feet, the display shows the Edwards map. When the virtual spaceplane is within the Southern California area and below 30,500 feet, the display shows the Southern California map. If the virtual spaceplane is anywhere outside of these two regions, the display shows the Mercator world map.

Trails on the display show the past and future trajectories of the virtual spaceplane. The past trail shows approximately three orbit trajectories and is green, which maintains

consistency with the rest of the cockpit. We calculated past trails using the virtual spaceplane's latitude and longitude. A future trail shows approximately one half orbit trajectory and is blue, which is the color code for direction. Future trails are computed by predicting the virtual spaceplane's direction out to 0.03 days; i.e., 43.2 minutes.

Ground locators show the location of ground entities on the map display. These locators look and function just like the ones we positioned on the earth. This allows selection of any ground entity at any time even if it is on the other side of the earth.

Waypoints are navigational aids used by the autopilot for flying routes. We only put these waypoints on the Edwards AFB map because we only use our autopilot for terminal operations. Each waypoint on the Edwards map has exactly the same position as the ones in the virtual environment, except for their altitude.

Navigation Panel Buttons

There are three buttons on the left of the panel, "STDN," "DSN," and "Target." "STDN" displays Satellite Tracking and Data Network sites. "DSN" displays Deep Space Network sites. "Target" displays ground military targets such as nuclear facilities and mobile missile launchers. All three buttons are radio type buttons, which means that only one or none are active at any one time.

Mission Panel

The mission panel (Figure 41) organizes and provides information about the virtual spaceplane's payload and controls for accomplishing various missions. Geometry on the panel consists of dynamic text and buttons, which we divide into two categories (payload operations and external view).

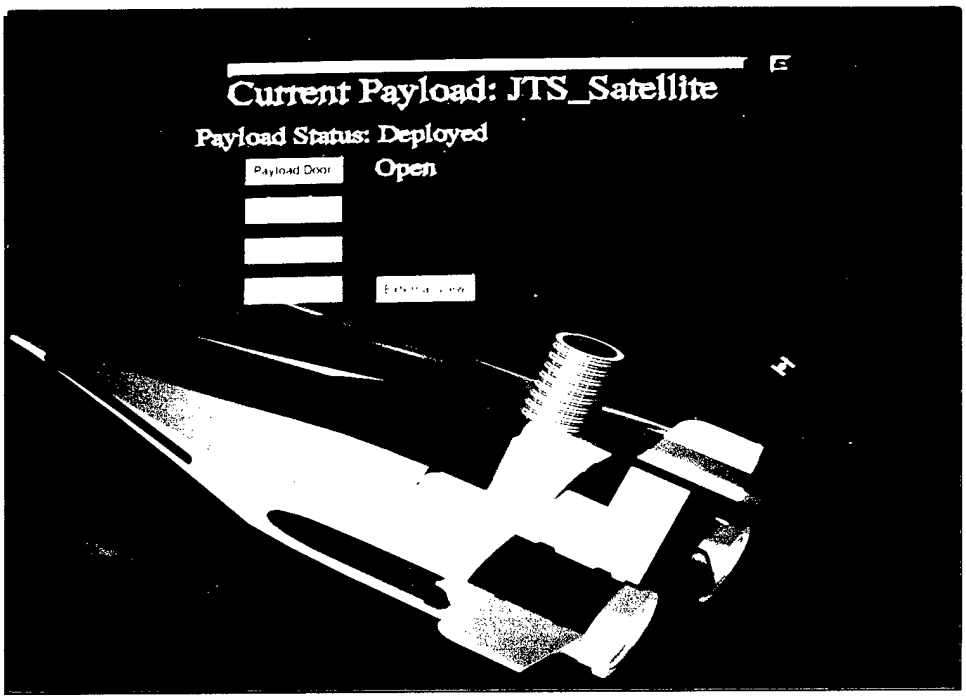

Figure 41: Mission Panel

Mission Panel Buttons and Text

At the very top of the panel is a string of dynamic text identifying the current contents of the cargo bay. Below that and to the left is a string of dynamic text indicating the status of the payload. There are three identifiable states for cargo ("Stored," "Ready" and "Deployed"). "Payload Door" is the first button in the payload operations category. It is a toggle button for opening and closing the cargo bay door. There is a string of dynamic text to the right of this button identifying the status of the cargo bay door ("Open," "Opening," "Closed," and "Closing"). Three buttons below the "Payload Door" button represent different mission involving payload. "Deploy Sat" is a push-to-activate button, which deploys a satellite from the cargo bay out into space. "Docking" initiates automatic docking maneuvers with a space station. "Recce" automatically maneuvers the virtual spaceplane in the correct orientation for conducting reconnaissance missions over a specified ground target. Currently, the virtual spaceplane only simulates deploying a

satellite. Therefore, the "Docking" and "Recce" buttons are disabled. "Deploy Sat" is inactive, which indicates what our current mission is. Although this button is inactive, we disable its functionality as long as the cargo bay door is closed. As soon as the cargo bay door is open and the satellite is ready, the user can then launch a satellite by pressing the "Deploy Sat" button.

Providing an external view of the virtual spaceplane is the second functional category and contains only one toggle button ("External View"). When the user presses this button, the cockpit camera offset resets to a distance of 100 meters behind the virtual spaceplane. By clicking on the virtual spaceplane and dragging the mouse cursor, the user can change the position of the cockpit. The viewpoint always faces the virtual spaceplane regardless of its position. This functionality allows the user to view payload operations outside of the virtual spaceplane and at any orientation.

Target Panel

The target panel (Figure 42) organizes and provides information applicable to target selection, target identification, target viewing, target manipulation, and target interception. The virtual spaceplane's target panel is comprised of eleven sections (data, display, viewing buttons, manipulation buttons, cycle buttons, domain buttons, entity buttons, maneuvering buttons, intercept sub-panel, autopilot buttons, time slider, and delta V options).

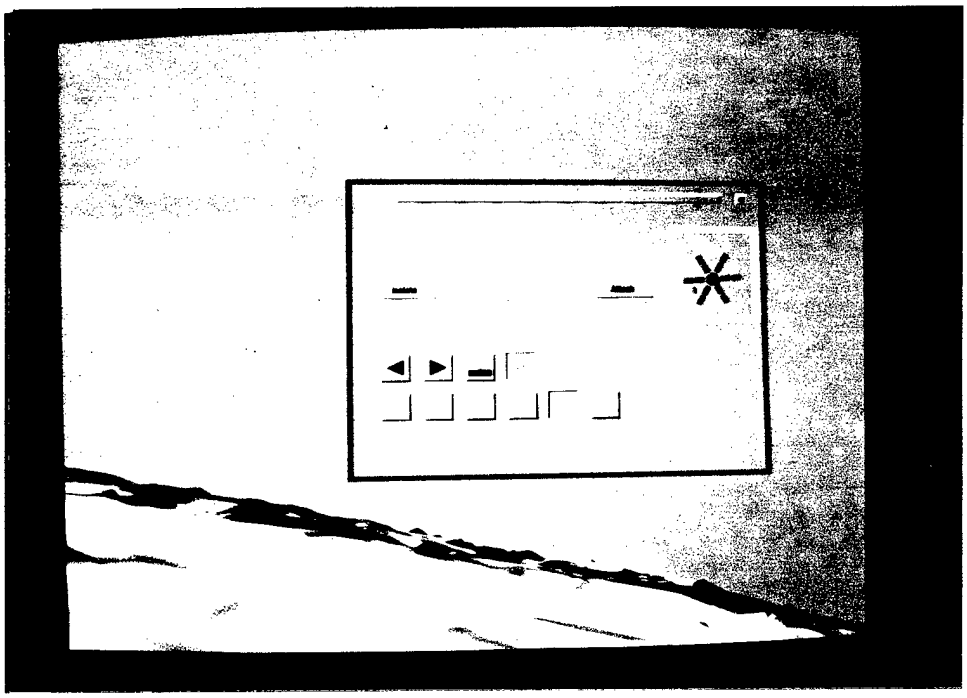

Figure 42: Target Panel

Target Panel Text and Display

The current target's name, latitude and longitude, altitude above the surface of the earth, and range from the virtual spaceplane are in the data section. Range is the vector length from the target position to the virtual spaceplane.

In the upper right hand corner of the panel is a display that presents a three-dimensional model of the current target. Instead of merely using a static model, we animate it by rotating it within the display window. A user can place a model of a target in the display either by selecting the appropriate buttons on the target panel or by selecting the target from the environment.

Target Panel Buttons

Four viewing buttons assist the user in manipulating the presentation of the current target in the environment ("Isolate," "Gryphon," "Reset," and "Attach"). "Isolate" is a toggle button that allows the user to single out the current target from the remainder of

the environment by turning off the trails and locators of all the entities except for the current target. "Attach" is a toggle button that allows the user to teleport the cockpit, including the viewpoint, to the current target while the virtual spaceplane remains on its current trajectory. Once attached to the current target the user can manipulate viewing the target by clicking on it and dragging the mouse cursor. Dragging the mouse cursor translates the viewpoint around the current target at a fixed distance. The current target always has the front of the cockpit directed towards it. This functionality is the same as the "External View" on the mission panel. Pressing "Attach" also enables "Reset" and "Gryphon." "Reset" is a toggle button that allows the user to reorient the cockpit to the initial attached view for the current target. "Gryphon" is a toggle button that returns the cockpit and viewpoint to the virtual spaceplane. It also inactivates "Attach," and disables itself and "Reset."

Manipulation buttons allow the user to turn on or off the trails and locators of the constellation of the current target. "Trails" is a toggle button that turns the trails on and off, and the "Locators" toggle button turns the locators on and off. For example, suppose the user selects a global positioning system (GPS) satellite and makes it the current target. By inactivating "Trails," the trails for all the GPS satellites are now off. They can turn them on again by activating "Trails." This same functionality is available for "Locators."

Cycle buttons are push to activate buttons that allow the user to cycle through the current target list. They can only cycle through the family of the current target. Icon design for this button is based on the previous and next symbology found on most

recording devices. We chose this design because the chance of a user exposed to its meaning are high and hence makes it easy to learn and remember.

Domain buttons are radio buttons that allow the user to determine the domain of targets to select from, those in space or ground. Pressing the ground domain button inactivates the space domain button and replaces the space entity buttons below it to the ground entity buttons. Its icon design consists of a green rectangle, representing the ground, and cross hairs, which represent a target. A space domain button functions in the same manner replacing the ground entity buttons with the space entity buttons. Its icon design consists of a blue arrow with a red orbit circling around it. This design is the old NASA logo.

Entity buttons are radio buttons used to select a particular constellation of targets. After selecting a constellation of targets, the current target becomes the first target associated with the constellation. Pressing the cycle buttons allows the user preview the targets in the current constellation. There are two groups of entity buttons, one for the space domain and the other for the ground domain. Table 17 lists the constellations of both the space and ground domain. Icon designs for space entity buttons are pictorial representations of the locator of the targets. This provides a direct one-to-one association between the locator in the environment and the image on the button. Icon designs for ground entity buttons are not as straightforward. Two of the ground entity buttons contain pictorial representations of a Satellite Tracking and Data Network station and a Deep Space Network station. The third button contains an abstract representation symbolizing all enemy ground targets. We could not use the ground locators for icon

designs because they do not provide any information as to the type of target it represents, only its alliance.

