THE VIRTUAL SPACEPLANE: A MODELING AND SIMULATION TOOL FOR ADVANCED PROTOTYPING, REQUIREMENTS DEVELOPMENT, AND TRAINING FOR THE MANNED SPACEPLANE PROJECT

Martin R. Stytz, Ph.D. and Sheila B. Banks, Ph.D.
Virtual Environments Laboratory
Human-Computer Interaction Laboratory
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
mstytz@afit.af.mil, sbanks@afit.af.mil

ABSTRACT

We present the requirements and design of the Virtual SpacePlane (VSP). The VSP application is a development and demonstration virtual prototype for the Manned SpacePlane (MSP) project. The VSP, to be completed in two years, will demonstrate the functionality and capabilities of the MSP throughout it entire flight regime, from takeoff through space operations and landing. The goals of the VSP project are to uncover, develop and validate the MSP's user interface requirements, design and implement an intelligent user interface, and to design and implement a prototype VSP that can be used to demonstrate Manned SpacePlane missions and to conduct preliminary training in support of the MSP. To achieve these objectives, the VSP must also be able to execute the planned MSP missions in a realistic and tactically sound manner within a distributed virtual environment. To quickly develop the functional VSP prototype, the VSP reuses code from the Virtual Cockpit, Solar System Modeler, and Common Object Database (CODB) systems that have been developed in our Labs over the past several years.

In this paper we present background to the VSP project, the known functional requirements for the VSP, its software architecture, and its baseline user interface design. We conclude the paper with a brief summary of the project's current status and discuss future VSP development.

ABOUT THE AUTHORS

Martin R. Stytz is an active duty Lieutenant Colonel in the U.S. Air Force serving as an Associate Professor of Computer Science and Engineering at the Air Force Institute of Technology. He received a Bachelor of Science degree from the U.S. Air Force Academy in 1975, a Master of Arts degree from Central Missouri State University in 1979, a Master of Science degree from the University of Michigan in 1983, and his Ph.D. in Computer Science and Engineering from the University of Michigan in 1989. He is a member of the ACM, SIGGRAPH, SIGCHI, the IEEE, the IEEE Computer Society, the IMAGE Society, and the Society for Computer Simulation. His research interests include virtual environments, distributed interactive simulation, modeling and simulation, user-interface design, software architecture, and computer generated forces.

Sheila B. Banks is an active duty Major in the U.S. Air Force serving as an Assistant Professor of Computer Engineering at the Air Force Institute of Technology, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH. She received a Bachelor of Science in Geology, Magna Cum Laude from University of Miami, Coral Gables, FL in 1984 and a Bachelor of Science in Electrical Engineering, Summa Cum Laude from North Carolina State University, Raleigh, NC in 1986. Also from North Carolina State University, Raleigh, NC, she received a Master of Science in Electrical and Computer Engineering in 1987 and her Doctor of Philosophy in Computer Engineering from Clemson University, Clemson, SC in 1995. Her research interests include artificial intelligence, intelligent computer generated forces, associate systems, distributed virtual environments, intelligent human computer interaction, and man-machine interfaces.

THE VIRTUAL SPACEPLANE: A MODELING AND SIMULATION TOOL FOR ADVANCED PROTOTYPING, REQUIREMENTS DEVELOPMENT, AND TRAINING FOR THE MANNED SPACEPLANE PROJECT

Martin R. Stytz, Ph.D. and Sheila B. Banks, Ph.D.
Virtual Environments Laboratory
Human-Computer Interaction Laboratory
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
mstytz@afit.af.mil, sbanks@afit.af.mil

INTRODUCTION

The advent of requirements for worldwide deployment of space assets in support of Air Force operational missions has resulted in the need for a spacecraft that can support these missions with minimal preflight and minimal in-orbit support from a mission control center. As a result, successful mission accomplishment will depend almost completely upon the Manned SpacePlane crew and upon the on-board capabilities of the spaceplane. To insure that the crew has the needed capabilities on-board the craft, Phillips Laboratory initiated the Virtual SpacePlane (VSP) project. The Virtual SpacePlane is a virtual prototype of the Manned SpacePlane (MSP) and must be able to perform MSP missions in a realistic and tactically sound manner within a distributed virtual environment (DVE). The goals of the VSP project are to develop and validate the MSP user interface requirements, design and implement an intelligent user interface, and to design and implement a prototype virtual spaceplane that can be used to demonstrate MSP missions and to conduct preliminary training in support of the Manned SpacePlane.

The Virtual SpacePlane is based upon the existing Virtual Cockpit (12,17, 21, 22), Solar System Modeler (18), and Common Object Database (CODB) (20) systems that have been developed in our Labs. The VSP, to be completed in two years, will demonstrate the functionality and capabilities of the MSP throughout its entire flight regime, from takeoff through space operations and landing. Because we anticipate that VSP application requirements will expand and evolve over the life of the project, we based the VSP's software architecture upon the CODB architecture (20). To achieve accurate and high fidelity performance for the VSP throughout its operational regime, the system integrates existing aerodynamics and astrodynamics models into a single seamless high fidelity model of the VPS's dynamics. To achieve our project's goals, we must continually develop and refine system requirements in several areas. These areas include the types of missions to be conducted, the weapons and sensors that the spaceplane will employ, and the data, controls, and displays required by the onboard astronauts. Based upon these requirements, we designed a prototype system interface.

The next section contains a brief overview of relevant background material and projects. The third section in the paper presents a discussion of the VSP requirements and their implications for the VSP application. The fourth section presents our architectural and user interface solutions to the VSP requirements. The final section contains a summary and suggestions for future work.

