A Portable Animation Facility for the Design of Autonomous Underwater Vehicles

by

Georgina Bronwyn Hackett
B.A.Sc., University of British Columbia (UBC), 1995

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Mechanical Engineering.

We accept this thesis as conforming
to the required standard

[redacted]

Dr. M. Nahon, Supervisor (Dept. of Mechanical Engineering)

[redacted]

Dr. Z. Dong, Member (Dept. of Mechanical Engineering)

[redacted]

Dr. J. Collins, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. M. Seto, External Examiner (International Submarine Engineering)

Supervisor: Dr. M. Nahon

Abstract

The development process of an Autonomous Underwater Vehicle (AUV) typically requires evaluation of the stability and controllability of a large number of candidate vehicle configurations. The three-dimensional animated simulation presented in this thesis provides a computer-based tool with which initial screenings of candidate vehicle configurations can be performed. The three dimensional animation includes an animated AUV and instrument panel. The application is written with Open Inventor, a C++ graphics library developed by Silicon Graphics and based on OpenGL. The entire package was initially developed on a Silicon Graphics Indy workstation and then ported to a PC running Windows NT and equipped with an OpenGL compatible graphics card.

Examiners:

[redacted]

Dr. M. Nahon, Supervisor (Dept. of Mechanical Engineering)

[redacted]

Dr. Z. Dong, Member (Dept. of Mechanical Engineering)

[redacted]

Dr. J. Collins, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]

Dr. M. Seto, External Examiner (International Submarine Engineering)

Table of Contents

List of Figures

List of Tables

Acknowledgements

A number of people assisted me with this work. I gratefully acknowledge their generous donations of time and expertise.

Special thanks are extended to Dr. Meyer Nahon for providing me with the opportunity to participate in this exciting project, and for his guidance and insight throughout.

Thanks are extended to Kai Benndorf from the University of Paderborn (Germany) for allowing me to use his Open Inventor extension to wxWindows. And to Kai and Julian Smart, from the Artificial Intelligence Applications Institute (UK), for their patience and long distance assistance with setting up wxWindows.

I wish to thank Vince den Hertog of International Submarine Engineering for traveling to Victoria to evaluate the animation from the perspective of an AUV designer.

I am grateful to my family and friends for their caring support and encouragement throughout the past two years.

Chapter 1

Introduction

1.1 Background

Autonomous Underwater Vehicles (AUVs) are designed to complete any number of tasks intelligently and independently. Potential exists for AUVs to carry out oceanographic surveys, and be equipped to monitor various environmental concerns such as oil spills, toxic waste levels, and fish stocks, or become underwater workhorses for the oil industry [1]. The military has a continuing interest in AUV technology for mine countermeasures and surveillance, [2].

Currently, AUV design relies on past experience, tow tank experiments, and hardware testing, all of which are taxing on a designer's available resources. The development process of an AUV typically requires many iterations in order to arrive at some 'optimal' design. During this process, the stability and controllability of a large number of candidate vehicle configurations must be evaluated in typical missions. Field testing is risky as communication with the vehicle is limited by a low bandwidth acoustic link. It is difficult to know where the vehicle is, communicate with it, and

recover it if control is lost. An intuitive understanding of vehicle behavior in its own environment is hampered by the designer's inability to watch the vehicle in operation. A three-dimensional animation can provide an efficient means to perform an initial screening of candidate vehicle configurations, and allow a better use of resources to discard grossly inadequate designs. Simulation can be effectively used as a filter to reduce the experimental effort that still must be performed for the detailed design of the AUV.

1.2 Related Work

Research into the development of virtual worlds for simulation of Autonomous Underwater Vehicles (AUV) is ongoing at a number of institutions.

Viewing an AUV as it carries out a mission has been the secondary goal of several research projects. At the MIT Department of Ocean Engineering Design Laboratory, a simulation model was developed to test AUV navigation applications [3]. Its focus is largely on map decomposition and environment customization. The simulator and virtual environment were developed for portability between UNIX based operating systems. The work was extended with the development of the NavViewer, a visualization tool for developing a map based AUV navigation algorithm, [4].

In Japan, Kuroda, Aramaki, and Ura, [5], are developing a "synthetic world" for an actual AUV. The AUV hardware and software systems are integrated with the virtual world. This allows the vehicle to interact with the virtual world while operating in a test pool. The real-time rendered environment provides a view of the virtual environment in which the AUV is operating. This allows real-time testing of the actual AUV software and hardware in a safe environment. This large scale

system is intended to make use of several networked systems and be available to multiple users.

Research at Imetrix is aimed at developing a virtual environment to aid in training ROV pilots, [6]. Although this work focuses on Remotely-Operated Vehicles (ROVs), rather than AUVs, it provides some insight into the capabilities of Open Inventor. The system utilizes multiple rendering windows to provide the virtual version of the equipment available to ROV pilots including several viewpoints of the environment, vehicle instrumentation, and vehicle controls. The software operates the vehicle dynamics simulation and a training tutorial module.

Design of a user interface enhances the ease with which a user interacts with an application. An interface designed to increase the usability of the Autonomous Underwater Vehicle Controller software was developed at Texas A&M University. The interface provides access to functions for mission planning, execution, and post-processing [7]. A world view of the AUV and its environment was developed as a part of the system. Integrated with either a 2D or 3D world view is instrumentation which can present data from any of the 21 software subsystems.

A simulation and 3D visualization system is under development at Florida Atlantic University (FAU). The software is being developed jointly with the Naval Postgraduate School specifically for FAU's AUV *Ocean Explorer*. It is intended for motion visualization, in-lab debugging, and testing mission specific strategies [8].

At the University of Victoria, Daniel Wall created a simulation package to assist with the development of AUVs and the intelligent controllers to guide them [9]. The main component of the work is a Graphical User Interface (GUI) based program to adapt the simulation source code for the AUV dynamics simulator, the sensor models, and the intelligent controllers. Missions are executed off line. Mission histories can be

viewed in an AutoCAD wireframe animation, or in single frame stills. To verify the tool, an AUV was developed with a controller to dock it at a simulated underwater docking platform.

The largest body of work involving 3D visualization and simulation of Autonomous Underwater Vehicles is from the Naval Postgraduate School. A simple 3D visualization system developed for mission planning and control is presented in [10]. Since that time Don Brutzman has developed an integrated simulation and virtual world for the development of AUVs. The system was developed for use during any phase of research involving the development, testing, and post mission analysis of the NPS AUV *Phoenix*. A more detailed description of the NPS Virtual World is presented in Chapter 2.

1.3 Goals

The 3D AUV Viewer (Viewer) developed in the present work is an animation facility targeted at a single end user whose immediate concern is understanding the dynamic behavior of various AUV designs. It provides a window into the remote and inaccessible environment in which an AUV operates. Intended as an initial evaluation tool, it does not require sonar models, collision avoidance capability, or hardware in the loop extensions. These tools are left to detailed design phases and more complex simulation tools such as those described in [5], [8], [11].

An instrument panel provides the designer with information for understanding vehicle motion. The information is presented in an easily synthesized format, and is organized in such a way that the designer is able to choose what information is displayed, the format in which it is displayed, and where in the panel it is displayed.

The flexibility provided by the panel ensures the animation can be customized to suit a variety of analysis tasks, and the preferences of various users.

An important feature of the facility is that it is based on software which is widely available and that appears likely to become a standard for animation. Another prime factor in this decision is that the hardware and software be available at a reasonable cost. In order to reach a wide range of users the software will be a PC based system, but also be compatible with other common operating platforms.

To maintain ease of use the Viewer will have a straight forward layout, and make use of standard graphical user interface tools. With the advancements in 3D rendering abilities of graphics capable PCs, wireframe models are abandoned for more realistic, animated, 3D creations of the vehicle and its surroundings.

Figure 1.1 provides a general overall layout of the simulation and animation package. The focus of this thesis work is on components 3 and 4 of Figure 1.1. These components encompass the animation of the AUV and the instrumentation, as well as rendering of the AUV environment. The Viewer developed in this work is configured to operate from a mission data file which is generated by the dynamics model, component 1 of Figure 1.1. The motion data is stored in a data file, and used by the animation. In the future, the animation could be provided with a real-time interface to the dynamics model, should that prove advantageous. The present set up provides some flexibility in the use of the animation to visualize simulation generated data or logged data from a real vehicle. Future development of the dynamics model includes adding thruster models and enhancing the environment model (component 2 of Figure 1.1).

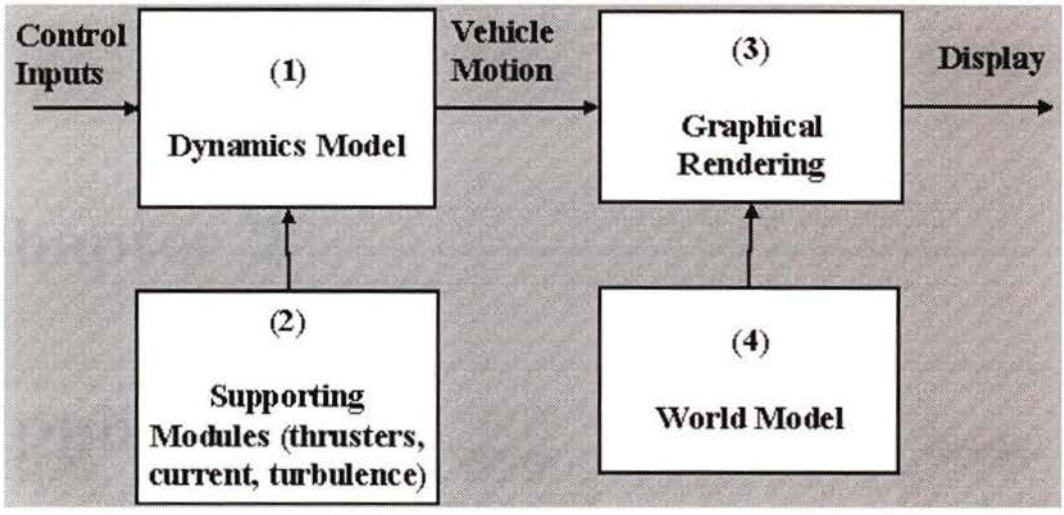


Figure 1.1: Layout of the AUV simulation and animation package.

1.4 Thesis Layout

Chapter 2 discusses the NPS Virtual World, outlines the features of the Viewer developed for this project, and presents several applications for it. Chapter 3 provides details regarding the implementation of the Viewer and the accompanying instrumentation. Also presented are the performance tests and results for the SGI based animation. Chapter 4 outlines the graphical user interface. Chapter 5 identifies the hardware and software requirements of the PC based animation. Chapter 6 presents conclusions and recommendations.

Chapter 2

Applications

This chapter outlines the vehicle kinematics, and presents a sample of the 2D data plots traditionally used to analyze vehicle motion. Because the NPS Virtual World was initially evaluated as a potential base for our design tool, the chapter continues with a description of the NPS Virtual World and a discussion of its advantages and disadvantages for our project. Following sections outline the features of the Viewer that was developed for our project. The Viewer has potential uses as a visualization tool during vehicle field trials in addition to its function as a design evaluation tool. These alternatives are also described.

2.1 Vehicle Kinematics

For the analysis of AUV motion in 6 DOF, two reference frames are defined. There is an inertial reference frame which is fixed to the ocean surface. It is a right handed coordinate system with the X_I - Y_I plane being coincident with the mean sea surface. The X_I axis is directed along true North and the Z_I axis points vertically down. The

Y_I axis completes a right handed coordinate system. The second reference frame is fixed to the vehicle and moves with it. The origin of the body fixed frame is generally at the center of gravity of the vehicle with the x_b axis pointing forward, the y_b axis directed starboard, and the z_b axis pointing down. Table 2.1 presents the variables used to describe the vehicle kinematics. The reference frames and the motions observed in each are shown in Figure 2.1.

DOF		Linear and Angular Velocity (Body Frame)	Position and Euler angles (Inertial Frame)
1	motion in x direction	u	X_I
2	motion in y direction	v	Y_I
3	motion in z direction	w	Z_I
4	rotation about x	p	ϕ
5	rotation about y	q	θ
6	rotation about z	r	ψ

Table 2.1: AUV motion variables.

For the purposes of animating vehicle motion, we need to know the three positions and three Euler angles shown in Table 2.1, as a function of time. In the present work, these are stored as columns in a data file. Other information may be stored in the file, such as the six velocities shown in Table 2.1, for display on the instruments. For the purposes of performing vector transformations between the inertial and body frame, it is also useful to define a rotation matrix, $\mathbf{T}_1$.

The matrix $\mathbf{T}_1$, in Equation 2.1, is used to transform vectors from the body fixed frame to the inertial frame. It is derived from a specific ordering of sequential rotations

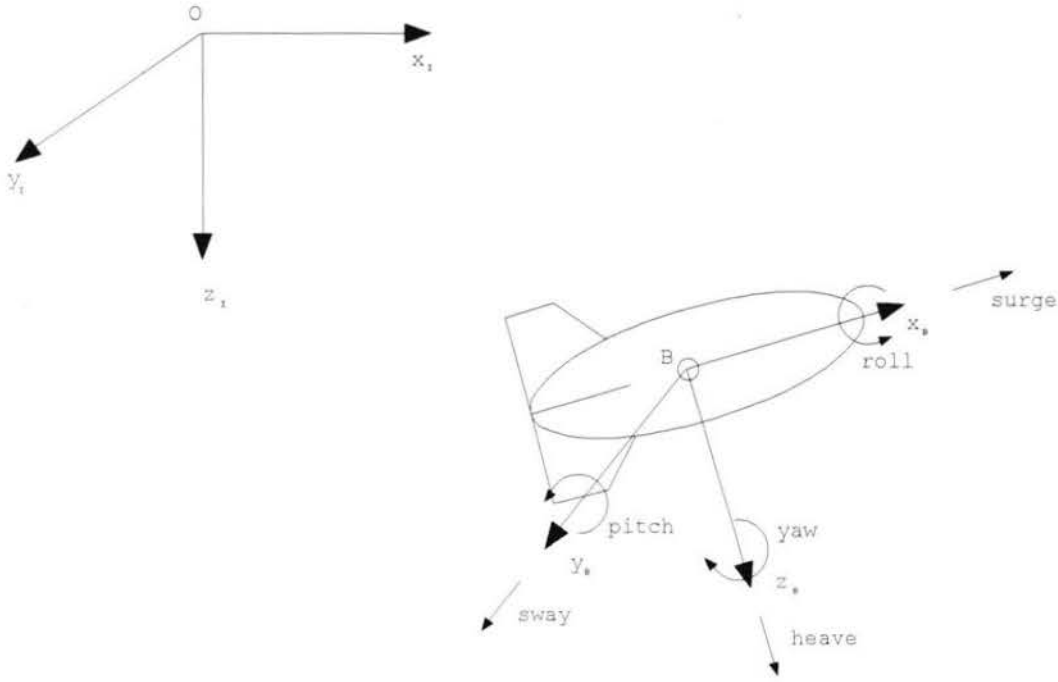


Figure 2.1: Reference frames.

specified in terms of Euler angles, ϕ, θ, and ψ. The order is an accepted standard for most vehicle motion and control applications. First an intermediate reference frame is created by translating the inertial frame, without rotating it, so its origin coincides with the origin of the body frame. The new frame is rotated about its Z axis by an angle ψ. The resulting frame is rotated about its Y axis by an angle θ. The last frame produced is rotated about its X axis by an angle ϕ. The resulting reference frame coincides with the body fixed co-ordinate system of the vehicle. The rotation matrices which define these rotations are as follows:

$$C(x,\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix}, C(y,\theta) = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix}, C(z,\psi) = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

$$s\cdot \equiv \sin(\cdot), \quad c\cdot \equiv \cos(\cdot)$$

The composite rotation matrix, $\mathbf{T}_1$, is then formed by multiplying the three individual rotation matrices:

$$\mathbf{T}_1 = C^T(z,\psi)C^T(y,\theta)C^T(x,\phi) \tag{2.1}$$

$$\mathbf{T}_1 = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\phi c\theta \end{bmatrix} \tag{2.2}$$

2.2 2D Plots

Traditional analysis of AUV motion demands an ability to integrate large volumes of information, provided in 2D plots, to understand the 3D behavior of the vehicle. Figures 2.2-2.7 are examples of such plots for a spiral dive maneuver of the ARCS vehicle. Details regarding the ARCS vehicle and the dynamics simulation using to generate this motion data are in Section 2.5.

The inertial position and linear velocity components of the AUV are shown in Figure 2.2, while the orientation and angular rates are provided in Figure 2.3. Obviously, trying to synthesize a view in one's mind of how the vehicle is moving, from these plots, is quite difficult even for this relatively simple maneuver. In cases where the vehicle behaves erratically or in an unstable manner, this synthesis becomes even more difficult. A 3D animation incorporates all of the position and orientation information into an animated model of the AUV in question. Figure 2.4 shows the control deflections used throughout the maneuver. The velocities, orientations, control surface deflections, as well as other data can be represented in instrumentation to provide a more immediate understanding of vehicle behavior.

Figure 2.5 shows the total velocity throughout the maneuver, where the total velocity is the square root of the sum of the squares of the three velocity components expressed in the inertial (or body) frame. Figure 2.6 presents the inertial position of the AUV in the X-Y and the X-Z plane to better understand the maneuver performed. Finally, Figure 2.7 presents the inertial position data in a 3D plot, and includes projections of the maneuver onto each plane. It should be noted that, in Figures 2.2 - 2.7, positive values of z imply a downward displacement, thus making interpretation of the plots which include z somewhat counterintuitive.