Table 17: Target Constellations

Space Domain	Ground Domain
Global Positioning Satellites (GPS)	Targets*
Defense Meteorological Satellite Program (DMSP)	Satellite Tracking and Data Network (STDN)
Defense Satellite Communications System (DSCS)	Deep Space Network (DSN)
Tracking and Data Relay Satellite (TDRS)	
Molniya	
Space Station	
Joint Tracking Satellite (JTS)	

*Note: Targets include Nuclear Facilities and Mobile Launch Sites

Maneuvering buttons change the position and orientation of the virtual spaceplane with respect to the current target. "Intercept" is a toggle button that displays the intercept sub-panel. When a user activates "Intercept," the autopilot system engages to fly the virtual spaceplane to the current target. "LockOn" is a push-to-activate button that orients the virtual spaceplane to face towards the current target.

Target Panel Sub-Panel

We display the target panel sub-panel (Figure 43) whenever the user activates the "Intercept" button. The sub-panel contains the autopilot buttons, time slider, and delta V options.

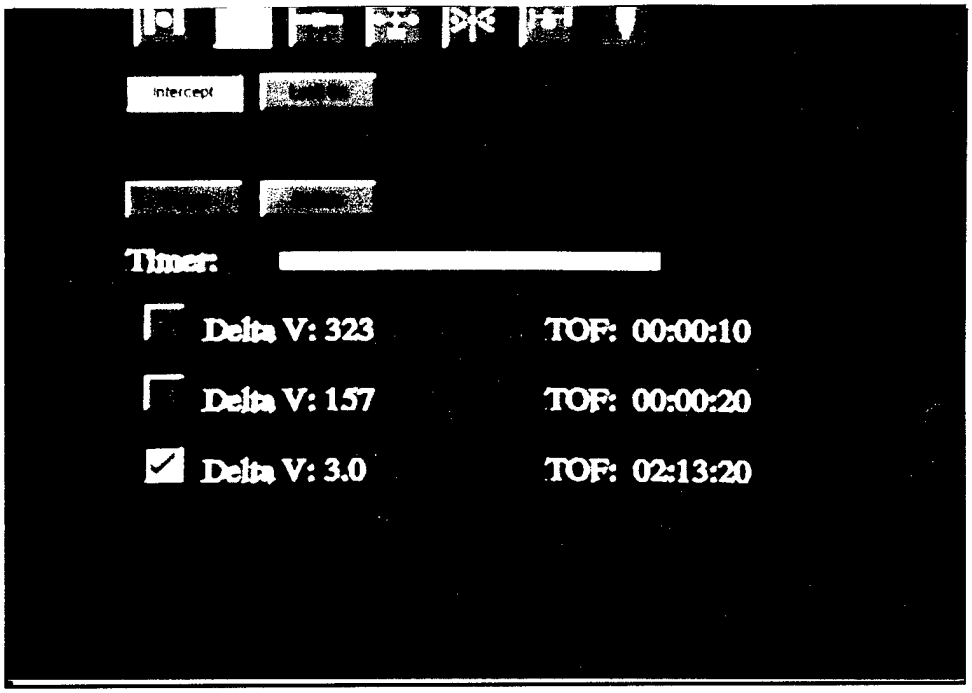

Figure 43: Target Panel Sub-Panel

Three buttons affect the autopilot system. "Engage" is a push-to-activate button that turns the autopilot on, activation automatically maneuvers the virtual spaceplane to intercept the current target, and the flight indicator on the space-flight control system switches to "Automatic." The maneuver begins with a re-orientation of the virtual spaceplane to ensure the main engines fire in the correct direction. During the intercept maneuver, the user now has no control over the virtual spaceplane's orientation until after it is complete. For more information about the specifics of the autopilot system, refer to Johnson [Johnson97]. "Engage" initializes as a disabled button until the user chooses a delta V option, at which point it becomes inactive.

Delta V options (Figure 44) are a group of three options used to select an intercept flight profile. The display for each option consists of a radio button for selection, a delta V amount and time of flight. Table 18 lists each option and their contents.

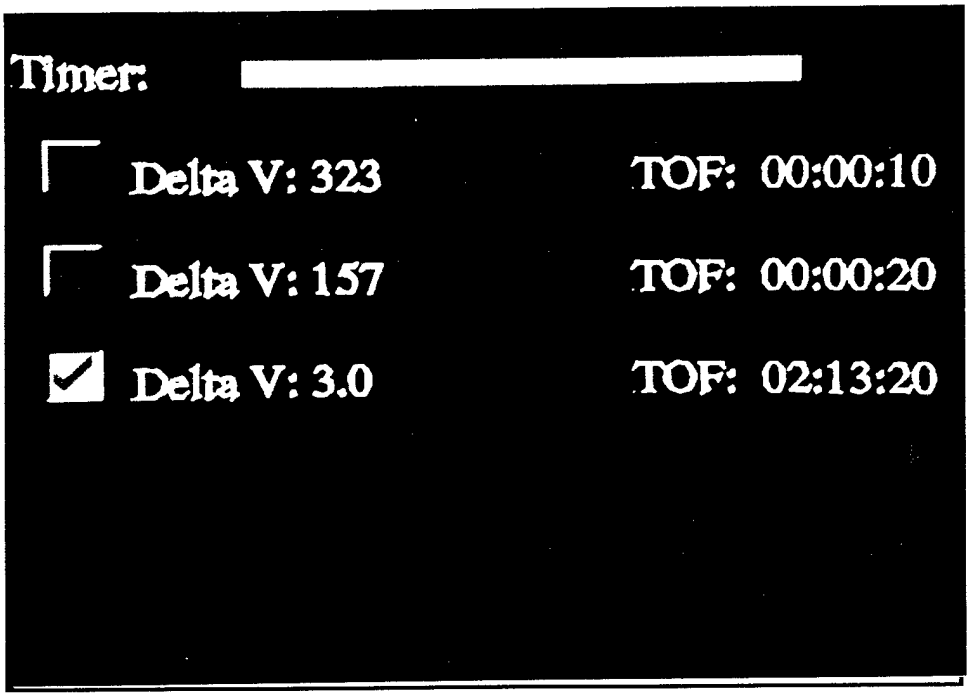

Figure 44: Delta V Options

Table 18: Delta V Options

Option	Contents
One	Greatest delta V and shortest amount of time
Two	Average delta V and mean time
Three	Lowest delta V and longest amount of time

A user can select an option by clicking the left mouse button on a radio-type selection button. At this point, the "Engage" button becomes inactive and the autopilot system is ready to initiate the maneuver. When the user presses "Engage," the autopilot system uses the selected option to maneuver the virtual spaceplane to intercept. Icon design for selection buttons consists of a plain face for unselected buttons, while a selected button displays a check mark. Because the selection buttons are small, it may be easy for the user to overlook which option they chose. Using the check mark icon reinforces visual feedback to the user making it clear which button was pressed.

"ReCalc" resets the autopilot system to perform new calculations for delta V and time of flight for the current target.

A time slider (Figure 45) represents the time of flight selected by the user for an intercept flight. Its length represents the complete time of flight. Yellow on the slider represents the portion of flight until initial burn, while cyan represents the portion of the time of flight until final burn. Color determination is based completely on appearance; i.e., the colors provide no symbolic representations.

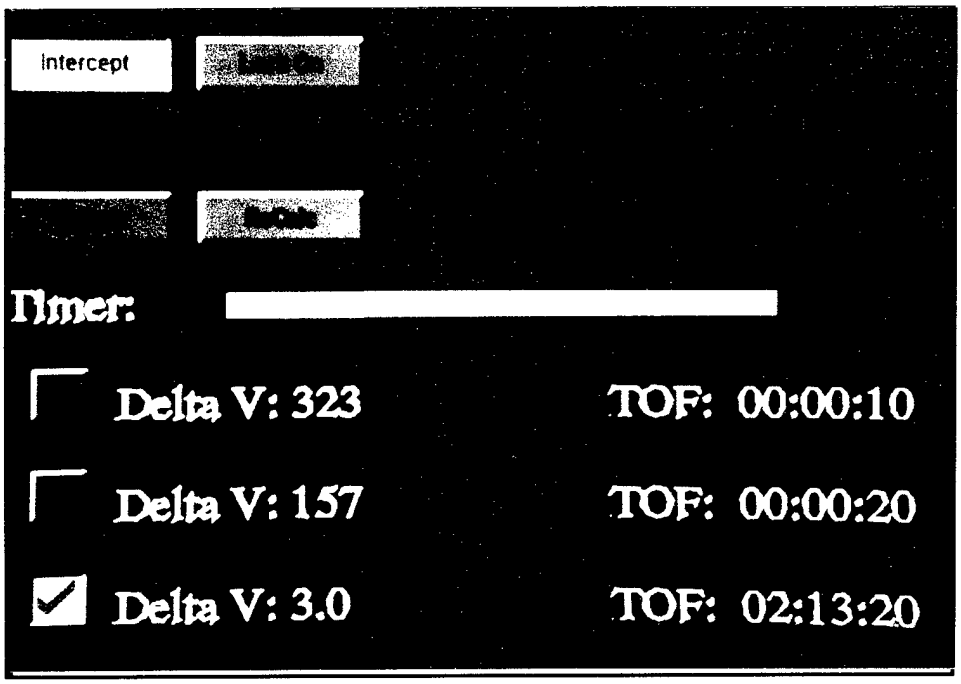

Figure 45: Time Slider

Trajectory Panel

The trajectory panel (Figure 46) presents information about various environmental parameters associated with orbit entry and re-entry. Its purpose is to provide the user with an indication of adverse atmospheric effects on the virtual spaceplane, e.g. dynamic pressure, hull temperature, and specific energy. Orbit entry is the transition from the atmosphere of the earth to space, while re-entry is the transition from space to the earth's

atmosphere. The panel contains a trajectory display centered on the panel and two categories of buttons.

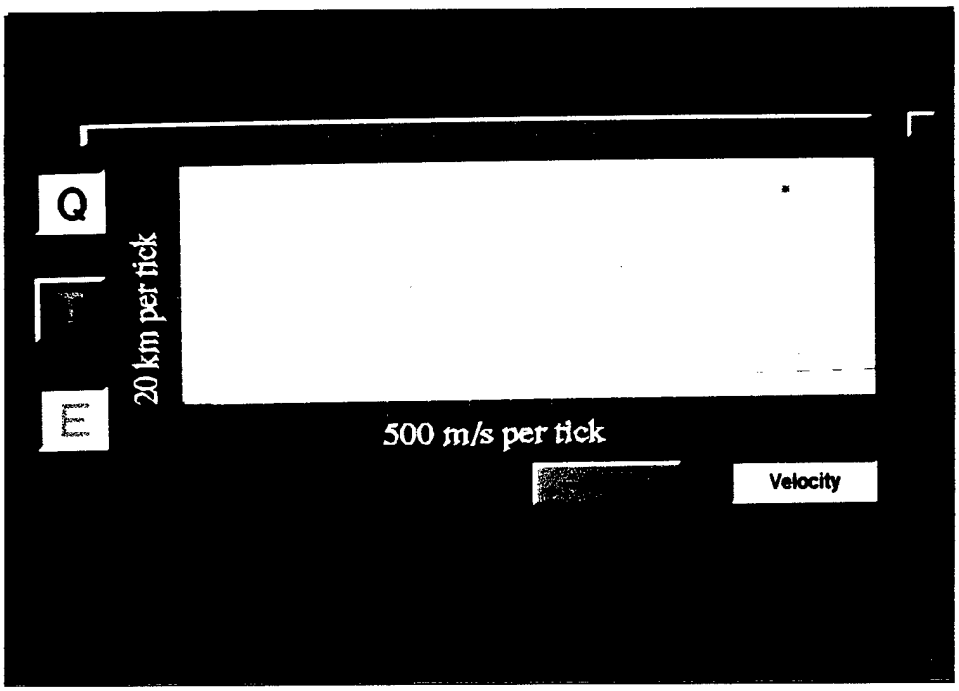

Figure 46: Trajectory Panel

Trajectory Panel Display

A trajectory display is a plot of altitude (in meters) versus velocity (in mach or meters per second). A green asterisk shows the current state of the virtual spaceplane and a set of lines represents various parameters. The blue lines represent lines of constant dynamic pressure (q), which is inversely proportional to altitude (because of the atmosphere) and directly proportional to the velocity of the virtual spaceplane. q provides an indication of engine performance capabilities. Rocket engines perform better when q is low; i.e., flying in a vacuum, while jet engines perform better when q is high. q is also applicable to the structural integrity of spacecraft. A structure can withstand a certain level of q before it weakens and breaks. We measure q in units of kPa (kilo Pascal).