BACKGROUND

This section discusses the topics of Distributed Interactive Simulation (DIS), the Common Object Database architecture, the Photorealistic Reconfigurable Virtual Cockpit, and the Solar System Modeler. These systems formed the basis for the VSP.

Distributed Interactive Simulation

The most widespread use of network technology for distributed virtual environments relies upon the current DIS suite of standards (3, IEEE Standard 1278-1993). DIS was designed to link distributed, autonomous hosts into a real-time distributed virtual environment via a network for exchanging the data that describe events and activities. DIS takes the concept of environmental distribution to its extreme; there is no central computer, event scheduler, clock, or conflict arbitration system. Each entity is responsible for processing the information that it receives and for insuring that all remote hosts have current and accurate information about its state. Stytz presents additional information concerning DIS and DVEs (19) and Blau discusses DIS as well (2).

Distributed interactive simulation technology uses heterogeneous hosts and a common virtual environment description to insert a variety of both human and computer controlled actors into a common shared synthetic environment. In a DIS networked virtual environment, the networked computer hosts cooperate to form a common shared environment wherein each host has its own local model of the environment and

there are no clients, servers, or central database description. The hosts are interconnected using networks that carry broadcasts of each data item that is dispatched from a host. Communication between the hosts is accomplished using the DIS suite of protocols. Within the DVE one (or more) entities in the distributed virtual environment are assigned to a single host. The responsible host propagates each one of its entities through the virtual environment using the appropriate dynamics model for its motion. To minimize the bandwidth required by each entity, dead-reckoning is used. Dead-reckoning significantly reduces the number of broadcasts required to propagate an entity.

The Common Object Database Software Architecture

The VSP's architecture is based upon the Common Object DataBase (CODB) as described by Stytz, et.al. (20) and illustrated in Figure 1 below. The Common Object DataBase is a data-handling architecture that uses classes, data containers, and a central runtime data repository to store and route data between the major objects in an DVE application. The Common Object DataBase contains the entire state of the DVE, as well as all the public or exported information for each object within the VSP application. This architecture reduces the coupling in a simulation application by reducing the amount of information that a class must maintain about other classes. To acquire public data from one or more application objects, an object must only access the container in the CODB where the information resides. The Renderer object portion of the architecture manages the rendering operations on the host computer. The Environment Database contains the graphical models for all of the entities to be portrayed in the DVE as well as the terrain model(s) for the virtual environment (VE). The User Interface object contains all of the selection and interface update software for the application. The World State Manager (WSM) handles incoming and outgoing network traffic from a simulation application and also maintains the world state for all entities controlled outside of its host. The WSM always has the most accurate description of the external world state. The WSM further minimizes coupling within the VSP by using it for the DVE interface. The dynamics and sensor models are used by the application to accurately propagate the entity through the virtual environment and realistically portray data that the real-world system would be able to sense. All other objects in the application are represented by the Application Objects object. One of the chief advantages of the CODB architecture is that it enables rapid switching between aircraft configurations and dynamics models with minimal impact to the virtual environment application's performance.

