



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

**Modeling and 3D Visualization for
Evaluation of Anti-Terrorism/Force Protection Alternatives
Phase II Final Report**

Don Brutzman, Curt Blais and Terry Norbraten, Editors

21 November 2006

Approved for public release; distribution is unlimited.

Prepared for: Naval Facilities Engineering Service Center, 1100 23rd Avenue Ocean
Engineering Dept. C51, Port Hueneme, CA 93010

and

The Navy Modeling and Simulation Office
1333 Isaac Hull Ave., Stop 5012
Washington Navy Yard, D.C. 20376-5012

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

COL David A. Smarsh, USAF
Acting President

Dr. Leonard A. Ferrari
Provost

This report was prepared for and funded by the Naval Facilities Engineering Service Center, Port Hueneme, California, and the The Navy Modeling and Simulation Office, Washington Navy Yard, D.C.

.

Reproduction of all or part of this report is authorized.

This report was prepared by:

Donald P. Brutzman, Ph.D.
Associate Professor

Curtis L. Blais
Research Associate

Terry D. Norbraten
Research Associate

Reviewed by:

Released by:

Rudolph P. Darken, Ph.D.
Director, MOVES Institute

Dan C. Boger
Interim Associate Provost and
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 21 November 2006	3. REPORT TYPE AND DATES COVERED Project Final Technical Report, 01 OCT 2005 – 30 SEP 2006	
4. TITLE AND SUBTITLE: Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives Phase II Final Report			5. FUNDING NUMBERS N0002506WR06603 and N0001406WR20154	
6. AUTHORS: Don Brutzman, Curtis L. Blais and Terry D. Norbraten, Eds.				
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER NPS-MV-06-002	
9. SPONSORING /MONITORING AGENCY NAME AND ADDRESS Naval Facilities Engineering Service Center (NFESC) 1100 23 rd Avenue Ocean Engineering Dept. C51 Port Hueneme, CA 93010 Navy Modeling & Simulation Office (NMSO) 1333 Isaac Hull Ave., Stop 5012 Washington Navy Yard, D.C. 20376-5012			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES: The views expressed in this report are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT <p>Modern Modeling and Simulation (M&S) techniques offer flexible, economical capabilities for assessing naval installation security systems, equipment and Concepts of Operations (CONOPS). These tools are useful for assessing risk and vulnerability in a broad range of operational situations and in response to a spectrum of threat scenarios. Of particular interest to both military and homeland-defense analysts is the combined shore-side and water-side protection of naval and harbor facilities.</p> <p>In August of 2005, the NPS MOVES Institute was funded by the Naval Facilities Engineering Service Center (NFESC) to investigate and develop such an analytic tool. This report describes the work accomplished during Phase II of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives project in order to achieve that goal.</p> <p>Waterside protection includes surveillance (detection and assessment), delay (e.g., barriers), and warning and response means (e.g., patrol craft). The purpose of the Phase II effort was to develop an analysis tool that supports assessment of the effectiveness of various sensor, barrier, and response systems to enable decision-makers to make good judgments on what to purchase and employ. For example, if there is no physical barrier in a port to protect naval assets then when does a threat need to be detected to permit sufficient time to intercept/neutralize and how many patrol craft and/or weapon stations are needed to provide an acceptable level of protection? Alternatively, if a barrier is employed that effectively stops all small boats for a designated period of time, then when does detection need to occur and how many patrol boats are needed for the same level of protection? With various surveillance system assets (including surface and/or subsurface sensors), how much time is available between detection/reporting and response?</p> <p>The selection of effective combinations of sensors, barriers, and response systems requires a tool that can represent all these various assets and physical factors, providing insights into the most effective combinations that provide an acceptable level of protection at the least cost (in terms of manpower and dollars) and least risk (in terms of lives and infrastructure).</p>				
14. SUBJECT TERMS Anti-Terrorism/Force Protection (ATFP), Installation Security, Harbor Defense Modeling and Simulation (M&S), Discrete Event Simulation (DES), Design of Experiments (DOE), Simulation Analysis			15. NUMBER OF PAGES 171	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Modern Modeling and Simulation (M&S) techniques offer flexible, economical capabilities for assessing naval installation security systems, equipment and Concepts of Operations (CONOPS). These tools are useful for assessing risk and vulnerability in a broad range of operational situations and in response to a spectrum of threat scenarios. Of particular interest to both military and homeland-defense analysts is the combined shore-side and water-side protection of naval and harbor facilities.

In August of 2005, the NPS MOVES Institute was funded by the Naval Facilities Engineering Service Center (NFESC) to investigate and develop such an analytic tool. This report describes the work accomplished during Phase II of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives project in order to achieve that goal.

Waterside protection includes surveillance (detection and assessment), delay (e.g., barriers), and warning and response means (e.g., patrol craft). The purpose of the Phase II effort was to develop an analysis tool that supports assessment of the effectiveness of various sensor, barrier, and response systems to enable decision-makers to make good judgments on what to purchase and employ. For example, if there is no physical barrier in a port to protect naval assets then when does a threat need to be detected to permit sufficient time to intercept/neutralize and how many patrol craft and/or weapon stations are needed to provide an acceptable level of protection? Alternatively, if a barrier is employed that effectively stops all small boats for a designated period of time, then when does detection need to occur and how many patrol boats are needed for the same level of protection? With various surveillance system assets (including surface and/or subsurface sensors), how much time is available between detection/reporting and response?

The selection of effective combinations of sensors, barriers, and response systems requires a tool that can represent all these various assets and physical factors, providing insights into the most effective combinations that provide an acceptable level of protection at the least cost (in terms of manpower and dollars) and least risk (in terms of lives and infrastructure).

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

The Naval Postgraduate School Modeling, Virtual Environments, and Simulation (MOVES) Institute thanks the many people who made the Phase II effort of this ongoing project such a success.

To Robert Taylor, Naval Facilities Engineering and Service Center (NFESC), and his assistant Alexandra Devisser for sponsoring the work and providing management guidance throughout the life of this project.

To John Moore and his team at the Navy Modeling & Simulation Office for his continued support of the MOVES Institute at NPS for this and countless other ongoing M&S projects.

To Alan Hudson, Yumetech, Inc. and his team for many hours spent on software engineering of the Xj3D browser, SavageStudio and 3D model archives and for many hours of consultations.

To Chris Greuel, Dan Ancona and Carlos Newcomb and the Planet 9 team for superb 3D modeling of Pearl Harbor and other ports used in this project.

To Dallas Meggit, Dennis Garrood, Roger Christiansen and Mario Pozzo and the Sound and Sea Technologies (S&ST) team for expert project guidance and sub-contracting support.

To Rick Goldberg of Aniviza, Inc. for his many hours spent developing Viskit and the many improvements made in support of LT Sullivan's thesis research.

To Tony Parisi and Keith Victor and the Media Machines, Inc. team for great work in incorporating X3D support in open source Flux Studio code base.

To Robert Landsdale, Andrew Grieve and the Okino Computer Graphics, Inc. team for their contributions in support of X3D rendering.

To Len Daly of Daly Realism for his contributions in help file set and documentation throughout the project duration.

To Margaret Bailey, Doug Nelson and the Sonalysts, Inc. team for their support of sonar physics modeling.

To LT Patrick Sullivan, USN for his research efforts which propelled the ATRP project into the forefront through countless demonstrations to various agencies and commands.

To LCDR Travis Rauch, USN for his research into SMAL which greatly simplified auto-generation of 3D scenarios.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

1.0	INTRODUCTION.....	1
1.1	BACKGROUND	1
1.2	PROJECT OBJECTIVES.....	3
	1.2.1 Phase I Project Objectives.....	3
	1.2.2 Phase I Project Accomplishments	3
	1.2.3 Phase II Project Objectives.....	4
	1.2.4 Phase II Project Accomplishments.....	5
1.3	PROJECT MANAGEMENT ORGANIZATION	9
	1.3.1 Naval Postgraduate School (NPS)	9
	1.3.2 Naval Facilities Engineering Service Center (NFESC)	9
	1.3.3 Sound and Sea Technologies (S&ST).....	9
	1.3.4 Yumetech, Inc.....	10
	1.3.5 Aniviza, Inc.....	10
	1.3.6 Planet 9 Studios	10
	1.3.7 Media Machines	10
	1.3.8 Okino Computer Graphics, Inc.....	10
	1.3.9 Daly Realism.....	10
	1.3.10 Sonalysts.....	11
1.4	ORGANIZATION OF THIS REPORT.....	11
2.0	NAVAL INSTALLATION SECURITY MODELING & SIMULATION.....	13
2.1	NAVAL INSTALLATION SECURITY M&S OBJECTIVES	13
2.2	NAVAL INSTALLATION SECURITY M&S REQUIREMENTS.....	13
2.3	AT/FP ANALYSIS TOOL SOFTWARE REQUIREMENTS	14
3.0	PHASE II PARTNER CONTRIBUTIONS.....	17
3.1	INTRODUCTION.....	17
3.2	NAVAL POSTGRADUATE SCHOOL.....	17
	3.2.1 LCDR Travis Rauch, USN Thesis	17
	3.2.2 LT Pat Sullivan, USN Thesis	18
3.3	SOUND & SEA TECHNOLOGIES (S&ST).....	18
	3.3.1 Overview of S&ST Contributions	18
	3.3.2 S&ST Team	20
3.4	ANIVIZA	21
	3.4.1 Overview	21
	3.4.2 Activities Performed	21
	3.4.3 Aniviza Team.....	23
3.5	DALY REALISM.....	23
	3.5.1 Overview	23
	3.5.2 Previous Work.....	23
	3.5.3 Scope of Work	24
	3.5.4 Activities Performed	24
	3.5.5 Deliverables Completed.....	25
	3.5.6 Recommendations for Future Work	25

	3.5.7	Daly Realism Team	26
3.6		MEDIA MACHINES.....	26
	3.6.1	Overview	26
	3.6.2	Scope of Work	26
	3.6.3	Activities Performed	26
	3.6.4	Media Machines AT/FP Team	27
3.7		OKINO COMPUTER GRAPHICS	28
	3.7.1	Overview	28
	3.7.2	Primary Task Groups.....	28
	3.7.3	Main Development Achievements	30
	3.7.4	Okino AT/FP Team.....	31
3.8		PLANET 9 STUDIOS.....	31
	3.8.1	Overview	31
	3.8.2	Previous Work.....	31
	3.8.3	Scope of Work	32
	3.8.4	Activities Performed	33
	3.8.5	Deliverables Completed.....	36
	3.8.6	Recommendations for Future Work	38
	3.8.7	Planet 9 Studios AT/FP Team.....	40
3.8		SONALYSTS.....	41
	3.8.1	Overview	41
	3.8.2	Progress to date:.....	41
	3.8.3	Recommendations for future work	41
	3.8.4	Sonalysts Team.....	42
3.9		YUMETECH.....	42
	3.9.1	Progress to Date	42
	3.9.2	Future Development	44
		3.9.2.1 Location Set-up	44
		3.9.2.2 Barrier Representation	45
		3.9.2.3 Property Editor.....	45
		3.9.2.4 Statistical/Visualization Tool Improvements	46
		3.9.2.5 3D Viewer Improvements.....	46
		3.9.2.6 Automating the Integration of Savage Studio with the Savage Library	47
		3.9.2.7 Day/Night/Weather Effects.....	48
	3.9.3	Yumetech, Inc. Team.....	48
4.0		SUMMARY AND CONCLUSIONS	49
4.1		M&S WORKSHOP CONCLUSIONS FROM (BRUTZMAN ET AL. 2006).....	49
4.2		PHASE II CONCLUSIONS AND RECOMMENDATIONS	50
APPENDIX A.		NAVAL INSTALLATION SECURITY MODELING AND SIMULATION WORKSHOP	51
		ATTENDEE LIST:	51
		NAVAL INSTALLATION SECURITY MODELING AND SIMULATION WORKSHOP AGENDA:.....	52
		Workshop Day 1: May 9, 2006 Tuesday	52

Workshop Day 2: May 10, 2006 Wednesday.....	53
Workshop Day 3: May 11, 2006 Thursday (organizers only).....	53
APPENDIX B. INSTALLATION/OPERATION TUTORIAL AT NPS MOVES OPEN HOUSE, AUGUST 7, 2006.....	55
APPENDIX C. AT/FP PROJECT FLYER.....	57
APPENDIX D. MODELING AND SIMULATION WORKSHOP CD-ROM.....	61
APPENDIX E. DISKIT SENSOR AND MOVER DYNAMICS: LOGARITHMICALLY RANGED ATTENUATED TRANSMISSION LOSS SONAR	63
E.1 DISKIT SENSOR AND MOVER DYNAMICS.....	64
E.1.1 Overview	64
E.1.2 DISMover3D.....	65
E.1.3 Sensors, Targets and Mediators	67
E.1.4 ScenarioManager	73
E.1.5 Example Multisectioned Log Range Attenuated Transmission Loss Sonar (MiltiLRATL).....	74
APPENDIX F: PLANET 9 PRESENTATION SLIDESETS	103
F.1 INTRODUCTION.....	103
F.1 BUILDING GEO-REGISTERED X3D	104
F.2 3D GEOSPATIAL DATA INTERFACES AND TOOLS	111
APPENDIX G: PHASE II TECHNICAL SUMMARY OF X3D	
MODEL CONSTRUCTION.....	117
G.1 PLANET 9 STUDIOS ART TEAM	117
Overview	117
Terrain	117
Buildings	122
Aids to Navigation.....	125
Compass Rose.....	129
Port Security Barrier.....	131
X3D Model Locations	133
Planet 9 Studios AT/FP Art Team.....	133
GLOSSARY OF TERMS AND ACRONYMS.....	135
REFERENCES.....	143
INITIAL DISTRIBUTION LIST	145