Light blue lines represent lines of constant specific energy. Specific energy is the sum of the potential and kinetic energy per unit of mass. We calculate potential energy with the equation

$$PE = g \bullet h$$

where g is the gravitational constant and h is altitude. Kinetic energy uses the equation

$$KE = \frac{1}{2} \bullet V^2$$

where V is velocity. "Specific" and "per unit mass" are related terms and mean that mass is assumed to be constant so it is factored out. Total energy for the virtual spaceplane is equal to the potential energy plus the kinetic energy, which always remains constant in the absence of friction. After an engine burn in preparation for re-entry, the virtual spaceplane retains an amount of energy and places the virtual spaceplane in a trajectory that will intersect the earth. On a re-entry trajectory, the virtual spaceplane gradually converts potential energy (height) into kinetic energy (velocity), but its total energy remains constant. If we neglect the atmosphere, the virtual spaceplane would follow one of the constant energy lines until it hit the earth. As the altitude decreases and the atmospheric density increases, atmospheric drag; i.e., friction, slows the virtual spaceplane and the green position indicator starts to deviate from the constant energy line. This friction generates heat, which transfers some of the total energy into the atmosphere, thereby reducing the kinetic energy of the virtual spaceplane. This is important because the information tells the user if the virtual spaceplane has enough energy to make it to the landing site without fuel.

Orange lines represent the lines of equilibrium hull temperature. These give an indication of the virtual spaceplane's hull temperature during re-entry. The hull temperature increases because of the friction generated by the atmosphere. If the virtual spaceplane takes a re-entry trajectory too steep, the hull temperature may exceed safety limits and burn up. If the virtual spaceplane takes a re-entry trajectory too shallow, the hull temperature would not be hot enough. In this case, the lower temperature would indicate that the virtual spaceplane may skip on top of the atmosphere and continue back out into space. Currently we have no equations for calculating these lines. Until then, we will simulate their calculations to show that we can display this functionality.

Plot scaling changes dynamically to keep the green asterisk within the bounds of the plot. Scales along the horizontal axis represent velocity or Mach, while the scales along the vertical axis represent altitude. At high altitudes and speeds, we increase the scale of the plot and at low altitudes and speeds we decrease the scale of the plot. Lines on the plot change their shape and position as the x-axis changes due to the dependency that Mach has with altitude. This dependency also affects the position of the asterisk.

Trajectory Panel Buttons

"Mach" and "Velocity" are radio type buttons that change the units of the x-axis between Mach number and meters per second. We give the user the ability to determine what values to display because velocity is more useful in space than Mach, which losses it is meaning in a vacuum.

Three buttons on the left are toggle type buttons with "q" representing dynamic pressure, "E" representing energy, and the "T" representing temperature. Functionality of these buttons is to control the display of their respective lines on the plot. We label

them with the first letter of their respective variable and they are color coded to match their respective plot lines.

Virtual Environment Panel

A Virtual Environment Panel (Figure 47) provides the capability to modify the state of the entire virtual environment. Users can change the rate of time, modify ambient light levels, manipulate trails, locators, and constellations, and teleport to and from space.

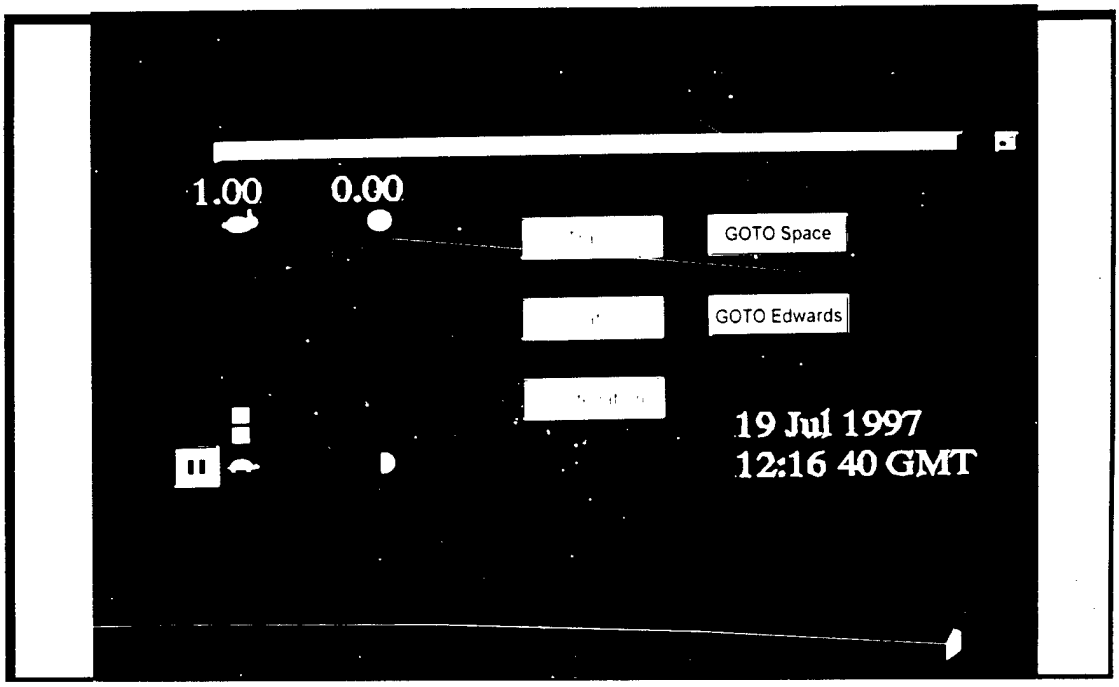

Figure 47: Virtual Environment Panel

Virtual Environment Panel Control Sliders

The virtual environment panel contains two control sliders, a time control slider and an ambient light control slider. Missions for the military spaceplane can range from 72 hours to weeks; therefore, it is generally impractical for a user to participate in a virtual simulation for that length of time. Users can change the rate of time using the time control slider on the left side of the panel. At the top of the slider is a numeric value

representing the time factor. A time factor of one is a standard 24-hour cycle. To increase (decrease) the rate of time a user clicks on the slider and drags the mouse cursor to the top (bottom). There is a rabbit icon at the top symbolizing a faster rate, while the turtle at the bottom represents a slower rate. At the bottom of the slider is a pause button, whose icon is the standard pause symbol found on many recording devices. When the user presses the pause button, the slider bar and time-factor flash indicating that the simulation is paused. This functionality is found in most recording devices and provides an excellent visual cue to the user.

There are two light sources used in the virtual spaceplane simulation, directed light from the sun and ambient light. Our virtual spaceplane initializes with ten percent ambient light, which provides good contrast between the day and night side of the earth. We give the user the ability to increase the ambient light level in case they do not want to fly in the simulated darkness. To the right of the time control slider is the ambient light control slider with the percent of ambient light displayed at the top. Users can change ambient light level in the same manner as changing the rate of time. At the bottom of the slider is a half moon icon, which indicates no ambient light and provides the greatest contrast between day and night. Under these conditions, the only light is coming from the directed light source at the sun. At the top of the slider is a full moon icon, which indicates the ambient light is at the same brightness as the directed light. This condition eliminates all contrast in the virtual environment.

Virtual Environment Panel Buttons

Three toggle buttons ("Trails," "Locators," and "Constellations") in the center of the panel allow the user to turn on and off trails, locators, and constellations. These buttons allow the user to set preferences that affect the look and feel of the simulation.

There are two push-to-activate buttons on the right of the panel that teleports the virtual spaceplane to and from space. "GOTO Space" transports the virtual spaceplane to a predetermined orbit, and "GOTO Edwards" transports the virtual spaceplane to the Edwards AFB runway. This functionality provides flexibility in achieving a variety of objectives, particularly training objectives. For example, if a user needs more practice on take-off and orbit entry the virtual spaceplane can be reset to the runway at any time in the simulation and repeat the process.

Virtual Environment Panel Text

Time is a very important variable in nearly all military operations. In the virtual spaceplane, time controls the propagation rates of every entity. It also affects burn times for orbital changes. In the bottom right corner of the panel we display the current date and time. We use the Greenwich Mean Time as our base time because the virtual spaceplane will be crossing multiple time zones and this is the standard for military operations.

Engineering Panel

The Engineering Panel (Figure 48) provides diagnostic and time critical information. Diagnostic information consists of consumable status, onboard equipment status, and payload status. Time critical information consists of emergency procedures for

equipment failures, below minimums on consumables, etc. This information presents itself to the user via a simulated HyperText Markup Language (HTML) browser. Its geometry consists of a display (representing the browser window) and three push-to-activate buttons ("Home," previous, and next).

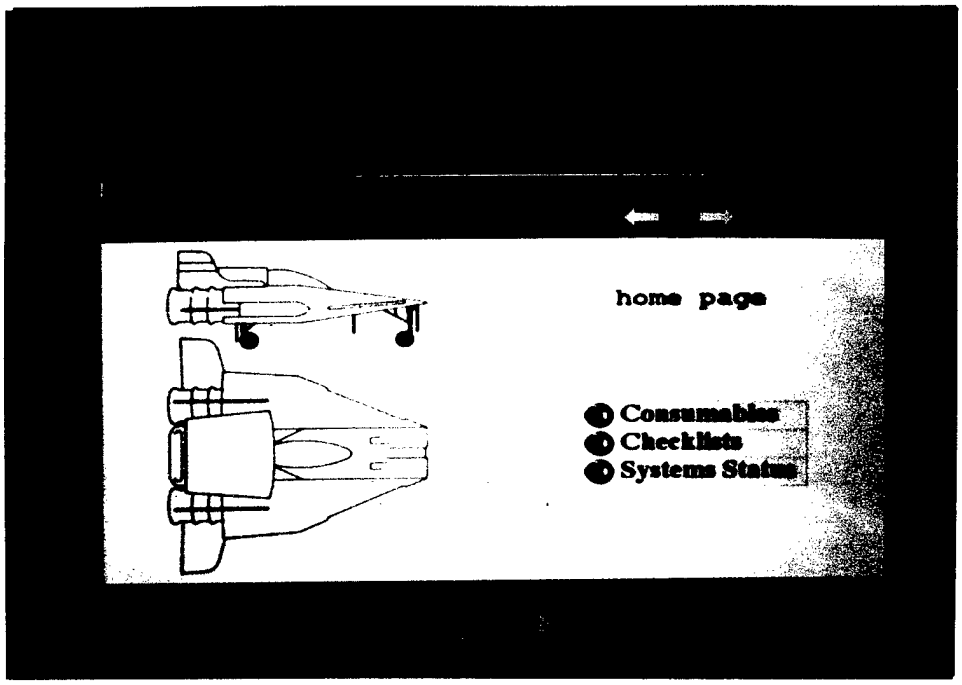

Figure 48: Engineering Panel

Button interaction functions just like those of existing browsers. Pressing the "Home" button resets the browser to the homepage. Previous and next buttons take you to the previous and next pages respectively. For an in-depth review of the functionality of the browser and the contents of the webpages, refer to Johnson [Johnson97].