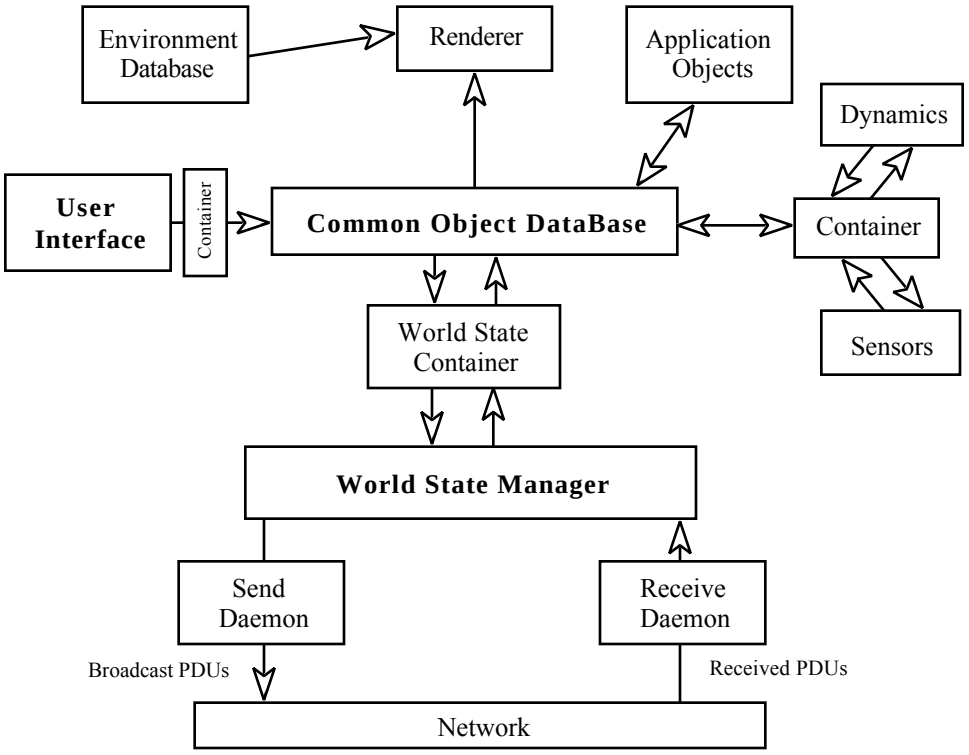

Figure 1: Common Object DataBase Architecture

The Photorealistic Reconfigurable Virtual Cockpit

The Virtual Cockpit (VC) is a cockpit and aircraft simulation project intended for use by military pilots for tactical training at their individual units (12,17,21,22). The VC project addresses issues in a low-cost, portable distributed interactive simulation (DIS) capable flight simulators for aircrew training. To accurately model the aircraft for both the user and other DVE participants, the VC must present realistic cockpit displays and utilize accurate aerodynamics, onboard sensor, and weapons models. To achieve these capabilities, issues must be addressed in the following areas: 1) 3D computer graphics, 2) human-computer interaction, 3) virtual environments, and 4) aerodynamics, onboard sensor, and weapons models. To adequately address these issues, the VC requires a high resolution, visually realistic, functional portrayal of the cockpit interior and a user interface for immersive operation of its controls. The user interface must support a range of interaction that permits useful work to be accomplished in a manner that emulates the operation of the actual aircraft's controls. The VC has only minimal physical instrumentalities for control of the aircraft, typically only a throttle and stick. All other components of the aircraft cockpit are represented using 3D graphical models.

A major factor that leads the user to interact with the VC as though it were real and the user were present within the cockpit is the visual fidelity of the objects depicted in the DVE. Visual fidelity is the amount of detail represented in the object. Visual fidelity increases as the amount of graphical modeling detail increases up to the limits of the display hardware. In the VC, photorealistic portrayal requires the most detailed 2D and 3D graphical models that the hardware can support. High fidelity model detail directly competes with available computational resources for the following: 1) simulating aircraft aerodynamics, 2) performing collision resolution for interaction with other entities in the DVE, 3) portraying the environment outside the cockpit, 4) supporting human-computer interaction, 5) portraying other virtual environment actors, and 6) satisfying network traffic requirements. To achieve acceptable visual realism and satisfy the other six requirements we used a combination of polygonally-defined objects and texture mapping. The polygonal objects form the basis of the VC's instruments, consoles, switches, and knobs. We used these same techniques for the visual components for the VSP.

An important capability developed for the VC, which is relevant to the VSP project, is its capability for rapid reconfigurability between high fidelity representations of an aircraft. Rapid reconfigurability allows a single host computer to fulfill multiple aircraft roles within the duration of a single exercise by virtue of its ability to rapidly change its appearance and flight behavior at any time as needed. The main obstacles in achieving this capability were the definition of system components, development of a reconfigurable aerodynamics model, and specification of a software architecture that supports rapid reconfigurability. The VC's implementation is based on a loosely coupled implementation of the CODB architecture, a highly-parameterized aircraft aerodynamics model, several different multi-resolution 3D graphical models of aircraft cockpits, and a parameterized basic sensor model. The system components and 3D graphical models were defined using text descriptions and models of the interfaces for the F-15E and F-16 cockpits. The aerodynamics reconfigurability and fidelity objectives for the VC were attained using a parameterized aircraft aerodynamics model developed by the USAF Wright Laboratory at Wright-Patterson AFB, OH. Because this model is parameterized, we can use data files at runtime to reconfigure the VC to represent different aircraft in the DVE. The VC sensor model was developed in our lab.

The Solar System Modeler

The Solar System Modeler (SM) (18) is a virtual environment application that permits interactive visualization of the Solar System, the near-Earth space environment, and satellites in their correct orbits in deep space and around the Earth. We should note that other systems have been developed to aid in the analysis and visualization of spacecraft data (1,4,5,6,7,8,10,11,13). The Solar System Modeler project has investigated the interface, physical modeling accuracy, and rendering quality required to accurately emulate the Solar System, visualize its components, facilitate user interaction, and improve user understanding of the space environment of our Solar System. To facilitate user interaction, the Solar System Modeler provides direct manipulation functions that allow a user to interact with the planets, comets, asteroids, planetary moons, and multiple satellites. User understanding is aided by a variety of user controls, display devices, and icons. The Solar System Modeler allows the user to view satellites, planets, moons, and other Solar System components from a space-based point of view. The system also functions as a network actor in a distributed virtual environment by broadcasting satellite positions and orientations using the DIS protocols.

The SM achieves accurate physical modeling of satellite orbital motion by using published orbital element descriptions and orbital mechanics propagation code. The Solar System Modeler realistically models planetary motion using celestial mechanics and published astronomical tables of planetary ephemerii. The simplest and most intuitive description for elliptical satellite orbital motion uses six orbital elements, as portrayed in Figure 2. These "classical" orbital elements consist of the following: the semi-

major axis, which is one-half the length of the orbit; the eccentricity or shape of the orbit; the inclination of the orbit, which is the angle between the equatorial plane and the orbital plane; the ascending node, which specifies the planetary longitude at which the satellite's orbit crosses the equatorial plane; the argument of perigee, which specifies the angle between the ascending node and the point of perigee; and the time of perigee passage, which is the time at which the satellite passes closest to the Earth.

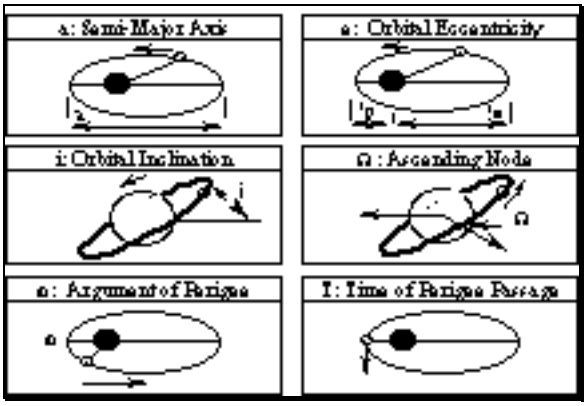

Figure 2: Classical Orbital Elements.