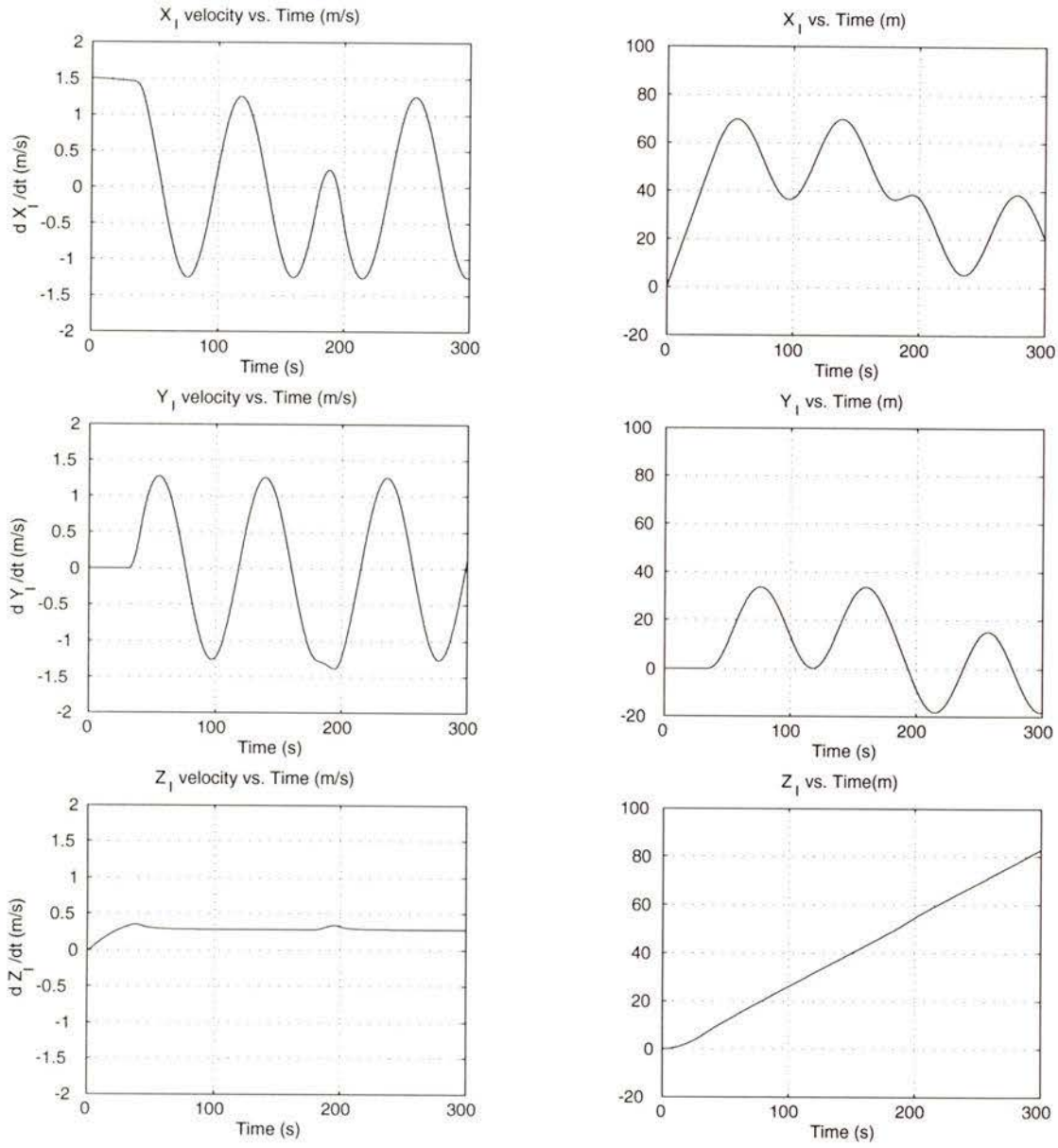


Figure 2.2: Variables: $\dot{\mathrm{X}}_I$, $\dot{\mathrm{Y}}_I$, $\dot{\mathrm{Z}}_I$, X_I, Y_I, Z_I vs. time.

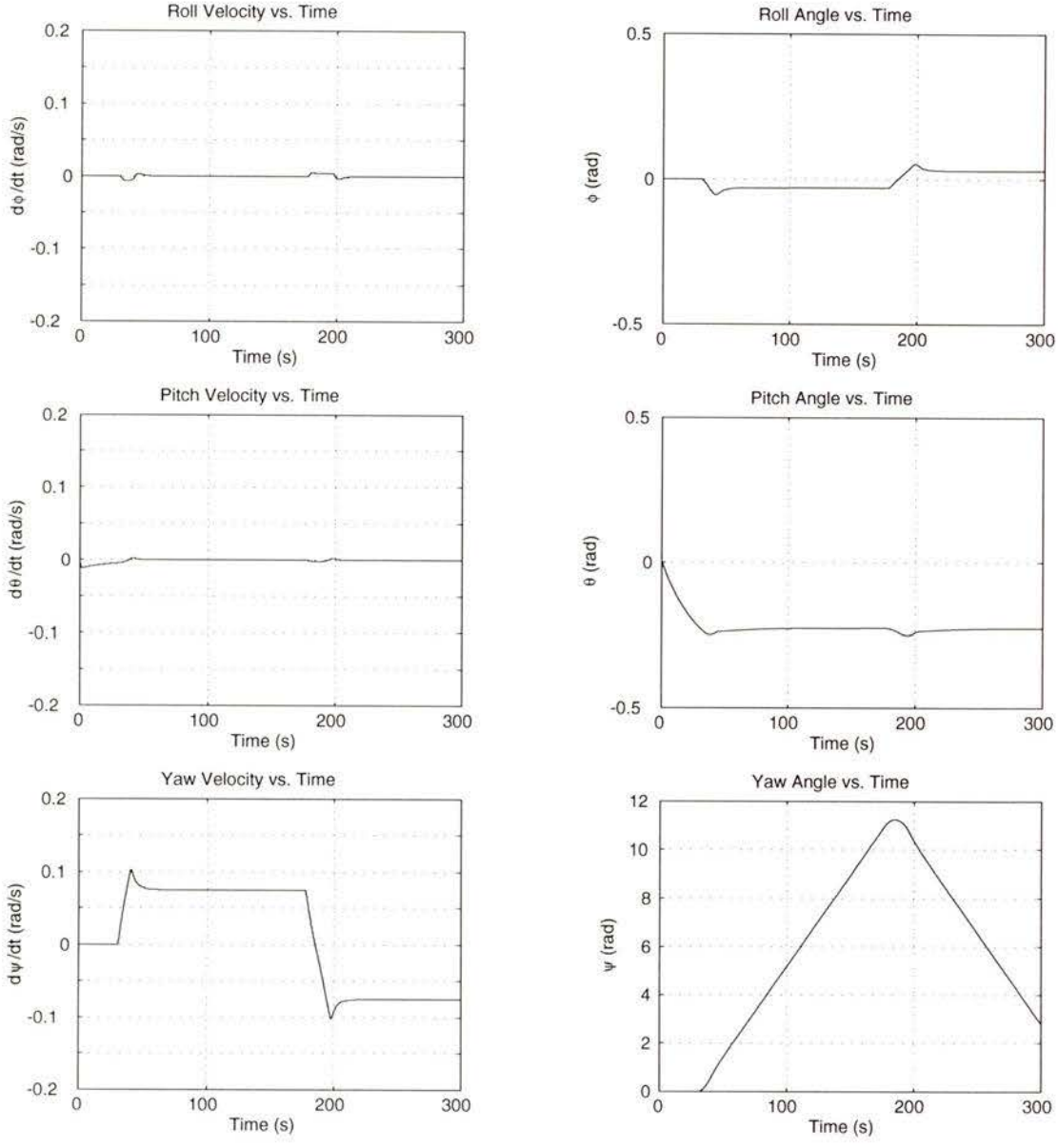


Figure 2.3: Variables: $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, ϕ, θ, ψ vs. time.

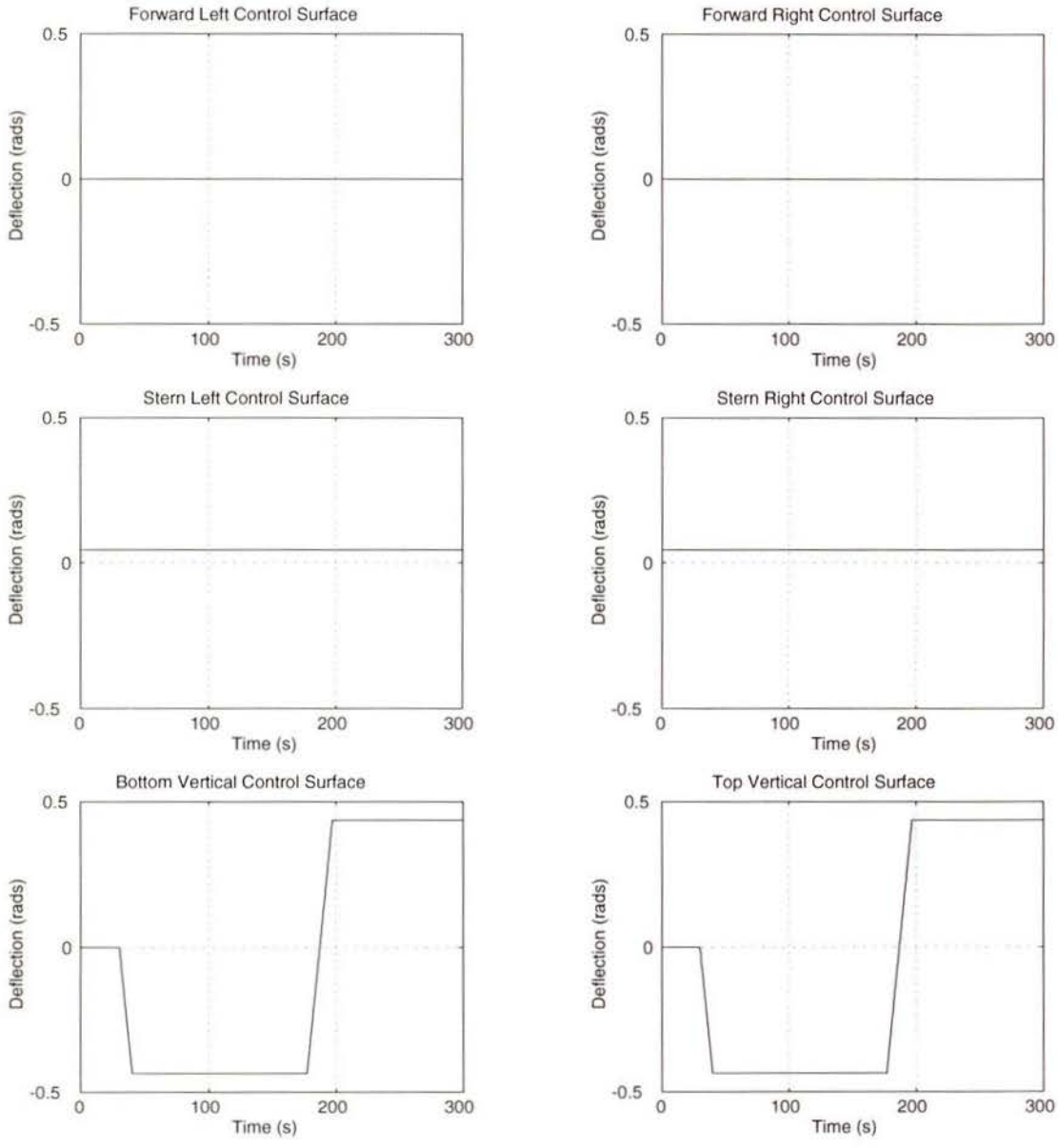


Figure 2.4: Control surface deflections vs. time.

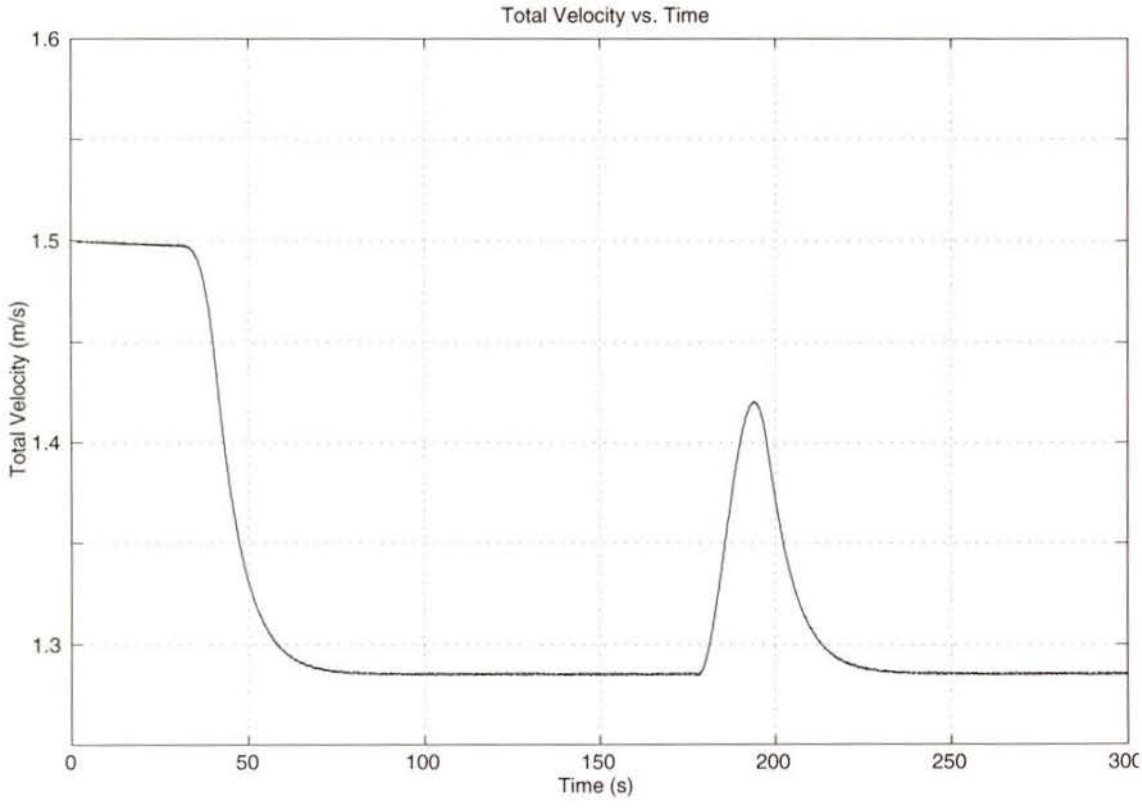


Figure 2.5: Total velocity vs. time.

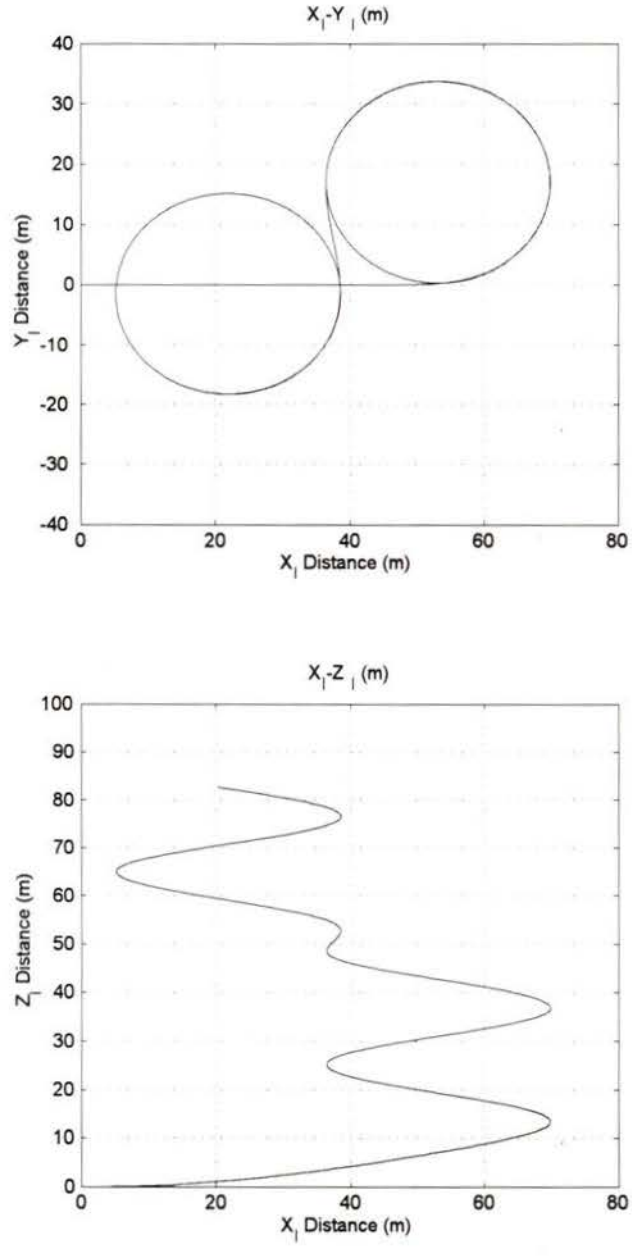


Figure 2.6: The inertial position in the X_I-Y_I and X_I-Z_I planes.