Toolbars

There are two toolbars in the virtual user interface (Figure 49), which are located on the main console in front of the cockpit. Each toolbar is a set of toggle buttons with icons depicting which panel they represent.

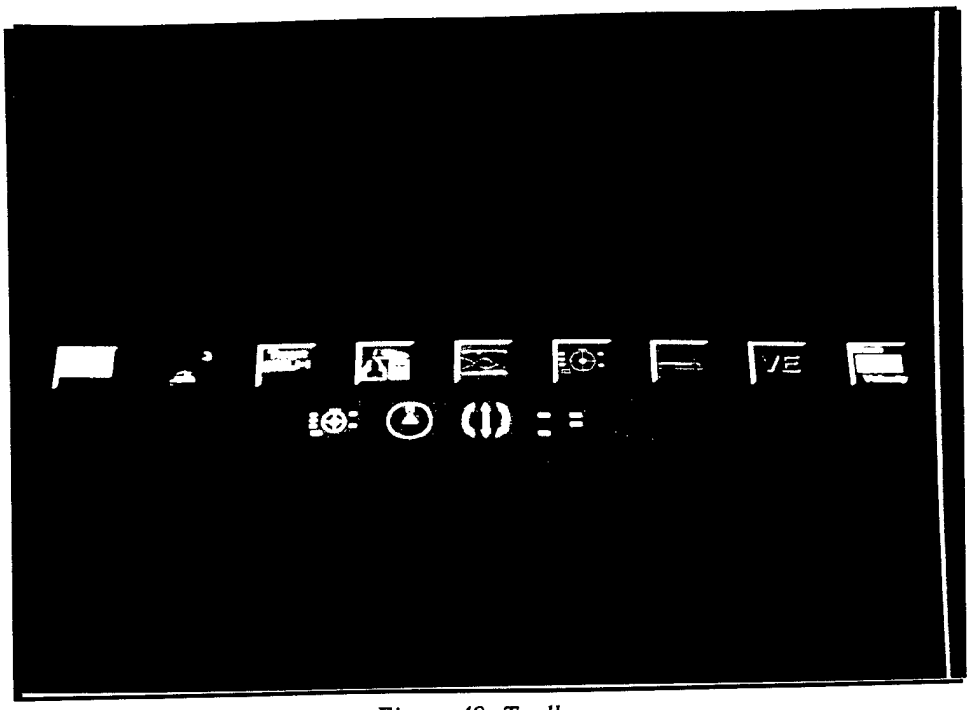

Figure 49: Toolbars

On the top row is a toolbar for minimizing and maximizing console panels. Pressing one of these buttons removes its associated panel from the cockpit. Pressing it again brings the panel back to its previous location. Icons on each of the buttons are either a pictorial or an abstract representation of a particular console panel. A pictorial representation is a replica image of the panel. An abstract representation is a symbolic image of a panel's functionality or purpose. Table 19 lists which buttons contain an abstract or pictorial icon design.

Table 19: Toolbar Icon Designs for Panel Buttons

Button Representation	Type of Icon
Aero-Flight Panel	Abstract representation of the atmospheric regime.
Space-Flight Panel	Abstract representation of the space regime.
Target Panel	Pictorial representation.
Engineering Panel	Pictorial representation.
Navigation Panel	Pictorial representation.
Flight Control System	Pictorial representation.
Mission Panel	Abstract representation of moving cargo.
Virtual Environment Panel	Abstract representation of a virtual environment
Trajectory Panel	Pictorial representation.

A toolbar for the aero-flight and space-flight control systems is on the second row. The first button has an icon representing the entire control system. This particular button minimizes or maximizes the entire control system onto the control panel. Remaining buttons on the toolbar toggle various components on the control system. All of the icons on these buttons are pictorial representations of a particular component of the flight control systems.

CONCLUSION

This chapter presented the steps and decisions made concerning the implementation of the virtual interface design. We began with a discussion of the synch and stabilize method used in the design process. Then, we explained how and why the individual components where integrated into the major sections of the virtual user interface. Chapter Six, Results, will determine the success of the overall interface design.

CHAPTER SIX – RESULTS

INTRODUCTION

This chapter will review the completion of the requirements presented in Chapter Three, Requirements. It will also present the results of an interface heuristic evaluation and determine the virtual spaceplane's degree of illusion.

COMPLETION OF REQUIREMENTS

This section presents a review of the overall virtual spaceplane requirements. In keeping with the same style as Chapter Three (Requirements) we only discuss the user interface results in detail. Refer to Johnson and Rothermel [Johnson97][Rothermel97] for results concerning the remaining requirements.

SIMULATED CAPABILITIES

Table 20 lists the capability requirements and corresponding method of completion.

Table 20: Completion of Capability Requirements.

ID	Requirement	Resolution
Flight Characteristics		
1.1	Maneuvering on runways	TaxiProp
1.2	Flight through the atmosphere	AeroProp
1.3	Maneuvering in space	AstroProp
1.4	Transition between flight regimes	OrbitEntryProp, common interfaces of PropModels
Manual Operation		
1.5	Manually operate in the atmosphere	Input methods of AeroProp
1.6	Manually operate in space	Input methods of AstroProp

Automatic Operation		
1.7	Automatically takeoff	TAKEOFF mode of Autopilot
1.8	Automatically fly specified routes	FLYROUTE mode, Route of Waypoints
1.9	Automatically enter orbit	ENTERORBIT mode of Autopilot
1.10	Automatically modify orbital parameters	HOHMANN and RENDEZVOUS modes
1.11	Automatically reenter the atmosphere	REENTER mode of Autopilot
1.12	Automatically land	LANDING mode of Autopilot

SUPPORTED MISSIONS

The missions successfully integrated into the virtual spaceplane are shown in Table 21. The current implementation places many restrictions and/or makes assumptions concerning parameters affecting the missions. For example, the rendezvous mission does not include any time restrictions and the satellite deployment mission does not model the boost to a higher orbit typical of many satellite launches.

Table 21: Completion of Mission Requirements.

ID	Requirement	Resolution
Supported Missions		
2.1	Rendezvous with orbiting object	Auto RENDEZVOUS capability, Autopilot
2.2	Deployment of satellite	Payload capability of SimGryphon

USER INTERFACE

The architectural design succeeds in providing a framework within which we developed, implemented, modified, and tested a user-centered virtual interface. Initially, very little was known about the style, methods, and functionality of the interface, yet the architecture provided the flexibility needed to experiment and firewalls to prevent changes from accelerating the system integrity towards chaos. Interface instantiations for fulfilling each interface requirement are in Table 22.

Table 22: Completion of Interface Requirements.

ID	Requirement	Resolution
Interaction Methods		
3.1	All functionality via three button mouse	Left button – geometry selection Middle button – viewpoint orientation Right button – field of view
3.2	HMD with head tracking	N-Vision HMD and Ascension Bird™
3.3	Auxiliary functionality via keyboard	Shortcut key commands for supporting the design process.
Configurable Cockpit		
3.4	Selectively display information	Minimizable panels
3.5	Modify location of information	Movable panels
Displayed Information		
3.6	Gryphon state in atmosphere	Aero-Flight Control System, Aero panel
3.7	Gryphon state during entry/reentry	Control Systems, Trajectory, Aero panels
3.8	Gryphon state in space	Space-Flight Control System, Orbit panel
3.9*	State of consumables	Engineering and Payload panels
3.10	Target information	Target panel
3.11	Locating/acquiring targets	Target panel, selection of locators
3.12	System management and diagnosis	Engineering and Payload panels
3.13*	Investigate hyper-text paradigms	Engineering panel
3.14	Minimize obstruction of view	Transparent panels
Controlling the Gryphon		
3.15	No throttle and stick	Mouse interaction, Control Systems
3.16	Change state in the atmosphere	Aero-Flight Control System, Autopilot
3.17	Change state in space	Space-Flight Control System, Target panel

*Note: Lt. Troy Johnson will address the completion of these requirements.

Our architecture succeeds by isolating user interaction from all components of the virtual spaceplane, with the exception of the main Sim class. Sim's ***TranslateInputs ()*** routine polls the various input devices and either directly causes changes in the virtual environment (as in the case of keyboard input) or generates events processed independently of their source. Generally, events are generated when the mouse selects geometry in the virtual environment.

To simplify interaction development, we divide the event processing into two stages. First stage occurs in the Sim and causes changes to entities and properties of the virtual environment. Input parameters are placed in their respective CODB containers and accessed by the Sim class. ***TranslateInputs ()*** uses these parameters and the index values

assigned to objects to process the appropriate functions that effect entities outside the virtual user interface. ***TranslateInputs ()*** also passes the index values to the Cockpit class, which processes the appropriate functions that effect the virtual user interface; i.e., the second stage.

We perform this event staging for two reasons. First, Sim is the only class with access to all the entities in the virtual environment. Processing inputs in the Sim class makes it easy to try different versions of a prototype component because access to all the entities is centralized in a single class. Second, the virtual user interface is a dynamic, complex component of the virtual environment that underwent continuous, radical changes. Processing inputs that effect the user interface in the Cockpit class restricts access from the rest of the virtual environment, except via the Sim class. Had we incorporated access to all components in the virtual environment, event processing would have been connected to all the components making debugging errors and prototype difficult and slow.

VIRTUAL ENVIRONMENT

Our virtual spaceplane succeeds in developing several aspects of the virtual environment previously unexplored at the AFIT Virtual Environments Lab. Integration of geographically accurate terrain into a simulation based on a round-earth-coordinate-system (WGS84) results in the ability to immerse the user in the atmosphere, in space, or anywhere in between. Visible atmospheric transition increases the realism by making the environment change as expected. The environmental requirements summary is in Table 23.

Table 23: Completion of Environmental Requirements.

ID	Requirement	Resolution
Environment		
4.1	Convincing terrain near Edwards AFB	DTED based with LOD and textures
4.2	Model Earth, Sun, Moon	EarthProp, SunProp, MoonProp
4.3	Earth orbiting objects	GPS, DMSP, DSCS, TDRS, Molniya, Space station
4.4	Day/night, atmospheric/space transition	Calculation of sky coefficient, fading stars

MISCELLANEOUS

Miscellaneous requirements, and their implementation, are in Table 24.

Table 24: Completion of Miscellaneous Requirements.

ID	Requirement	Resolution
Miscellaneous		
5.1	Accept remote entities via DIS	DISEntityManager, DISEntity, DISEntityProp
5.2	Transmit Gryphon state via DIS	BroadcastSimObject in Sim
5.3	Mean of 15 frames per second	

HEURISTIC EVALUATION

Appendix C contains the compiled results of three heuristic evaluations conducted by the design team. Below is a summary of the results of each heuristic evaluation component.