These elements are the basic foundation for computing orbital behavior but these elements ignore the effects of perturbations upon spacecraft behavior. However, accurate static orbits that take these perturbation effects into account can be computed using the NORAD SGP4 model (9), which replaces the time of perigee passage with mean anomaly. The SGP4 model, which is what the SM uses, calculates spacecraft position and velocity based on elapsed time from the time of perigee passage.

The SM virtual environment contains a 3D texture-mapped model of the Earth, a texture-mapped 3D model of an orbiting Moon, a starfield, and a 3D representation of the Sun. The Earth model is anchored at the origin of our Earth-Centered Inertial (ECI) coordinate system and only needs be rotated not propagated. The ECI Cartesian coordinate system's origin is the center of Earth, with the x-axis intersecting the equator at the vernal equinox, the z-axis intersecting the North Pole, and the y-axis perpendicular to the x-axis in the equatorial plane. The angular rotation per frame is computed based on the frame rate at which the simulation executes. Similarly, modeling the Moon's orbit is straightforward, based on the algorithm provided by Meeus (14). To accurately model the position of the Sun, we use an algorithm that assumes a purely elliptical orbit of the Earth, as described in Meeus (14) and in Seidelmann (16). Using the current epoch time, the geometric mean longitude and anomaly of the Sun are computed. Using the mean anomaly and the Sun's equation of center, the true longitude and anomaly are determined. Then, the eccentricity of the Earth's orbit and the true anomaly are used to compute the distance, R, from the Earth to the Sun. Finally R, the true longitude and the eccentricity are used to compute the ECI coordinates of the Sun.

Our interface paradigm for the SM is a "Pod" centered around the user's viewpoint that the user enters to travel throughout the virtual space environment (23). The controls are arrayed around the user on sets of 3D panels, with each panel devoted to a specific category of tasks. The SM's Pod consists of four panels: a left, center, bottom, and right panel. Each panel is comprised of sub-panels. The sub-panels are separate software objects that in turn contain the objects for controlling the Pod's capabilities as well as information display. Each panel holds a set of objects and displays related to the general functionality of the panel. The panels' objects allow the user to modify the attributes of the distributed virtual environment, such as moving the viewpoint, enabling the display of visual features, and attaching to actors. The buttons, displays, and panels were modeled so that they do not obstruct the user's view of the environment. All of the panels, controls and displays are within easy reach of the user. The SM interface forms the basis of the VSP interface.

REQUIREMENTS AND THEIR IMPLICATIONS

The basic set of requirements for the Virtual SpacePlane stems from several anticipated operational conditions. The VSP requirements stem from the following Manned SpacePlane system requirements: 1) to operate in air and in-orbit; 2) to perform military space missions; 3) to achieve rapid vehicle turn-around time, high availability, and field repairability; 4) to operate under human control; 5) to perform a wide variety of missions or mission subcomponents semi-autonomously (without assistance from mission control); 6) to achieve maximum automation so that crew size and crew workload are minimized; and 7) to fly using a self-contained virtual cockpit that can be used on the ground or within the real-world spaceplane. Using these real-world spaceplane requirements as a basis, we derived the following seven VSP application requirements.

The *first* of these operational considerations is that the VSP must operate autonomously. The expected modes of use and mission profiles of the MSP indicate that it may have to perform many of the functions in-orbit that are currently performed by mission control. Some of these autonomous tasks include fault detection and diagnosis, system repair, orbit re-planning, re-entry re-planning, mission re-planning, and consumables re-planning. These tasks are all data intensive and can quite easily overwhelm the crew unless the interface and the data management facilities are designed to pro-actively assist the crewmember in the tasks at hand while also considering the mission objectives and other relevant parameters. To accomplish the tasks mentioned above, the crew will require a capability to

access information about the spaceplane in real-time; browse technical manuals in-orbit; assess tradeoffs between time, fuel consumption, and mission objectives when re-planning missions; and assessing the adequacy of consumables for performing re-planned missions. This level of activity is far above the current level of astronaut in-orbit activity, which largely consists of executing checklists provided by mission control and looking to mission control to make decisions concerning mission accomplishment and mission plan changes. One subsidiary requirement is that the VSP requires an on-board intelligent system to assist the crew by performing routine analysis of mission and orbit parameters and assessing the impact of action or inaction in a circumstance. Without VSP-based intelligent assistance, the crew will be overwhelmed with information and their ability to perform the mission will suffer.

A *second* VSP requirement is that the application exhibit correct dynamics performance throughout its flight regime and be able to rapidly respond to changes in the dynamics performance of the real-world MSP. These requirements must be met because the real-world spaceplane is still being designed and the performance criteria can be expected to change. As a result, the VSP must be able to switch between aerodynamics and astrodynamics code that models its dynamic flight performance within each flight regime. The switch between codes on launch or landing must not be detectable by the user and the VSP should not have a break in its flight performance. In addition, the dynamics models will need to be able to be modified in their performance to match the changes in the aerodynamics or astrodynamics performance of the actual MSP. To satisfy this requirement the VSP application must have a parameterized flight model for aerodynamic and astrodynamic flight that can be readily altered.

A *third* requirement is that the spaceplane must be able to operate within a distributed virtual environment. Therefore, the VSP application must be able to operate within both a DIS-defined virtual environment and a DOD High Level Architecture (HLA) defined virtual environment. As a result, the architecture must be able to switch between application software that can send and receive using the data transmission formats used in each type of DVE. The VSP must transmit its position, velocity, orientation, orbital parameters, and possibly individual crewmember actions. In addition, the VSP must be able to receive and process information relating to the position, velocity, and orientation of other entities, atmospheric and solar weather conditions, and commands from crewmembers.