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Overall Software Architecture for the Anti-Terrorism/Force Protection (AT/FP) Analysis Tool (From Phase II SOW 2006)	6
Figure 2.	POA&M for Phase II Efforts (Part I) (From Phase II SOW 2006)	7
Figure 3.	POA&M for Phase II Efforts (Part II) (From Phase II SOW 2006)	8
Figure 4.	May 2006 M&S Workshop CD Cover Image	61
Figure 5.	Simkit, Viskit, Diskit Platform Relationships.....	64
Figure 6.	DISMover3D Entity in Event Graph Form.....	65
Figure 7.	DISMover3D Parameters.....	66
Figure 8.	State Variables for the DISMover3D Entity	66
Figure 9.	Diagram of Cancelling Invalid Pending Events Due to Change in Target Velocity Vector State.....	68
Figure 10.	Simple <i>SonarMediator</i> Event Graph	73
Figure 11.	Looking Down on “Sweep”	74
Figure 12.	Baffle Geometry divided into triangular sections, viewed from above	75
Figure 13.	Side View of Approximation Geometry. First cut, “watermelon” slices.	75
Figure 14.	Further Simplification of Volume Geometry.....	76
Figure 15.	Final Geometry of Volume Space.....	76
Figure 16.	EnterRange and ExitRange Point Depicted	79
Figure 17.	Energy Potential Calculus.....	79
Figure 18.	Showing Comparison between an Inner and Outer Approximation to the Sphere by a Facet.	87
Figure 19.	CheckDetection Event Graph	95
Figure 20.	Adding the <i>MultiLRATLSonar</i> as a drop in component.....	101
Figure 21.	Debug Output from a Random Run of <i>IndianIslandSonarTest</i> in Viskit	102
Figure 22.	Oahu and Pearl Harbor Terrain Grids Optimized	118
Figure 23.	Oahu and Pearl Harbor Terrains Drapped with Imagery	119
Figure 24.	Oahu and Pearl Harbor Terrain Grids Designated.....	119
Figure 25.	Pearl Harbor Buildings Grouped into Five Separate Files Determined by Location.	122
Figure 26.	Example Building/Terrain File Dependency Chart	124
Figure 27.	Example ATON X3D Code (VRML Syntax) with One Range Light and Two Lights	128
Figure 28.	Example ATON X3D Scene with One Range Light (in distance) and Two Lights	129
Figure 29.	Example Compass Rose X3D Code (VRML Syntax)	130
Figure 30.	X3D Example Compass Rose Scene First Looking North, then Northwest..	130
Figure 31.	Example PSB X3D Code (VRML Syntax).....	132
Figure 32.	X3D Example PSB Scene First with One Section, then Three Sections.	132

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1.	List of Phase II Accomplishments by Partners (From Phase II SOW 2006).....	5
Table 2.	Table showing sample values of <i>AT</i> used for various frequencies of interest.	91
Table 3.	Various Resolution Version of the Oahu Terrain	120
Table 4.	Individual Grid Sections of the Oahu Terrain.....	120
Table 5.	Example File Names for the Single Pearl Harbor Grid Section A1	121
Table 6.	Aids to Navigation X3D Proto Models - RangeLight	126
Table 7.	Aids to Navigation X3D Proto Models – Daybeacon.....	126
Table 8.	Aids to Navigation X3D Proto Models – LightPrototype	127
Table 9.	Aids to Navigation X3D Proto Models – LightedBouyPrototype	127
Table 10.	Aids to Navigation X3D Proto Models – LightPostPrototype.....	127
Table 11.	Compass Rose X3D PROTO Attributes and Options.....	130
Table 12.	Port Security Barrier X3D PROTO attributes and options:.....	131

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 INTRODUCTION

Modeling and Simulation (M&S) techniques offer flexible, economical capabilities for assessing naval installation security systems, equipment and Concepts of Operations (CONOPS). These tools are useful for assessing risk and vulnerability in a broad range of operational situations and in response to a spectrum of threat scenarios. Of particular interest to both military and homeland-defense analysts is the combined shore-side and water-side protection of naval and harbor facilities.

1.1 BACKGROUND

The primary products for this phase of work are the Sullivan thesis, the Rauch Thesis, the M&S Workshop and the software and models distributions. This report provides additional amplifying information.

The Systems Engineering tasks associated with the naval installation security span the breadth of Navy CONUS and OCONUS bases. They must cover many existing “legacy” and proposed new systems. The Systems Engineering effort includes analysis, trades, and requirements definition and refinement. The outputs will provide the basis for recommendations for procurement, training methods, concepts of operation – in other words, all standard outputs from the Systems Engineering process for systems that are to meet the operational needs of the naval installation security initiative.

The challenges facing the naval installation security problem are complicated by the widely varying nature of the threats to be addressed, by the diversity of existing systems, equipment and Concept of Operations (CONOPS), and by the fact that there are more than 100 U.S. naval facilities, each of which can be expected to have a different set of Anti-Terrorism/Force Protection (AT/FP) requirements and solutions for harbor defense and installation security. As a result, it is simply not practical to conduct these analyses on a purely empirical basis by installing and trying different combinations of equipment and systems. A better, more cost-effective approach for Systems Engineering analysis is needed (sponsor sets the requirements).

A widely accepted methodology for dealing with complex systems is the use of M&S. M&S tools allow a user – from the analyst to the civilian administrator to the military operator –

to assemble simplified representations of actual systems that allow an understanding of underlying relationships among sensors, combatants and their behaviors, all against the backdrop of 3D, immersive displays of actual locations such as a harbor and surrounding areas.

The abstract from (Harney 2003) describes the genesis of this project:

Despite the many advances achieved within both Modeling and Simulation and Information Technology over the past several decades, practical application of such technology remains under-utilized by operational units in the United States Navy. Furthermore, when such technology has been deployed in the last decade it has been to exercise operator proficiency or increase C4I battlespace awareness. Few tools have allowed operational warfighters to run ‘what-if’ simulation scenarios to aid in development of tactical plans for executing published doctrine.

The approach taken in this thesis is to select an exemplar warfare area, in this case Anti-Terrorism and Force Protection for Navy ships, and through research and development to identify, develop, and deploy the necessary modeling and simulation (M & S) technologies to demonstrate a prototypical planning tool that can be used by today’s deployed warfighter. All research and work is conducted in a web-based, ‘user-centric’ fashion utilizing a combination of user-driven and agent-based control of entities for simulation iterations, along with various open source technologies which include Extensible 3D Graphics (X3D), Scalable Vector Graphics (SVG), and Extensible Markup Language (XML). Conventions are demonstrated for the integration of the many academic disciplines utilized during this research to achieve automatic generation of tactically significant scenarios. In order to give the end-user the greatest insight towards potential drawbacks in the tactical planning against surface-borne terrorist threats, various 2D and 3D media provide both real-time and non-real time scenario playback.

The result of this work is a fully integrated, prototypical, Java-based application that demonstrates how various Open-Source, web-based technologies can be applied in order to provide the tactical operator with tools to aid in Force Protection planning. Scenarios can be auto generated, viewed, analyzed, and manipulated by end users with little to no computer experience necessary beyond requirements for operation of a desktop personal computer (PC) in the Information Technology for the 21st Century (IT-21) environment at sea. This approach has broad applicability to improve the tactical awareness and defensive posture of ships defending against terrorist attacks in port.

1.2 PROJECT OBJECTIVES

The primary intended outcome of the project is to:

- Communicate general goals for improving naval installation security through M&S
- Define potential goals for M&S programs that support implementation of naval installation security systems
- Discuss candidate M&S requirements for naval installation security studies and analyses
- Describe and demonstrate relevant M&S capabilities and approaches
- Assess the state-of-the-art in M&S as related to naval installation security
- Identify potential areas for data interchange and collaboration through data and model sharing
- Identify the most productive areas for further M&S development
- Explore how to create broader-based tool support for tactical analysis of harbor risk and vulnerabilities.

1.2.1 Phase I Project Objectives

During the Phase I effort, the decision was made to integrate the extended capabilities for a AT/FP Visualization and Analysis Tool into a different established code base, the Autonomous Unmanned Vehicle (AUV) Workbench (AUVW). This tool provides 2D and 3D mission planning and mission execution with integrated vehicle dynamics and basic sensor physics. The current open source code base provides a more extensive framework for addition of capabilities and features to meet requirements of the AT/FP analysis tool.

1.2.2 Phase I Project Accomplishments

Accomplishments from the Phase I effort conducted through the first two quarters of fiscal year 2005 set the foundation for continuing work. This work included:

- 3D modeling of NAVMAG Indian Island, Washington, including development of Ammunition Pier, nearby buildings, and surrounding terrain

- 3D modeling and texture mapping of NAVSTA Bremerton, Washington, including development of piers, near shore buildings, and surrounding terrain
- 3D modeling of port security barriers and a selection of water craft
- Further development of existing software infrastructure in Xj3D, including initial efforts to integrate the open-source Open Dynamics Engine (ODE) software library
- Gathering geospatial information sets for multiple locales
- Gathering technical information for barriers and sensors
- User interface design for the overall planning and assessment tool
- Establishing working relationships and coordination mechanisms across the project team (NFESC, Sound&Sea Technologies, NPS, Planet 9, and Yumetech)

1.2.3 Phase II Project Objectives

Waterside protection includes surveillance (detection and assessment), delay (e.g., barriers), and warning and response means (e.g., patrol craft). The purpose of the proposed effort is to develop an analysis tool that supports assessment of the effectiveness of various sensor, barrier, and response systems to enable decision-makers to make good judgments on what to purchase and employ. For example, if there is no physical barrier in a port to protect naval assets then when does a threat need to be detected to permit sufficient time to intercept/neutralize and how many patrol craft and/or weapon stations are needed to provide an acceptable level of protection? Alternatively, if a barrier is employed that effectively stops all small boats for a designated period of time, then when does detection need to occur and how many patrol boats are needed for the same level of protection? With various surveillance system assets (including surface and/or subsurface sensors), how much time is available between detection/reporting and response? The selection of effective combinations of sensors, barriers, and response systems requires a tool that can represent all these various assets and physical factors, providing insights into the most effective combinations that provide an acceptable level of protection at the least cost (in terms of manpower and dollars) and least risk (in terms of lives and infrastructure).

1.2.4 Phase II Project Accomplishments

Allocation of AT/FP Visualization and Analysis Tool SOW Tasks to Performers									
SOW Para	Task	NPS	Planet 9	Yumetech	Aniviza	Daly Realism	Sonalysts	Media Machines	Okino
4.1	Port and Port Facility Modeling								
4.1.1	Indian Island and Bremerton Visualization Improvements	X	PRIME	X		X			
4.1.2	Pearl Harbor and Port Hueneme Visualizations	X	PRIME	X		X			
4.1.3	Assess Shore-Side integration	PRIME							
4.1.4	Physics-based models	PRIME		X	X	X			
4.1.4.1	Sonar modeling	PRIME					X		
4.1.5	DNC load/display	X		PRIME		X			
4.1.6	Navy C2 in tool modeling	PRIME							
4.1.7	X3D Tool Updates	X						PRIME(1)	PRIME(2)
4.2	Analysis Tool Development								
4.2.1	Integration with AUVW	PRIME	X	X	X	X			
4.2.2	GUI for scenario set-up	X		PRIME	X	X			
4.2.3	Simkit scenario creation	X			PRIME	X			
4.2.4	Experimental design tool	X			PRIME	X			
4.2.5	Analysis report-writing	PRIME							
4.2.6	Web3D 2D/3D UI Working Group	X	X	PRIME	X	X			
4.2.7	NMCI/IT-21	PRIME		X	X				
4.2.8	Configuration control	PRIME	X	X	X	X			
4.2.9	Remainder FY05 activities	X	X	X	X				
4.2.10	Computing Cluster	PRIME							
4.3	Education and Training								
	Instructional materials	X	PRIME			X			
	Conduct training	PRIME	X						
	Documentation	X	X	X	X	PRIME			
4.4	Team coordination and management	PRIME	X	X	X	X	X	X	X
4.5	Record follow-on requirements	PRIME	X	X	X	X			

NOTE: "PRIME" means primary responsibility for execution of the task.

(1) VizX3D product update; (2) Polytrans product update

Table 1. List of Phase II Accomplishments by Partners (From Phase II SOW 2006)

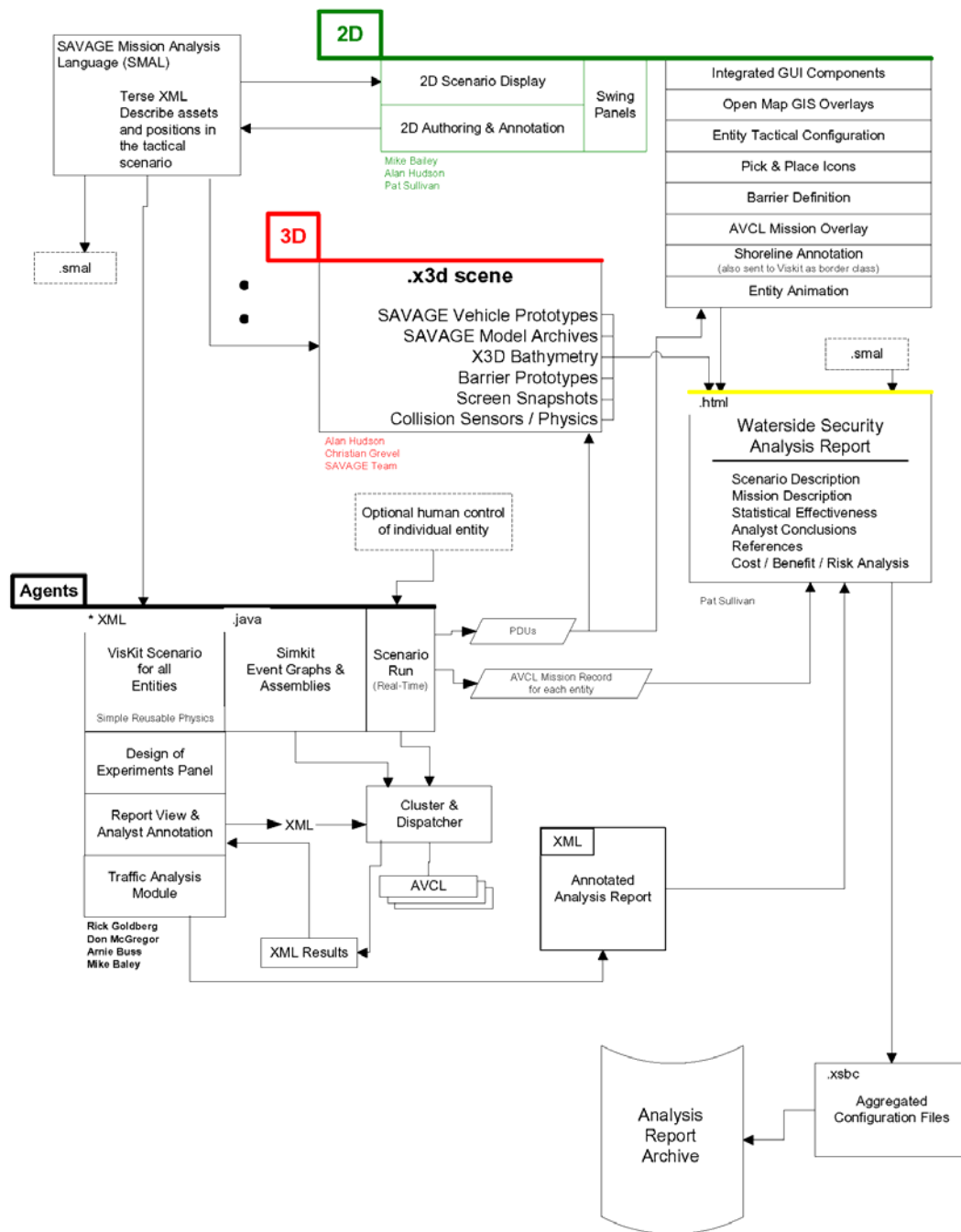


Figure 1. Overall Software Architecture for the Anti-Terrorism/Force Protection (AT/FP) Analysis Tool (From Phase II SOW 2006)