SIMPLE AND NATURAL DIALOGUE

All the information presented to the user appears in a natural and logical order. In addition, dialogues do not contain irrelevant or rarely needed information. Sequences involving user interaction and information presentation matches the expected operation profiles in a natural and logical order. Research based on space shuttle operations [NASA94] and interaction by the design team is the primary foundation for determining the correct sequence of events.

We group information in the virtual user interface according to mission functionality. Because of this grouping, the user can present relevant information pertaining to a specific aspect of a mission profile, and hide the information that does not. For example, during a satellite deployment, the user may not want to see target information. In this case, they would present information appropriate to satellite deployment, e.g. the mission panel, and hide information associated with target acquisition; i.e., the target panel.

The console panel paradigm maintains the task grouping, while information on the panel is separated spatially according to the requirements of a particular task. This structure breaks down a particular mission task into sub-components. As an example, the target panel contains many sub-components associated with a particular aspect of target acquisition. Structuring the target panel in this manner makes finding the right information and controls easier than if all the sub-components where arranged at random or in an illogical order.

Determination of what the user needs was accomplished during discussion among the members of the design team and was based on information obtained from *The Space Shuttle Operator's Manual* [Joels88], the *Shuttle Crew Operations Manual* [NASA94], *System Requirements for a Military Spaceplane* [DOD97a], and *Technology Roadmap for a Military Spaceplane System* [DOD97b]. The purpose of this determination is to reduce the information overload already present in current space shuttle operations [NASA94], which allows the user to concentrate on the high level goals associated with a mission profile. The computer performs any functionality that would otherwise require extensive interaction on the part of the user; e.g., calculations.

The implementation of the color codes seems to be the key element in presenting information in the virtual user interface. Changing the color of buttons when pressed makes it easier to determine subtle difference in complex displays. As an example, the target panel is a very complex display with a lot of buttons. The subtle color changes between the active and inactive buttons makes it easier for the user to quickly determine the status of these components.

SPEAKING THE USER'S LANGUAGE

Much of the terminology used in the virtual user interface comes from personal flight experience among the design team and the *Shuttle Crew Operations Manual* [NASA94]. Consequently, dialogue in the user interface expresses itself clearly in words, phrases and concepts that are familiar to the user. For example, information on the trajectory panel presents terminology consistent with an astronaut's knowledge of space flight. Symbols and icons are taken from these sources as well as from within the virtual environment, which presents information consistent with the user's domain knowledge.

MINIMIZE USER MEMORY LOAD

Information in the user interface is clearly visible or easily retrievable, which reduces the amount of information the user would have to remember from one part of the dialogue to another. By basing the design on what the user sees in the environment and what they will do functionally, the user will always receive redundant information from multiple sources about the current state of the virtual spaceplane. As an example, when the user selects a target in the virtual environment, the target panel is updated to contain

the information relevant to that particular target. If the user begins to perform other operations, all they have to do is return their attention to the target panel to refresh their memory.

CONSISTENCY

Buttons and text all work in harmony with each other, which eliminates the chances that the user will ever wonder whether different words, situation, or actions mean the same thing. For example, the previous and next buttons on the target panel cycle through the target associated with a particular constellation of targets. The previous and next buttons on the engineering panel cycle through the different HTML pages. Because both sets of buttons look alike, they have similar functionality. Maintaining consistency throughout the virtual user interface is a result of the design specifications. By creating these specifications and then applying them to the interface design, we ensure that commands have the same effect at all times, information is in the same location at all times, and information is formatted the same and standardized.

FEEDBACK

Our virtual user interface always informs the user about what is going on, through appropriate feedback within a reasonable time. Therefore, the user never has to wonder what is going on. For example, during atmospheric flight the user can receive information from a variety of sources about the current direction, orientation, and speed of the virtual spaceplane. These sources include the view changes associated with the virtual environment, the altitude direction indicator, the navigation panel, and the aero-

flight control system. Each user interaction results in a visible change to the interface either through a change in state of an object or by a change in the orientation of the virtual environment.

HELP AND DOCUMENTATION

Currently there is no online help and documentation system available. As it is well beyond the goals of this research. However, the functionality for further development is in this area exists in the HTML interface on the engineering panel. There is some documentation available as a README.TXT file for the first time user.

CLEARLY MARKED EXITS

Exits in the virtual spaceplane are not as we expect in a window operating system. Here exits allow that user to reset the state of the simulation in either space or the Edwards runway. Exits are also implied within the button interaction paradigm. If the user crashes the virtual spaceplane, pressing the "GOTO Edwards" button on the virtual environment panel can reset the simulation.

SHORTCUTS

Shortcuts in the virtual spaceplane are not like the keyboard shortcuts in current user interfaces. Our shortcuts involve speeding up time to achieve results faster. The ability to "jump" to and from space also serves as a shortcut.

DEGREE OF PRESENCE

Based upon the three aspects of virtual environment technology (user presence, environment type, and interface), we can estimate the degree of presence a user may feel while interacting with the virtual space plane. Figure 50 is an estimate of the degree of presence achieved by the virtual space plane.

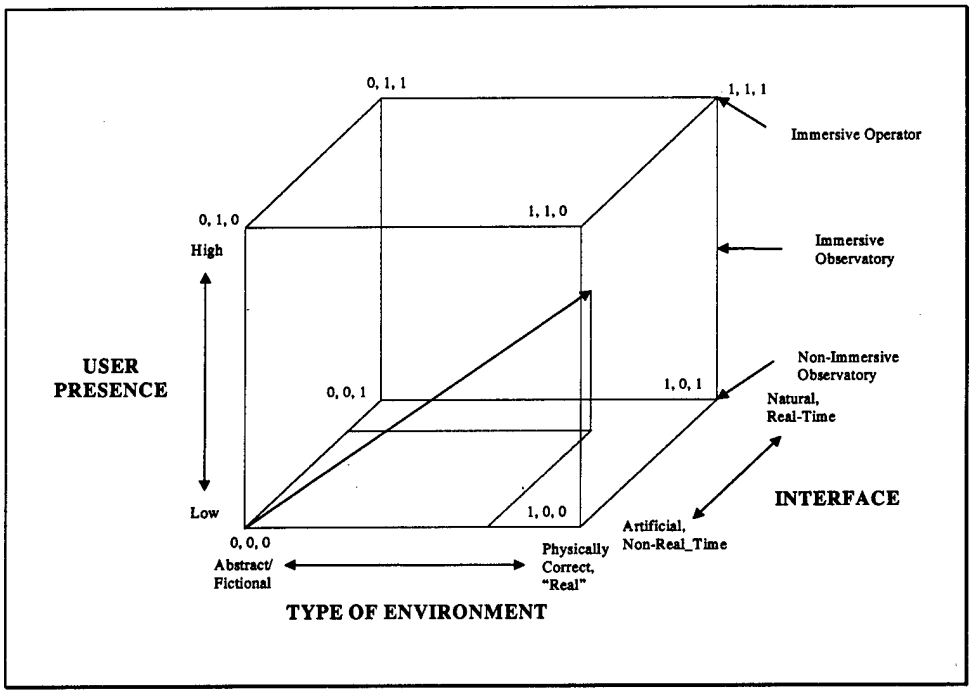

Figure 50: Dimensions of the Virtual Space Plane

We estimate that the virtual space plane has a degree of presence of around (0.75, 0.5, 0.8). Due to the number of dynamic entities and the amount of texture memory rendered in the virtual environment there are some compromises in the overall appearance of the environment. These compromises in appearance are necessary because the large number of entities and their realistic behavior takes up a lot of processing power from the hardware.

Achieving a high level of user presence relies on the type of interaction devices used in the simulation. Currently, the simulation uses a mouse and head mounted display to

achieve the effect of user presence. Unfortunately, this setup does not provide the maximum amount of interaction to achieve a high level of user presence. Using a mouse-based interaction weakens the degree of illusion, because in the real world we do not have mouse cursors that help us select on objects. The implementation of a dataglove would resolve this problem; however, for now we only have an average user presence. With a dataglove, this level should increase to around 0.75.

Since the virtual space plane interface has no real world counterpart, its design facilitates user interaction based on anticipated needs and functionality. A user has all the necessary tools to interact with the environment in a convincing way, but because there is no association to a real interface, this area did not receive a high rating.

CONCLUSION

In this chapter, we presented the results of the virtual user interface design from three aspects. A review of the overall virtual spaceplane requirements, with emphasis on the user interface, showed we accomplished our main objectives. A heuristic evaluation showed the user interface complies with standardized design principles. An estimate of the degree of presence presented the visual effectiveness of the overall simulation. In the next chapter, we present recommendations for improving the user interface.

CHAPTER SEVEN – RECOMMENDATIONS AND CONCLUSIONS

INTRODUCTION

As with any project, there are always new ways to improve. This chapter presents recommendations, which can improve the interaction techniques associated with the virtual spaceplane simulation. The last section of this chapter presents the conclusion of this thesis.

REQUIREMENTS AND CAPABILITIES

LOOK-AT MENUS

Look-at-menus are an interaction technique that replaces mouse interaction and adds additional interaction associated with data glove technology. Head orientation can be used instead of the mouse cursor or traditional hand position to control the cursor used to select an item from a menu. To select the user turns their head instead of moving their hand. The picking ray, used in selection and intersection testing, is fixed relative to the head; i.e., the origin of the head-tracking receiver. This functionality gives an intuitive way to select an item simply by looking at it. To confirm selection, the user presses a physical button or, with pulldown menus, releases the menu, which activates the last item the user looked at [Mine97].

VIRTUAL TECHNOLOGIES SOFTWARE/HARDWARE

The following subsection provides a comprehensive list of software and hardware devices specifically design for virtual environment interaction. Implementation of these devices would increase both the naturalness and user presence aspects of the degree of presence estimate.

CyberGlove®

CyberGlove® (Figure 51) is a lightweight glove with flexible sensors which accurately measures the position and movement of the fingers and wrist. To accomplish tracking, there are mounting provisions for the Ascension tracker, which attach to the wristband [VTP97]. The basic design of the CyberGlove® provides a graphical representation of the user's hand inside the virtual environment. Intersection testing between the CyberGlove® and objects can be used to activate interface functions. For example, when the user intersects the payload door with the virtual hand, it would activate the function that opens the payload door.

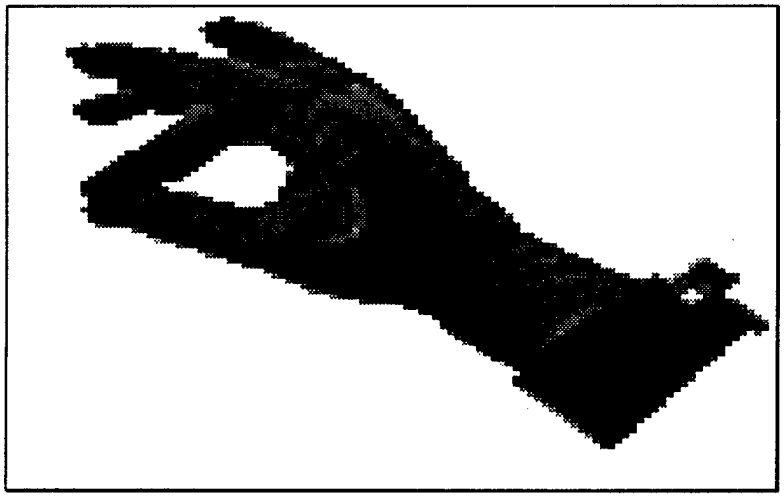

Figure 51:Virtual Technologies CyberGlove®

CyberTouch®

CyberTouch® (Figure 52) consists of a tactile feedback option for the CyberGlove®. It features small vibrotactile stimulators on each finger and the palm of the CyberGlove®. Each stimulator can be individually programmed to vary the strength of touch sensation. The array of stimulators can generate simple sensation such as pulses or sustained vibration, and they can be used in combination to produce complex tactile feedback patterns. Future programmers for the virtual spaceplane can even design tactile sensation representing the perception of touching a solid object in virtual spaceplane simulation [VTP97].