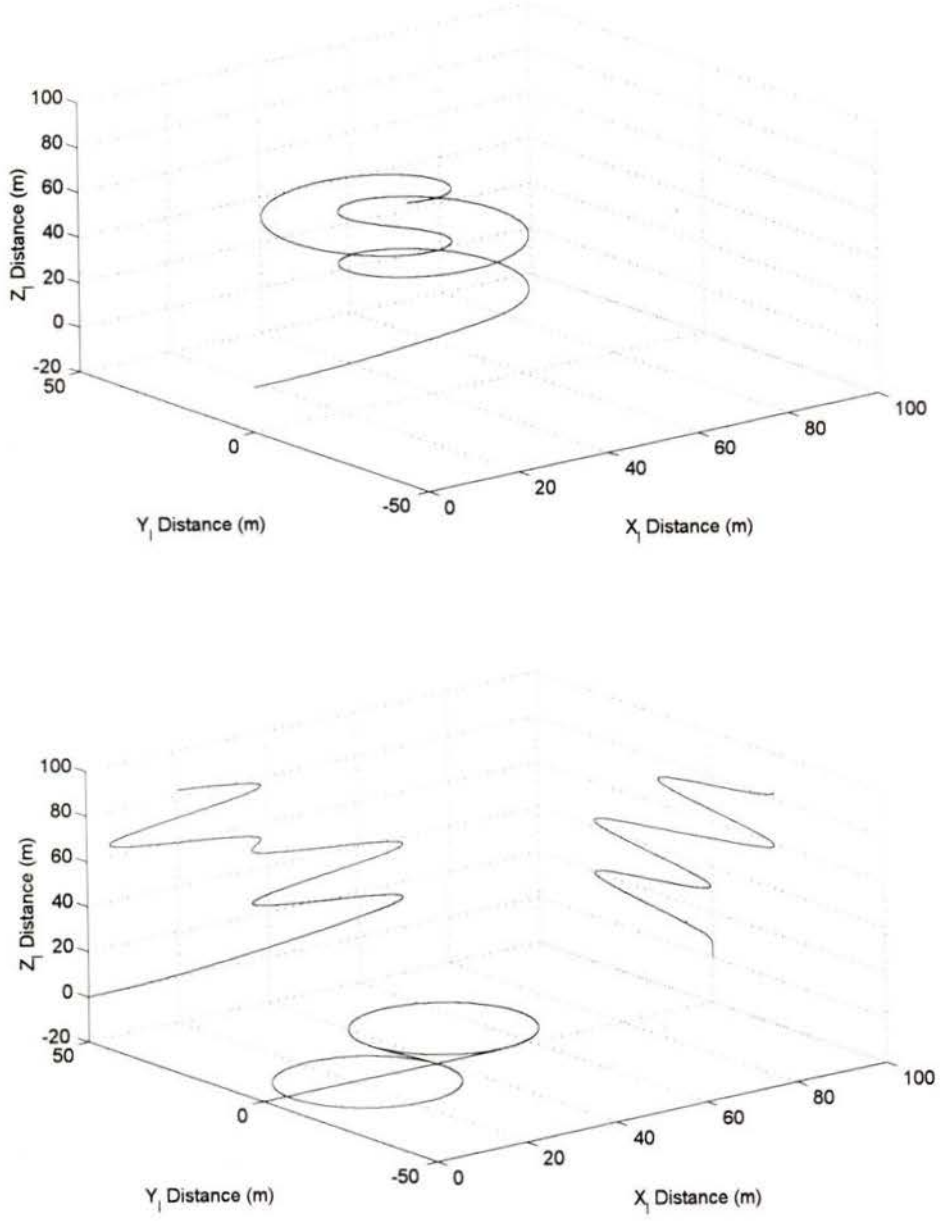


Figure 2.7: Spiral maneuver.

2.3 NPS Simulator

2.3.1 NPS Virtual World Research and Implementation

The objective of the NPS Virtual World research is to reproduce real-world robot behavior with complete fidelity in the laboratory [11]. The Integrated Simulator supports complete scientific visualization of actual NPS AUV vehicle performance [12] and reduces the need for access to the vehicle for testing in order to complete the range of research being carried out for various vehicle systems. Using the simulation, developmental AUV software can be evaluated in a timely and intuitive manner. End to end testing of all hardware and software components of the AUV can be completed with the 3D window connected to the vehicle itself, or identical components in the lab. Finally, the system can be used to perform post-mission playback of telemetry and sensor data for analysis in the laboratory. By enabling the visualization of all aspects of design, development, and testing, an NPS researcher has the ability to observe the behavior of the vehicle and evaluate performance without relying solely on interpreting 2D plots of complex AUV behavior.

The NPS software is freely available via anonymous file transfer protocol from [13]. The package is a 20 MB download of source code, executables, and documentation. Two Silicon Graphics (SGI) computers, one (preferably both) with the Open Inventor Execution Only Environment Version 2.1 installed, are required to run the virtual world. If the graphics source code is to be modified, the Open Inventor Development Library Version 2.1 must also be installed.

Successful operation of the virtual world simulation is a three step process. The first step is to set up the virtual world on the most graphics capable SGI. Command

line options are used to indicate initial camera view, to toggle texture rendering, and to specify the viewer IP address.

The virtual environment is shown in Figure 2.8. It includes a textured ocean floor and a surface piercing platform with a ROV circling it. The AUV model is

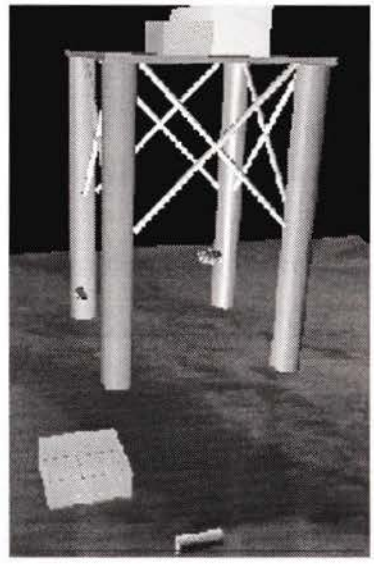


Figure 2.8: NPS Virtual World [13].

of the NPS AUV *Phoenix*, shown in Figure 2.9. This vehicle has four paired control surfaces and bidirectional twin propellers. It has a hull length of 2.2 m, and a design depth of 6.1 m. All systems are powered by a pair of lead-acid gel batteries providing an endurance of 90-120 minutes with speeds up to 2 knots ($\sim$1 m/s). On board systems include sonars, rotational gyros, Global Positioning System, and an Ethernet local-area network connection to onboard computers and external networks. A more detailed description of the vehicle can be found in [14].

The virtual *Phoenix* is represented by simple shapes. Animated features include

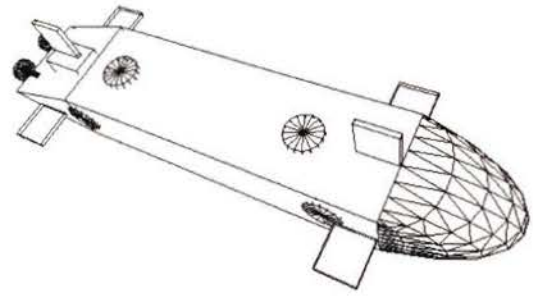


Figure 2.9: NPS AUV - Phoenix [13].

rotating control surfaces, and an indication of thrust using animated cones.

Once in operation, communication with the virtual world is through keyboard commands.

The next step is to set up the dynamics simulation. This is executed on a second SGI networked with the graphical viewer. Running *dynamics* with the full sonar model requires the Open Inventor Execution Only Environment. There is a no-sonar option which runs the dynamics simulation and does not require Open Inventor. Setup of the dynamics simulation is completed with a text based menu which allows the user to set the protocols for communication with the graphics computer, to set environment variables such as current, and to set a number of operating variables (sound effects). Several of these options can also be set with command line switches. The dynamics simulation is started, but waits for the third and final component to be run, before beginning a simulation.

The final program needed to operate the system is the *execution* program. This program can be run on the same workstation as the virtual world. *Execution* is the communication level between the vehicle, its software (tactical), and the virtual

world, or the simulation and the virtual world. The program operates identically on the *Phoenix* and on a workstation. Data recorded by this program can be plotted for further analysis of the simulation.

2.3.2 Evaluation of the Virtual World

An evaluation of the NPS Virtual World was undertaken to determine whether the virtual world would meet the goals outlined for the current project. If a desired feature existed in some form in the NPS Virtual World, the ease with which it could be adapted or extended was considered. In particular, since we had already developed a vehicle dynamics model [15], we intended to replace the model of the *Phoenix* with our own of the ISE ARCS vehicle. If a desired feature was non existent, then further consideration was given to how the feature could be added. The following summary is based on the trial period of the Virtual World available in July 1997 and the hardware available at the Space and Subsea Laboratory at the University of Victoria.

One of the first considerations is the ease of program set up and user interaction. All three processes which run the virtual world are initiated through command line keywords. A text based menu assists the user through some of the set up procedure. Generally it is necessary to have the keywords, file names, and set up procedures at one's finger tips in order to start a virtual world simulation correctly. It became apparent that this could become a lengthy process when setting up a mission, and daunting for a first time or infrequent user. A goal of the current project is to make the set up as quick and simple as possible using common GUI tools. Secondly, as outlined in the previous section, the NPS Virtual World runs on two networked SGIs. The requirement proved challenging to overcome as a second SGI which met the

hardware and software requirements had to be found outside the Space and Subsea Laboratory. Arrangements also had to be made to operate the Virtual World at times when its use wouldn't interfere with other work being done on the second workstation. Porting the NPS system to a single platform would have posed complex challenges.

Each component of the NPS system prints long streams of data related to its task in its shell window. The volume of data from the three concurrent processes, and the speed at which it is output, is too much for any operator to synthesize. It is also distracting, prompting this user to iconify the windows during a simulation. An instrument panel would enhance the designer's ability to understand the information currently output to the screen. The instrument panel envisioned for the current project will be flexible enough to display any information available from the simulation in a format which suits individual preferences and tasks. This is the largest single component missing in the virtual world and required in the current project.

The Virtual World was developed around the *Phoenix*. As such, the vehicle geometry is coded directly into the viewer program. To meet the current guideline of interchangable vehicle models, the permanent vehicle would have to be replaced by a more generic vehicle building program. This process would remove a large component of the existing virtual world code. Animated sonar modeling is included in the virtual world, and is not required in the Viewer. This component would be removed as well. Both of these changes to the Virtual World code would leave not much more than a shell of it, and the task of linking in-house code to the existing structure.

Finally, the NPS Virtual World offers a set of predefined camera view points, and one which can be positioned by the user. The viewpoint is changed with keyboard shortcuts. The viewpoints offered match the list of desired follow points for our project. The camera setup would have to be extended in order to include the ability

to customize the camera location for each following scheme. In keeping with the theme of simple interaction, a GUI based method of switching and customizing cameras is desirable.

The NPS Virtual World is a comprehensive and complex simulation platform. In order to use it for the current project, components such as the cameras would have to be extended. Other features such as a GUI and instrumentation would have to be created. The task of extending, adapting, and adding to the virtual world code would leave it unrecognizable. It would also restrict the development of the Viewer to an existing structure. Considering the number of features that would ultimately have to be created from scratch whether the NPS world was used or not, and remembering that what exists would have to be changed substantially to work on a stand alone PC, it was decided that an in-house program would ultimately be the best use of our time and available resources.

2.4 Design Tool

As discussed in Chapter 1, the purpose of the present work is to develop a tool intended for the vehicle designer, as opposed to a mission planning/testing tool such as the NPS system. Several broad criteria were established for the development of the 3D animated design tool. Given the advancement in graphics rendering capabilities on PCs, and their continued low cost, it is considered valuable to develop a facility that will be PC based, but also be compatible with other common operating platforms. An important feature is that the animation software be widely available and likely to become a standard. It is equally important that the software and hardware required by both the user and developer be available at a reasonable cost. To reduce

the operational challenges to the user, a familiar Graphical User Interface (GUI) is to be included. The GUI is the most platform dependent component of the application making it necessary to consider cross platform compatible components. The combination of these criteria will produce an application which is available on a range of platforms for a diverse set of users and resources.

2.5 Features of the Viewer

The driving program for the AUV animation is the dynamics simulation program described in [15]. The model is applicable to streamlined undersea vehicles. It determines the forces and moments acting on a vehicle by summing the effects of its constituent components, including the hull, foreplanes, and tailplanes. The simulation currently models the ARCS AUV, Figure 2.10, built by International Submarine Engineering, Port Coquitlam, B.C.

The ARCS vehicle was initially developed to complete under-ice hydrographic surveys for the Canadian Hydrographic Service. Later it was acquired by the Department of National Defense and used for testing prior to the development of the THESEUS vehicle. ARCS is 6.5 m long, 0.7 m in diameter, displaces 1840 kg, and has a top speed of 5 knots. It has a range of 235 km with a 10 kWh Nickel cadmium battery. Vehicle systems include a Motorola 68030 microprocessor for vehicle control, an acoustic telemetry link, an inertial navigation unit, and doppler sonar. A rear mounted propeller provides forward motion while three pairs of control surfaces are used to change the vehicle orientation. The forward control surfaces, used differentially, roll the vehicle while the horizontal stern control surfaces are used together to pitch the vehicle. The vertical surfaces (rudder) change the heading of the vehicle.

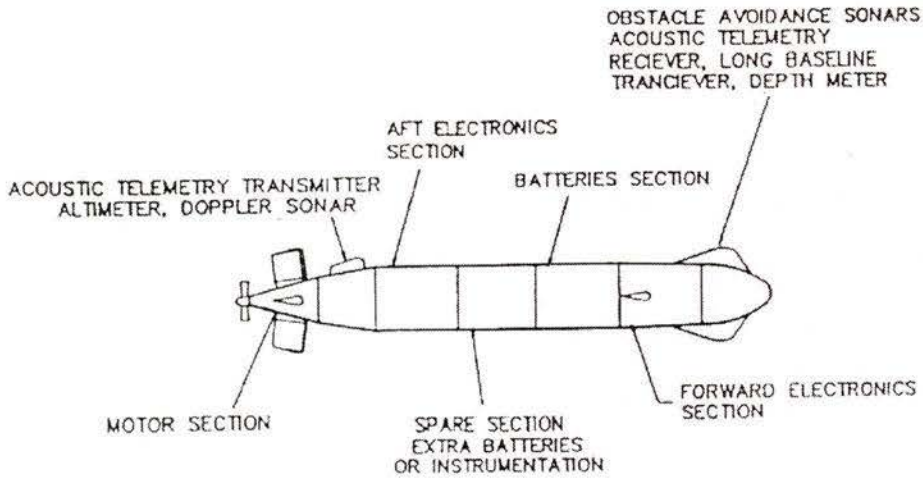


Figure 2.10: ARCS AUV.

The three-dimensional Viewer, Figure 2.11, provides a window into the virtual environment of the AUV, as well as a panel of instruments. The environment consists of a contoured ocean floor and a grid representation of the ocean surface. The animation can be observed from several viewpoints, including wingman and trailing views. A second rendering window, placed to the right of the virtual world Viewer, contains the instruments. This panel is capable of rendering several instruments at one time and provides the designer with the flexibility to choose what information is displayed. Several special purpose instruments exist to combine related information into an easily understood format. Common user interface tools give the designer control over the panel layout, the rendering features of the environment, and the camera viewpoint. Various simulation data files can be loaded through the menu system, as can modified vehicle models. A more detailed discussion of the various components

in the Viewer can be found in following chapters.

2.6 Other Applications

This thesis focuses on the development of an application to assist designers visualize the behavior of various AUV designs. The Viewer provides a 3D window into the underwater environment and a modifiable set of instruments to assist the understanding of the behavior being viewed. The combination of these resources makes the 3D viewing system a powerful tool which can be adapted to other scenarios that would benefit from a virtual view point.

2.6.1 AUV Field Trial

Experience with vehicles such as the THESEUS AUV from ISE has proven that high bandwidth real time communication with the vehicle is necessary during sea trials and when fine tuning the control system, [16]. During sea trials, communication at depths less than 10 m can be established through an RF coaxial cable between the vehicle and a surface float. A radio links the float to the support ship. At depths greater than 20m a fiber optic cable can provide communication from the vehicle to a float, and then from the float to the support ship, as shown in Figure 2.12. Although pulling a cable and a surface float is a hindrance and removes autonomy from the vehicle, it is essential during testing.

The arrangement described relays data from AUV mounted instruments to the support ship in real time. The real world data would be arranged in the same format as the dynamics simulation data file and used in its place to operate all components of the Viewer. Because communication is facilitated though a high speed data link

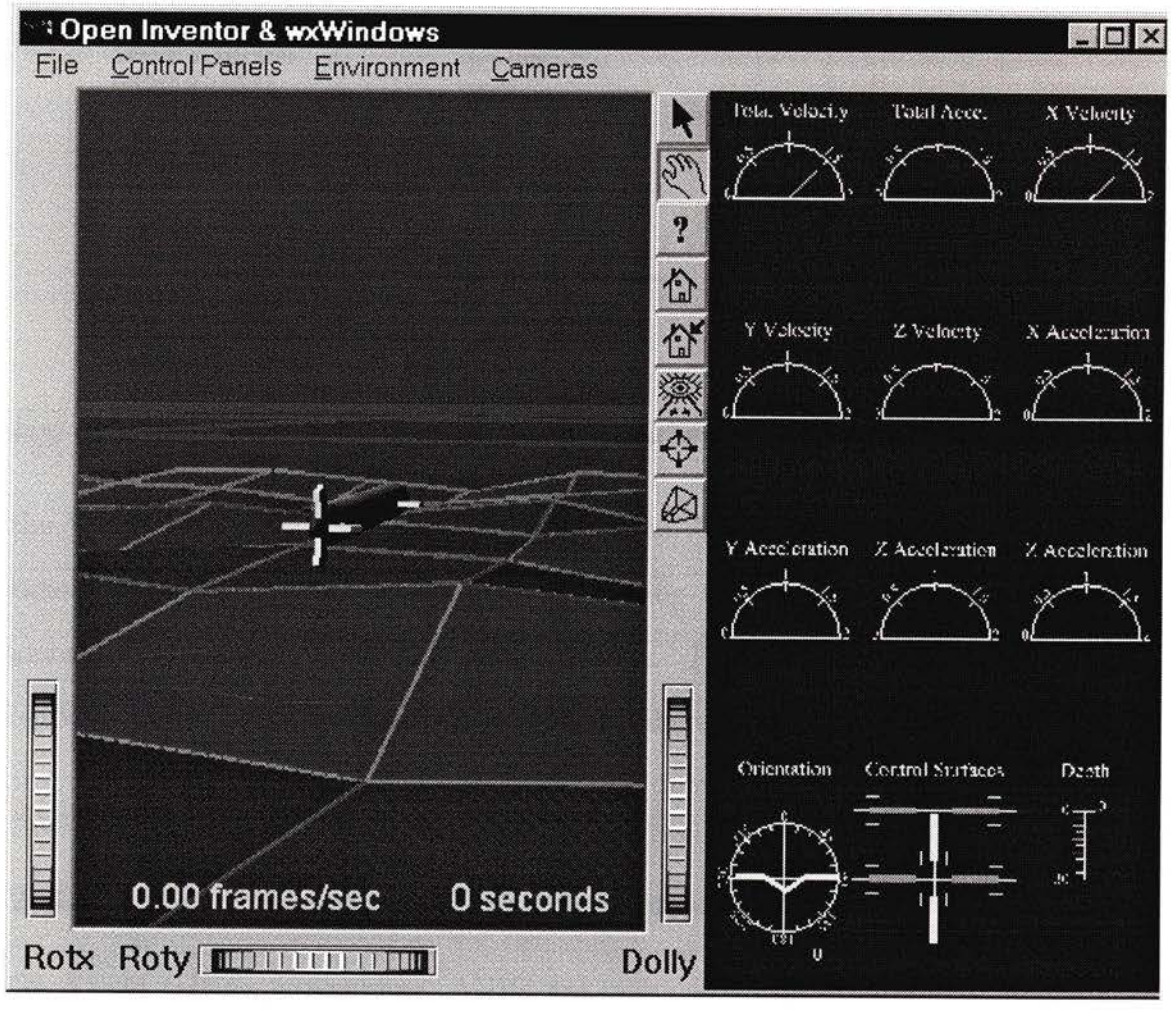


Figure 2.11: The 3D AUV Viewer.