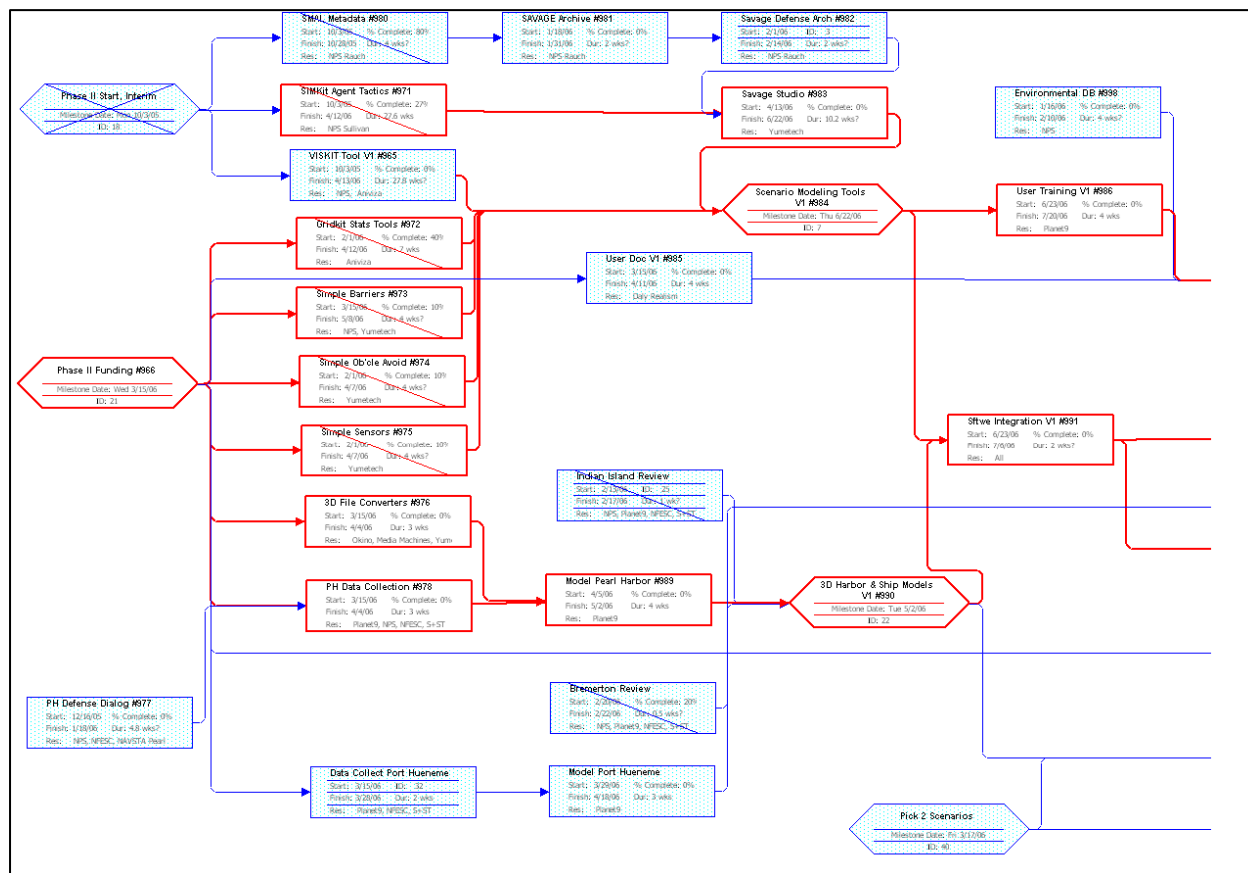


Figure 2. POA&M for Phase II Efforts (Part I) (From Phase II SOW 2006)

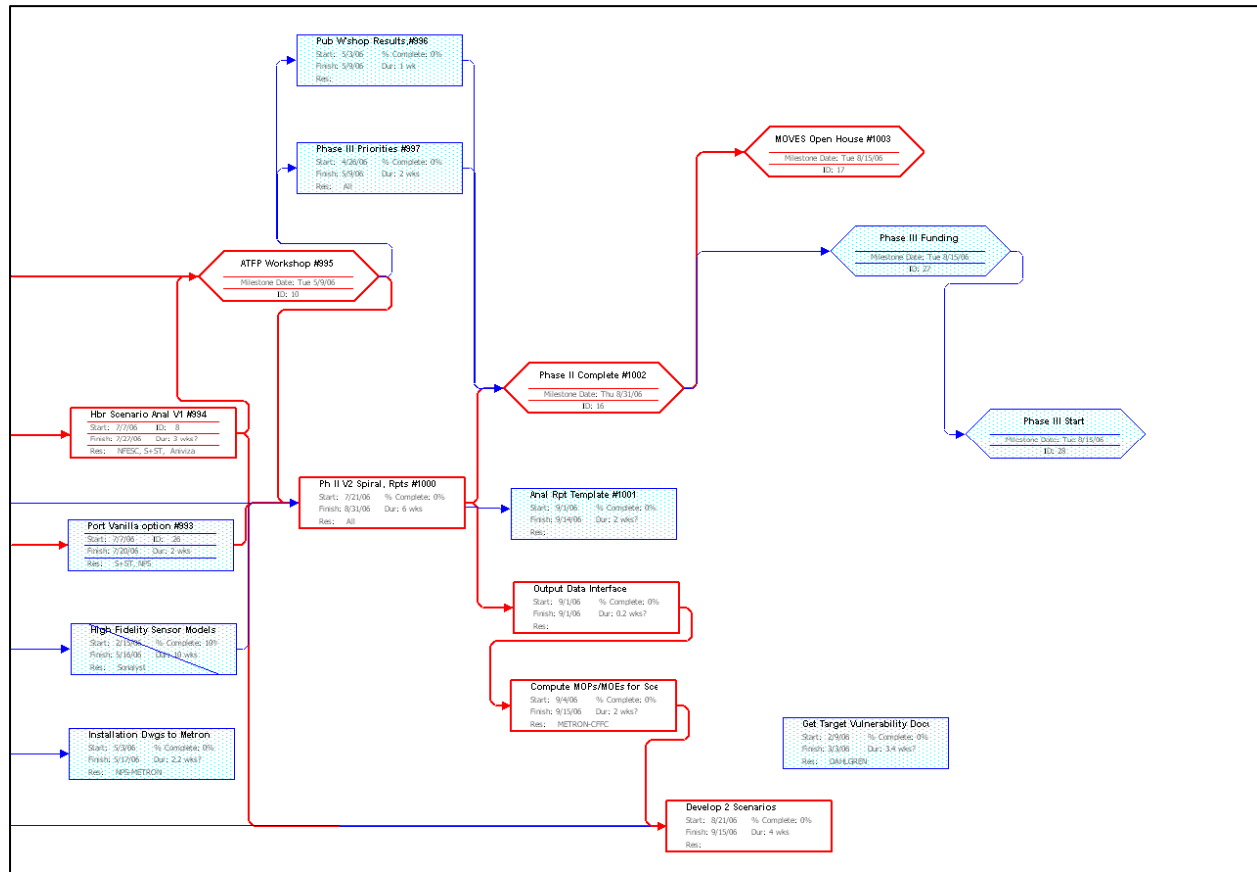


Figure 3. POA&M for Phase II Efforts (Part II) (From Phase II SOW 2006)

1.3 PROJECT MANAGEMENT ORGANIZATION

1.3.1 Naval Postgraduate School (NPS)

Donald P. Brutzman, Ph.D., NPS Principal Investigator (PI)

Arnold Buss, Ph.D., Simkit Software Engineer

Mike Bailey, Viskit Software Engineer

Don McGregor, High Performance Computing (HPC) Cluster Engineer

Jeff Weekley, 3D Modeler

LCDR Travis Rauch, USN, Thesis Researcher

LT Patrick Sullivan, USN, Thesis Researcher

Curt Blais, Project Support, Final Report Editor

Terry Norbraten, Project Support, Final Report Editor

1.3.2 Naval Facilities Engineering Service Center (NFESC)

Mr. Robert Taylor

Alexandria De Visser

1.3.3 Sound and Sea Technologies (S&ST)

Dallas Meggit

Dennis Garrood

Mario Pozzo

Roger Christiansen

Denise Bjorling

1.3.4 Yumetech, Inc.

Alan Hudson, CEO

Justin Couch

Stephen Matsuba

1.3.5 Aniviza, Inc.

Rick Goldberg, CEO

1.3.6 Planet 9 Studios

David Colleen, CEO

Chris Greuel, 3D Model Engineer

Dan Ancona, Documentation and Training

Carlos Newcomb, 3D Imagery

1.3.7 Media Machines

Tony Parisi, CEO

Keith Victor, Software Engineer

1.3.8 Okino Computer Graphics, Inc.

Robert Landsdale, CEO

Andrew Grieve

1.3.9 Daly Realism

Leonard Daly, CEO, Software Documentation

1.3.10 Sonalysts

Margaret Bailey

Doug Nelson, Physics Modeling

1.4 ORGANIZATION OF THIS REPORT

Chapter 1 is the introduction of this report and covers the overview and objectives of Phase II efforts. Chapter 2 covers the objectives and requirements of the May 2006 M&S Workshop. Chapter 3 highlights two thesis abstracts and the Phase II partner contributions and final reports submitted by each. Chapter 4 covers the summary and conclusions of this final report. Appendix A lists the attendees and the agenda of the May 2006 M&S Workshop. Appendix B lists the MOVES Open House 2006 tutorial agenda for the AT/FP Analysis Tool. Appendix C contains the AT/FP Project Flyer. Appendix D contains information on how to obtain a copy of the FOUO May 2006 M&S Workshop. Appendix E contains a white paper covering Diskit Sensor and Mover dynamics by Rick Goldberg, Aniviza, Inc. Appendix F contains Planet 9 slidesets detailing 3D model construction techniques. Appendix G concludes this report with a technical summary of Phase II efforts from Planet 9 Studios.

THIS PAGE INTENTIONALLY LEFT BLANK

2.0 NAVAL INSTALLATION SECURITY MODELING & SIMULATION

2.1 NAVAL INSTALLATION SECURITY M&S OBJECTIVES

Broadly stated, the objectives of naval installation security M&S are to:

- Develop open-source/open standards (nonproprietary) modeling and simulation tools to evaluate contributions of system and equipment alternatives to Naval and U.S. Coast Guard installation security effectiveness. This is envisioned to include a series of tools of differing complexity and fidelity for different applications.
- Develop and evaluate concepts of layered defense using existing, emerging and potential future physical security and Command, Control, Communications, Computers and Intelligence (C4I) systems and equipment.
- Facilitate the evaluation of equipment and systems in the models by providing an industry standard (e.g., Microsoft EXCEL®) interface for outputting simulation initial conditions and results. This interface will provide the user an efficient and tailored way of reporting and displaying the data and will facilitate the use of data post-processors for the generation of user-defined Measures of Performance (MOPs) and Measures of Effectiveness (MOEs).

2.2 NAVAL INSTALLATION SECURITY M&S REQUIREMENTS

A preliminary set of M&S requirements to meet the objectives listed above include:

- Perform physically-based statistical assessment and visualization to evaluate the effectiveness of sensors, barriers, and response systems for naval installation security.
- Be a true tool set (i.e., not a site-specific simulation), structured so the users can select model fidelity and scale into a simulation that provides a realistic solution to their particular problem.
- Provide realistic, extensible, 3D visualization models of bases and surrounding environment, including bodies of water, together with high-fidelity, physics-based sensor, dynamics and damage assessment models.

- Include the capability for the design of problems and scenarios and implementation of these problems on clusters of computers or a dedicated DoD supercomputing facility.
- Support training in a dynamic, realistic environment for boat handling, weapons, tactical control, and all other areas of waterside and shoreside security and response.
- Allow quantitative evaluation of the performance of sensors and systems of sensors through “hardware-in-the-loop” simulations.
- Support the conduct of pre-mission planning/post mission analysis.
- Provide quick results to common naval installation security problems through the use of a library of pre-worked simulations.
- Provide tools to allow easy generation of a notional harbor (in 3D), suitable objects and behaviors for training and demonstration purposes.
- Implement the M&S tools in an open-source software environment to eliminate dependence on proprietary software, or a single or limited number of vendors, and to eliminate recursive DoD costs for such software.
- Provide interfaces to internal calculated results and external programs such as MathWorks’ MATLAB® and Microsoft EXCEL® for user defined report generation.

2.3 AT/FP ANALYSIS TOOL SOFTWARE REQUIREMENTS

AT/FP Harbor Security Visualization and Analysis Tool Development – In addition to visualization of the environment to aid in understanding employment of security resources, an analysis tool is needed to configure and run experiments to evaluate the effectiveness of different combinations of AT/FP assets against a variety of threats. Refer to Figure 1 for an overview of the major software components of this tool. The NPS team will perform the following subtasks to design, develop, test, and demonstrate an AT/FP Harbor Security Visualization and Analysis Tool:

Integration with AUVW: Integrate AT/FP modeling with the AUV Workbench code base to create the AT/FP Harbor Security Visualization and Analysis Tool.

GUI for Scenario Set-up: Design and begin development and testing of a user interface to facilitate selection of a locale and configuration of platforms, sensors, countermeasures, threats, and other assets involved in AT/FP studies.

Simkit Scenario Creation: Design, develop, and test scenario simulation using the Simkit Discrete Event Simulation (DES) Application Program Interface (API). Simulation modeling will use the Viskit visual event graph tool for retention and reuse of modeling components.

Experimental Design Tool: Create an experimental design and execution harness for conducting analyses using the AT/FP Harbor Security Visualization and Analysis Tool. Utilize low-cost computer clusters for heavy-duty computational performance.

Analysis Report-writing: Design and develop an analysis report-writing capability to facilitate preparation of reports providing analysis results from the tool. Target audiences include AT/FP acquisition officers, AT/FP harbor supervisors, and AT/FP officers on ships entering port.

Web3D 2D/3D UI Working Group: Participate in the 2D/3D User Interface (UI) Working Group in the Web3D Consortium. The GUI design of the AT/FP Harbor Security Visualization and Analysis Tool is critical to its rapid adoption and effective employment. This is a sophisticated and complicated area of software design; however, the solution in the tool development will be evolving as problems are resolved. Participation in this group will ensure that best-practice design patterns are utilized and combined repeatably. Web3D Consortium membership is required for participation.