A *fourth* requirement is that the VSP have an accurate, high fidelity presentation of the ground, the Earth's surface as seen from orbit, and the contents of the space environment. As a result, the VSP must support automatic level of detail switching so that the 3D computer graphics computational burden is minimized. Other orbiting bodies, such as the Moon and Earth orbiting satellites, must be portrayed within the VSP application and they must also propagate properly in their orbits. The stars must be properly positioned relative to the Earth and to each other. Finally, the launch and landing locations, Earth-bound sensor targets, and sensor targets in space must be modeled in detail and across the electromagnetic spectrum.

A *fifth* requirement is that the VSP support rapid prototyping of the cockpit's user interface and of the software architecture and its components. A significant portion of this requirement is assisting the user in accomplishing tasks, and identifying and eliminating extraneous data from the cockpit displays. The requirement for support of rapid prototyping arises from several considerations. First, at this time, no decision concerning the astronaut suit that will be worn in-orbit has been made. The crew may be in anything from a full-pressure suit environment to a shirt-sleeves environment. As a result, the interface may need to be able to be tested by users in an environment that ranges from very little capability for haptic operation because large, cumbersome gloves are being worn, to one that permits the same level of haptic interaction as in an office. Second, there is no experience in designing a spaceplane interface to guide us in developing the VSP interface or to guide in the development of the MSP. The only starting point we have for VSP interface development is the cockpit layout of the NASA Space Shuttle. However, the shuttle cockpit interface is unsuitable for the VSP. The current suite of displays for the shuttle are generally crowded with information, most of which are irrelevant to the tasks that the astronauts perform in orbit. One task the VSP must support, then, is identification of information that is important to the crew and of the best means for presenting that information to the crew when needed. As a result, the interface will evolve as usability analysis (15) points out flaws in the placement or presentation of information to the crew. Third, the overall physical design, missions, sensor suite, and operational philosophy for the MSP has not been identified, much less specified. As a result, the dynamics models, sensor models, and even crew compartment layout will change over time and new systems will need to be integrated into the VSP architecture. Given that this project is undertaken to identify alternatives and explore as much of the design space as possible, rapid integration of software is of equal priority with run-time performance.

Sixth, the crew of the VSP must be able to change its landing point and orbit as directed by mission control. This requirement means that the crew must have sufficient computational resources on-board to rapidly perform a series of orbital change computations, an interface that will allow them to specify the required

changes, and an interface that will allow them to evaluate tradeoffs between solutions.

Seventh, the VSP must contain a suite of simulated sensor systems that accurately emulate the sensors that the MSP is expected to carry. Some of these sensors will be a permanent part of the spaceplane, and hence a permanent part of the VSP, while others will be mission specific. The VSP user interface and software architecture must be able to support integration of both temporary and permanent sensors into the application and the presentation of the simulated sensor outputs.

Satisfaction of these requirements calls for an architecture and an interface that supports experimentation, rapid prototyping, rapid interchange of software components, and an interface that supports layout experimentation as well as experimentation with presentation and interaction components. The next section presents our solution to these challenges.

A SOLUTION

The solution we propose is composed of four major components: 1) the Common Object DataBase, 2) an Information Pod approach to the interface, 3) reuse of code from the VC and SM projects, and 4) reconfigurability as demonstrated in the VC. We should emphasize that our architecture and design were not developed to maximize code reuse, but rather to meet the requirements outlined in the preceding section.

Design Overview

The design we developed, shown in Figure 3, uses the CODB as its core, in that all data that moves between application components passes through the CODB using a container. The DVE Object class encapsulates all the functionality required to communicate with the CODB and to represent a generic object for the virtual