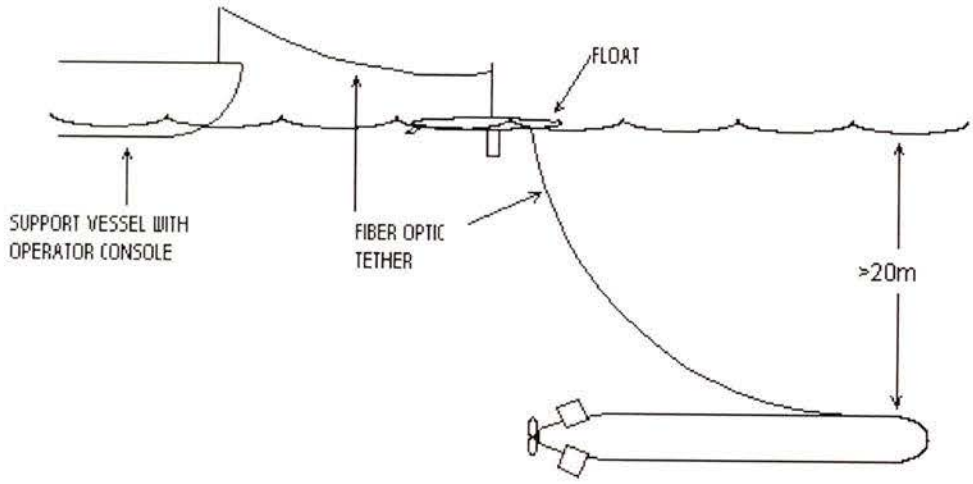


Figure 2.12: Fiber optic communication with THESEUS [16].

the Viewer would have satisfactory update rates. The Viewer would provide a more timely understanding of vehicle behavior during the mission. Problems could be identified visually, and possibly solved at sea. Fine tuning could be carried out without returning to shore for data analysis, thereby reducing delays during trials, and minimizing onshore analysis by narrowing down the search for trouble spots.

2.6.2 Flight Recorder Data Analysis

Once field testing is complete, AUVs are intended to be programmed and left to carry out missions with minimal surface support. What little interaction is available is carried out with low bandwidth acoustic modems at rates of 50 bits/s, [17] to 5,000 bits/s [18], depending on range and sophistication of the communication system. More details regarding acoustic communication can be found in [19]. In this case the Viewer could be used to assist with processing the telemetry data recorded by the

AUV once the vehicle is recovered.

The Viewer would use the flight recorder data in place of the dynamics simulation data, and would provide a visual account of how the AUV responded to its programming and its environment. The instrument panel would provide a more in-depth understanding of why the AUV behaved as it did. The analysis process would be less tedious with the Viewer, and areas requiring further study could be identified more quickly than with traditional analysis tools.

2.6.3 Mission Support Tool

Changing the vehicle model with ease is a feature of the Viewer. With this *plug and play* ability it is a simple process to adapt the field trial tool to show any vehicle being tested. A second vehicle of interest is the DOLPHIN built by ISE. This vehicle operates near the surface with a surface piercing mast. Communication between the vehicle and the surface vessel is accomplished through a digital radio link. A real-time link to the vehicle means telemetry data can be relayed to the Viewer at a rate that smoothly updates the animation. The Viewer would then provide that long sought window into the working environment of this undersea vehicle.

Several extensions to this use are also possible. Testing is underway to use the DOLPHIN to pull a tow fish. The arrangement eliminates the surface tow ship, isolating the tow fish from ship motions [20]. It also means the vehicles could complete missions faster, or be used with multiple DOLPHINs to complete surveys of larger areas.

The Viewer, with a few modifications, could be used to observe the behavior of the DOLPHIN, the towfish, and the cable between them. It would provide a real-

time, intuitive understanding of how the vehicles interact with each other. A further possibility is to acquire information about the bottom profile as it is measured by the sonar and using the data to construct a realistic representation of the actual working environment in the Viewer.

It is envisioned that multiple vehicles, be they DOLPHINs or other AUVs will eventually be used in groups, interacting with each other to gather data in a larger volume of ocean, and to complete tasks in shorter amounts of time. Extension of the Viewer to display multiple vehicles would improve the researchers' and oceanographers' understanding of where the vehicles are relative to each other, how they interact, and whether the desired task is progressing satisfactorily.

The Viewer provides a range of information to better understand underwater vehicles operating in the open ocean. The emphasis of the Viewer is on observation of overall vehicle behavior, not on testing individual systems, software or hardware. By understanding the overall behavior, we will gain confidence in the ability of vehicles to operate autonomously.

Chapter 3

The 3D AUV Viewer

This chapter discusses the requirements of a three - dimensional graphics program and outlines the Open Inventor toolkit used to create the Viewer. The chapter also outlines the development of the Viewer and presents the results of rendering timing tests.

3.1 Open Inventor

A 3D graphics application must meet some general guidelines for development of a Viewer. In order to gain insight into a real world situation, it is important that the scene be rendered rapidly and realistically. A graphics library that is an industry standard increases the life of the application. Portability to various platforms not only meets a range of researchers' needs but also increases the flexibility of uses.

The above criteria are met by Open Inventor [21], a C^{++} graphics toolkit based on OpenGL [22]. Both libraries were originally developed by Silicon Graphics for the UNIX environment. Template Graphics Software Inc. now develops and distributes

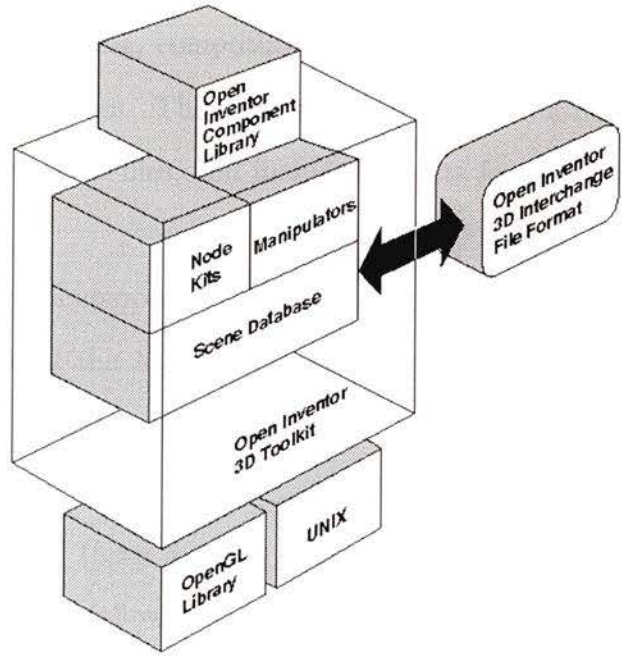


Figure 3.1: Components of the Open Inventor toolkit. [21].

Open Inventor for multiple platforms including PCs. The toolkit provides a set of building blocks which packages the individual functions of OpenGL into objects which can be used to create 3D scenes and interactive applications. The intent of Open Inventor is to focus on what is to be rendered, rather than how it will be rendered.

Figure 3.1 provides a picture of the UNIX version of Open Inventor. The three components of the library are the scene database, the manipulators, and the node kits. A separate Component Library handles user events, ensuring portability of the Open Inventor Library to any platform which supports OpenGL.

The scene database, shown in Figure 3.1, stores dynamic representations of 3D scenes as graphs of objects called nodes [23]. Each node performs a specific function

necessary to create a 3D scene. A set of actions provided by the database perform operations such as rendering, computing a bounding box, searching, or writing to a file, when applied to scenes. The database also defines methods for writing scenes to and reading scenes from files, and includes objects for animation, and monitoring purposes.

The two higher level components of the Inventor toolkit indicated in the diagram are the Node Kits and the Manipulators. Node kits provide a systematic way to produce groupings of nodes which create higher level objects. Manipulators facilitate interaction with the 3D scene.

The combination of these features provides a powerful tool for creating a 3D Viewer for an AUV. The following sections will describe the features of the Viewer.

3.2 The Main Window of the Viewer

The Viewer is split into two rendering windows contained within the main application window. The AUV and its virtual environment is rendered in an Open Inventor Examiner Viewer window that fills 60% of the application window. An Examiner Viewer has scene manipulation tools which appear in toolbars, referred to as decoration, around the outside edge of the scene. The tools and the mouse provide the user with the ability to change the orientation of the entire scene, and zoom in and out. The manipulation tools are functional while the animation is paused or stopped. Once animating, the camera has precedence over the viewpoint, and will override changes made with the tools. A popup menu provides an additional set of capabilities including rendering the scene as a wireframe or low resolution drawing. The decoration can be turned off with this menu.

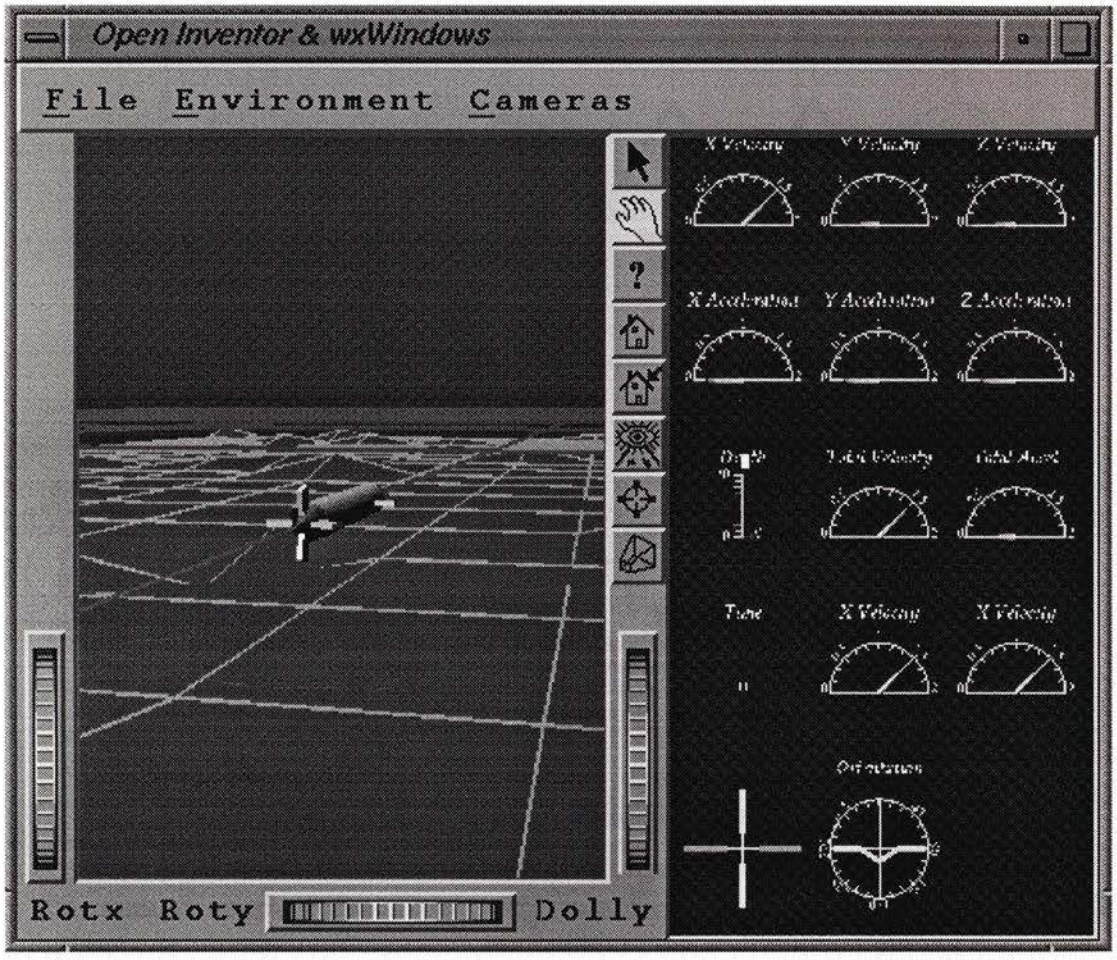


Figure 3.2: The Viewer with Open Inventor scene tools.

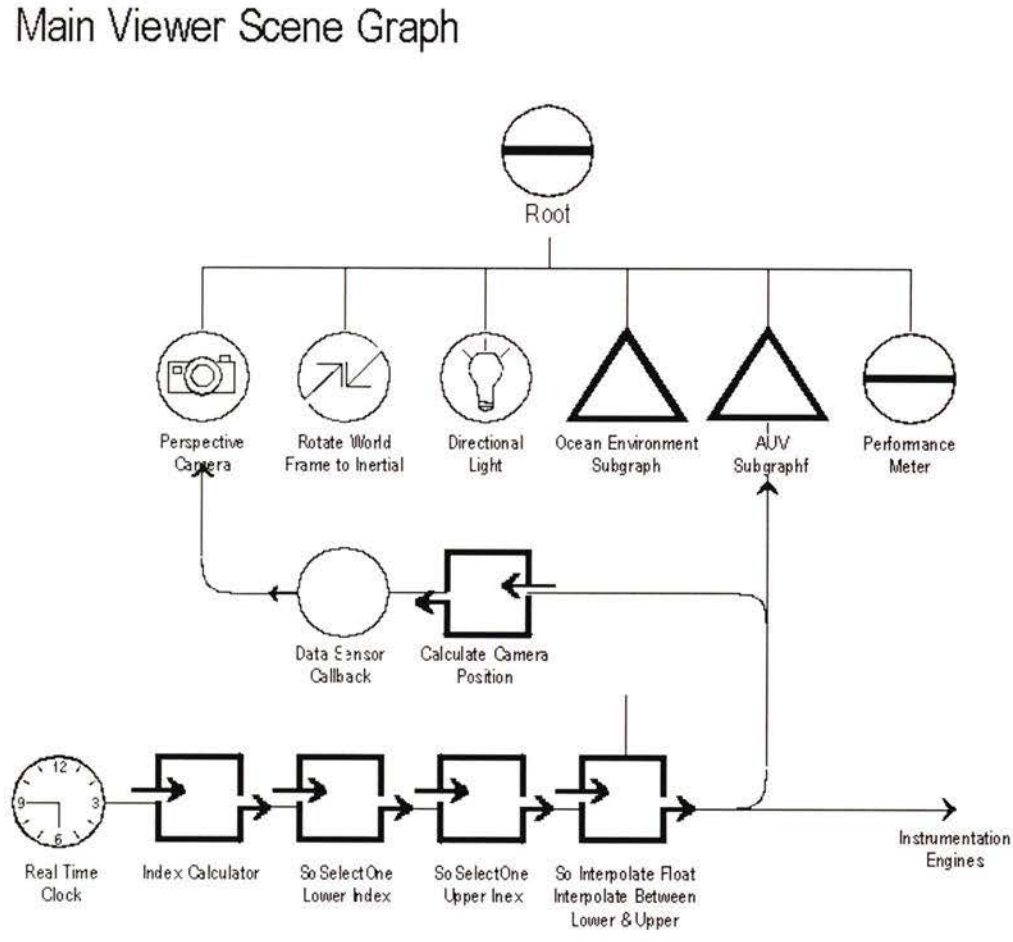


Figure 3.3: The main window scene graph.

The main scene includes a perspective camera, and a directional light for illumination. The inertial reference frame is as defined in Figure 2.1. The AUV and its environment are built in separate subgraphs and added to the scene in the main window. The main window includes a heads up display which contains the frame rate counter and the elapsed time of the animation. A system of Open Inventor engines and sensors animate the AUV, the camera, and the instrumentation in real time from a data file.

Hull_shape:	CYLINDERBODY (or BOXBODY)
Hull_length_x:	4.98
Hull_width_y:	0
Hull_height_z:	0
Hull_diameter:	0.70
Tail_length_x:	0.88
Tail_width_y:	0
Tail_height_z:	0
Nose_length:	0
control_surfaces:	6

Table 3.1: Example data file: ARCS dimensions.

3.2.1 AUV Subgraph

The geometric characteristics of the AUV are stored in an ASCII file which is simple to understand and alter. The AUV is represented by standard 3D shapes (e.g. cones and cylinders) and the dimensions in the geometry file reflect these approximations, as seen in Figure 3.4. A portion of the geometry file is given in Table 3.1. Currently the Viewer supports two basic hull shapes: cylindrical and rectangular. The position $(X_I,\ Y_I,\ Z_I)$ and orientation (φ, θ, ψ) of the AUV are specified in transformation nodes. With each update from the simulation data file the AUV is redrawn in its new position and orientation.