NMCI Port: Identify expected user operational hardware/software configurations and determine the most effective means for deploying (or making available) the software, data, and analytical capabilities into those environments (e.g., NMCI environment).

Configuration Control: Maintain the code base under configuration management and prepare software installation packages.

Remainder FY05 Activities: For FY2005, complete subtask 4.1.1 and commence subtasks 4.1.2 plus all follow-on subtasks.

Computing Cluster: Obtain and configure hardware and software for a high performance cluster environment to support rapid execution of AT/FP analyses.

3.0 PHASE II PARTNER CONTRIBUTIONS

3.1 INTRODUCTION

This effort was only able to get underway with the contributions of each and every component/partner/researcher assigned to this project. Listed in this section are abstracts from two theses, written by Naval Officers who conducted research on vital pieces of this project, and the Phase II final reports generated from work performed in support by contributing partners.

3.2 NAVAL POSTGRADUATE SCHOOL

3.2.1 LCDR Travis Rauch, USN Thesis

Abstract from (Rauch 2006):

Visualizing operations environments in three dimensions (3D) supports the warfighter's ability to make rapid, well-informed decisions by presenting complex systems in a naturalistic, integrated display format. Unfortunately, constructing these environments is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability of personnel with minimal graphics experience to create virtual environments quickly and accurately by leveraging data-driven customization of content from model archives with the data available in the command center. Practical 3D visualization depends on standardized scene autogeneration.

The Extensible 3D (X3D) Graphics family of specifications is approved by the International Standards Organization (ISO) as the Web-based format for the interchange and rendering of 3D scenes. Previous work has demonstrated that an archive of X3D scenes, such as the Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) library, can be used to autogenerate sophisticated 3D tactical environments. Assembling and making sense of the data necessary to autogenerate a 3D environment requires context and good documentation, best accomplished through metadata. Metadata also supports data-centric, component-based design; key philosophies in promoting interoperability of networked applications. Coupled with recent developments in X3D, enhanced features of the Savage X3D Model archives are now sufficiently mature to support rapid generation of tactical environments.

This thesis proposes an XML metadata standard to collect and organize the information necessary to create and populate a tactical 3D virtual environment: the Savage Modeling and Analysis Language (SMAL). The logical extension of a well designed standard is the ability to cross the boundaries of usage, allowing simulators to share data with command and control (C2) suites and mission

planning tools based on the construction of a virtual scene. SMAL provides the informational “glue” necessary to perform tactical modeling, simulation, and analysis using networked, physics-based X3D virtual environments.

3.2.2 LT Pat Sullivan, USN Thesis

Abstract from (Sullivan 2006):

The individuals charged with the task of planning, developing and implementing force protection measures both at the unit and installation level must consider numerous factors in formulating the best defensive posture. Currently, force protection professionals utilize multiple sources of information regarding capabilities of systems that are available, and combine that knowledge with the requirements of their installation to create an overall plan. A crucial element missing from this process is the ability to determine, prior to system procurement, the most effective combination of systems and employment for a wide range of possible terrorist attack scenarios.

This thesis is inspired by the work done by James Harney, LT, USN: “Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships Using X3D Graphics and Agent-Based Simulation” (Harney 2003). The thesis will expand the Anti-Terrorism Force Protection Tool developed during the original thesis by including the capability of testing force protection measures in multiple scenarios by utilizing models of force protection equipment and forces, virtual worlds of existing naval facilities, and terrorist agents that exhibit intent and behavioral characteristics which can test the effectiveness of the force protection equipment used.

The result of this work is a scalable and repeatable methodology for generating large-scale, agent-based simulations for AT/FP problem domains providing 3D visualization, report generation, and statistical analysis.

3.3 SOUND & SEA TECHNOLOGIES (S&ST)

3.3.1 Overview of S&ST Contributions

The Naval Facilities Engineering Service Center (NFESC) is responsible for planning and executing a comprehensive Anti-Terrorism/Force Protection (ATFP) Ashore program to develop, evaluate, deploy and sustain components, subsystems and systems to reduce the vulnerability of naval facilities and assets worldwide to attack by terrorists.

As part of the planning tasks, the ATFP Ashore System Engineering analysts must consider many existing and proposed systems, installed in a multitude of different configurations, at in excess of 200 Naval installations, each with a specific set of requirements applied over a wide range of threat scenarios. The tasks include analysis, trade studies and requirements definition, with the objectives of providing recommendations for system configurations for procurement, measures of performance (MOP), and assessment of the effectiveness (MOE) of planned and installed systems, development of concepts of operation and support of training for ATFP response forces.

It is simply not practical to conduct these analyses on a purely empirical basis, by installing and trying different combinations of equipment and systems. It is necessary to obtain site-specific data on the physical conditions, local threats and resultant vulnerabilities of each site before applying either material or non-material solutions. However, it is equally obvious that it is not cost-effective to develop a set of material options and methods for every site by physical trial and error alone. A better, more cost-effective approach for Systems Engineering analysis is required.

A widely accepted methodology for dealing with complex systems is to use a Modeling and Simulation (M&S) approach. M&S tools can provide a tool set that allows the user - from the analyst to the civilian administrator to the military operator – to assemble simplified representations of an actual system that allows an understanding of underlying relationships among sensors, combatants and their behaviors, all against the backdrop of 3D, immersive displays of actual locations such as a harbor and surrounding areas.

Sound and Sea Technology personnel conducted a survey (Garrood 2006) of available M&S software within the DoD community and prepared a companion white paper (Garrood, et al 2006) on the results. These papers are available as appendices to this report.

The requirements for the CY2006 software development from the white paper are presented elsewhere in this report; however the conclusions and recommendations are repeated here and served as guidance for the development work summarized in this report.

Quoting from the summary section of the (Garrood 2006) white paper:

There are a number of M&S software systems that have been developed for similar, but limited, physical security programs. A review of extant anti-terrorism M&S software has shown that the technical approach in the ongoing program of M&S development at the Naval Postgraduate School (NPS) is the only M&S effort that is on a clear path to meet the ATFP Ashore program requirements. Beginning with the waterside, NPS has demonstrated substantial progress toward extending their work to the required capabilities over the entire ATFP Ashore spectrum of terrestrial, air and waterside threats, systems and components.

The Naval Postgraduate School M&S tool has a robust, open-source architecture embodied in software and resources tailored to the ATFP Ashore requirements. It has been extended to become the evaluation and assessment tool required by the ATFP Ashore Systems Engineering team to perform most of the analytic studies necessary for defining ATFP Ashore system risk, vulnerability and consequence assessments for Naval installations.

In short, the NPS M&S effort is both necessary and sufficient to meet the ATFP Ashore program requirements.

Therefore, the current project in place with NPS should be extended and accelerated to meet all the requirements of the tool set and ensure that documentation and user training keep pace with tool development.

3.3.2 S&ST Team

Dallas Meggit

Dennis Garrood

Mario Pozzo

Roger Christiansen

<http://www.soundandsea.com>

3.4 ANIVIZA

3.4.1 Overview

Aniviza, Inc. provided technical support, implementation, and improvement of Viskit and related projects, including cluster operations, designs of experiments, physically based sensor implementations with test scenarios, scenario entities, geometric utilities, user interfaces and package installers.

3.4.2 Activities Performed

Improvements to Gridkit, the cluster component to Viskit's experimental design feature included transitioning from an interpreted to a compiled runtime to support more complex parameterization of SMAL entities. This required some redesign of the Gridkit boot loader, which sets up the runtime environment for each node run in order to import compiled classes from Viskit XML entities and assemblies as opposed to translating these on-the-fly with the interpreter. The benefits of using the Java™ Beanshell interpreter were mainly that it simplified re-designation of the class pool for each replication without having to reload a new Java™ Classloader each time; this saved implementation complexity as well as runtime startup overhead. However, the interpreter has limitations as to how many parameters a class can consume at about 1/10th that of compiled code, so the Beanshell interpreter was replaced with a standard Javac compiler so more complex entities can now be loaded on the grid.

Other considerations for analysis of scenarios were addressed, including whether the current design of experiments (DOE) graphical user interface (GUI) was sufficient to set up parameters for a nearly orthogonal Latin hyper-sample (Chioppa 2002). The current mode takes parameters into linearized differentials where the user sets high and low endpoints, but does not consider the use of other shaped random variates as parameters; part of the difficulty with varying input parameters is that an event-graph agent designer may not see a particular variate as needing to be non-constant, however it may be selected in the Viskit DoE panel to be an independent variate, on the other hand, some parameters may have been already been designated as non-constant variates for the entity by use of an explicit randomizer for the parameter by the designer. If the designer of the entity was correct, then all one should want to do is set some ranges in the Viskit Assembly Editor, run either locally or on the grid, and get the same if not

faster results. A specialized random number can now be used to create nearly orthogonal Latin hyper-samples from ordinary Viskit assemblies for cluster runs without interaction with the Viskit DoE.

Further work needs to be performed on the DOE panel to selectively display potential parameters (based on entity listener patterns) and also to verify and validate proper implementation of the Latin hyper-sample algorithm.

Part of the Diskit package includes support for mover and sensor kinematics. Sensors were designed to be pluggable into any scenario; however, all sensors so far have been simple enter/exit ranges. This is insufficient for analysis that requires more accurate assumptions about, for instance, detectability by sonar in a shallow harbor, where such ranges may vary, or be intermittent. At the core of a sonar model is some Figure of Merit (FOM) for how much signal is returned in a meaningful way to a particular operator, which then describes the range of the sensor at that exact moment. If the FOM is positive, then a detection has happened; likewise if it goes negative after being previously detected, then it is undetected (i.e. contact is lost). To maximize the number of possible sensor configurations for sampling the sonar, e.g. side-scanning vs. omni-directional, or skyward for radar, a geometrically based scan approximation is utilized. This algorithm estimates the attenuated transmission loss of a sonar ping, also optimizes scheduling for detection tests, pings, depending upon its most maximum range and desired scan shape, while still being a drop-in replacement for any simpler existing Diskit sensor.

One component to the FOM calculation is noise sampled at a location. In the MultiLRATLSonar for example, noise is parameterized by a random variate. In the sample test case, a normal random variate is used, however, it is possible to take geo-referenced sample data of noise using an InterpolatedXYVariate, which calculates a noise level based upon an interpolation of closest sample data. The design of the InterpolatedXYVariate intended for fast updates to the dataset, so that noise from moving objects could be simulated inexpensively. See Appendix E.

Another component to the FOM calculation is target strength (TS). Target strength depends on the relative rotation of the target and its size, which can now be accessed via SMAL. The current implementation, however, is assuming constant TS as more work was needed to easily obtain the rotation of a Mover.

Adding vehicle kinematics and propagation-based sensor predictions to a DES system is highly unusual (and perhaps unique). These capabilities greatly improve the fidelity of the most critical interactions being modeled.

Getting Viskit updates out to a user base has so far been via a Concurrent Versioning System (CVS). This is insufficient for deployment to end users. A new auto-installer builder has been incorporated into the regular Viskit distribution tasks. Previous installers have either relied upon using commercial freeware that have become obsolete, or upon being bundled with other installers. Viskit now has an open-source auto-installer builder as part of the build process.

3.4.3 Aniviza Team

Rick Goldberg, CEO

<http://www.aniviza.com>

3.5 DALY REALISM

3.5.1 Overview

Daly Realism is an Internet Consulting company that provides business solutions to its clients. Its focus is on secure web sites that deliver the right user experience. The company uses the latest web technologies, including interactive 3D graphics to complete its solutions. The principal is a professional member of the Web3D Consortium.

3.5.2 Previous Work

Daly Realism has worked with NPS on a code and documentation review of the SAVAGE library. All of the X3D code was reviewed to determine compliance with the X3D specification. Code that was not compliant was identified and the changes needed to make it compliant were documented. The documentation structure and navigation was reviewed and improvements were recommended and implemented.

3.5.3 Scope of Work

Daly Realism's Statement of Work (SoW) identified one major task, one minor task, and a number of small project-administrative tasks. The major task was to develop application documentation for the user-facing applications (SAVAGE Studio and Viskit). The minor task was to develop training and instructional materials. The project-administrative tasks include monthly progress reports, regular meeting participation, and maintain a list of future improvements.

During the project kickoff meeting, it was determined by Sound & Sea Technology, NPS, and NFCSE that providing the materials listed below satisfied the SoW for the major and minor tasks

- a. Viskit help, including a tutorial covering the various uses of Viskit
- b. SAVAGE Studio, including a tutorial covering the various uses of SAVAGE Studio
- c. Frequently Asked Questions (FAQ) and answers for high-level questions on the project and application

3.5.4 Activities Performed

All of the application help was developed in HTML & CSS to work with the embedded JavaHelp™ system. Viskit help comprises 90 cross-referenced help pages in standard help hierarchal format with screen captures to illustrate the processes. Included in the 90 pages are 16 pages of tutorials showing the step-by-step use of Viskit. The help for SAVAGE Studio comprises 39 cross-referenced help pages in standard help hierarchal format with screen captures to illustrate the processes. Included in the 39 pages are 5 pages of tutorials showing the step-by-step use of SavageStudio. The help for Viskit and SavageStudio is included in all distributions of the applications.

3.5.5 Deliverables Completed

The following items were delivered on this project.

- a. Viskit Help – 90 HTML pages plus 55 images
- b. SAVAGE Studio Help – 39 HTML pages plus 20 images
- c. FAQ – 1 HTML page
- d. Monthly status reports – 7 reports
- e. Weekly meeting attendance – for duration of project
- f. Occasional program review meetings at NPS – 2 trips to Monterey
- g. NPS Open House and ATFP tutorial – 1 trip to Monterey
- h. Contributions to the Final Report

3.5.6 Recommendations for Future Work

The applications for this project are built and distributed using the Open Source model. That model has been shown to be highly responsive to user questions and bug reports if the user and developer communities are large enough. If clients are willing to pay for support than the size of those communities is not an issue (for those clients). To help build the community the following suggestions are offered:

ATFP web site: SourceForge is an excellent location for developers and downloads of installation packages; however, it does not provide for the necessary capabilities to support a web-based user community. The ATFP site needs to offer a threaded discussion or email list that is open to all users. The site can also host the FAQ, on-line help, tutorial, and other use information.

Context-Sensitive Help: Providing help to the user that is sensitive to the user's current situation is very useful for improved usability. Ideally the help that is provided is akin to an electronic expert in that the help sub-system is completely aware of the steps the user has already completed and what the user needs to do next.