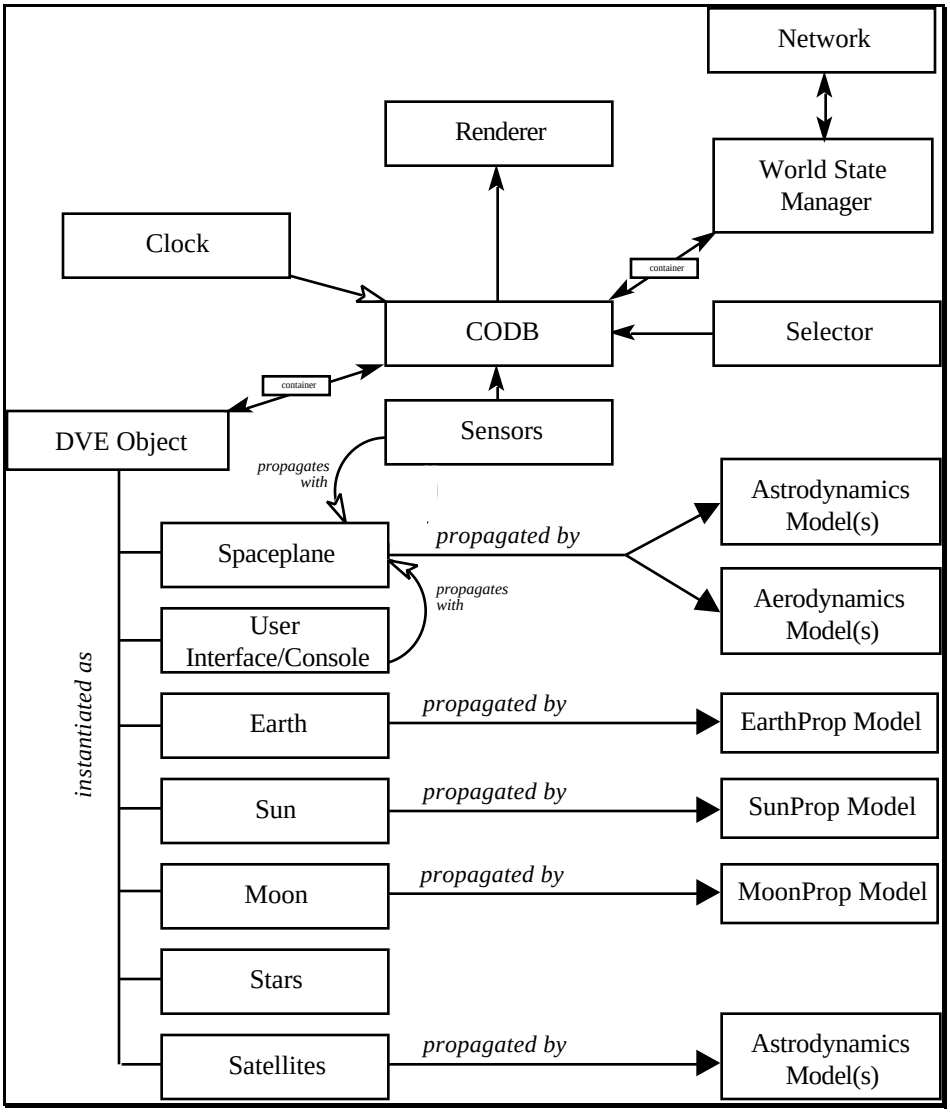

Figure 3: Virtual SpacePlane Design.

environment. Because there are many different orbital propagation models that we must employ, propagation is not part of the base class definition other than as a placeholder; the actual dynamics model(s) used for an object are defined when the object's class is defined. Within the DVE Object class are the classes for the spaceplane, the spaceplane operator's console, the Earth, the Moon, and the Sun. As in the VC, the CODB provides an inheritable, reusable interface to the computer hardware, input and output devices, and Silicon Graphics' Performer software.

The VSP's DIS interface is our World State Manager 3.0 (WSM). The WSM provides a protocol data unit send and receive capability. The World State Manager accepts containers from the CODB with data for VSP entity state PDUs and dispatches containers to the CODB with data received in entity state PDUs. Because of this clean separation between the communications facilities and the remainder of the application, we can readily adapt the VSP to the HLA architecture, and this work is currently underway.

The Renderer class manages all of the rendering functions. The Renderer relies upon Performer to execute most of the culling and drawing operations as well as to maintain the scene database. The Clock object provides the application with the current Julian date for astrodynamics calculations and could serve to synchronize the VSP with an external clock in the HLA. The Selector object manages intersection testing and interface component picking for the VSP cockpit and the Console object. The Stars object is responsible for the starfield in the VSP. The VSP has over 33,000 stars. Each star is positioned at its correct declination and right ascension with an accurate color and apparent magnitude using data in the Astronomical Data Center's *Catalog of 33,342 Stars*.

The various astrodynamics models in the VSP are responsible for moving the celestial objects through their orbits. The VSP's Aerodynamics model is currently the one developed for the VC with its parameters altered to portray high Mach flight. The Astrodynamics models were ported from the SM and the one used for an object depends upon the type of object being propagated. For example, the Satellite propagation model uses the NORAD SGP4 propagation code (9) whereas the Moon propagation model uses code from the book by Seidelman (16). Because of the wide variety of dynamics models used in the VSP, careful selection of the VSP application's coordinate system was paramount to realistic and computationally inexpensive portrayal all entities. Therefore, like the SM the VSP uses an Earth Centered Inertial coordinate system for the space environment model in the VSP.

The SpacePlane object encapsulates all of the aspects of the VSP except for the cockpit user interface. It uses a variety of aerodynamics and astrodynamics objects to compute the motion of the VSP during a mission from launch through landing. The Sensors object is attached to the SpacePlane object and is responsible for computing the entities that are visible to the VSP with its onboard sensor systems. The User Interface/Console object manages the display of all of the VSP data and, using the Selector object, determines the user's commands to the system. The User Interface/Console object is of critical importance to achieving the VSP's objectives and is described further in the following subsection.

The VSP User Interface

The central issue for the design of the VSP interface is the portrayal of capabilities of an interface that uses technologies that do not yet exist. Our design assumes that the real-world spaceplane will use display technology beyond that of the CRT, such as very large flat-panel displays. One of our working assumptions for the VSP interface design has been that the current controls and displays in the NASA Space Shuttle will not be adequate for the VSP, however they do provide a starting point for assessing VSP information display requirements. We also assumed that no amount of display space will be adequate to support the continuous display of all of the mission data. As a result, the displays must be reconfigurable in real-time as the VSP moves through its mission profile and in response to user activity. Additionally, because the MSP crewmember will operate with little or no ground support, automatic information management capabilities are required in the VSP. It will also be necessary for the interface to anticipate the user's needs and retrieve additional relevant information and present it to the crewmember.

Given the assumptions above and the requirements presented earlier, we decided to base the user interface on a modified VC Pod interface and to postpone physical implementation of the interface hardware. We believe that this approach will provide maximum flexibility for experimentation with the interface elements and with novel selection devices. This freedom will also allow us to modify the VSP interface to match that of the MSP as it develops and to prototype MSP interfaces without the expense incurred by manufacturing the display consoles and their components.