The data required in the geometry file to construct a control surface is given in Table 3.2. The default control surface is the top vertical surface. The dimensions of each surface are defined in the default orientation. The appropriate rotations are then

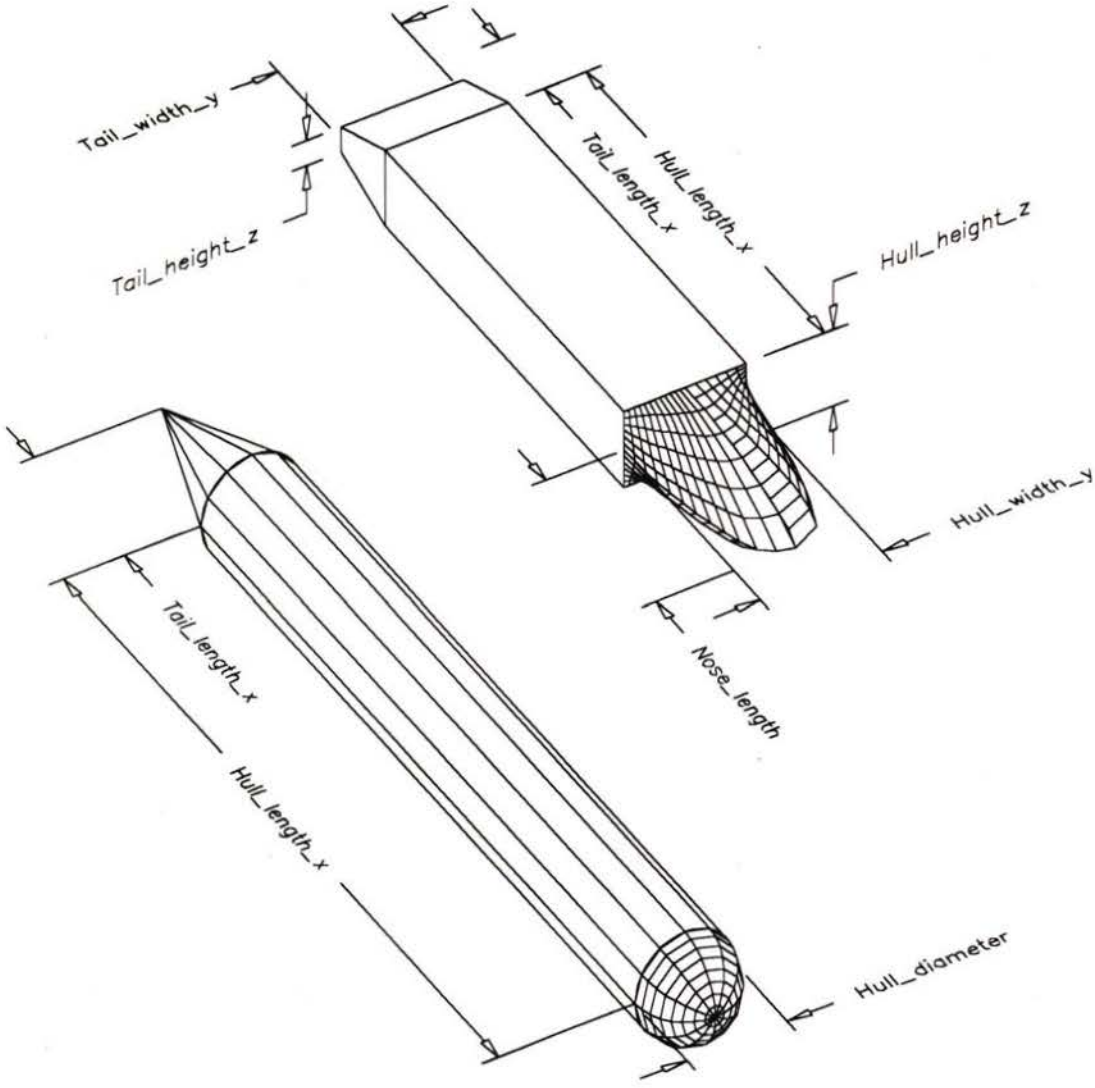


Figure 3.4: Dimensions required in the AUV geometry file.

Control_surface_#:	1 (front right horiz +y)
Control_surface_x:	4.88
Control_surface_y:	0.345
Control_surface_z:	0
Control_surface_length:	0.435
Control_surface_thickness:	0.1
Control_surface_width:	0.36
Control_surface_axis:	1.57
Control_surface_offsetAngle:	0
Control_surface_sweepAngle:	0.3

Table 3.2: Example data file: an ARCS control surface.

applied to orient the surface. The Control_surface_axis field is used to define a vertical surface (0°) or a horizontal surface (90°). Other control surface configurations can be specified by indicating a total angle from the default, or by indicating a standard axis and an offset angle. Sweep angles, defined in Figure 3.5, are measured from the vertical axis. All angles are declared as positive rotations in radians. Finally, the control surface is positioned with the values found in Control_surface_x, Control_surface_y, and Control_surface_z. These variables specify the location of the center of the control surface, including fairings. The position is measured from a reference point on the centerline of the AUV at the stern. The control surface number is the index to the surface deflection data in the simulation file. The surface deflections are updated from the data file. The new deflections modify each control surface through each surface deflection node, Figure 3.6.

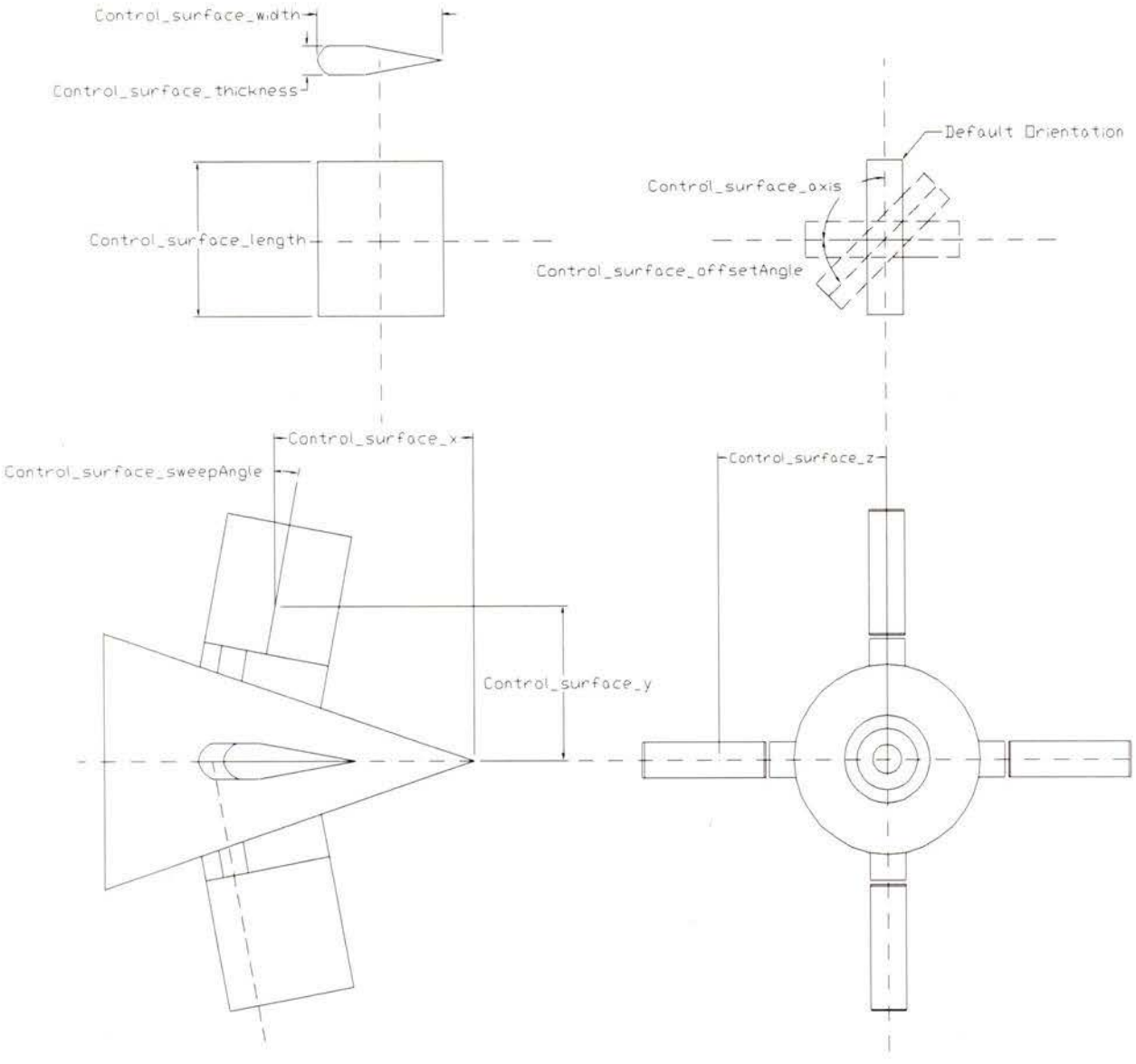


Figure 3.5: Diagram of data required to construct control surfaces.

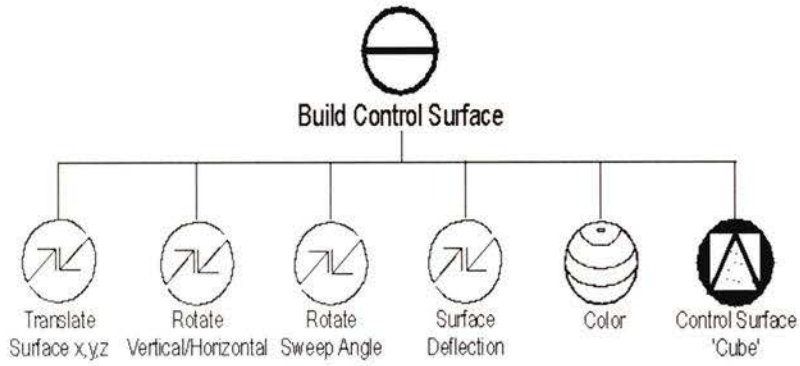


Figure 3.6: Scene graph for one control surface.

3.2.2 Environment Subgraph

The virtual environment for the AUV is divided into a 15 × 15 grid. The gently contoured ocean floor can be represented by a shaded surface, a grid, or set of dots. A grid can be rendered slightly above the shaded floor to assist with interpreting environment depth and dimension. The ocean surface is represented by a blue grid with the same dimensions and grid spacing as the floor below it, as can be seen in Figure 3.2. The depth of the virtual environment is initially set at 30 m, but can be changed through the user interface.

3.2.3 Update Engine

A set of engine nodes connect the AUV and the instrumentation to the simulation data file. The network of engines begins with an elapsed time engine which acts like a stopwatch. By default the engine is connected to the computer clock. Buttons on the GUI allow the user to start, pause, stop, and reset the engine. The output of the engine is the time elapsed since it was started or reset.

The remainder of the engine network selects two lines of data from the file which correspond most closely to the current time indicated by the output of the elapsed time engine. The data in the chosen time steps is linearly interpolated using the current time as the weighting. The interpolation engines update the various rotation and translation fields throughout the scene graphs resulting in animation.

The advantage of this node network is that it forces the animation to run in real time. Once the scene is updated, the engine network uses the current value of the elapsed time engine for the next loop. In this manner it intelligently chooses the appropriate line of data from the simulation file, skipping over data which is outdated. The disadvantage to the network is that the animation can only be paused or reset, like a stop watch.

3.2.4 Cameras

The Viewer has several predefined viewpoints which can be selected from a pulldown menu. Options are also available to create a customized viewpoint. Two types of cameras provide different schemes for following the vehicle. The coordinates of the first type of camera are specified in a reference frame whose origin remains at the vehicle's mass center (i.e., it translates with the vehicle), but which always remains aligned with the inertial frame. The camera is oriented such that it always looks at the center of mass of the vehicle. With this setup, the camera effectively tracks the vehicle but remains unaffected (in the inertial frame) by its changes of orientation. The distance between the camera and the vehicle's mass center will remain fixed, but the camera will appear to change its position relative to the vehicle as the vehicle changes orientation. The second type of camera is a body fixed camera. Its position

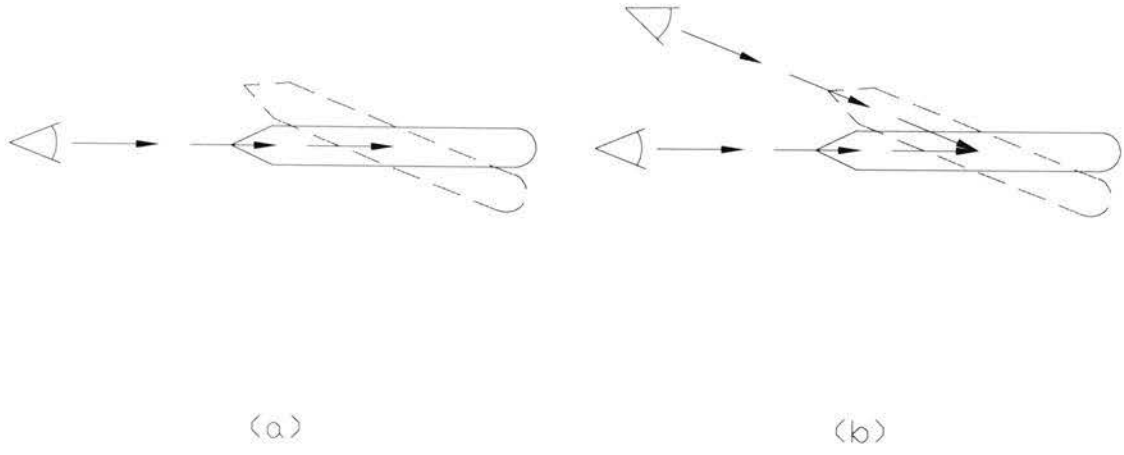
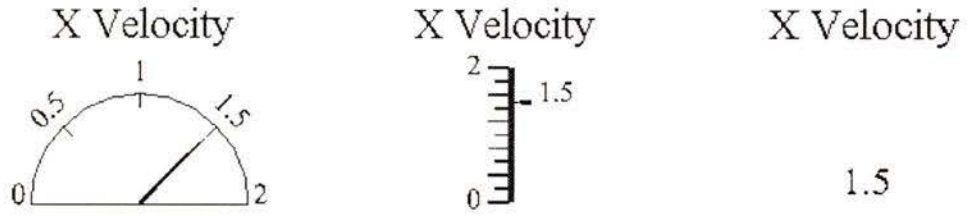


Figure 3.7: Camera following points.

remains at a point in the vehicle's x-y (horizontal) plane of symmetry, which is fixed relative to the vehicle's mass center. Again, the camera is always directed toward the center of mass of the vehicle. Because the camera's position is specified in the body frame, that position will change (in the inertial frame) with changes in orientation of the vehicle, due to movements of the plane of symmetry. This scheme tracks the vehicle and will follow changes in the vehicle's orientation. A diagram of the schemes for following the vehicle is shown in Figure 3.7.

3.3 The Viewer Instrument Panel

The instrumentation panel is displayed in the remaining 40% of the application window. It is rendered in a Walker Viewer to restrict the manipulation of the whole



(a) Dial Display (b) Gauge Display (c) Digital Display

Figure 3.8: The three instrument display styles available.

panel to translations in X, Y, and Z. The constraints further the 2D appearance of the panel, and prevent accidental changes in orientation of the 2D instruments. The Viewer decoration is turned off but can be turned on through the popup menu if required.

The Viewer instrumentation is provided to help understand the behavior observed in the viewing window. The panel is organized to provide complete flexibility to the designer to choose what information is displayed, where it is displayed, and in what format it is displayed. This flexibility promotes customization for both designer preference, and current usage.

The information presented on the panel is only limited by the output from the simulation. A sample of the information available for display in the general purpose instruments includes inertial velocity and acceleration, body frame velocity and acceleration, total inertial velocity, and total inertial acceleration. The information can be viewed in the three formats shown in Figures 3.8a, 3.8b, and 3.8c. The styles are interchangeable.

The general purpose instruments occupy twelve spaces, three across and four

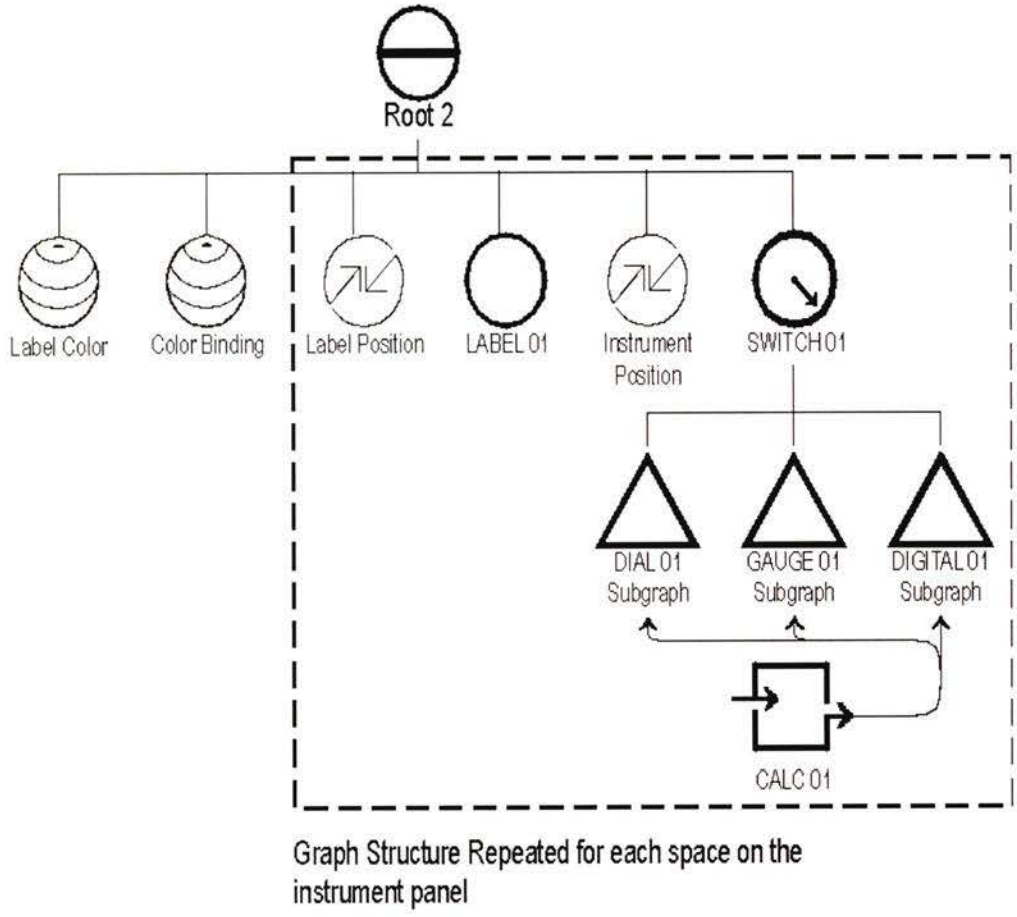


Figure 3.9: A fragment of the instrument panel scene graph.

down, on the instrument panel. A switching node in each space, shown in Figure 3.9, maintains subgraphs for a dial, a gauge, and a text read out. The user interface provides the mechanism to switch between the instrument styles in each position, or turn them off. It also enables the designer to choose the information to be represented in each position. The set up allows flexibility in the type of information shown, amount of information shown (number of active slots), and how it is represented (instrument style).