Facility Building Tool: Daly Realism does not believe that a user-oriented tool that builds a port facility should be a recommendation for future work. On occasions a tool of this

type was discussed, but dismissed as beyond the scope of this project. ATFP is used by professional doing critical risk-assessment of vital facilities. Allowing non-experts to build the port runs the risk of severely incorrect decisions being made based on incorrect simulations. Developing a tool that would allow a sufficiently trained individual to modify features of an existing port is useful. This can allow quick response to changes in the local port environment, such as dredging, new or changed piers or berths, or changes to breakwaters.

3.5.7 Daly Realism Team

All work on this project was performed by Leonard Daly, President.

<http://www.realism.com>

3.6 MEDIA MACHINES

3.6.1 Overview

Media Machines is a leading provider of technology and solutions for real-time 3D communication. The company is spearheading the development of standards and technologies that lower the barrier of entry and total cost of ownership for developing real-time, rich media applications. The company believes that 3D graphics, integrated with rich media sources such as hypertext, audio and video, represents the next step in human-computer interaction. The company is an organizational member of the Web3D Consortium.

3.6.2 Scope of Work

Visualization capabilities of the ATFP Harbor Security Visualization and Analysis Tool are being provided using the Extensible 3D Graphics (X3D) international standard for 3D graphics on the Web. Enhancement of X3D authoring tools is an essential part of the development work in order to facilitate development of the visualizations.

3.6.3 Activities Performed

X3D Tool Updates: Updated the Flux Studio (formerly Vizx3D) Authoring Tool to add support for Amendment 1 to the X3D Specification, specifically [aligning with overall Project SOW para 4.1.7]:

Script and Proto Editing. The tool had the capability to Import, Export, and Edit Scripts and Protos in the Vizx3D Beta. The remaining tasks were:

- To support editing of the Proto Body using the native Vizx3D nodes and Render the Proto Body within Vizx3D. The Proto Body consisted of Generic Nodes which provided the user the ability to edit any of the Fields of the Nodes, but there was no Node Specific GUI that Vizx3D provided for the Native Vizx3D Nodes. Also, these Generic Nodes were not rendered inside of Vizx3D. The company also provided a GUI that allows users to specify the IS/Connect constructs within the Proto Body.
- Provided export for Protos and Scripts in the Classic VRML encoding.

Provided support of IMPORT and EXPORT statements.

- For export, provided a GUI that supports specification of which Nodes in the scene will be exported via the Export Statement.
- For import, for each Inline Node, provided support for looking into the Inlined Content (if present) to generate an Import Statement that corresponds to the Export statement found in the Inlined Content.

Provided support for Import, Export, Render Elevation Grid, and Triangle Set Nodes.

Fixed Import and Export of Extrusion Node (including Rendering within Flux).

Provided support for Import, Export, Render of new CAD component Nodes and provided edit capability for Quad Geometry Nodes.

Provided support for Cubic Environment Maps to include generation of Maps within Vizx3D, similar to the current support for generating Spherical Environment Maps. Included Rendering within Flux and support for rendering within Vizx3D.

3.6.4 Media Machines AT/FP Team

Tony Parisi, President and CEO

Keith Victor, Vice President of Engineering

<http://www.mediamachines.com>

3.7 OKINO COMPUTER GRAPHICS

3.7.1 Overview

In one sentence, Okino has allocated **all** of its primary programming resources to the X3D project from March through to September 2006, well beyond what we could ideally allocate to one single project. In real figures, we had to steal development time from 2 other key projects (XAML and v5 release cycle) in order to achieve our lofty goals for the X3D project. Relatively speaking, the 100 hours of invoiced work time covers one weekend of work, and just touches on some of the overall time allocated to this March-September sub-project.

We are highly motivated and (in essence) fanatical about getting our bidirectional X3D + Classic VRML pipeline 100% perfected. Starting in 1999 VRML2 turned out to be one of our most important conversion pipelines for our PolyTrans product, and hence we likewise see the need and reason to allocate all of our programming resources to X3D + Classic VRML during 2005 and 2006. We decided, from a business standpoint, to allocate 2005 and 2006 to the development, completion and refinement of this X3D project. We are now at that completion point as of September 26th 2006.

3.7.2 Primary Task Groups

In basic terms, our time allocation has been spent on these distinct portions of the X3D project:

1) Addition of new capabilities (import + export):

- 3D point sets (including new internal PolyTrans + NuGraf UI display, options and save/load)
- 3D polylines (including new internal PolyTrans + NuGraf UI display, options and save/load)
- Classic VRML support
- ZLIB compressed output capabilities for VRML1, VRML2 and X3D and Inventor2

- Removal and refinement of all known problems, on 32-bit and 64-bit platforms.
- Migration to using the faster Microsoft MSXML v6

2) Coordination of the release of OpenVRML v0.16 from Braden McDaniel. Okino developed the X3D + Classic VRML code for OpenVRML v0.15 during 2005 (our primary task during 2005). Braden then took all of our code and integrated it "his way" into v0.15 from January through to March 2006. He also fixed all known bugs in the toolkit. From March to September 2006 we then had to re-integrate his new v0.16 initial release BACK into our code line, and then make all changes necessary to make the new codeline compatible to what we consider "proper X3D support", as it had existed in the Okino version of OpenVRML back in December 2005. The sheer, ultra complexity of the OpenVRML toolkit, and its 1 hour compile + link times, made this the most horrendous project ever taken on at Okino, bar none. As of September 26th we finally believe the v0.16 toolkit is commercially viable for our customers to use. Okino does not release any software until we can personally guarantee a software solution - at this point in the evolution of our own X3D converters + OpenVRML combo; we believe the end to end solution is finally working nicely.

3) Porting of OpenVRML to the Visual Studio VC2005 compiler. OpenVRML is a very heavy templated toolkit and hence refused to compile, far less run, on VC8. This was a real thorn in our side all during this project. We would rather have kept with VC7.1 but in order to even think of porting to 64-bit we needed to first get the codeline running on VC8 Win32. The VC7.1 version of v0.16 was working by May 17 2006, and the VC8 version by September 23rd 2006.

4) Porting of Okino X3D + Classic VRML + VRML2 code to 64-bit architecture. This was tied in directly to the initial port of all the code (import and export) to VC8. The exporters were functional by Siggraph 2006. However, the first successful execution of the 64-bit importer code (with no known crashes) only occurred on September 23rd.

5) Re-engineering of our various installers to support the new VC8 + 32/64-bit versions of the VRML2+X3D importer. This turned out to be a real fiasco. A simple task turning into a complete task. Our new code requires MSXML v6 for 64-bit. The MSXML installer requires

Windows Installer v3.1. Windows Installers v3.1 requires that Installshield install it before it executes our main Okino installer. However, a "chicken before the egg" problem occurs with these dependencies. In the end we finally opted to use MSXMLv4 on 32-bits and MSXML v6 on 64-bits, both of which have been proven to be functional. This may allow us to use the stock Microsoft installers without requiring the end users to upgrade their entire operating system first. This is just one classic problem which has caused the X3D project to consume almost every hour of our development time this summer.

3.7.3 Main Development Achievements

- 3D point sets (including new internal PolyTrans + NuGraf UI display, options and save/load)
- 3D polylines (including new internal PolyTrans + NuGraf UI display, options and save/load)
- Classic VRML support
- ZLIB compressed output capabilities for VRML1, VRML2 and X3D and Inventor2
- Migration over to an "Okino qualified" OpenVRML v0.16 toolkit, which officially includes all of the Okino X3D and Classic VRML extensions + bug fixes.
- Final release version of our X3D+VRML2+Classic-VRML import and export converters using the first stable release of OpenVRML v0.16
- Porting of code to VC8 32-bit and 64-bit.
- Modified installers to support this new version of Okino X3D+VRML support.
- Migration to using the faster Microsoft MSXML v6

3.7.4 Okino AT/FP Team

Robert Lansdale, CEO

Andrew Grieve

<http://www.okino.com>

3.8 PLANET 9 STUDIOS

3.8.1 Overview

Planet 9 Studios is a 3D products and content company focused on providing real business solutions for the Internet. The company has produced over 250 virtual worlds for a variety of applications such as marketing, advertising, product visualization, training, architectural simulation, military simulation and entertainment. It is constantly incubating new software products for companies and helping them to reach market. The company is an Organization Member of the Web3D Consortium.

3.8.2 Previous Work

Planet 9 Studios has worked with the US Navy for several years, developing high-fidelity models and software systems for a variety of needs. This includes development of world-class models as part of the Anti-Terrorism/Force Protection (AT/FP) team.

In October 2004, the company was tasked with the development of a fully textured Extensible 3D (X3D) model of the Al-Basrah Oil Terminal (ABOT) and the surrounding area for the purpose of evaluating scenarios in the protection from surface threats. This project was originally known as Gas and Oil Platforms (GOPLATS).

In April 2005, the company was contracted to develop two additional X3D models for the AT/FP effort, during Phase I of this project. These were high-fidelity models of two Navy facilities in Washington State, specifically NAVSTA Bremerton and NAVMAG Indian Island. These models included geo-referenced terrain as well as a number of photo-realistic shore-side 3D buildings and structures. Several models of watercraft were also developed as part of this deliverable. These combined models were used as the primary test-bed scenarios in the continuing development of the AT/FP software.

3.8.3 Scope of Work

In February 2006, Planet 9 Studios received a scope of work (SOW) for Phase II of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives. The scope was an order of magnitude greater than Phase I, which included revisions of Phase I deliverables, as well as the development of new X3D models of waterside buildings and terrain at both Pearl Harbor and Port Hueneme. Additionally, the SOW included supporting roles in the tasks of analysis tool development, training and documentation, as well as the necessary team coordination and management.

However, the full level of funding required to meet every task described in the SOW was not available. Planet 9 Studios was awarded a Purchase Order for approximately 48% of the total amount quoted as being required to complete all of these tasks. The company worked with the customer and the team to prioritize the tasks, and made a determination that the following items would be undertaken with the available funds:

- Pearl Harbor – construction of X3D model for Waterside Security Visualization
- On-site Training – provide one on-site training in the use of AT/FP software
- Team Coordination – project management, reporting, and conference attendance

Those items removed from the list of expected deliverables were the following:

- Indian Island and Bremerton – enhancement of Phase I modeling
- Port Hueneme – construction of X3D model for Waterside Security Visualization
- Analysis Tool – contribution to design, development, testing, and demonstration
- Off-site Training – provide off-site training in the use of AT/FP software

As the project proceeded, some additional task items were requested by the customer. Where possible these requests were accommodated by making non-critical adjustments to the requirements of the existing tasks. These are called out below.

3.8.4 Activities Performed

Pearl Harbor Waterside Security Visualization (Bug #989) – Planet 9 Studios was given notice to proceed on March 7, 2006. Prior to this, the company had received the prerequisite technical drawings and other source data from Naval Facilities Engineering Command (NAVFAC). A company photographer was dispatched to Pearl Harbor to take pictures of the buildings and structures located in the area of interest. These photos would serve as both a visual reference and as the source for texture maps to be applied to the 3D building models, in order to give them a photo-realistic appearance. (Data collection was identified as Bug #978)

A geo-referenced X3D terrain model of Oahu was developed using 10-meter SDTS Digital Elevation Model (DEM) source files. This was draped with color-corrected 30-meter Land Remote-Sensing Satellite (LANDSAT) imagery. To the immediate area around Pearl Harbor, topographic data was integrated into the greater terrain model. This higher-resolution area was draped with 1-meter imagery originating from Space Imaging, and included the Naval Station, Naval Shipyard, SUBASE, FISC, and Ford Island facilities. The resulting geometry was then optimized for real-time rendering, including the addition of Levels of Detail (LOD) for increased efficiency, and geo-referenced within the Universal Transverse Mercator (UTM) coordinate system. The rendering system, Xj3D, required these to be converted to a terrain specialized form of the X3D node, called “GeoLOD”.

Upon the terrain were constructed X3D models of the majority of Navy buildings visible from the water, as well as piers and wharves attached to the facilities. The location and footprint of each structure was extracted from the provided computer-aided drafting (CAD) files, which had been aligned with the geo-referenced terrain. The footprints were extruded and modified to create a representational geometric model of each structure. To these were applied texture maps derived from the location photographs, thereby resulting in a photo-realistic model for use in the AT/FP simulation software. For ease of management, the buildings were grouped into a limited number of separate files based upon location. It was determined that providing each individual structure in a separate file, as originally requested, would have been too burdensome given the amount of work that would have been required. This allowed for the assigned funds to be redirected to other additional tasks.

Similar to the topographic terrain, bathymetric geometry was developed using vector-based CAD files as the source data. These describe the depth of the harbor floor with contours at regular intervals. From these, a 3D mesh was created and optimized. However, this model was not integrated into the final scene. Higher resolution bathymetric data in Digital Nautical Chart (DNC) format, supplied by the Naval Undersea Warfare Center (NUWC), was able to be loaded directly into Xj3D by the team at Naval Postgraduate School (NPS).

A series of Aids to Navigation (ATON) X3D PROTO models was produced to allow the virtual waterways to be populated with charted marks as they are in the actual world. The selection included various buoys, lights, daybeacons, and range lights that can now be positioned and oriented at precise locations within a given scene. When applicable, certain models contain switches to allow assignment of a few specific attributes such as port (green) vs. starboard (red), light on vs. light off, and brightness of light glow. An X3D scene was laid out using these ATON models to reflect the actual lay out of marks at Pearl Harbor, according to GIS data describing the exact location and identity of these. This data was obtained from the public website of the National Oceanic & Atmospheric Administration (NOAA).

As an addendum to the SOW (via Bug #1009), Planet 9 Studios was asked create a 2D compass rose for general direction finding, to be displayed in a Heads-Up Display (HUD) manner over any given X3D scene. This consisted of a texture mapped compass face which rotated in direct correlation with the orientation of the user's viewpoint. While the visual components were complete with basic functionality in place, the file was not finished as of this report. There remains an issue of gimbal lock in the compass rotation, due to the fact that it is tied to the viewpont orientation via the X3D "ProximitySensor" node. The visual artifact is not noticeable when the viewpoint is parallel to the ground, but becomes apparent when viewpoint is pitched up or down. A quaternion approach may need to be implemented in order to alleviate this issue.

The request was also made (via Bug #537) of Planet 9 Studios to provide a copy of its previously existing X3D PROTO of the Port Security Barrier (PSB) for release into open source, thereby allowing it to be freely used and modified. A portion of the funding was redirected from other tasks to provide compensation for this transfer of intellectual property. The PSB model was reviewed and released, after minor refinements.

Finally, Planet 9 Studios coordinated with project partners to achieve necessary results. The company worked closely with Yumetech not only to integrate the X3D models into its Xj3D real-time rendering environment, but also to sort out various technical issues, including geo-location, naming conventions, headers, LODs, metadata, lighting, and bug identification. To a lesser extent Planet 9 Studios also worked with Daly Realism, providing some minor assistance with the Help System they developed for the software.