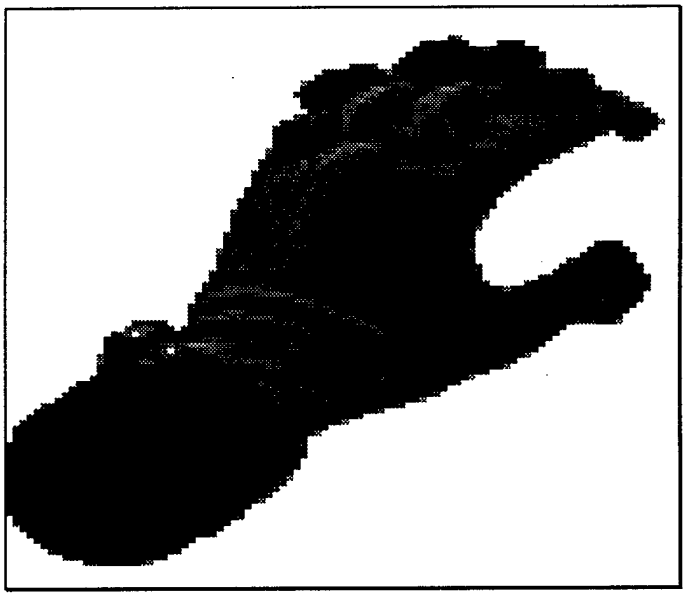

Figure 52: Virtual Technologies CyberTouch®

GesturePlus®

GesturePlus® (Figure 53) is a gesture recognition system that uses joint measurements from the CyberGlove® to recognize user defined hand formations. During a system training procedure, the GesturePlus® system receives information necessary to

associate the user's hand formations with the output symbols of the user's choice. GesturePlus® will allow the user to define a customized gesture library for the virtual spaceplane application. After the training session is complete, the system is ready to output the recognized symbol whenever the user makes the corresponding hand formation [VTP97].

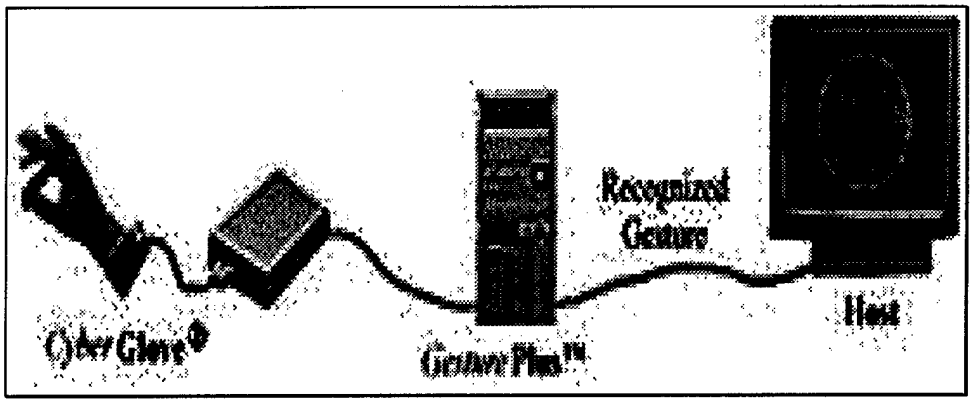

Figure 53: Virtual Technologies GesturePlus®

VirtualHand® Toolkit

The VirtualHand® Toolkit acquires hand and finger position data from the CyberGlove® and generates a graphical hand on the computer monitor, which reproduces the movement of the user's physical hand and fingers. It allows easy integration of CyberGlove® related function calls into the virtual spaceplane's application software. It also includes support for the CyberTouch® vibrotactile feedback option for the CyberGlove®, GesturePlus® gesture recognition system, as well as the Ascension® tracking sensors. This toolkit provides a set of CyberGlove® hand model calculations, display, calibration, data acquisition and three-dimensional tracking functions [VTP97].

USER TESTING

The biggest drawback to designing the virtual user interface using Discount Usability Principles was the lack of test user to perform usability tests. Getting feedback and advice from actual astronauts and pilots is the only way to ensure a complete and viable user interface to the virtual spaceplane.

CONCLUSIONS

The completion of virtual user interface has fulfilled all of the requirements and goals of this thesis effort. This effort saw the development of an architecture that supports the rapid design and prototype of interface components without complex interconnection with the overall simulation architecture. This design also manages the information flow associated with user interaction in an efficient and transportable manner. It has developed a set of standardized specifications for the design of interface components. These specifications ensure consistency throughout the virtual user interface and the interactions of the user. The effort put forth in this thesis saw the completion of the virtual user interface during the implementation process. Finally, the results verified this effort against the accomplishment of the levied requirements, heuristic guidelines, and the degree of presence affecting the user.

APPENDIX A: DESIGN SPECIFICATIONS

COLOR CODES

Type/Uses:

- Red
  - Systems and Conditions Warning
  - Enemy Locators and Trails
  - ADI Altitude Warning Lines
- Yellow
  - Systems and Conditions Caution
  - Unknown Forces Locators and Trails
- Green
  - Systems and Conditions All Clear
  - Friendly Forces Locators and Trails
  - Virtual Hud Interface
  - ADI Altitude All Clear Lines
- Light Blue
  - Environmental Representation of the Sky
- Dark Blue
  - Future Trails
  - Constellation Lines
  - Constellation Names
  - Dynamic Pressure Lines
- Light Gray
  - Indicates Active Button State
  - Text Color for Disabled Buttons

Medium Gray

Indicates Inactive Button State

Base Color for Console Panels

Frame Color for Displays

Dark Gray

Indicates Disabled Button State

Brown

Environmental Representation of the Ground

Black

Text Color for Active and Inactive Buttons

Frame Color for Console Panels

Red Orange

Temperature Lines

Cyan

Energy Lines

Off-White

Console Panel Text

TEXT FORMAT

Types/Uses

Helvetica

Button Text

Constellation Names

Status Indicators

Flight Mode Indicator

Times Elfin

Console Panel Text

Virtual HUD Interface

BUTTONS

Types/Uses

Radio

Selects an item out of a group of related items. Only one item in the group can be selected at any given time, or none of the items can be selected.

Push-To-Activate

The application is active as long as the user continues to press the button. When the user releases the button, the application is inactive.

Toggle

The application is active when the user presses the button.

When the user releases the button, the application remains active until the user presses the button again.

Sizes

Miniature

0.5 x 0.5 cm

Small

1.0 x 1.0 cm

Regular

1.0 x 3.0 cm

Long

0.5 x 16.0 cm

Color

Active

Light Gray

Inactive

Medium Gray

Disabled

Dark Gray

Text

Active

Black Helvetica

Inactive

Black Helvetica

Disabled

Light Gray Helvetica

PANELS

Types/Uses

Console

Used to group buttons, text, and displays related to a specific category.

Sizes

Regular

20.0 x 10.0 cm

Large

20.0 x 15.0 cm

Sub-Panel

20.0 x 10.0 cm

Color

Base

Medium Gray, 30% Opaque

Frame

Back, 100% Opaque

Text

Off-White, Times Elfin

DISPLAYS

Types/Uses

Qualitative

Shows information according to conditions; i.e., overall situation awareness in a single indicator.

Quantitative

Shows information in amounts or portions; i.e., information about the measurable value of some variable.

Hybrid

Shows information in both a qualitative and quantitative format.

Sizes

Depends on the application, but does not exceed the size of the panel it is attached to.

INDICATORS

Types/Uses

Status

Shows the current state of a variable specific to a particular application.

Sizes

Regular

1.0 x 3.0 cm

Color

Warning

Red, 30% opaque

Caution

Yellow, 30% opaque

All Clear

Green, 30% opaque

Text

Green, Helvetica

Green, Times Elfin

LOCATORS

Types/Uses

Space

Identifies entities in space.

Ground

Identifies entities on the earth.

Display

Identifies entities on a display.

Sizes

Space

Approximately 40 km.

Ground

Approximately 5 km.

Display

Approximately 1.0 cm

Shapes

Space

Symbolic three-dimensional representation of the entity it represents.

Ground

Wire-frame dome.

Display

Wire frame dome.

Symbolic three-dimensional representation of the entity it represents.

Flat 2D geometric symbols, e.g. asterisk.

Color

Enemy Forces

Red

Unknown Forces

Yellow

Friendly Forces

Green

TRAILS

Types/Uses

Past

Shows the previous trajectory of an entity.

Future

Shows the expected trajectory of an entity.

Orbit

Shows the orbit trajectory of a space entity

Transition

Shows the transition from one orbit to another

Color

Enemy Forces

Red

Unknown Forces

Yellow

Friendly Forces

Green

Future

Blue

Virtual Spaceplane

White

Transition

Blue

POINTERS

Types/Uses

Directional

Shows the direction the user must look to find the entity it represents.

Sizes

Regular

Approximately 1.0 x 1.0 cm

Shapes

Flat 2D geometric symbols.

SLIDERS

Types/Uses

Time

Shows the proportional relationship between two related time periods.

Control

Controls the value of a parameter by changing the length of the slider bar.

Sizes

Regular

Approximately 0.5 x 10.0 cm.

ANIMATED MIMICS

Types/Uses

Roll

Shows the direction and roll rate of the virtual spaceplane

Shape

Arrow head designed to reinforce direction of movement

Color

Green

Overall color scheme

White

APPENDIX B: TRANSLATE INPUTS METHOD

```
void Sim::TranslateInputs( )
{
    // Open the keyboard container and read the information
    keyboard_data = (keyboard_struct*)Kb_CODB->BeginRead(KeyboardStruct);

    switch (keyboard_data->value)
    {
        .
        .
        .
    }

    // Close the keyboard container
    Kb_CODB->EndRead(KeyboardStruct);

    // Open the mouse container and read the information
    mouse_data = (mouse_struct*)Mouse_CODB->BeginRead(MouseStruct);

    // Check if the left mouse button is down
    if (mouse_data->buttons & PFUDEV_MOUSE_LEFT_DOWN)
        .
        .
        //initialize variables during the first frame the left mouse button is down
        if (select_on)
        {
            .
            .
            // Make a call to the selection manager to return the index of the selected
geometry
            index = sp_sel_mgr->GetIndex(mouse_data->normx, mouse_data->normy, hitpoint);
            .
            .
            // If a valid index is returned process the value to invoke the callback
functions
            if (index != -1)
                {
                    ProcessSelection(index, hitpoint);
                }

            // If we are moving a panel, initialize the appropriate variables
            .
            .
        } // Completed initializing variables during the first frame the left mouse
button is down

        // The following apply as long as the left mouse button is down
        if (the Cockpit->maneuver_on)
        {
            Update the appropriate control system
            Maneuver the virtual spaceplane with the appropriate propModel
        }

        if (theCockpit->throttle_on)
        {
            Update the throttle control slider
            Set the throttle in the appropriate propModel
        }

        if (theCockpit->time_on && !theClock.IsPaused( ))
        {
             Update the time slider on the God Panel
```

```
            Set the time factor in SimClock
        }

        if (theCockpit->ambientLight_on)
        {
            Update the ambient light slider on the God Panel
            Set the abient light setting in Renderer
        }

        if (theCockpit->panel_on)
        {
            Reposition the selected panel based on updated mouse values
        }

        if (theCockpit->rotating_orbit_display)
        {
            Implement trackball procedures to rotate the orbit display on the Space-
Flight Panel
        }

        if (targetCameramoving)
        {
            Implement trackball procedures to rotate the viewpoint around an attached
target
        }
    }
    else  // The left mouse button is not pressed
    }
        // Do some things as soon as the left mouse button is released
        if (!MOUSE_LEFT_UP)
        {
            if (theCockpit->panel_on)
            {
                Reset the variables used to reposition a panel
            }

            Switch (….)
            {
                End all maneuver commands to the appropriate propModel
            }

            // Reset any other variables as needed
    } // End left button

    if (mouse_data->buttons & PFUDEV_MOUSE_MIDDLE_DOWN)
    {
        Tell renderer to simulate head movement inside the cockpit with the middle mouse
    }  // End middle button

    if (mouse_data->buttons & PFUDEV_MOUSE_RIGHT_DOWN)
    {
        Tell renderer to change the field of view
    }  // End right button

    // Complete any clean up procedures for the next frame.

    // Close the CODB mouse container
    Mouse_CODB->EndRead(MouseStruct);

    Reset the virtual spaceplane and the space-flight control system due to the reality
disconnect
```