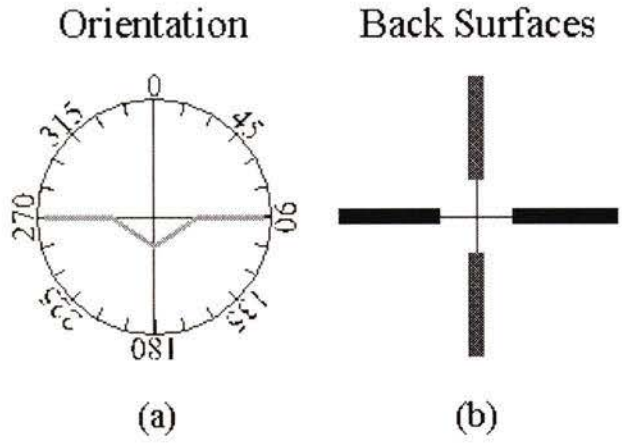


Figure 3.10: Special purpose instruments.

A timer indicates the duration of the animation. It is intended to provide an indication of how far into a mission the vehicle is and therefore prepare the designer for upcoming maneuvers. It can also be used to understand how long a particular maneuver takes to complete, or how long a behavior lasts before stabilizing or amplifying. A depth gauge indicates the total depth of the virtual ocean environment and provides a visual account of the location of the AUV in the water column.

A set of special purpose instruments provides a concise representation of all the information relative to understanding the current status of the AUV. These instruments are always present on the panel.

The orientation indicator in Figure 3.10a reduces the amount of information to be synthesized by combining the roll, pitch, and yaw motion into one instrument. Modeled on the artificial horizon indicator of an aircraft, the center object rotates as the vehicle rolls, and indicates pitch motion by translating vertically. The outside ring rotates to indicate changes in heading, or yaw. A stationary cross behind the

instrument marks the zero point for the indicators.

The Back Surfaces indicator in Figure 3.10b represents the control surfaces diagrammatically with a cross marking the neutral positions of each. As a control surface is deflected, the corresponding indicator surface moves. This instrument would be particularly useful in detecting mechanical failures during field trials.

3.4 Performance Tests and Results

The Viewer was developed on an SGI equipped with the R4600 SC 134 MHz CPU, and the 'XZ' 24 bit HW accelerated Z buffer. On this platform the Viewer, running from a data file, reached frame rates of 1.6 to 1.8 fps. The animation is choppy and disjointed at this frame rate, and is only moderately acceptable for use as a design tool. Timing tests were performed to determine effect of individual components on the frame rate. The tests and results discussed here are for the simulation running on the SGI. Some of the tests were repeated for the PC version of the Viewer.

Each of the tests were conducted using a 60 second segment of the spiral dive maneuver shown in Figure 2.7. All of the Viewer features are used during this maneuver. An Open Inventor data sensor was used to record frame rate information when the AUV was set in motion.

To determine the effect of subgraphs on the frame rate the Viewer was run for various test cases where a major subgraph or set of sub graphs were commented out. The trial runs are listed in Table 3.3 and the results are plotted in Figure 3.11.

Although the highest frame rate achieved was approximately 14 fps it was for an unrealistic case in which the vehicle has static control surfaces, no surrounding environment, and no corresponding instrumentation. The more practical scenario is

Case	Description:	fps
1	Complete Viewer: AUV scene + 14 instruments	1.8
2	AUV not rendered	1.8
3	Control Surfaces not animated	1.9
4	Control Surfaces not rendered	1.9
5	AUV + Control Surfaces not rendered	1.9
6	Environment not rendered	1.9
7	Whole AUV scene not rendered	2.0
8	12 Instruments Active (No Special Purpose Instruments)	2.6
9	9 Instruments Active (No Special Purpose Instruments)	3.1
10	6 Instruments Active (No Special Purpose Instruments)	4.1
11	3 Instruments Active (No Special Purpose Instruments)	5.3
12	1 Instrument Active (No Special Purpose Instruments)	6.6
13	No Instruments rendered in panel	9.6
14	No Instruments rendered and Control Surfaces not animated	12.0
15	No Instruments or environment rendered	11.3
16	Instruments / Environment not rendered; Control Surfaces not animated	14.4
17	No AUV, Control Surfaces, or Instruments rendered	14.3

Table 3.3: Component effect on frame rate.

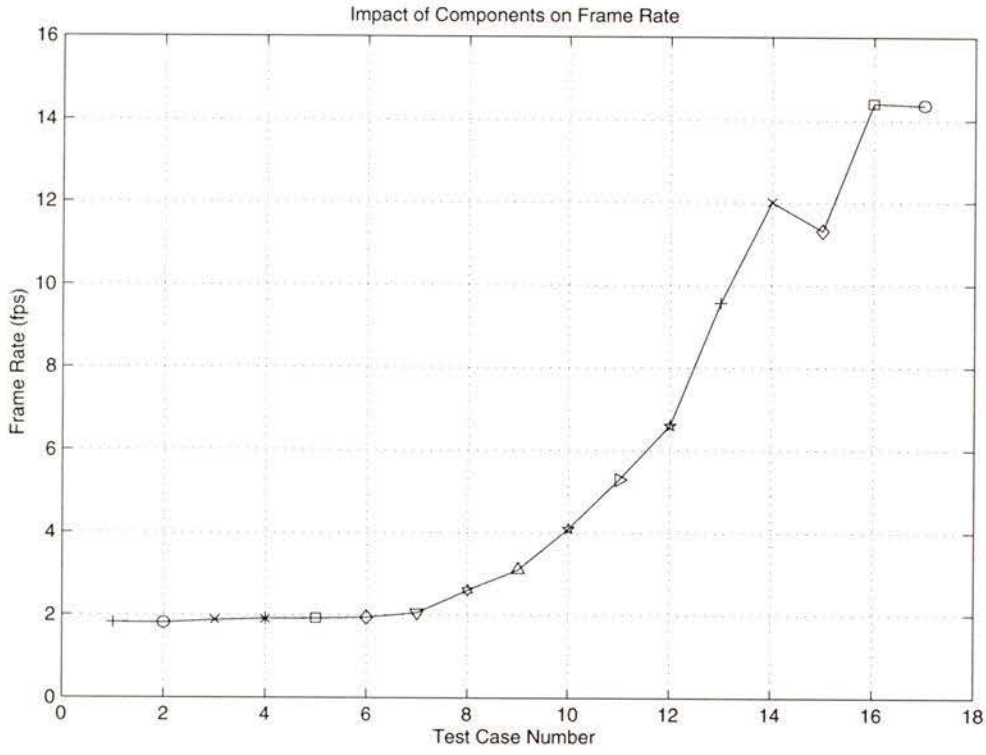


Figure 3.11: Effect of components on frame rate.

the test run without an instrument panel which achieved a frame rate of almost 10 fps. As can be seen in Figure 3.11, removing single components from the main scene causes insignificant changes to the overall frame rate. The instrument panel has the largest impact on the frame rate.

To gain an understanding of the effect of the number of active instruments in the panel and the impact of the individual instrument styles the following series of cases were tested:

- progressively commenting out each position and its corresponding instruments
- progressively commenting out the dial instruments

- progressively commenting out the gauge instruments
- progressively commenting out the digital instruments

In the final three cases the two instrument styles not being tested were commented out.

As shown in Figure 3.12, the frame rate drops from a high frame rate when no instrumentation is displayed to a rate of approximately 6 fps when one instrument is displayed. In the complete panel test the three instrument styles are available, but only dials are rendered. The final four instruments in this case are the depth gauge, the timer, the orientation instrument and the back surfaces indicator. The complete animation runs at approximately 1.8 fps.

The digital readout instrumentation is the simplest style of all three instruments and results in the highest set of frame rates when rendered alone. The dials have one moving part and consequently have a slightly lower frame rate than the digital instrumentation. The gauges have moving text and an animated indicator making them the most complicated instrument style. This is reflected in the lowest set of frame rates.

In all cases the frame rate decreased by approximately one-half when one instrument was rendered in the panel as opposed to none, regardless of instrument style. In another series of tests one dial was rendered in the main window alone, in the instrument panel alone, and then in both. The Viewer was initially tested using its default layout; a perspective camera in an Examiner Viewer that is 60% of the overall window size, and an orthographic camera in a Walker Viewer. With this arrangement the following frame rates were recorded: 18.4 fps for a dial in the main window, 21.2 fps for a dial in the instrument panel, and 10.7 fps when a dial was rendered in each.

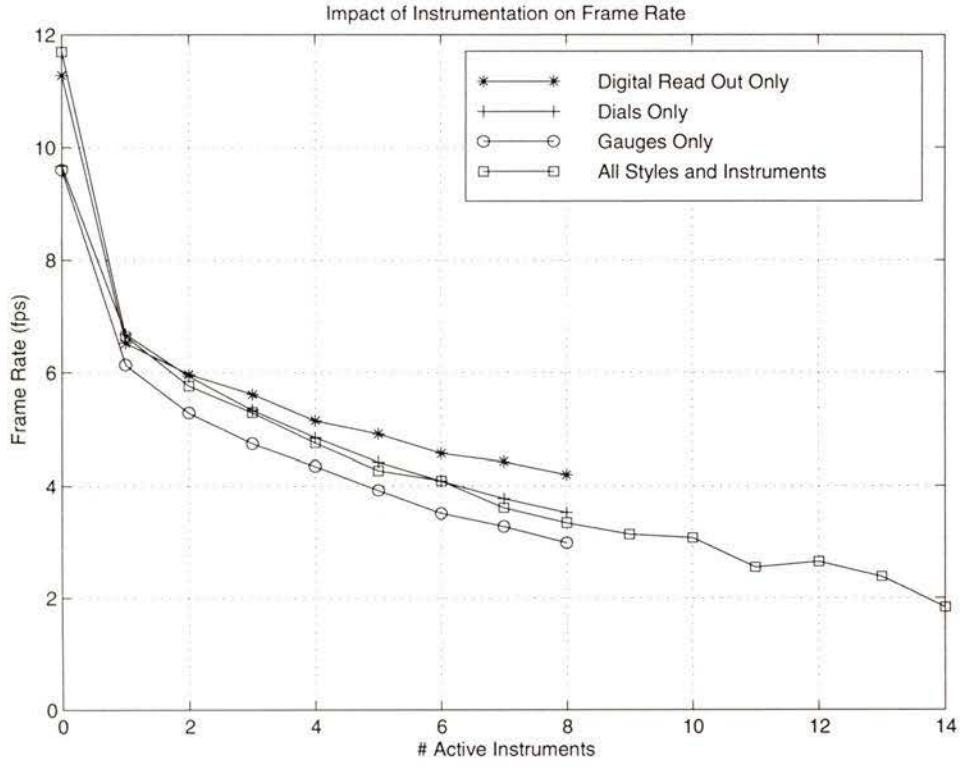


Figure 3.12: Impact of instrumentation on frame rate.

Further tests included using the same cameras in each window, using the same viewer styles for each window, and changing the window sizes to a 50/50 split. These tests determined the viewer style and camera type have no effect on the frame rate and that smaller windows increase the frame rate. When identically sized windows were used the following frame rates were recorded: 20 fps for a dial in the main window, 20 fps for a dial in the instrument panel, and 11.3 fps when a dial was rendered in each. Slight variations were seen in the frame rates for windows with viewer decoration on as opposed to off.

3.5 Discussions with an AUV Designer

The Viewer was developed in consultation with ISE, and once complete it was evaluated by an AUV designer. The following is a summary of the comments and suggestions made during the evaluation, and how they were incorporated into the Viewer.

Overall the Viewer was considered a good tool because it is simple, intuitive, and easy to use. The flexibility afforded in the instrument panel is advantageous and worth keeping. The vehicle geometry file is straight forward and easily modified. The camera options provide the customization necessary for specialized observations, and default viewpoints that would be beneficial for various tasks.

Several suggestions were made regarding the instrument panel, and as a result the number of general purpose instruments was reduced from twelve to nine. A further reduction to six general purpose instruments may be considered. The timer was moved from the instrument panel to the heads up display in the main window.

The special purpose instruments were rearranged to include a text read out of the heading in the orientation instrument, as seen in Figure 3.13a. A set of forward control

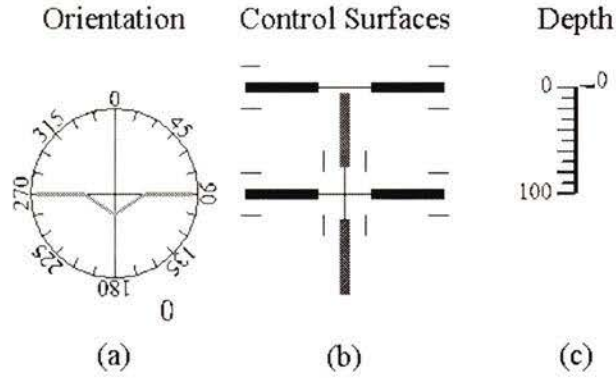


Figure 3.13: Final layout of the special purpose instruments.

surfaces was added to the surface deflection instrument, Figure 3.13b. Markings indicating the maximum deflection for each surface were also added. To complete the set of information provided by the special purpose instruments, the depth gauge was moved to a permanent location next to the control deflection indicator, Figure 3.13c.

Additional suggestions to be incorporated in the future include creating more realistic control surfaces (airfoils) to improve the visual accuracy of the vehicle model. Also, including a path tracking instrument to display the actual vehicle path and its desired path would be beneficial when evaluating controllers. It is recommended that a future version include a way to interactively change setpoints to see how the vehicle reacts.

The final arrangement of the Viewer, shown in Figure 3.14, is based on the above suggestions. The completed Viewer runs at approximately 1.8 fps on the SGI.

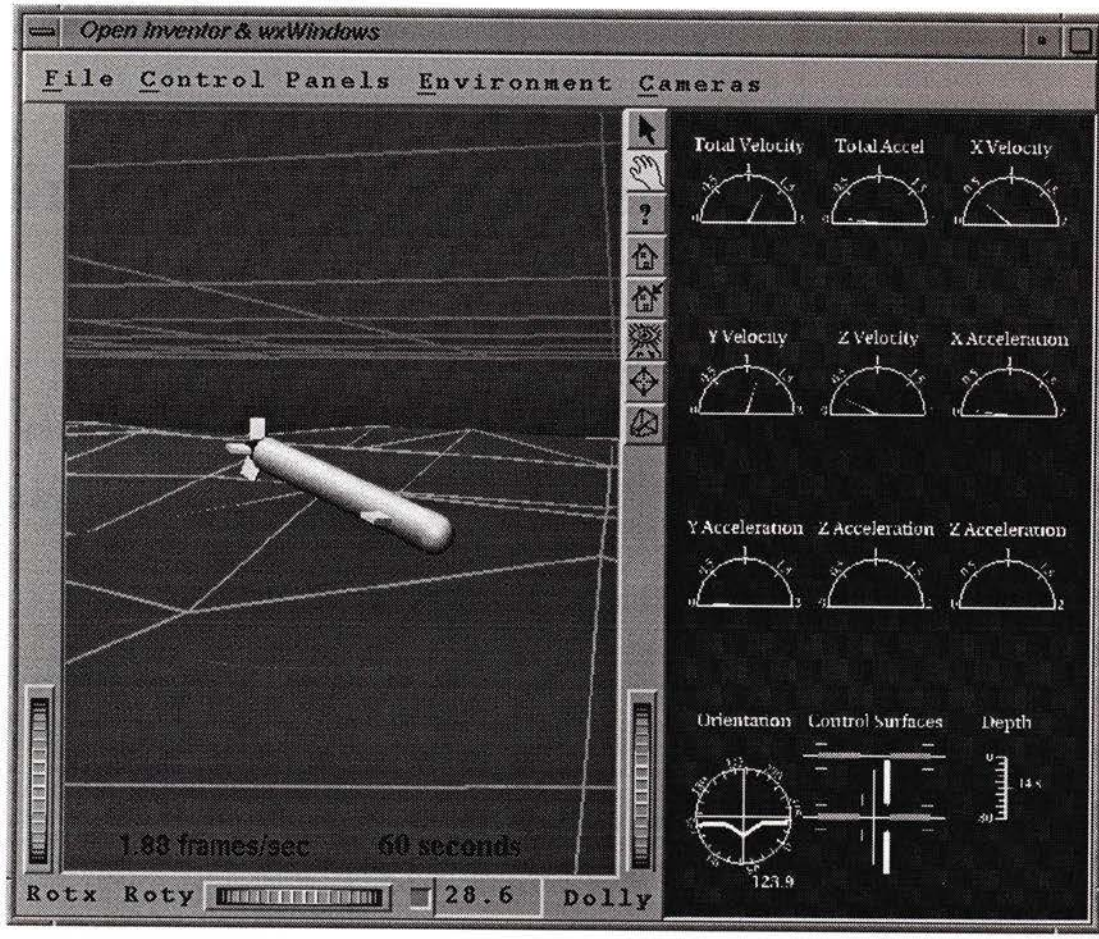


Figure 3.14: The final configuration of the Viewer - SGI version.

Chapter 4

Graphical User Interface

The intent of the user interface is to facilitate interaction between the application and the user. In choosing a toolkit to develop the GUI for the Viewer the following criteria were used:

- low cost to both the developer and the end user
- cross platform compatibility
- ease of implementation
- compatibility with Open Inventor or support for compatibility

This chapter presents background information on typical user interface libraries for Windows and UNIX applications, and the challenges posed by these libraries for the current project. Further sections outline the cross platform compatible library chosen to create the GUI for the Viewer. Finally, the components of the GUI developed for the Viewer are discussed.

4.1 Typical GUI Development Tools

Both Windows and UNIX applications use visual tools to facilitate interaction between the user and the application. These tools are standard in their function regardless of the platform; however their appearance varies slightly between platforms, and the libraries used to create them are worlds apart.