Training and Documentation (Bug #986) – Planet 9 Studios was originally asked to provide a hands-on tutorial targeted towards prospective force protection officers and users of the AT/FP software, with an emphasis on more advanced, analyst type levels of technical experience. Based on expected changes in the prospective audience, the focus of the tutorial was changed to include a more general overview of the code and data structures that the system runs on. The objective was to give prospective users a good idea of the underlying system and allow more advanced users a glimpse into the ease with which the system can be extended.

The process involved collaborating with LT Pat Sullivan of the Naval Postgraduate School (NPS) on basic VisKit examples, as well as obtaining a simple DES example used to illustrate the basic principles underlying the simulation system. The two example scenes illustrate the most simple event graphs and assemblies possible to run the system with. Most of the material in the AT/FP software tutorial slides was generated from these examples.

The first milestone was the initial draft of the tutorial, delivered on September 12, 2006. The second milestone was the delivery of the second version of the tutorial, delivered on August 1, 2006. The primary deliverable was the tutorial itself, which was made available via the Planet 9 Studio website as well as presented in person at the August 7, 2006 AT/FP software tutorial session at the MOVES Open House, fulfilling our supporting role in developing the project documentation. The tutorial was developed using the open-standards based s5 presentation system.

Future directions for work should primarily include feedback from a broader array of potential users. Future versions of the tutorial will be more hands on and comprehensive. Delivery of this sort of tutorial can play a critical role in detecting and repairing usability issues.

Team Coordination and Management – Planet 9 Studios began the project by contributing to the Plan of Action and Milestones (POA&M). The regular administration of this

project included weekly teleconferences with the team, with occasional project review meetings onsite at the Naval Postgraduate School (NPS) in Monterey, CA. Technical issues were reported, assigned, and tracked using Bugzilla software. Detailed status reports were submitted after the end of each month, as well as contributions to this final report. Additionally, Planet 9 Studios was asked to participate in a selection of professional conferences during the period of performance.

Representatives of the company attended the Anti-Terrorism/Force Protection (AT/FP) Ashore Modeling and Simulation (M&S) Workshop at NPS on May 9-11, 2006, an informational forum of M&S professionals working in the service of naval installation security. David Colleen, CEO, Planet 9 Studios, gave a presentation entitled, “3D Geospatial Data Interfaces & Tools” which discussed the various methodologies used by the company to provide solutions to its customers within the M&S market.

For the MOVES Open House, held at NPS on August 8-10, 2006, Planet 9 Studios was again in attendance. A tutorial for the AT/FP software was held on the previous day during which Dan Ancona, Planet 9 Studios, presented the instructional “Port Security Simulation with SAVAGE Studio”, demonstrating fundamentals of the software with examples and code. Christian Greuel, Planet 9 Studios, presented a high-level overview of the production process for creating geo-referenced models, entitled “Building Geo-Registered X3D - Port & Harbor Models... Accurately Located”. The company also participated in the Demo Night by showcasing several examples of work that have been completed for the various phases of the AT/FP project.

3.8.5 Deliverables Completed

In the course of its performance of the contract for Phase II of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives, Planet 9 Studios completed and delivered the following items:

- Input to the Plan of Action and Milestones (POA&M)
- Geo-referenced Buildings and Terrain, X3D models with texture maps:
 - Pearl Harbor / Oahu terrain, including GeoLODs, and piers & wharves

- NAVSTA/SUBASE/FISC buildings
 - Naval Shipyard buildings
 - Ford Island buildings
 - Extra buildings (Lochs)
 - Ford Island Bridge
- Aids to Navigation, X3D PROTO models with texture maps:
 - Daybeacon
 - Lighted Buoy
 - Light Post
 - Light
 - Range Light
 - Danger Daybeacon (non-PROTO)
- Aids to Navigation, Sample Layouts, X3D models:
 - Pearl Harbor Navigation Aids (Geo-referenced)
 - Navigation Aids Example (Generic example)
- Buoys, X3D models with texture maps:
 - Marker Buoy
 - Mooring Buoy
- Port Security Barrier, X3D PROTO model, released as open source
- Compass Rose, X3D PROTO and example, VRML format (incomplete code)
- AT/FP software installation script
- Slide-sets from AT/FP Ashore M&S Workshop presentations
- Slide-sets from AT/FP software tutorial presentation
- Findings from Savage / SavageDefense archives file verification

- Monthly Status Reports
- Contributions to Final Report

3.8.6 Recommendations for Future Work

Much progress has been made in the first two phases of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives effort. To build upon this success, the following enhancements are suggested.

Items removed from Phase II – The project would benefit from attending to those tasks which were not able to be addressed within the allotted funding for Phase II. These include:

- Indian Island and Bremerton – enhancement of Phase I modeling
 - increased fidelity of terrain imagery
 - addition of more site-specific buildings
 - inclusion of foliage
 - other
- Port Hueneme – construction of X3D model for Waterside Security Visualization
- Analysis Tool – contribution to design, development, testing, and demonstration
- Off-site Training – provide off-site training in the use of AT/FP software

Conversion of existing ports – Planet 9 Studios has previously developed a variety of US Navy specific 3D content, to which the company has maintained ownership of the intellectual property rights. Approximately half of this data exists in X3D format, while the other half is in the older VRML97 format. So that these models might achieve the widest possible use, it is suggested that they are 1) geo-referenced, 2) converted to X3D format, if applicable, and 3) moved into the realm of open source, to be served from the Savage and SavageDefense (FOUO) X3D Archives. The models to which this currently applies are of the following locations:

- Al-Basrah Oil Terminal (ABOT)
- Friday Harbor, WA (civilian)
- MCAS Miramar

- NAS North Island (rough)
- NAVMAG Indian Island (earlier work)
- NAVSTA Norfolk (coming soon)
- Pearl Harbor
 - Terrain
 - Ford Island Buildings
 - Ford Island Bridge
 - Arizona Memorial
- Port Hueneme (rough)
- SUBASE Bangor
 - Marginal Wharf
 - Delta/Drydock
 - Service Pier
 - Explosives Handling Wharf
- Washington Navy Yard
- Yokosuka, Japan

Creation of new ports – In addition to these pre-existing assets, Planet 9 Studios would be able to provide any number of new X3D port facilities. This could include any or all real-world facilities, either CONUS or OCONUS, that would benefit from utilizing the AT/FP visualization system. It could also include generic, non-specific ports that might be used for examples and software training scenarios.

Aids to Navigation (ATON) – Create a comprehensive system of X3D PROTO models with attributes adherent to International Hydrographic Organization (IHO) S-57 (<http://www.caris.com/s-57>). This standard, prepared by the IHO Committee on Hydrographic Requirements for Information Systems (CHRIS), is for the coding and exchange of hydrographic digital data. The X3D PROTOS would include defined options such as numeric designation,

types of sounds, and precise light flashing characteristics. The system would include a more complete selection of ATON types (e.g. Cans/Nuns, Mileboards, Warning Markers, etc). See Appendix G.

Automation – The following production items would benefit from development of automation processes:

- Automatic editing of X3D files to include multiple alternate URLs for textures maps
- Automate editing of X3D files to include multiple alternate URLs for ExternProto files under the “ExterProtoDeclare” node
- Automate separation of individual geometries in the scene (e.g., platforms, buildings, piers, etc.) into separate files that can be inlined and geo-positioned into other scenes and not tied specifically to the subject scene.

Miscellaneous – In addition to the above, the following tasks are also suggested as future work for this continuing effort:

- Addition of visual effects, explosions, gun fire, time of day, weather.
- Fix all older X3D headers, e.g. ABOT, Bremerton, Indian Island, boats
- Move Compass Rose to HUD layer
- Write a “How to Build an X3D City Model” white paper.

3.8.7 Planet 9 Studios AT/FP Team

David Colleen, Chief Executive Officer

Christian Greuel, Director of Art & Production

Dan Ancona, Software Engineer

Danny Lee, 3D Artist

Carlos Newcomb, 3D Artist

Ken Rhee, 3D Artist

Alberto Rodriguez, Office Manager

<http://www.planet9.com>

3.8 SONALYSTS

Noteworthy contributions to the sonar model design were provided by Sonalysts, Inc. All design work was coordinated by NPS. See Appendix E for a rigorous design summary of sonar physics modeling.

3.8.1 Overview

For more than 25 years, Sonalysts has developed solutions in computer software design and implementation, telecommunications research and analysis, prototype development and manufacturing, multimedia design and editing, animation, intelligent training systems, weather products, commercial nuclear power safety and quality assurance, and naval systems analysis and operations research.

3.8.2 Progress to date:

Sonalysts researched unclassified sources for parameters and techniques appropriate for acoustically modeling shallow, noisy waters at high frequencies. Sonalysts provided Aniviza with a description of the sonar equation and value estimates for various parameters. Cylindrical spreading plus frequency dependent attenuation was selected to provide initial estimates of acoustic transmission loss. While quick to calculate and reasonably accurate, a more sophisticated alternative to represent transmission loss has also been considered. The Comprehensive Acoustic Simulation System (CASS) was studied as a more sophisticated alternative to the aforementioned cylindrical spreading plus frequency dependent attenuation. A CASS input stream was developed to estimate transmission loss in the vicinity of Port Townsend/Indian Island.

3.8.3 Recommendations for future work

Refinement of the spreading + attenuation model should be possible using CASS results as a guide. Furthermore, the CASS results themselves can be improved by using measured sound speed profiles, bathymetry, and bottom type for the various harbors of interest. Unclassified descriptions of appropriate sonar systems should also be sought.

3.8.4 Sonalysts Team

Margaret Bailey

Doug Nelson

<http://www.sonalysts.com>

3.9 YUMETECH

3.9.1 Progress to Date

At the end of this phase of project, Yumetech made significant progress on a number of key areas. As well as the development directly related to the ATFP simulation system, the company made significant updates to its Xj3D and Aviatrix toolkits. These changes significantly improved the performance of the software and made future changes to the software easier to incorporate. Yumetech also added a number of features to the ChefX3D toolkit to expand its functionality.

MOVES Institute members added invaluable input with regards to software bugs and implementation problems through regular telephone conference calls. Moreover, MOVES content provided useful material for testing the Xj3D source code. Yumetech was able to make adjustments to the code to correct the bugs discovered in these tests.

At the beginning of this project, simulation developers had to employ several separate and distinct software tools to generate a scenario. One of these components—a 3D modeling tool—typically requires a fairly expert user. At the end of this Phase, Yumetech has successfully created a fully integrated tool that allows a non-expert user to author a scenario using one of the pre-defined ports. Moreover, one can change all SMAL parameters and some agent specific simulation parameters and then can launch a 3D overview of a simulation and/or run the scenario for statistics analysis.

- a. Yumetech has accomplished the following during this project:
- b. Upgraded the ATFP 3D visualization software to Xj3D version 2.0.
- c. Upgraded the ATFP 3D visualization software to Aviatrix3D 2.0.
- d. Added the prototype 3D editing viewer.

- e. Provided zoom and pan capabilities in main view window.
- f. Developed an OrthoViewpoint for top-down photos of 3D scene.
- g. Completed the integration of the Pearl Harbor Model into the software.
- h. Added screenshots and conversion factors capabilities.
- i. Fixed Xj3D issues discovered with new Pearl Harbor model.
- j. Created an end-user install package to facilitate software installation.
- k. Re-architected ChefX3D to support new features
 - i. Added support for segment tools (Fence, Barrier).
 - ii. Added support for segment property panels.

Yumetech completed the following assigned tasks during this project:

- a. Port and Port Facility Modeling.
 - i. Indian Island and Bremerton, Washington Waterside Security Visualization.
 - ii. Pearl Harbor and Port Hueneme Waterside Security Visualization.
- b. Analysis Tool Development.
 - i. Integration with AUVW.
 - ii. GUI for Scenario Set-up.
 - iii. Configuration Control.
- c. Follow-on Requirements.

The Phase Two work required that certain components be completed before others. Because of task priorities and additional requirements arising from the development process, the following tasks were not completed in this phase:

- a. DNC Load/Display.
- b. Physics-Based Models (some work on this component was done, but the task is not completed).
- c. NMCI/IT-21.

3.9.2 Future Development

Yumetech sees the next phase of development on this project as the one that will bring the ATFP simulator to a fully deployable stage. As well as completing the tasks left from the Phase Two work, the company has identified a number of areas that would further enhance the usability and flexibility of this already invaluable tool. These areas are:

- a. Location set-up.
- b. Barrier representation.
- c. Property editor.
- d. Statistical/Visualization tool improvements.
- e. 3D viewer improvements.
- f. Automating the Integration of Savage Studio with the Savage Library.
- g. Environmental Effects.

3.9.2.1 Location Set-up

Currently, creating new scenario locations such as port facilities and military bases requires 3D modeling experts to create these models at a premium cost. Yumetech proposes the development of a set of modeling tools that will allow non-3D graphics experts to create their own scenario locations. These tools will make use of geo-spatial and satellite data as well as a model database that will allow end-users to easily create terrain models and place buildings, port facilities, and other assets with relative ease. This work can be integrated with the X3D Earth effort to facilitate this task.

Yumetech also sees the following components as key elements of this task:

- a. Fences—create a fence model/behavior.
- b. Checkpoints—Create a check point model/behavior.
- c. Building Authoring—Develop an easy interface to create simple buildings.

The Google *Sketchup* interface is a good example.

3.9.2.2 Barrier Representation

The current barrier models need further improvement to be effective in simulation assessment. The current barrier models need a SMAL representation. Moreover, the top-down visual representation of the 3D model needs significant improvement. Finally, the visual response of boat models should use real-time physics for realistic response characteristics.

3.9.2.3 Property Editor

Savage Studio needs to parse the Viskit event graph file and use that information to populate a new tab on the property editor panel. Currently, a restricted interface is in place to handle Patrol Zones. This interface needs to be generalized in order to make the system more useful.

Moreover, the current system parses an XML instance to develop the parameter space in the following manner:

```
<Communications channel="2" address="124.134.89.2"/>
```

Yumetech proposes that parsing a schema would greatly improve the utility of the property editor. Such a scheme would allow the addition of *appInfo* components for adding features such as Data Editors (e.g., File Dialog boxes). The *appInfo* tag identifies what special editor the system must use in a particular instance. Making this change would also allow the system to pull the allowed values to populate a combo box. A possible scheme example would look like the following:

```
<xs:element name="Channel">
  <xs:complexType mixed="false">
    <xs:attribute name="channel" type="xsd:integer" appInfo="NumberEditor">
    <xs:attribute name="addressed" type="xsd:string"
      appInfo="InternetAddressEditor">
  </xs:complexType>
</xs:element>
```

3.2.9.4 Statistical/Visualization Tool Improvements

The current system requires users to select entities to track in the statistics tool in Viskit. A better way to accomplish this task would be to allow users to select entities for analysis using Savage studio. This change would provide end-users with a more intuitive, graphical interface for this task.