Our initial concept for the VSP operator's console is presented in Figure 4. The panels shown there are notional and user-configurable. Different panels sets can appear at any location in the interface depending upon the task at hand and the information required. The console wraps around the operator and surrounds the user with relevant data by placing it on both the flat, table-top like portion of the interface as well as on the vertical portion of the interface. The far left side of the console contains fault annunciation and diagnosis displays. These displays are always presented because

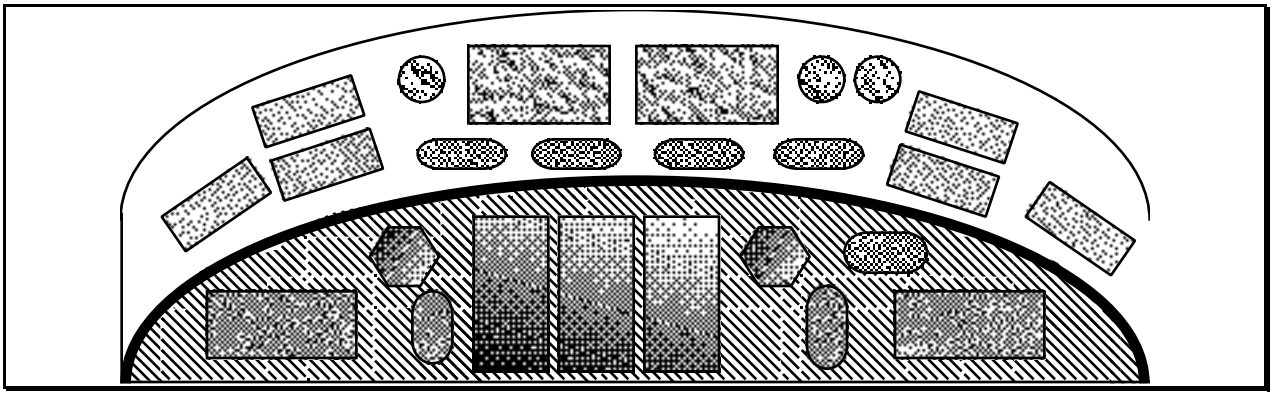

Figure 4: Conceptual Virtual SpacePlane Cockpit Interface

of their critical nature. These sets of panels contain a 3D display of the VSP and its systems. When a fault occurs, the affected system(s) are highlighted on the display. By selecting the affected system, a HTML, WWW-like interface on that portion of the console appears. This display contains information about the affected systems, related systems, diagnosis and fault-isolation tests, and possible repair procedures. We intend that all checklists for the VSP will use HTML for markup and a WWW-like browser for their display.

The paradigm that we adopted for flying the VSP is that of flight management and not flight control. Therefore, we do not provide a throttle and stick for the user. Instead, we provide interface devices that allow the user to control orientation, velocity, and VSP orbit by inserting the desired final state or input into the system and then allowing the VSP software to compute the direction and duration of the rocket firings. Input may be accomplished using virtual sliders, buttons, direct manipulation, or some combination thereof. To assist the user in maintaining awareness of the VSP and its environment, we show the location of the VSP in its orbit along with target objects projected onto a depiction of the Earth as a globe. The interface will provide displays to show the user information regarding fuel status, remaining velocity change capability (delta-V), consumables status, possible recovery bases, solar wind, other near-Earth orbital weather information, and additional information of interest.

Because the VSP interface must be used for mission evaluation and for preliminary MSP training, it must be complete and functional. By using the Pod interface and the CODB architecture, we have sufficient flexibility in the design and implementation of the VSP to allow us to replace individual interface elements or the entire interface if needed. The CODB allows us to separate the calculations for the display from the display elements themselves, thereby permitting us to freely experiment with the display without affecting the underlying code. The Pod interface software provides us with the software infrastructure needed to allow us to rapidly reconfigure the interface in real-time while the VSP is operating within the DVE.

SUMMARY AND FUTURE WORK

In this paper we presented the requirements and design of the VSP as derived from the MSP project's objectives. The VSP requirements coupled with the need to rapidly assemble a prototype, led us to reuse code from the Virtual Cockpit, Solar System Modeler, and Common Object DataBase (CODB) applications. We presented a review of the highest level requirements for the VSP as they are currently known and described their implications for the VSP project. We also presented a brief description of the VSP software architecture and the user interface design. The VSP project is currently underway and the initial interface work is nearly complete. We have identified appropriate astrodynamics code and the initial software integration based upon the VSP architecture is nearly concluded.

One problem we have encountered that was unexpected is difficulty in inserting a WWW-like browser into a VE application. Because of this difficulty and of the need to realize a functional prototype quickly, we have foregone development of a WWW-like browser and have instead decided to simulate its operation using texture maps and clickable icons within the texture map display. This is an acceptable short-term solution, but we must implement a suitable browser or find an alternative in the near future.

One issue that must be addressed is that of space weather. Currently, we lack the capability to generate space weather within the VSP application. We are currently in the process of locating available space weather code and determining its suitability for incorporation into a DVE.

In this same vein, we also lack accurate sensor models and the aerodynamics model must be improved. The current sensor models that we use in the VSP are rough approximations of the actual devices and may depart significantly from actual performance characteristics. We hope that in the near future we will be able acquire more accurate and representative parameters for these systems. The current aerodynamics model uses

approximate, but realistic, parameters for MSP atmospheric flight.

A final important topic is the evaluation of the user interface and its suitability for the intended space operations. Thus far, we have confined our evaluation efforts to the application of usability heuristics as described by Nielson (15). While these techniques are suitable for the initial phases of interface development and for most of the interface components, they are not sufficient for those components of the interface that will be used frequently or that will be used to manage critical situations. Therefore, we plan to identify these critical and frequently used components and to test them thoroughly.