APPENDIX C: HEURISTIC EVALUATION RESULTS

SIMPLE AND NATURAL DIALOGUE

- Is the information the user needs – and no more – presented at exactly the time and place where it is needed? *Currently, the user has to determine what categories of information; i.e., panels, to display and where. Sub-categories of information; i.e., sections of information on the panels, are presented accordingly. Transition to and from space swaps out certain panels and the status indicators automatically present warnings. In addition, the target panel presents target information after the user selects a target and the space-flight panel updates its orbital parameters.*
- Can the user access information objects and operations in a sequence matching the way they will most effectively do things? *Yes. Sequences for landing, takeoff, target intercept, etc. are in a logical manner.*
- For items that appear in groups, are they:
  - Grouped appropriately? *Yes.*
  - Enclosed by lines or boxes? *Panels create groups of information according to mission functionality, which separates them from other categories. The subcategories of information on sub-panels are separated spatially as well as information on the flight control systems.*
  - Move or change together? *Yes. The panel paradigm accomplishes this.*
  - Alike with respect to shape, color, size, or typography? *Yes. The only inconsistency is the differences in fonts between dynamic (Times Elfin) and static (Helvetica).*
  - Prioritizing attention by making the most important dialogue stand out from the background? *Panels have no prioritization factors. Information and objects on panels do present visual clues to user as to based on color coding, active or inactive states, and dynamic text.*
- Are the colors appropriate and do they help information stand out from the background? *Yes.*

SPEAKING THE USER'S LANGUAGE

- Is terminology based on the user's language? *Yes. Aeronautical concepts are used during atmospheric flight (ADI, waypoints, etc.), while astronautical concepts are used during space flight (delta V, time of flight, etc.).*
- Do icons represent symbols appropriate to the user's language? *Yes. Most are based on appearance of the item it represents. Satellite icons look like locators and locators look like satellites. Some panel icons reflect the appearance of the panel, while others reflect a panel's functionality.*

MINIMIZE USER MEMORY LOAD

- Does the interface make it easier for the user to recall information? *Yes. In many cases icons look like objects or panels.*
- Is hidden information readily accessible by the user? *The transparent panels and cockpit prevent hidden information. The only problem is when a panel is out of the user's view.*
- Are icon meanings easy to remember? *Yes.*
- Are tasks grouped appropriately and labeled clearly? *Yes.*

- Are meanings clear without resorting to help menu or system? *Once functionality of an object is learned, meanings are clear.*

CONSISTENCY

- Do commands have the same effect every time the user initiates them? *The only difference found is the change in how the control system functions in both space and atmosphere. This difference appears fine because of the drastic differences between flight operations on both regimes.*
- Is information in the same location on all groups pertaining to the interface or other interface elements in the simulation? *Yes. All standard elements are in consistent locations.*
- Does the information have the same format on all groups pertaining to the interface or other interface elements in the simulation? *Yes. Sizes, colors, fonts, styles, and metric values are consistent throughout the interface.*
- Is the interface standardized with other interface elements in the simulation? *Yes. Primarily color.*

FEEDBACK

- Does the interface provide the user with information about what it is doing and what the user input for all commands? *For the most part, yes. One flaw in the system is that not enough information concerning the autopilot system is being presented to the user, e.g. distance to waypoints.*
- Is the response time of the interface approximately 0.1 seconds or less? *There are no noticeable delays.*
- If not, does the interface provide responses to the user? N/A

HELP AND DOCUMENTATION

- Is there a help system for this interface? *There is no online help system available. There is a keyboard command that presents a README.TXT script in another window, but this is more for debugging and maintenance.*

CLEARLY MARKED EXITS

- Does the interface provide the functionality to bring the user back to a previous state? *To some extent, yes. Users have the ability to reset the simulation either to a space orbit of on the Edwards runway. There is functionality to step back from button interactions (cycling through targets, resetting panels, etc.), however, if the user deploys the satellite he cannot get it back.*
- Is this functionality clearly visible or obscure? *It is not always clear that some other button or the same button will reverse the action. User needs prior knowledge of this functionality.*
- Can the user recover from a fatal mistake? *If the user crashes the virtual spaceplane, they can reset to either space or the Edwards runway. If the simulation crashes; e.g., a core dump, the only choice is the restart the simulation.*

SHORTCUTS

- Is functionality available to the user allowing them to take shortcuts in the virtual environment? *Yes. Reset buttons provide the most obvious method of shortcuts.*
- Does the user have the capability to speed up time to achieve faster results? Yes. The user can increase or decrease time in the simulation enabling them to achieve their objectives faster.

BIBLIOGRAPHY

[**Adams96**] Adams, Terry A. *Requirements, Design, and Development of a Rapidly Reconfigurable, Photo-Realistic, Virtual Cockpit Prototype.* AFIT/ENG/GCS/96D-02. M.S. Thesis, Air Force Institute of Technology, Air University, 1996.

[**Appino92**] Appino, Perry A., J. Bryan Lewis, Lawrence Koved, Daniel T. Ling, David A. Rabengorst, and Christopher F. Codella. "An Architecture for Virtual Worlds." *Presence*, Vol. 1, No 1, Winter 1992, pp. 1-17.

[**ATC96**] Ascension Technology Corporation. *The Flock of Birds: Installation and Operation Guide.* Burlington, VT: ATC, 1996

[**Banks92**] Banks, Sheila B. *Use of Color in Aircraft Cockpit Displays.* Clemson University, Clemson, SC: ? Department, July 1992.

[**Barnard95**] Barnard, Phil. "Study of Human-Computer Interaction." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Bannon95**] Bannon, Liam. "From Human Factors to Human Actors: The Role of Psychology and Human-Computer Interaction Studies in System Design." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Barrus96**] Barrus, John W., Richard C. Waters, and David B. Anderson. "Locales: Supporting Large Multiuser Virtual Environments." *IEEE Computer Graphics and Applications*, November 1996, pp. 50 – 57.

[**Bennett5**] Bennett, Kevin B., and Allen L. Nagy. *Spatial and Temporal Frequency in Animated Mimic Displays.* Wright State University, Dayton, OH: Psychology Department, 199?.

[**Bryson93**] Bryson, Steve. *Implementing Virtual Reality.* SIGGRAPH93, 20th International Conference on Computer Graphics and Interactive Techniques. Anaheim, CA. 1 – 6 August, 1993.

[**Bukowski97**] Bukowski, Richard, and Carlo Sequin. "Interactive Simulation of Fire in Virtual Building Environments." *SIGGRAPH '97 Conference Proceedings.* Los Angeles, CA: ACM SIGGRAPH, 3 – 8 Aug, 1997.

[**Cooperstock97**] Cooperstock, Jeremey R., Sidney S. Fels, William Buxton, and Kenneth C. Smith. "Reactive Environments." *Communications of the ACM*, Vol. 40, No. 9, Sept. 1997.

[**Coryphaeus95**] Coryphaeus Software, Inc. *Designer's Workbench 3.1 Reference 2nd Edition.* Los Gatos, CA: CSI, 1995

[**Cusumano97**] Cusumano, Michael A., and Richard W. Selby. "How Microsoft Builds Software." *Communications of the ACM*, Vol. 40, No. 6, Jun. 1997, pp. 53 – 61.

[**Davis92**] Davis, A. M., "Operational Prototyping: A New Development Approach," *IEEE Software*, Vol. 9, No 5, Sept. 1992, pp. 70 – 78.

[**Deitel94**] Deitel, H. M., and P. J. Deitel. *C++: How to Program.* Englewood Cliffs, NJ: Prentice Hall, 1994.

[**DOD97a**] U.S. Dept. of Defense. *System Requirements for a Military Space Plane.* Kirkland AFB: Phillips Laboratory Space Technology Directorate, 1997.

[**DOD97b**] U.S. Dept. of Defense. *Technology Roadmap for a Military Spaceplane System, ver 1.1.* Military Spaceplane Integrated Concept Team: Science and Technology Panel, Feb. 1997.

[**Ellis91**] Ellis, S. R., "Nature and Origins of Virtual Environments: A Bibliographical Essay." *Computing Systems in Engineering*, Vol. 2, No 4, 1991, pp. 321 – 347.

[**Erickson90**] Erickson, Thomas D. "Working with Interface Metaphors." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Foley92**] Foley, James D., et al. *Computer Graphics: Principles and Practice.* Reading, MA: Addison-Wesley Publishing Co., 1992.

[**Garcia96**] Garcia, Brian W. *Design and Prototype of the AFIT Virtual Emergency Room: a Distributed Virtual Environment for Emergency Medical Simulation.* AFIT/ENG/GCS/96D-07, M.S. Thesis, Air Force Instate of Technology, Air University, 1996.

[**Green85**] Green, Mark. "The University of Alberta User Interface Management System." *Computer Graphics*, Vol. 19, No 3, July 1985, pp. 205 – 213.

[**Hearn97**] Hearn, Donald, and M. Pauline Baker. *Computer Graphics: C Version.* Upper Saddle River, NJ: Prentice Hall, 1997.

[**Hiros97**] Hirose, Michitaka. "Image – Based Virtual World Generation." *Multimedia*, Vol. 4, No 1, January – March 1997, pp. 27 – 33.

[**Hong97**] Hong, Lichan, Shigeru Muraki, Arie Kaufman Dirk Bartz, and Taosong He. "Virtual Voyage: Interactive Navigation in the Human Colon." *SIGGRAPH '97 Conference Proceedings.* Los Angeles, CA: ACM SIGGRAPH, 3 – 8 Aug, 1997.

[**Hubbard95**] Hubbard, Philip M. "Collision Detection for Interactive Graphics Applications. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No 3, September 1995, pp. 218 – 230.

[**Joels88**] Joels, Kerry Mark, George P. Kennedy, and David Larken. *The Space Shuttle Operator's Manual.* New York: Ballantine Books, 1988.