The visualization tool for the review specific scenarios should allow end-users to select a specific run of a scenario and review it in the 3D viewer. This ability would allow users to examine and analyze anomalies in the statistical runs to determine weaknesses in the defense plan. This capability needs to be added to the current system.

The system would also benefit from a 2D View of the scenario. The system should provide 2D, so that top-down view of the scenario incidents can be viewed by the user. This ability might be accomplished using an orthographic view of the 3D scene.

3.9.2.5 3D Viewer Improvements

Yumetech has identified a number of areas to improve the usability of the 3D viewer. These include:

- a. Improving texture loading to improve performance on lower-end machines.
- b. Optimizing memory usage so larger areas can be modeled.
- c. Implementing a complete (MIL-STD)-2525A for Unit descriptor top-down view.
- d. Integrate X3D Binary generation and loading

Yumetech also believes that providing a tree view of the entities of a scene can make it easier to edit some worlds. This tree-view should allow deletion of entities. Moreover, selecting an entity should bring its parameters up for editing in the property editor.

Moreover, Yumetech believes that end-user utility can be significantly enhanced by employing a number of viewpoint interface enhancements. Better viewpoint selection can be created by assigning a keyboard interface for selecting high/med/low viewpoints. Usability can also be improved by implementing the ViewpointGroup node. Other viewpoint improvements include:

- a. Provide better options for selection.
- b. Provide an option for grid display and snap-to object.
- c. Finish implementation of undo/redo feature.
- d. Provide measurement features between two locations.

3.9.2.6 Automating the Integration of Savage Studio with the Savage Library

Yumetech has identified the improved integration of Savage Studio with the Savage X3D Archives as a key component of this phase of the project. This integration will facilitate the rapid deployment of new models and behaviors into the authoring tool, and will significantly improve the utility of this tool. An important task in this component is Savage/Savage Defense Automation; that is the automation of some tasks involved in maintaining Savage and Savage Defense libraries. These tasks include:

- a. Create a SMAL size checker to insure 3D models match simulation data.
- b. Multi-URL/fallback fill-in—fill-in URL fields with local and web fallbacks for URLs.
- c. Auto-generate top-down map view from the actual 3D model of the location.
- d. Auto-generate icons for Savage entities.

3.9.2.7 Day/Night/Weather Effects

The current system assumes clear weather in daytime conditions. Simulations would be greatly enhanced by implementing visual and simulation changes for difference in sensor performance in different conditions.

3.9.3 Yumetech, Inc. Team

Alan Hudson, President and CEO

Justin Couch, Software Engineer

Stephan Matsuba, CFO

<http://www.yumetech.com>

<http://www.xj3d.org>

4.0 SUMMARY AND CONCLUSIONS

4.1 M&S WORKSHOP CONCLUSIONS FROM (BRUTZMAN ET AL. 2006)

The primary intended outcome of the workshop was simply informed discussion of current Research and Development (R&D) projects. A secondary goal is continued broad sharing and promulgation of relevant technical information. Consensus of the attendees indicated that both goals were well achieved. There was also strong demand to conduct follow-on meetings to solidify the information exchange and opportunities for technical collaboration that were evident in the meetings. Presentations and demonstrations clearly showed that many tools had overlapping capabilities...

This event may have been the first time that a group of M&S practitioners performing related efforts in naval installation security have been brought together. This is a good start, and such efforts need to continue... Workshop participants saw a broad set of activities presented and demonstrated, cutting across a variety of exercises involving human participants, automated analysis, and real-world decision support. Common to all was importance of correlated 2D and 3D visualizations, representations of facilities and bases, modeling of sensors and environmental conditions, computing measures of interest, and modeling other aspects of the problem. Despite significant challenges, there are large opportunities for sharing of resources if practitioners are able to adopt specific standards and establish community contributions for interchange and reuse. Otherwise, massive overlap of human and monetary capital in duplicative efforts is likely to prevent ever-limited resources from establishing the more sophisticated models that are needed to solve complex real-world problems.

Specific conclusions gleaned from the Workshop include:

- The Workshop was exceptionally important and provided great value to the attendees.
- Free technical interchange without specific programmatic constraints allowed better exploration of potential technical capabilities.
- One or more sponsors with direct interest in these activities ought to participate to ensure continued progress.

- Further CNI and NAVFAC participation would be valuable.

Based on the clear group consensus which produced the workshop findings, we recommend the following actions be taken:

- Propose a special issue on Installation Security for the Journal of Homeland Security Affairs.
- Participants should consider attending the NPS MOVES Institute Open House tutorial session August 7, 2006, including project presentations and demonstrations during the Open House August 8-10.
- Establish the necessary organization to enable some form of working group to continue to address the issues raised in this Workshop.
- Plan a follow-on meeting to be held in 4-6 months.
- Candidate sponsors are requested to review this report, talk to participants and consider establishing a partnered activity by multiple sponsors in order to continue workshop efforts and technical collaboration.
- Various parties who have modeled certain ports ought to compare collected assets, evaluate what resources exist and determine how those resources might be best merged for broader use.

4.2 PHASE II CONCLUSIONS AND RECOMMENDATIONS

The reader is referred to the previous chapter (see Chapter 3) for detailed sub-contractor and partner conclusions extracted from their summary reports. The editors of this technical report feel that these are sufficient for the purposes of this report.

APPENDIX A. NAVAL INSTALLATION SECURITY MODELING AND SIMULATION WORKSHOP

ATTENDEE LIST:

LT Charles Adams	USS Bonhamme Richard
MAJ Darryl Ahner	US Army TRADOC Analysis Center, Monterey
Dan Ancona	Planet 9 Studios
Michael Ayling	Commander, Navy Installations Command
Doug Backes	US Pacific Fleet
Margaret Bailey	Sonalysts, Inc.
Manoj Bhardwaj	Sandia Laboratories
Curtis Blais	NPS MOVES
Joyce Borgen	Center for Asymmetric Warfare
Platt Brabner	21 st Century Systems, Inc.
Don Brutzman	Naval Postgraduate School MOVES Institute
Chris Carlson	Metron Corporation
David Colleen	Planet 9 Studios
Jeff Debrine	OPNAV N81
Alexandra Devisser	Naval Facilities Engineering Service Center
Ayman El-Swaify	Naval Facilities Engineering Command, Naval Information Technology Center
David Garvey	Boeing Phantom Works
Dennis Garrood	Sound & Sea Technologies
Rick Goldberg	Aniviza
Riley Goodin	Northrop Grumman Corporation
Chris Greuel	Planet 9 Studios
Christopher Guryan	ARES Corporation
Sean Harrigan	MOVES Institute
Alan Hudson	Yumetech
CDR John Inman	US Pacific Fleet
Ray Jakobovits	Metron Corporation
Dustin Kozal	21 st Century Systems, Inc.
Steve Kunkle	ARES Corporation
J. D. Miller	Johns Hopkins University Applied Physics Laboratory
Elizabeth Morin	Booz-Allen
Terry Norbraten	NPS MOVES
Robert Seligman	SAIC
LT Pat Sullivan	NPS MOVES
Pete Swan	MaK Technologies
Doris Turnage	US Army Engineer Research and Development Center
Jeffrey Weekley	NPS MOVES
Doug Weihnacht	Kinection
David Zeltzer	Northrop Grumman Corporation

NAVAL INSTALLATION SECURITY MODELING AND SIMULATION WORKSHOP AGENDA:

Workshop Day 1: May 9, 2006 Tuesday

0800 Registration

0830 Welcome to Monterey & NPS – Don Brutzman, NPS Principal Investigator

Workshop Objectives – Don Brutzman, NPS Principal Investigator

NPS Presentation and Demonstrations: 3D Modeling Applied to the AT/FP Problem and Interfacing to the Simulation for Data Mining – Don Brutzman and NPS Savage Team

1000 Break (Set up of NPS Wireless Guest Accounts)

1030 Invited Presentation: M&S Applied to AT/FP Harbor Defense Measures of Effectiveness and Measures of Performance
– Chris Carlson and Ray Jakobovits, Metron Corporation

1130 Lunch

1300 Invited Presentation: Application of the AVERT Model
– Christopher Guryan and Steve Kunkle, ARES Corporation

1400 Invited Presentation: Application of Simulation to Large-Scale Exercises
– Joyce Borgen, Center for Asymmetric Warfare

1430 Break

1500 Inserted Presentation: Graduate Education for Homeland Defense and Security – Dr. Paul Stockton, NPS Director, Center for Homeland Defense

1515 Invited Presentation: Using M&S Tools to Simulate Terrorist Attacks
– Doris Turnage, US Army Engineer Research and Development Center (ERDC), and MAJ Darryl Ahner, TRAC-Monterey

1630 Open Source and Open Standards for Long-term Project Success: Lessons from 3D Model Management
- Don Brutzman, NPS Principal Investigator

1700 Workshop Day 1 Summary and Wrap-up
– Don Brutzman, NPS Principal Investigator

1800 Social Hour and Dinner (Hula's in Monterey)

Workshop Day 2: May 10, 2006 Wednesday

0800 Session 2 Agenda and Objectives

– Don Brutzman, NPS Principal Investigator

0815 Invited Presentation: Physics Based Modeling and AT/FP Ashore M&S – Margaret Bailey, Sonalysts

0900 Invited Presentation: 3D Geospatial Data Interfaces and Tools –David Colleen, Planet 9 Studios

1000 Break

1030 Invited Presentation: Shipboard Area Protection Systems – Platt Brabner, 21st Century Systems, Inc.

1100 Invited Presentation: "GIS Central" NAVFAC's Approach to Centrally Hosting and Delivering the Navy's GeoReadiness Repository – Ayman El-Swaify, Naval Facilities Command, Naval Information Technology Center (NAVFAC NITC) (via conference call)

1200 Lunch

1330 Invited Presentation: Open Source Discrete Event "Extend" and Open Source, System Dynamics "Vensim" Efforts
- David Garvey, Boeing

1400 Survey of Additional Naval Installation Security Related Modeling and Simulation Efforts
- Dennis Garrood, Sound and Sea Technologies (S&ST)

1500 Break

1530 Invited Presentation: Tactical Decision-Making Training for Force Protection – Pete Swan, MaK Technologies

1630 Workshop Critique and Go-Forward Discussion – Don Brutzman, NPS Principal Investigator (Moderator)

1730 Workshop Day 2 and Public Attendee Session Concludes: Social Hour, NPS Trident Room (Basement of Herrmann Hall)



Workshop Day 3: May 11, 2006 Thursday (organizers only)

0700 Assemble report of Workshop presentations, findings and recommendations

1700 Workshop Complete

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. INSTALLATION/OPERATION TUTORIAL AT NPS MOVES OPEN HOUSE, AUGUST 7, 2006

	August 7-8, 2006 Tutorial Announcement: NPS Waterside Security (WSS) Anti-Terrorism / Force Protection (AT/FP) Analysis Tool	 THE MOVES INSTITUTE NAVAL POSTGRADUATE SCHOOL
---	--	--

Introduction: How can we plan for the defense of our nation's harbors and waterways in a way that shows us surprise scenarios that we never imagined? How do we graphically visualize the tactical execution of our force-protection plans? How do we compute statistical data to support findings of best-effort plans for our naval forces afloat? How do we use Java to model opponents, render entire harbors using interactive 3D graphics, and even run grid clusters to provide high-confidence analytic results? **This tutorial shows how.**

Eligibility: The tool and the instruction are open to all, but there will be a US Government only (For Official Use Only) session during the tutorial that will be closed to foreign attendees.

Location: The tutorial will be conducted in the Mechanical Engineering Auditorium, just outside the main doors to Watkins Hall, at the Naval Postgraduate School (NPS), Monterey, California.

Dates: The tutorial will run from 9am Monday August 7 through 11am Tuesday August 8 preceding the start of the Modeling, Virtual Environments, and Simulation (MOVES) Institute Open House.

Registration: To attend the tutorial, register for the MOVES Open House, scheduled for August 8-10, 2006 at the Naval Postgraduate School, Monterey, California.

Online registration: <http://gallery.bcentral.com/GID5061928DD447447-Conferences.aspx>

Maps and other information are available on the MOVES Institute web site: <http://www.nps.navy.mil/moves/>

Please **RSVP** your intention to attend this tutorial by sending e-mail to Terry Norbraten, MOVES Institute: tdnorbra@nps.edu

Tutorial Schedule

Tutorial Day 1: August 7, 2006 Monday, 0900-1700

0900 Project Overview: Associate Professor Don Brutzman

0930 Introduction to Harbor Modeling and Simulation using Agent-Based Tactics and Discrete Event Simulation (DES): LT Pat Sullivan

1030 Break

1045 Security-Assessment Demonstration for Bremerton & Pearl Harbor: LT Pat Sullivan

1230 Lunch

1330 Behavior Modeling using Viskit Event and Assembly Graphs: Dan Ancona

1415 2D/3D Scenario Generation using SavageStudio: Alan Hudson

1500 Break

1515 Building Geo-Registered X3D; Port & Harbor Models Accurately Located: Christian Greuel, Planet 9 Studios

1545 Design of Experiments (DOE) and Cluster Operations: Rick Goldberg

1615 Break

1630 Summary, Group Discussion, Conclusions and Next Steps: Don Brutzman

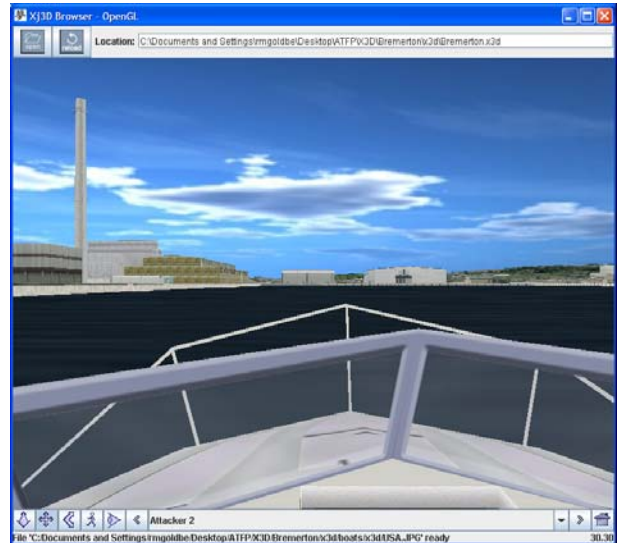
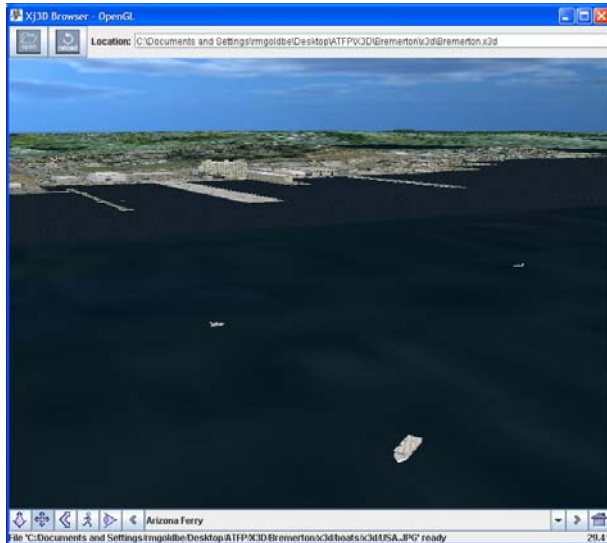
Tutorial Day 2: August 8, 2006 Tuesday, 0900-1200