REFERENCES

1. Bagiana, F. (1993) "Tomorrow's Space: Journey to the Virtual Worlds," *Computers & Graphics*, vol. 17, no. 6, pp. 687 - 690.
2. Blau, B.; Hughes, C. E.; Moshell, J. M.; & Lisle, C. (1992) "Networked Virtual Environments," *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, Cambridge, Massachusetts, pp. 157-160, 29 March - 1 April.
3. Blau, B., Moshell, J. M., & McDonald, B. (1993) "The DIS (Distributed Interactive Simulation) Protocols and Their Application to Virtual Environments," *Proceedings of the Meckler Virtual Reality '93 Conference*, San Jose, California, pp. 19 - 21.
4. Bricken, W. & Coco, G. (1994) "The VEOS Project," *Presence*, vol. 3, no. 2, pp. 111-129.
5. Eyles, D.E. (1987) "A Computer Graphics System for Visualizing Spacecraft in Orbit," *Proceedings of 1987 Conference on Spatial Displays and Spatial Instruments*, Moffett Field, CA.
6. Feng, X.; Niederjohn, R. J.; & McGreevy, M.W. (1991) "An Intelligent Control and Virtual Display System for Evolutionary Space Station Workstation Design," *Proceedings of the Fifth Annual Workshop on Space Operations Applications and Research (SOAR)*. pp. 582-587.
7. Hagedorn, J.; Ehrner, M.; Reese, J.; Chang, K.; & Tseng, I. (1986) "A Computer Graphics Pilot Project: Spacecraft Mission Support with an Interactive Graphics Workstation," *Proceedings of the Society for Computer Simulation Simulators Conference*, pp. 61-64.
8. Hallet, H. & Jahnke, R. (1987) "Space Simulation Using Computer Generated Imagery," *Proceedings of the Conference on Aerospace Behavioral Technology*, pp. 199-206.
9. Hoots, F. & Roehrich, R. L. (1980) "Models for Propagation of NORAD Element Sets," *Space Track Report Number 3*, Peterson AFB, CA.
10. Kobayashi, H.; Yamamoto, Y.; Oka, M.; Goma, K. (1973) "Interactive Graphic Display of Satellite Orbital Information," *Proceedings of the Tenth International Symposium on Space Technology and Science*, Tokyo, Japan, pp. 901-908.
11. Mara, S. (1993) "VISIM," *Journal of the British Interplanetary Society*, vol. 46, pp. 203-208.
12. McCarty, W. D.; Sheasby, S.; Amburn, P.; Stytz, M. R.; & Switzer, J. C. 1994. "A Virtual Cockpit for a Distributed Interactive Simulation Environment," *IEEE Computer Graphics and Applications*, vol. 14, no. 1, January, pp. 49-54.
13. McGreevey, M. (1993) "Virtual Reality and Planetary Exploration," in *Virtual Reality -- Applications and Explorations*, Alan Wexelblat (ed), Academic Press: New York, NY.
14. Meeus, J. (1991) *Astronomical Algorithms*. Willmann-Bell: Richmond, VA.
15. Nielsen, J. (1993) *Usability Engineering*. Academic Press Professional: Boston, MA.
16. Seidelmann, P. K. (ed.). (1992) *Explanatory Supplement to the Astronomical Almanac*. University Science Books: Mill Valley, CA.
17. Stytz, M. R.; Adams, T.; & Banks, S. B. (1997) "A Rapidly Reconfigurable Photorealistic Virtual Cockpit," *The SCS 1997 SMC Simulation Multiconference: 1997 Military, Government, & Aerospace Simulation Conference*, Atlanta, GA, 6 - 10 April, pp. 211-216.
18. Stytz, M. R.; Vanderburgh, J.; and Banks, S. B. (1997) "The Solar System Modeler," *IEEE Computer Graphics and Applications*, vol. 17, September, to appear, IEEE Press.
19. Stytz, M.R. (1996) "Distributed Virtual Environments," *IEEE Computer Graphics and Applications*, vol. 16, no. 3, pp. 19-31.
20. Stytz, M. R., Adams, T., Garcia, B., Sheasby, S. M., & Zurita, B. (1997) "Developments in Rapid Prototyping and Software Architecture for Distributed Virtual Environments," *IEEE Software*, vol. 12, to appear.
21. Stytz, M. R. & Diaz, M. (1996) "The Photo-Realistic Virtual Cockpit," *The SPIE/SCS Joint 1996 SMC Simulation Multiconference: 1996 Military, Government, & Aerospace Simulation Conference,* vol. 28, no. 3, New Orleans, Louisiana, 8 - 12 April, pp. 71 - 76.
22. Stytz, M. R. & Schneider, N. (1996) "Performing Entity Control Switching Between Computer-Controlled and Manned Aircraft Hosts in a Distributed Interactive Simulation", *The SPIE/SCS Joint 1996 SMC Simulation Multiconference: 1996 Military, Government, & Aerospace Simulation Conference*, vol. 28, no. 3, New Orleans, Louisiana, 8 - 12 April, pp. 89 - 94.
23. Stytz, M.R.; Banks, S.B.; Kesterman, J.J.; Rohrer, J.J.; & Vanderburgh, J.C. "The Information Pod: An Interface for Immersed Control in a Virtual Environment," *Interfaces '97 - Man-Machine Interaction*, Montpellier, France, 28 - 30 May 1997, pp. 24-27.