[**Johnson97**] Johnson, Troy. *The Virtual Spaceplane: Integrating Multiple Motion Models and Hypertext.* AFIT/GM/ENG/97D-01. M.S. Thesis, Air Force Institute of Technology, Air University, 1997.

[**Kaiser92**] Kaiser Aerospace & Electronics Co. *3SPACE User's Manual.* Colchester, VT: KAE, 1992

[**Kanade97**] Kanade, Takeo, Peter Rander, and P.J. Narayanan. "Virtualized Reality: Constructing Virtual Worlds from Real Scenes." *Multimedia*, Vol. 4, No 1, January – March 1997, pp. 34 – 47.

[**Kay90**] Kay, Alan. "User Interface: A Personal View." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Lawton93**] Lawton, Gina, David L. Morrison, and Peter L. Lee. "Managing Complexity: Display Design in Process Control." *HCI Applications and Case Studies.* Eds. M. J. Smith and G. Salvendy. Amsterdam: Elsevier, 1993.

[**Lemay97**] Lemay, Laura, and Charles L. Perkins. *Teach Yourself Java 1.1 in 21 Days.* Indianapolis, IN: Sams.net Publishing, 1997.

[**Lewis95**] Lewis, Clayton and John Rieman. "Getting to Know Users and Their Tasks." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Macedonia95**] Macedonia, Michael R., et al., "Exploring Reality With Multicast Groups," *IEEE Computer Graphics and Applications*, Vol. 15, No 5, Sept 1995, pp. 38 – 45.

[**Macedonia97**] Macedonia, Michael R., and Michael J. Zyda. "A Taxonomy for Networked Virtual Environments." *Multimedia*, Vol. 4, No 1, January – March 1997, pp. 48 – 56.

[**Marcus90a**] Marcus, Aaron. "Designing Graphical User Interfaces: Part I." *UnixWorld*, Vol. 7, No 8, August 1990, pp. 107-116.

[**Marcus90b**] Marcus, Aaron. "Designing Graphical User Interfaces: Part II." *UnixWorld*, Vol. 7, No 9, September 1990, pp. 121-127.

[**Marcus90c**] Marcus, Aaron. "Designing Graphical User Interfaces: Part III." *UnixWorld*, Vol. 7, No 10, October 1990, pp. 135-138.

[**Mine97**] Mine, Mark R., Frederick P. Brooks, and Carlo H. Sequin. "Moving Objects in Space: Exploring Proprioception in Virtual Environment Interaction." *SIGGRAPH '97 Conference Proceedings.* Los Angeles, CA: ACM SIGGRAPH, 3 – 8 Aug, 1997.

[**Moezzi96**] Moezzi, Saied, Arun Katkere, Don Y. Kuramura, and Ramesh Jain. "Reality Modeling and Visualization from Multiple Video Sequences." *IEEE Computer Graphics and Applications*, November 1996, pp. 58 – 63.

[**Moezzi97**] Moezzi, Saied, Li – Cheng Tai, and Philippe Gerard. "Virtual View Generation for 3D Digital Video." *Multimedia*, Vol. 4, No 1, January – March 1997, pp. 18 – 26.

[**Mount93**] Mount, F. E. "HCI in Space Systems." *HCI Applications and Case Studies.* Eds. M. J. Smith and G. Salvendy. Amsterdam: Elsevier, 1993.

[**Mountford90**] Mountford, S. Joy. "Tools and Techniques for Creative Design." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Myers95a**] Myers, Brad A. "State of the Art in User Interface Software Tools." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Myers95b**] Myers, Brad A., et. al. "Garnet: Comprehensive Support for Graphical, Highly Interactive User Interfaces." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**NASA94**] National Aeronautics and Space Administration. *Shuttle Crew Operations Manual.* July 1994.

[**Nelson90**] Nelson, Theodor Holm. "The Right Way to Think About Software Design." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Nielson93**] Nielson, Jakob, *Usability Engineering.* Academic Press, Boston, 1993.

[**Pausch97**] Pausch, Randy, Dennis Proffitt, and George Williams. "Quantifying Immersion in Virtual Reality." *SIGGRAPH '97 Conference Proceedings.* Los Angeles, CA: ACM SIGGRAPH, 3 – 8 Aug, 1997.

[**Reising82**] Reising, John M., and Gloria L. Calhoun. "Color Display Formats in the Cockpit: Who Needs Them?" 26th Proceedings of the human Factors Society, 1982.

[**Reising85**] Reising, John M., and Terry J. Emerson. "Colour in Quantitative and Qualitative Display formats." *Journal of the Institution of Electronic and Radio Engineers.* Vol. 65, No 11/12, Nov/Dec 1985, pp 397 – 400.

[**Renze96**] Renze, Kevin J., and James H. Oliver. "Generalized Unstructured Decimation." *IEEE Computer Graphics and Applications*, November 1996, pp. 24 – 32.

[**Rheingold90**] Rheingold, Howard. "An Interview with Don Norman." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Risch96**] Risch, John S., Richard A. May, Scott T. Dowson, and James J. Thomas. "A Virtual Environment for Multimedia Intelligence Data Analysis." *IEEE Computer Graphics and Applications*, November 1996, pp. 33 – 41.

[**Rothermel97**] Rothermel, Scott. *Architecture, Design and Implementation of a Rapidly Prototyped Virtual Environment for a Military Spaceplane.* AFIT/GCS/ENG/97D-17. M.S. Thesis, Air Force Institute of Technology, Air University, 1997.

[**Rumbaugh91**] Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. "Object-Oriented Modeling and Design." Englewood Cliffs, NJ: Prentice Hall, Inc., 1991.

[**Sellen90**] Sellen, Abigail, and Anne Nicol. "Building User-centered On-line Help." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**SGI95**] Silicon Graphics, Inc. *IRIS Performer™ Programmer's Guide.* Mountain View, CA: SGI, 1995.

[**Shneiderman98**] Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Reading, MA: Addison-Wesley, 1998.

[**Sternbach91**] Sternbach, Rick, and Michael Okuda *Star Trek the Next Generation: Technical Manual.* New York: Pocket Books, 1991.

[**Stokes90**] Stokes, Alan, Christopher Wickens, and Kirsten Kite. *Display Technology: Human Factors Concepts.* Warrendale, PA: Society of Automotive Engineers, Inc., 1990.

[**Stytz96a**] Stytz, Martin R., E. Block, B. Soltz, and K. Wilson. "Tools for Visualizing and Analyzing Large-Scale Virtual Environments," IEEE *Computer Graphics and Applications*, Vol. 16, No 1, Jan 1996, pp. 16 – 26.

[**Stytz96b**] Stytz, Martin R., "Distributed Virtual Environments." *IEEE Computer Graphics and Applications*, Vol. 16, No 3, May 1996, pp. 19 – 31.

[**Stytz97**] Stytz, Martin R., Terry Adams, Brian Garcia, Steven S. Sheasby, and Brian Zurita. "Rapid Prototyping for Distributed Virtual Environments." *IEEE Software*, Vol. , No , Sep/Oct 1997, pp. 83 – 92.

[**Tognazzini90**] Tognazzini, Bruce. "Consistency." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Vertelney90a**] Vertelney, Laurie, Michael Arent, and Henry Lieberman. "Two Disciplines in Search of an Interface: Reflections on a Design Problem." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Vertelney90b**] Vertelney, Laurie, and Sue Booker. "Designing the Whole – Product User Interface." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**VTP97**] "Virtual Technologies Software/Hardware." *Virtual Technologies Products.* Online. Internet. 14 November 1997. Available HTTP: www.thevrsource.com/virtech/vtsoftware.htm.

[**Wagner90**] Wagner, Annette. "Prototyping: A Day in the Life of an Interface Designer." *The Art of Human Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison – Wesley Publishing Co., 1990.

[**Weicha95**] Weicha, Charles, William Bennett, Stephen Boies, John Gould, and Sharon Greene. "ITS: A Tool for Rapidly Developing Interactive Applications." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Weitzman85**] Weitzman, Donald O. *Color Coding Re-Viewed.* McLean, VA: The MITRE Corporation, 1985.

[**White95**] White, George M. "Natural Language Understanding and Speech Recognition." *Readings in Human-Computer Interaction: Toward the Year 2000, 2nd ed.* Ed. Ronald M. Baecker [et al]. San Francisco, CA: Morgan Kaufmann Publishers, Inc., 1995.

[**Zelter92**] Zelter, D. "Autonomy, Interaction, and Presence." Presence: Teleoperators and Virtual Environments, Vol. 1, No. 1, Winter 1992, pp. 127 – 132.

VITA

Captain John M. Lewis was born on [redacted]. He graduated from Green Sea-Floyds High School, South Carolina, in 1985 and entered undergraduate studies at The University of South Carolina in Columbia, South Carolina. He graduated with a Bachelor of Science degree in Applied Mathematics in May of 1989. He was enrolled in the Reserve Officers' Training Corps while in college and received his commission upon graduation on 5 May 1989.

Captain Lewis's first assignment was as a student in the Basic Meteorology Program at Penn State University. He completed his first assignment in May 1990 with a Bachelor of Science degree in Meteorology. His second assignment was in Fort Riley, Kansas as the Officer in Charge, Cadre Weather Team. During this term, he served in Desert Storm with the Fourth Aviation Brigade, First Infantry Division (Mech) where he earned the Bronze Star Medal for meritorious service in combat. In December, 1992 he went to Kadena AB, Japan as the Officer in Charge, Special Operations Weather Team. During this term, he support the 353rd Special Operations Group. In December, 1994 he went to Barksdale AFB, Louisiana as a war planner for Headquarters, Eight Air Force. During this term he laid the foundation for restructuring military weather support at the Joint Task Force and Component Command level for wartime operations. In May, 1996 he enter the School of Engineering, Air Force Institute of Technology where he will receive a Master of Science degree in Interactive Graphic Meteorology. His follow-on assignment is to the Air Force Weather Agency at Offutt AFB, Nebraska.

Captain Lewis is married to the former Shanna L. Spear of Fort Leavenworth, Kansas. They have a Golden Labrador Retriever named K.C., short for Kansas City.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1997	Master's Thesis

4. TITLE AND SUBTITLE
REQUIREMENTS, DESIGN AND PROTOTYPE OF A VIRTUAL USER INTERFACE FOR THE AFIT VIRTUAL SPACEPLANE

5. FUNDING NUMBERS

6. AUTHOR(S)
John M. Lewis, Captain USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology,
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GM/ENG/97D-02

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
PL/VTS
Mr. Jerry Gibson
3550 Aberdeen Ave SE
Kirtland AFB, NM 87117

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

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

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The United States Air Force is evaluating the feasibility of designing a military spaceplane capable of accomplishing military objectives from a low earth orbit and atmospheric flight regimes. Current efforts are involved in determining the scientific, operational, and budgetary constraints associated with this concept. This thesis looks at the exploration of new interface techniques associated with the design of a virtual spaceplane and is a subset of the overall virtual spaceplane effort, which will assist researchers in determining the feasibility of a military spaceplane. Interface techniques are integrated into a virtual user interface that is designed to accommodate expected operations associated with atmospheric and low earth orbit military operations. We expect these operations to include satellite deployment and recovery, reconnaissance, and space station construction and resupply. The focus of the virtual user interface design effort involves the application and integration of current interface design methodologies and virtual environment technologies to support the functionality of a virtual spaceplane.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Virtual User Interfaces, Heuristic Evaluations, Flight Simulators, Spaceplanes,	155
	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	UL