The high level class libraries used to create GUIs are based on low level C libraries. In Windows this library is called the Windows Application Program Interface (API). It is a set of functions which provides the basis for creating GUIs and applications by handling tasks such as user interface controls, windows management and display operations. The Windows API is an industry standard for creating applications with a look and feel familiar to Windows users.

Equivalent libraries exist on the UNIX system, but are slightly more complex. The Motif Widget set is a widely accepted toolkit used to create UNIX applications. It is based on two lower level libraries: *XLib* and *X Toolkit Intrinsics* (Xt). A Motif application is built by making appropriate calls to the Motif library, Xt library, Xlib and the normal UNIX system libraries. A Motif Style Guide details the expected behavior and appearance of a Motif compliant application.

These low level libraries were not investigated further as high level, object oriented libraries called application frameworks exist for both platforms. Application frameworks considerably reduce the complication and effort required to create an interactive application by packaging common functions into C^{++} classes. Application frameworks can be used to implement fully functional, generic applications with only a few lines of code, and common tools are easily added. As a result, the development effort can focus on application specific features.

The Microsoft Foundation Classes (MFC) library is an example of a C^{++} application framework based on the Windows API. ViewKit is a C^{++} application framework based on the Motif Toolkit. Both integrate features typically expected of applications including standard user interface tools, print and print preview, inter-application communication, and on-line help. Both frameworks handle all low level details required to create a native application and package commonly required components into reusable classes.

Application frameworks have been incorporated into integrated development environments (IDE). These applications reduce the work involved in creating an application to a click of a mouse button. The IDE manages makefiles, application organization, and other house-keeping tasks. Again, application development is focussed on specific functionality, and not the supporting structure. RapidApp is an example of a UNIX based IDE. It uses the ViewKit framework and integrates Open Inventor applications. Microsoft Visual C^{++} (MSVC^{++}) is a windows IDE. It is based on the MFC and with additional classes MSVC^{++} incorporates Open Inventor graphics.

4.2 Cross Compatibility Issues

Development with the tools discussed previously would create a platform dependent application. Further options were explored to create an application that would satisfy our cross platform compatibility goal.

Several proprietary development libraries are available to create platform independent applications; however, the disadvantage of this software is the risk of discontinued support [24].

One well promoted option is Wind/U from Bristol Technologies. With this soft-

ware an application can be developed with MSVC^{++}, and then be re-compiled on a UNIX system and linked with the Wind/U libraries. The Windows API calls are mapped to the equivalent calls in X and Motif [24]. Wind/U supports Open Inventor.

A third route is to develop the application using RapidApp, and use NuTCRACKER to create a Win32-based UNIX environment. The source code is compiled and linked against the NuTCRACKER dynamic link libraries, and then run on a PC [25]. There is no reference to Open Inventor support with this software.

A major concern in all three cases is the price of the software, end user licences, and product support. The software also complicates the development process, and final distribution.

A freeware library called wxWindows was suggested as a possible alternative for creating a GUI for an Open Inventor application. Research into this library found that it met all four of the criteria established for the Viewer's user interface. As a result, wxWindows was used to develop the cross-platform compatible application.

4.3 wxWindows

wxWindows is a freeware C^{++} class library that provides the tools necessary to develop an application with a user interface which will run on more than one platform. wxWindows was originally developed at the Artificial Intelligence Applications Institute (UK) for internal use [26]. Recognized as a potentially useful tool it was released as public domain software. It is currently maintained by Julian Smart, its creator, and interested volunteers world wide. As a result, wxWindows has undergone several upgrades, and ports to several more platforms than the original version.

4.3.1 Availability and Support

wxWindows is freely available through anonymous file transfer protocol from the wxWindows web site, and several mirror sites. The package includes source code, extensive sample programs, and documentation in various formats. The wxWindows library must be compiled before it can be used to compile the sample programs or applications. Makefiles exist for all supported operating systems, and several standard compilers. Compiling the library is not without challenges; however, help can be accessed quickly and easily through the wxWindows mailing list (wxwin-users@wx.dent.med.uni-muenchen.de).

4.3.2 Portability

wxWindows calls the relevant Motif, XView, or Windows API rather than emulating their behavior, [27]. By using the native system APIs, the application doesn't require a runtime library, and has the appropriate appearance for each platform. Once compiled, the application will operate on any similarly configured system. wxWindows supports Linux, Sun, SGI, Windows 3.x, 95, NT, OS/2 and Macintosh. It has been tested with widely available C^{++} compilers for UNIX, and commercially available Windows C^{++} compilers.

4.3.3 Features

Once compiled wxWindows is fairly simple to use. It is a single, object-oriented interface to all of the native APIs it supports. This provides an advantage to developers who only have to learn one toolbox, but can develop applications for any number of platforms. Although it is recognized that wxWindows could be considered a propri-

etary toolbox, being an open, internet based project provides some protection from the transience of software companies.

The wxWindows application framework initially provides a main frame which can be resized, iconified, and closed. Subwindows are added to this main frame to create a unique application. Commonly used widgets are packaged in easily implemented objects. An example of this packaging is the wxButton object which creates a push button, and uses a call back function to specify the application response to the action. There is a rich set of user interface tools available. The standard pull down, pull right, and popup menus are easily added. Dialog boxes and pop up or static panels can be populated with any combination of buttons, check boxes, choice lists, radio boxes, and text boxes, as can be seen in Figures 4.1 and 4.2. Any tool commonly found in a commercial application can be created using the wxWindows library. There is also support for such options as audio output, printing, graphics, and interprocess communication.

4.3.4 wxWindows and Open Inventor

wxWindows doesn't directly support Open Inventor. At the University Paderborn, Kai Benndorf developed a wrapper class which maps the system dependent window calls of Open Inventor to the appropriate wxWindows classes. The wxWindows application treats these calls as it would any other and uses the appropriate API to create them on each platform. The user interface in the Viewer is created entirely with wxWindows.

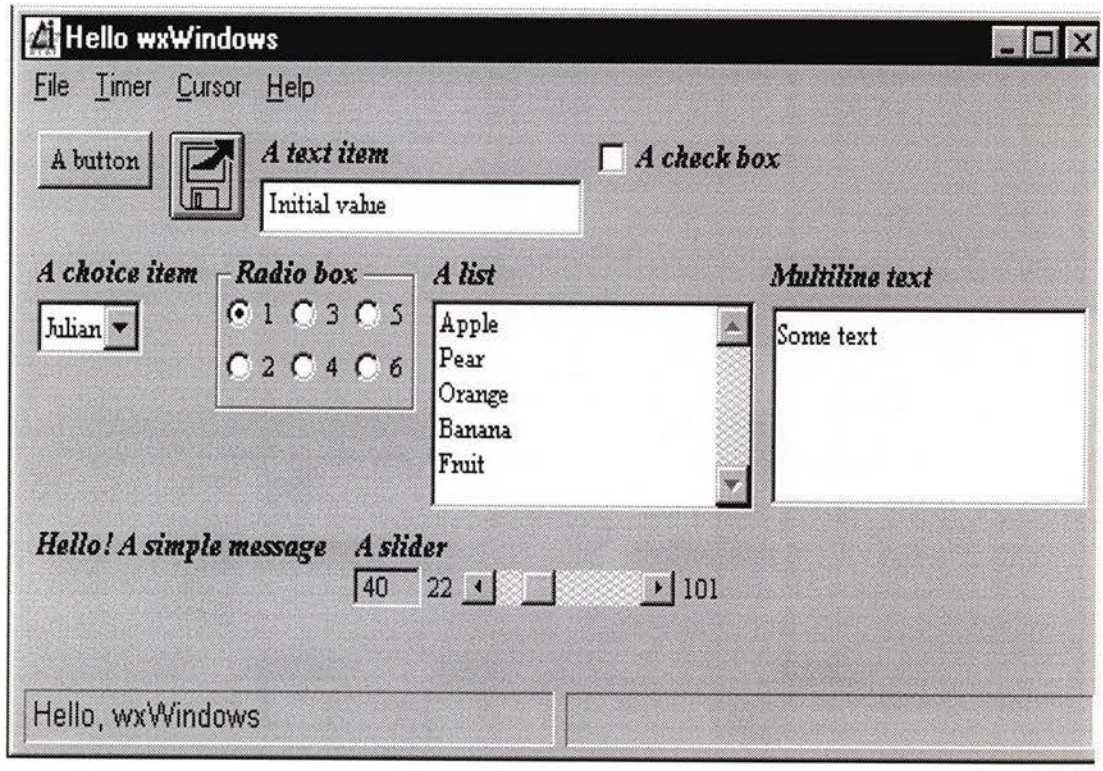


Figure 4.1: A sample of the tools available with wxWindows - Windows implementation.

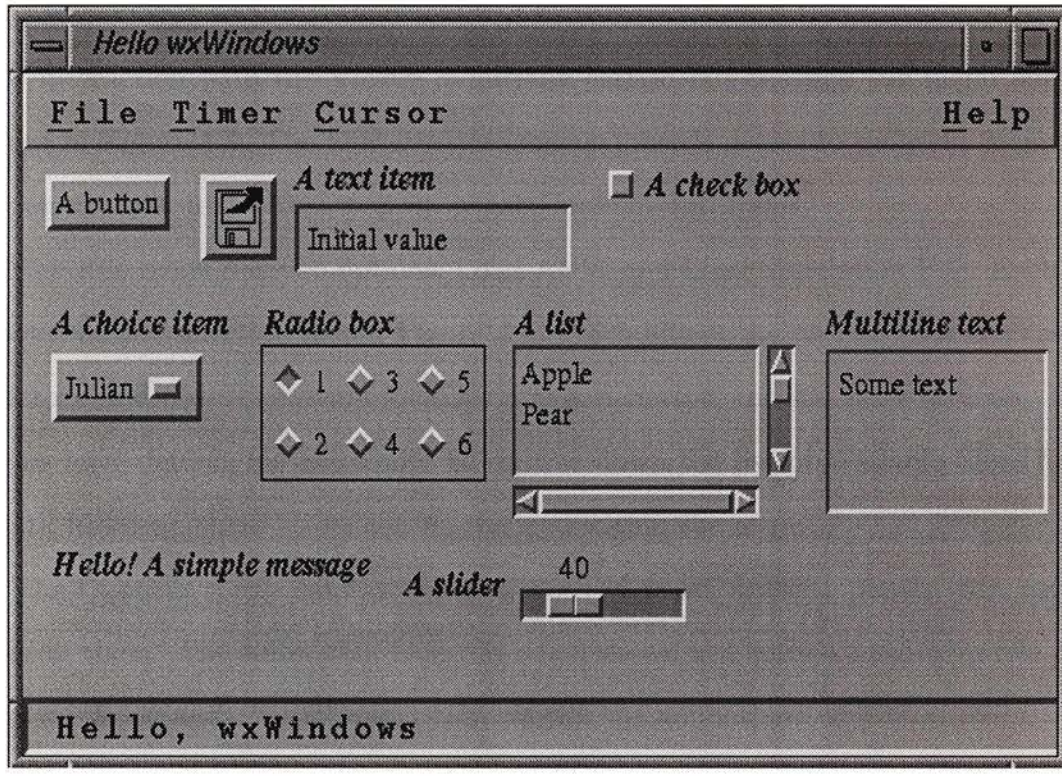


Figure 4.2: A sample of the tools available with wxWindows - Motif implementation.

Figure 4.3: The main menu bar in Viewer.

4.4 The Viewer GUI

User interaction with the Viewer is initially carried out through the menu bar displayed in Figure 4.3. The four available menus provide access to pullright menus for further choices, text boxes, pop up panels, and checked items.

The File menu has two options: Load AUV, and Load Simulation File. Each one activates a file selection menu, Figure 4.4, which allows the user to pick the desired directory and file. The menu includes a cancel button and file filter options. Both menus have default file extensions to narrow down the list of available files (*.auv, *.sim); however, these filters can be changed within the menu.

The Controls menu has two options, both of which activate panels with various tools on them. The Animation Controls panel shown in Figure 4.5 provides two push buttons, *Reset* and *Play/Pause,* which control the animation.

The Instrument Panel menu consists of choice lists and radio boxes, as can be seen in Figure 4.6. Each line corresponds to a space in the panel where an instrument can be rendered. The choice list allows the user to choose the information to be displayed by a particular instrument (e.g., a velocity, depth). The radio boxes are used to select the instrument display style. If the information hasn't been shown before, the default representation style will be rendered until a new style is selected.

The Environment pulldown menu, Figure 4.7, has a more complex set of options.

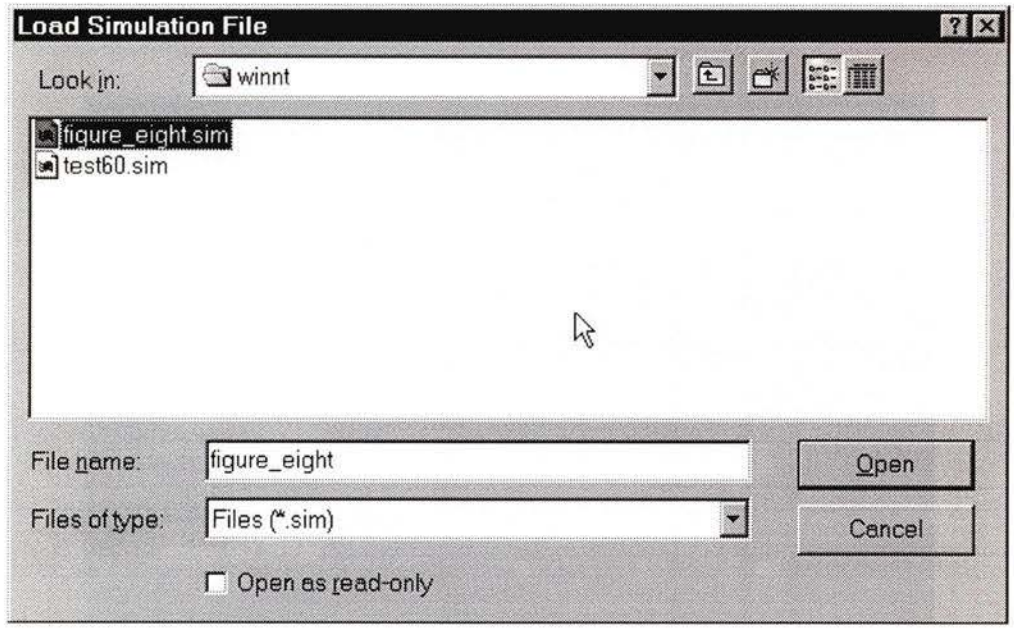


Figure 4.4: A file selection menu.

Figure 4.5: The animation control panel.

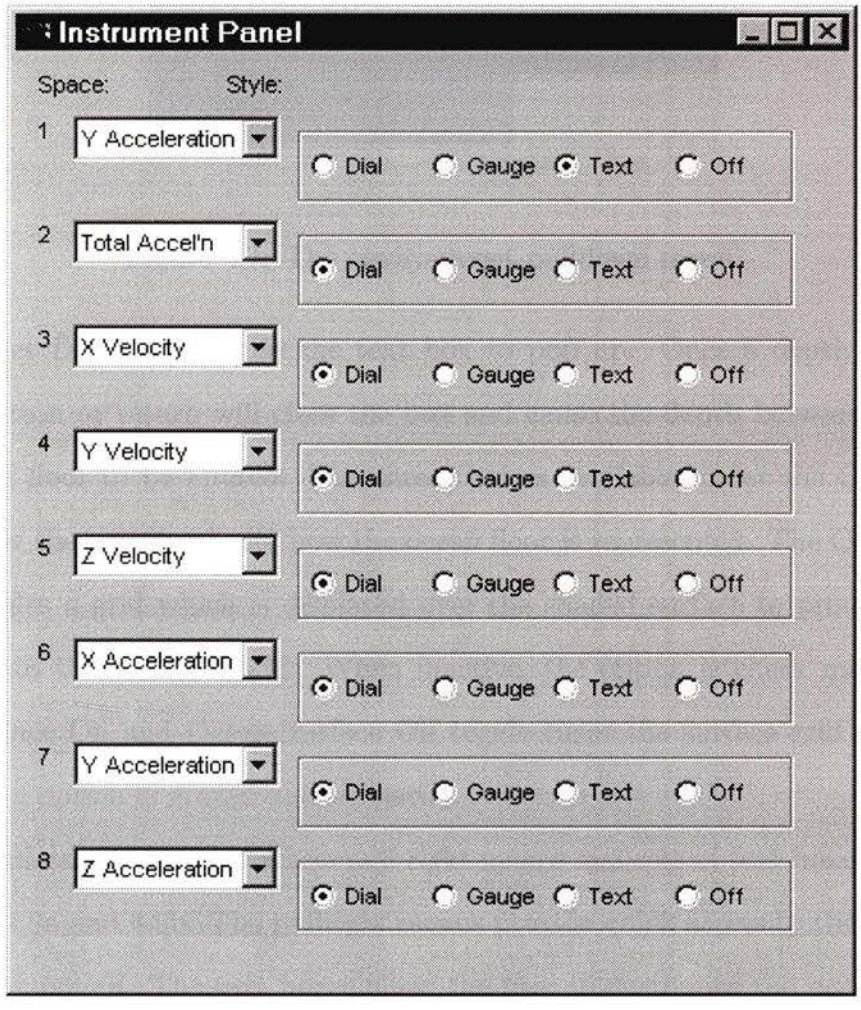


Figure 4.6: The instrument control panel.

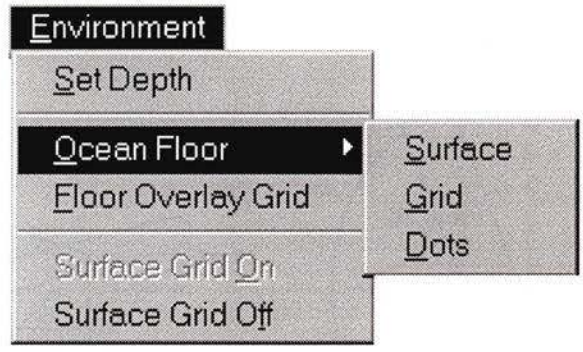


Figure 4.7: The environment pulldown menu.

Selecting Set Depth will cause the text box to pop up. Once a depth is entered, the OK button or return will close the box and cause the depth between the ocean surface and floor to be altered. The three choices available under the Ocean Floor option allow the user to specify how the ocean floor is represented. The Overlay Grid option toggles a grid which is displayed over the shaded surface to provide a sense of depth into the virtual world. When inactive, the option is check marked. The Ocean Surface On and Ocean Surface Off toggle turns the surface grid on and off. The inactive option is grayed out for clarity.

The Camera menu provides two pull right menus and sets of text boxes, as shown in Figures 4.9a and 4.9b. The pullright menus provide quick access to the predefined camera view points. The text boxes allow the user to customize the position of the camera. Figure 4.8a and Figure 4.8b show what information each text box modifies. The vehicle following schemes were described in detail in Chapter 3.

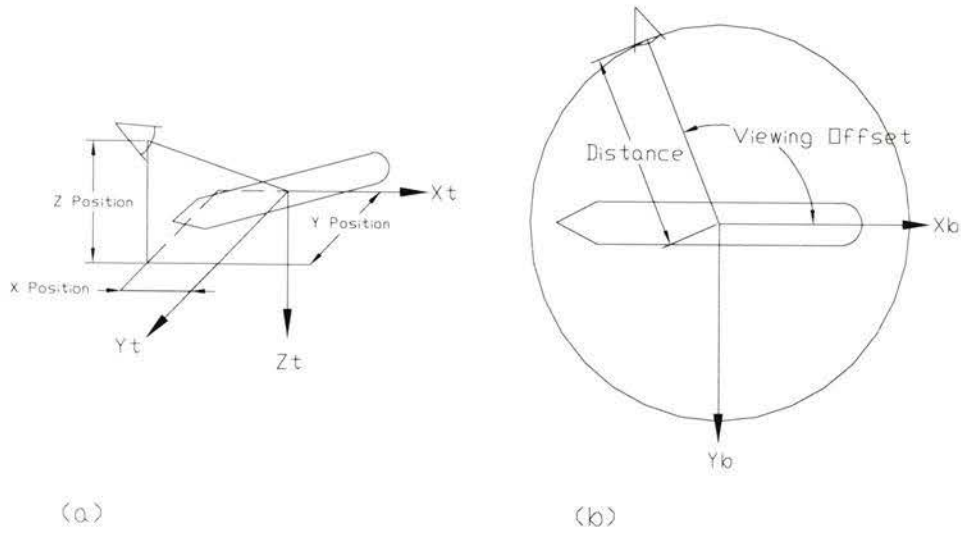


Figure 4.8: The camera customization variables.

Figure 4.9: The camera menus.

Chapter 5

Porting to a PC

The following chapter outlines the hardware and software required to run the Viewer on a PC. It also presents the frame rate results for the PC version.

5.1 Software

TGS licenced Open Inventor from SGI. It now develops and distributes Open Inventor world wide. Its versions are supported on various UNIX platforms and Windows 95 and NT.

Open Inventor can be downloaded from the TGS website. There is a free 30 day trial period of the entire library which can be accessed with a password issued by TGS. The graphics components of Open Inventor are identical on any platform. The code written on the SGI or PC is completely transferable between systems, with a line of code to indicate whether it should be a Windows, or UNIX rendering window.

Open Inventor 2.4.0 is a 20 MB download, and is 60 MB once expanded. Installation is matter of running the setup program as a system administrator. Currently,

some options in MSVC^{++} 5.0 must be set manually; however, complete instructions are included in the installation guide.

The Open Inventor distribution for the PC includes all of the sample programs in the Inventor Mentor [21], and the Inventor Toolmaker [28]. The application wizards, and sample programs are distributed for MSVC^{++} only.

Open Inventor is licenced for a specific host. As such, a password based on the volume serial number of the host computer is issued by TGS. Also, programs link with two MSVC^{++} dynamic link library files, MSVC50.dll and MSVCRT.dll, which are found on the local drive of the computer on which MSVC^{++} is installed. These files must be on the local drive of the computer running any Open Inventor application, and all of the sample and demo programs.

Because Open Inventor is multithreaded, the wxWindows library must be compiled as a multithreaded library as well. With wxWindows 1.68C this involved adding two flags to the makefile. With these changes, the wxWindows library compiled from the makefiles without a problem. The Viewer source code compiled on the PC with only minor changes to it.

The application executable and two inventor ASCII files can be distributed as a zipped file and installed in any directory.

5.2 Hardware

An investigation of available graphics cards and high end PCs found a range of hardware that would meet any budget. All the graphics cards considered support OpenGL in hardware. Several high end computers were also considered as they are tuned for 3D graphics applications, and are built around the specialized graphics cards.

Because the technology is constantly changing, as is the pricing, it was considered important to first evaluate a basic graphics card in a standard PC. Upgrades and higher end hardware purchases could then be made based on the evaluation and the actual requirements of the viewer once it was ported to the PC.

The basic evaluation card chosen for our tests was a Leadtek WinFast 3D L2300 3D accelerator card. This card was selected as the best overall and best value graphics card in a recent test of 26 graphics cards for PCs [29]. The card was installed in a Pentium 233 MHz PC with 32 MB RAM running Windows NT 4.0. This graphics card is based on the 3D Labs Permedia 2 chipset, and is the only slightly specialized component added to the PC to operate the animation. The card has 8MB SGRAM, and supports OpenGL rendering in hardware. It cost approximately $300 Cdn; it is therefore in harmony with the low cost project goal.

5.3 Performance

The PC version of the application, Figure 5.1, has the same look and functionality as the SGI version, Figure 3.14. The most dramatic difference between the two platforms is frame rate. The PC version of the AUV Viewer operates at a rate of 6.8 fps, as compared to a rate of 1.8 fps on the SGI. At this speed the animation is smooth, and acceptable for a computer based design tool.

Since advances in graphics cards for PCs are occurring at a tremendous rate, it is expected that this frame rate can be doubled on a yearly basis for the foreseeable future. Graphics cards based on the 3D Labs GMX 2000 chipset can already be purchased, and are about three times faster than the Permedia 2 boards (albeit at 10-20 times the price). It is expected that a more complex animation (e.g. including

textures or a very complex environment) would require a higher end board to maintain acceptable frame rates.

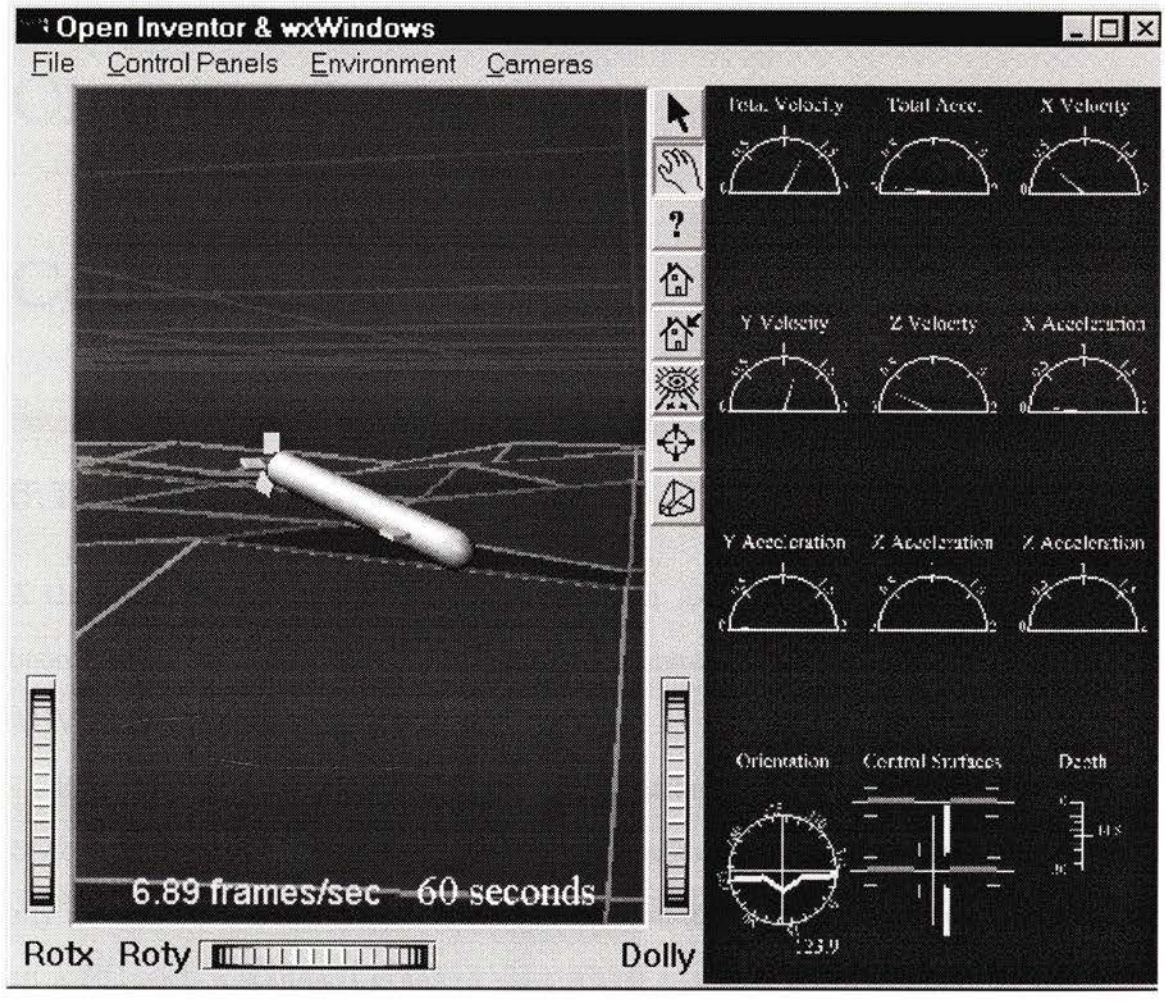


Figure 5.1: The Viewer - PC version.

Chapter 6

Conclusions

6.1 Conclusions

A three-dimensional graphical animation of an AUV in a virtual environment has been created to assist an AUV designer's understanding of the dynamics of various AUV configurations.

It is possible to develop 3D graphics applications on a PC and achieve acceptable performance rates. A PC based graphics application reduces the hardware cost generally associated with 3D graphics capable computers. The application runs smoothly on a PC with a standard graphics card which supports OpenGL, and an Open Inventor runtime licence.

An instrument panel includes both general purpose and special purpose instruments to assist an AUV designer's understanding of the behavior being observed. The panel is organized to provide the designer with complete control over what information is shown in the instrument panel, as well as where it is shown, and in what format it is displayed. The flexibility promotes customization for both designer preference

and current usage.

A standard set of graphical user interface tools provides a familiar method of interaction with the application.

The entire system has been written using cross-platform compatible libraries. It is portable to any platform which supports OpenGL. Alternate platform support ensures that the application is accessible to a variety of users.

The three-dimensional graphical simulation will enhance the AUV designer's ability to evaluate the stability and controllability of a large number of vehicle configurations. The animation reduces the amount of information to be synthesized by integrating it into an intuitve format. It is a tool that uses resources effectively. It will provide efficient means to rapidly evaluate AUV designs and arrive at an optimal design with minimal experimental effort.

6.2 Future Work

Considering the leap in performance from the SGI to the PC, further development of the animation should proceed on the PC. The low cost of the hardware required to operate the PC version of the animation also supports PC based development.

Future work includes linking the dynamics simulation of the ISE ARCS vehicle to the 3D animation to operate both components simultaneously. There is also potential to implement the Viewer as a field trial tool, and extend it to include multiple vehicles and towed vehicles.

References

[1] J. R. Fricke, " Down to the Sea in Robots," *Technology Review*, pp. 46–55, October 1994.

[2] A. M. Horwitz, " Standing on Guard for Thee: Canada's New Mine Countermeasures System," *The BC Professional Engineer*, pp. 8–26, January/February 1994.

[3] S. T. Tuohy, " A Simulation Model for AUV Navigation," in *Proceedings of the IEEE Oceanic Engineering Society Conference Autonomous Underwater Vehicles (AUV) 94*, (Cambridge), pp. 470–478, July 1994.

[4] S. Tuohy, D. Bruening, and N. Patrikalakis, " Visualization for AUV Non-Traditional Navigation Algorithm Development," in *Proceedings of the IEEE Oceanic Engineering Society Symposium Autonomous Underwater Vehicles (AUV) 96*, (Monterey), pp. 380–387, June 1996.

[5] Y. Kuroda, K. Aramaki, and T. Ura, " AUV Testing using Real/Virtual Synthetic World," in *Proceedings of the IEEE Oceanic Engineering Conference on Autonomous Underwater Vehicles (AUV) 1996*, pp. 365–371, 1996.

[6] B. Fletcher and S. Harris, " Development of a Virtual Environment Based Training System for ROV Pilots," in *Conference Proceedings IEEE/MTS Oceans 96*, (Fort Lauderdale), pp. 65–71, September 1996.

[7] E. L. Nelson, J. F. DeSoi, J. W. Mollenhauer, S. R. McClaran, K. P. Carroll, U. W. Pooch, and G. N. Williams, " User Interface Design Strategies for AUVC Software Development," in *Proceedings of the 1992 Symposium on Autonomous Underwater Vehicle Technology*, (Washington), pp. 152–157, June 1992.

[8] X. Chen, D. Marco, S. Smith, E. An, K. Ganesan, and T. Healey, " 6 DOF Nonlinear AUV Simulation Toolbox," in *Proceedings of the IEEE/MTS Oceanic Engineering Oceans '97 Conference*, (Halifax), October 1997.

[9] D. J. Wall, " A Development System For Unmanned Untethered Submersibles," Master's thesis, University of Victoria, 1994.

[10] M. J. Zyda, R. B. McGhee, S. Kwak, D. B. Nordman, R. C. Rogers, and D. Marco, " Three-Dimensional Visualization of Mission Planning and Control for the NPS Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, vol. 15, pp. 217–221, July 1990.

[11] D. Brutzman, " Virtual World Visualization for an Autonomous Underwater Vehicle," in *Procedings of the IEEE Oceanic Engineering Society Conference Oceans 95*, (San Diego, California), pp. 3–10, October 1995.

[12] D. P. Brutzman, Y. Kanayama, and M. J. Zyda, " Integrated Simulation for Rapid Development of Autonomous Underwater Vehicles," in *Proceedings of the IEEE Oceanic Engineering Society Autonomous Underwater Vehicle (AUV) 92 Conference*, (Washington), pp. 3–10, June 1992.

[13] D. Brutzman, " Virtual World for an Autonomous Underwater Vehicle." Web Site - http://www.stl.nps.navy.mil/ auv/uvw-tutorial.html.

[14] D. Kortenkamp, P. Bonasso, and R. Murphy, *AI-Based Mobile Robots*, pp. 1–20. MIT/AAAI Press, 1997.

[15] M. Nahon, " A Simplified Dynamics Model for Autonomous Underwater Vehicles," in *Proceedings of the 1996 IEEE Symposium on Autonomous Undersea Vehicles*, (Monterey), pp. 373–379, June 1996.

[16] B. Butler, " Field Trials of the THESEUS AUV," in *Proceedings of the 9th International Symposium on Unmanned Untethered Submersible Technology*, (Northeastern University), pp. 6–15, 1995.

[17] B. Butler and V. D. Hertog, " THESEUS: A Cable-Laying AUV," in *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Techology*, (University of New Hampshire), pp. 1–6, September 1993.

[18] L. Frietag, M. Johnson, and J. Preisig, " Acoustic Communications for UUVs," *Sea Technology*, vol. 39, pp. 65–71, June 1998.

[19] C. P. Sayers, R. P. Paul, L. L. Whitcomb, and D. R. Yoerger, " Teleprogramming for Subsea Teleoperation Using Acoustic Communication," *IEEE Journal of Oceanic Engineering*, vol. 23, pp. 60–71, January 1998.

[20] J. M. Preston, " Stability of Towfish used as Sonar Platforms," in *Proceedings of Mastering the Oceans Through Technology (Oceans 92)*, (Newport), pp. 888–893, October 1992.

[21] J. Wernecke, *The Inventor Mentor.* Adison-Wesley Publishiing Company, 1994.

[22] J. Neider, T. Davis, and M. Woo, *OpenGL Programming Guide - The Official Guide to Learning OpenGL, Release 1* . Addison-Wesley Publishing Company, 1993.

[23] P. S. Strauss and R. Carey, " An Object-Oriented 3D Graphics Toolkit," *Computer Graphics*, vol. 26, pp. 341–349, July 1992.

[24] Bristol-Technology, " Wind/U White Paper." Web Site - http://www.bristol.com/Bibliography/wupaper.html.

[25] DataFocus, " Products and Services." Web Site - http://www.datafocus.com/nutcracker/productsservices.htm.

[26] J. Smart, *wxWindows User Manual.* Artificial Intelligence Applications Institute, University of Edinburgh, 1996.

[27] K. Reichard and E. F. Johnson, " A Free, Portable GUI Toolkit," *Unix Review*, vol. 13, pp. 87–90, November 1995.

[28] J. Wernecke, *The Inventor Toolmaker.* Addison-Wesley, 1994.

[29] " Hardware Lab Report," *Byte*, March 1998.

Vita

Surname: Hackett **Given Names:** Georgina Bronwyn

Educational Institutions Attended:

University of British Columbia (UBC)	1991 to 1996
University of Victoria	1996 to 1998

Degrees Awarded:

B.A.Sc.	University of British Columbia (UBC)	1995

Honours and Awards:

Research Assistantship (University of Victoria)	1996-1998
IEEE Oceanic Engineering Society Scholarship	1995
Robert Allen Memorial Scholarship in Naval Architecture	1994

Publications:

D. Perrault, G. Hackett, M. Nahon, Simulation and Active Control of Towed Undersea Vehicles, In Proceedings of the IEEE/MTS Oceans '97 Conference, Halifax, NS

Partial Copyright License

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

A Portable Animation Facility for the Design of Autonomous Underwater Vehicles

Author [redacted]

Georgina Bronwyn Hackett

August 8, 1998