***Hands-on Scenario Analysis Session, Q&A: to be held in the Ingersoll Building Room
366 (ING 366)***

APPENDIX C. AT/FP PROJECT FLYER

NPS Waterside Security (WSS) Anti-Terrorism / Force Protection (AT/FP) Project

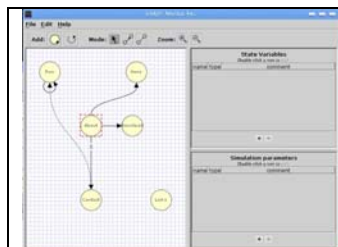
How can we plan for the defense of our nation's harbors and waterways in a way that shows us surprise scenarios that we never imagined? How do we graphically visualize the tactical execution of our force-protection plans? How do we compute statistical data to support findings of best-effort plans for our naval forces afloat? How do we use Java to model opponents, render entire harbors using interactive 3D graphics, and even run grid clusters to provide high-confidence analytic results? **This project shows how.**



The NPS waterside security project is a group effort. A top-notch team of government, industry and academic experts is using Java to produce a tactical application for use in defending national harbors and waterways. Scenarios can be autogenerated, viewed, analyzed, and manipulated by end users. Individual scenarios can be replayed from any vantage point using agent-driven X3D graphics models. Cluster-based computational assets use the Sun Grid Engine for massive replication of heavy-duty simulation scenarios, producing measures of effectiveness within statistically significant, analyst-specified confidence intervals.

Key technical features include:

- End-to-end open-source Java application, using Extensible Markup Language (XML) for all datasets
- ISO-Standard Extensible 3D Graphics (X3D) scenes using military model archives
- Xj3D open-source browser built with Java for OpenGL (JOGL) rendering speed
- Web-services queries for environmental forecasts and oceanographic-dataset updates
- Runs out-of-the-box on Windows, Linux, Mac OS X, Solaris SPARC, Solaris x86 operating systems, with NO Java recoding or X3D model adjustment required to achieve consistent operation throughout



In order to model realistic battle tactics for friendly forces and opponents, the waterside security project uses Viskit and Simkit, open-source Java 2™ packages built for visual creation of Discrete Event Simulation (DES) models. Simkit is used at NPS to make advanced simulation capabilities available to analysts, demonstrating meaningful real-world results. Simkit labs and tutorials are available online, downloadable at <https://diana.cs.nps.navy.mil/Simkit>.



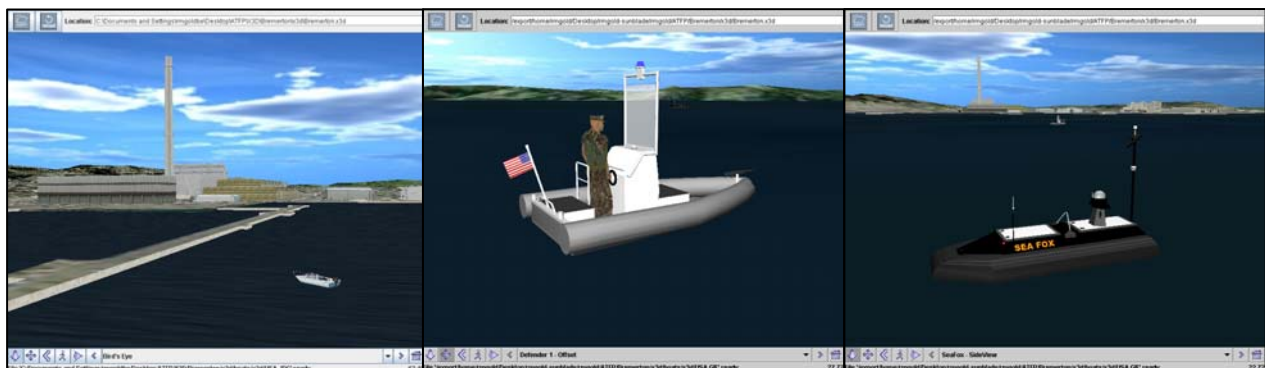
This work was publicly demonstrated 27 July 2005 as part of the Sun Microsystems JavaONE conference keynote session in San Francisco California. Seven thousand attendees in Moscone Center plus 250,000 remote attendees watching the webcast saw this agent-based 3D simulation running in real time. 900,000 lines of Java library code ran on a new Java Ultra 20 Solaris PC with exceptional performance. Viewable online at <http://java.sun.com/javaone/sf/2005> (view Webcasts, Day One, minute 1:23)

The production team putting all this work together includes the following Web3D Consortium partners:

- NPS MOVES Institute, Dr. Don Brutzman, <http://www.MovesInstitute.org> MOVES is currently partnering as a Sun Center-of-Excellence (COE) in Modeling and Simulation
- Planet 9 Studios, David Colleen, CEO, <http://www.planet9.com>
- Yumetech, Inc., Alan Hudson, CEO, <http://www.yumetech.com>
- Aniviza, Inc., Rick Goldberg, CEO, <http://www.aniviza.com>

Sponsors include:

- Naval Facilities Engineering Service Center (NFESC), <https://portal.navfac.navy.mil>
- Navy Modeling & Simulation Office (NMSO), <http://nmso.navy.mil>
- Web3D Consortium, <http://www.web3D.org>



Network connectivity is provided among multiple users via standards-based implementation of the IEEE Distributed Interactive Simulation (DIS) behavior protocol. This waterside security project will soon undergo initial user testing using naval officers at NPS, and then be tested using actual waterfront facilities. It is likely to provide significant improvements in the situational awareness and defensive posture of ships defending against terrorist attacks in port. The demonstrated scenario features friendly security forces defending against hostile entities in a simulated attack on Bremerton Washington harbor.



**May 2006 Project Update:
Modeling Pearl Harbor Waterfront**



THE MOVES INSTITUTE
NAVAL POSTGRADUATE SCHOOL

The NPS waterside-security team is currently demonstrating an updated set of software tools for modeling, simulation, visualization and analysis of harbor defense. Current capabilities are being tested using Extensible 3D (X3D) graphics models for Bremerton harbor, the ABOT Iraqi oil terminal and the Indian Island logistics pier. Pearl Harbor modeling is in progress. A tutorial course is simultaneously being developed in order to rapidly expose the combined efforts of 20 government and industry experts.


Three customers are envisioned for this integrated suite of analytic tools.

- *Port security investment:* how to best invest harbor-defense funds to maximize defense against risks
- *Port operations:* how to best deploy current assets on the water now for maximum defensive posture
- *Ship + harbor coordination:* help ships train sailors to be immediately effective upon entering port

This is an alpha-stage software release, being shown as a proof-of-concept tutorial to gain professional feedback. This work is also being demonstrated as part of an invitation-only industry workshop on modeling & simulation capabilities, hosted at NPS in Monterey California, 8-10 May 2006. All software and content models are being produced as open source. Use of open standards and unencumbered business-friendly licenses that protect government rights is expected to maximize potential growth and interoperability. Current work remains unclassified with access restrictions designated For Official Use Only (FOUO). Initial release is scheduled for 10 August 2006 at the NPS MOVES Open House. <https://www.MovesInstitute.org>

Risk models are connected and run in a complex adaptive multi-agent system. Simulations are either visualized “live” in real time on the desktop, or massively replicated for statistical analysis using low-cost computer clusters. Such Monte Carlo repetition lets analysts confidently determine whether defensive improvements are truly effective, using either commodity computers or high-performance computing assets.

In addition to tool development, the group is modeling the Pearl Harbor waterfront for in-depth risk analysis. The next major milestone will support automatic creation of detailed analyst-annotated risk-analysis reports.

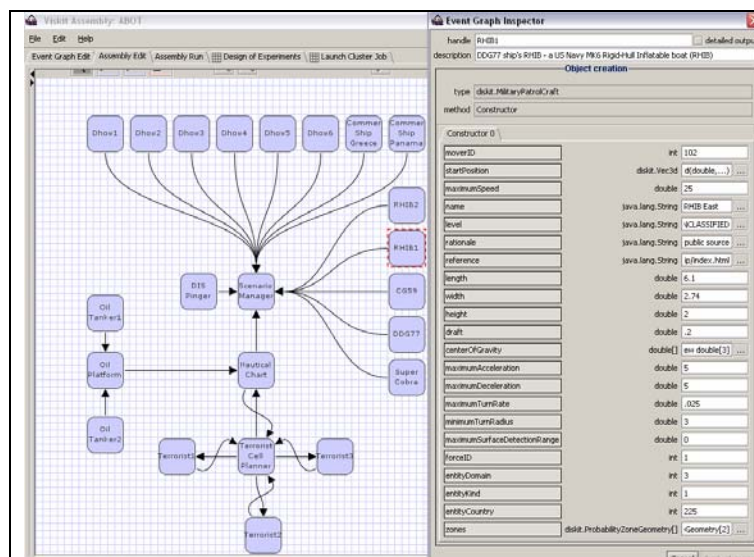
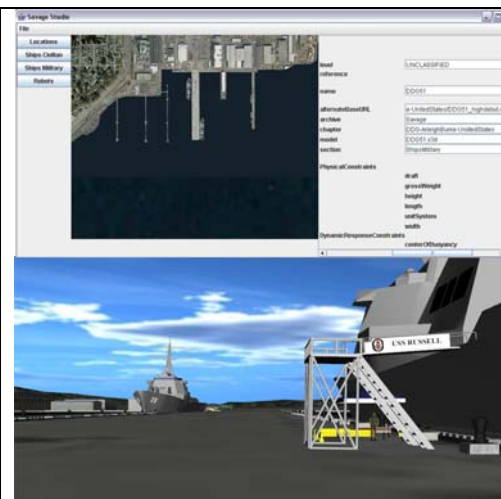
	<p>A Navy Lieutenant master's student from NPS and a professional photographer from Planet 9 Studios were given official port access to Navy facilities in Pearl Harbor, shooting 2,700 photographs in 4 days. These are being assembled into a high-fidelity X3D model of the Navy-controlled port.</p> <p>This real-world study evaluating Pearl Harbor is the first large-scale test of this application and research. Results are being geared to support analysts responsible for port security.</p> <p>This approach is repeatable for other ports and harbors, adding a tool-based suite of new capabilities to homeland defenders.</p>
---	--

The new Savage Studio authoring tool supports scenario creation with 2D/3D “pick and place” functionality. In other words, users can lay down a harbor-defense scenario by selecting ship assets from a menu, then drag and rotate ship icons into position. Simulations are then ready to start.

The just-published Savage Modeling and Analysis Language (SMAL) is used to embed well-defined metadata annotation capabilities within each model, suitable for further tool exposure using the Extensible Markup Language (XML).

Thus models “know what they are” and analysts merely need to customize capabilities to match the current scenario.

Use of ISO-standard X3D graphics means that the growing ship-model library can remain royalty free, open source, broadly interoperable, and approved for Navy use.



“Intelligent” adversaries are modeled using an intuitive flowchart-style tool that lays out tactics for “good guy” and “bad guy” behaviors using terms similar to those used by actual warfighters.

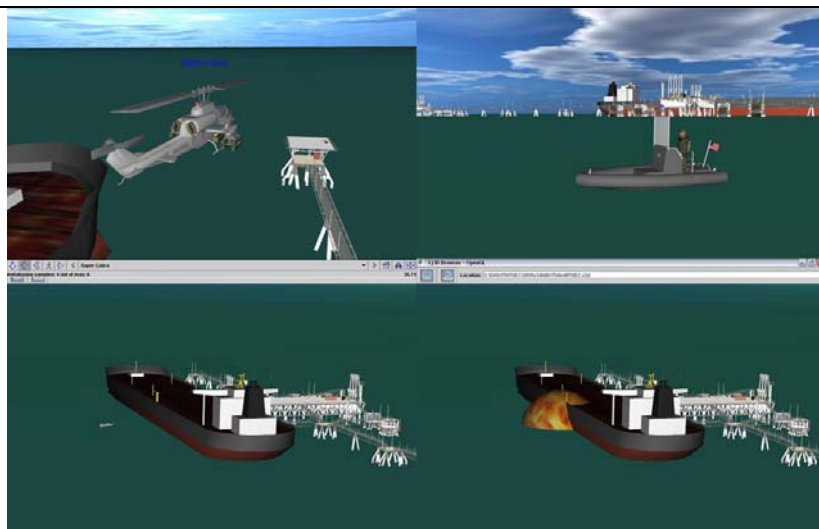
Libraries of tactical agent-based behaviors are being developed by active-duty Naval officers. Scenario creation and design of experiments for new ships and ports is thus simple and repeatable.

Terrorist models can also improve tactics and their probability of success over replications, exposing potential areas of vulnerability. Simulation insights thus enable analysts to recommend prioritized harbor improvements.

The ability to accurately reproduce simulated and actual scenarios is expected to increase user confidence that the tool provides satisfactory and dependable analysis results.

Navy and Coast Guard sailors can further use X3D playback to visualize their own roles in harbor defense (and even points of view for potential adversaries) as scenarios progress.

Project partners are documenting the process and software tools, allowing for repeatable approaches for all future work and easier integration by engineering and fleet users.



Inquiries are welcome. For further info, contact Don Brutzman (brutzman@nps.navy.mil), 1.831.656.2149.

APPENDIX D. MODELING AND SIMULATION WORKSHOP CD-ROM

U.S. Government agencies and their Contractors may obtain a copy of the Workshop CD via request to the NPS MOVES Institute.



Figure 4. May 2006 M&S Workshop CD Cover Image

Contact:

Terry Norbraten tdnorbra@nps.edu or tdnorbra@nps.navy.mil

700 Dyer Road, Wa-267
Naval Postgraduate School
Monterey, CA 93943-5001

Comm: +1.831.656.7593
DSN: 756.7593
Fax: x7599

THIS PAGE INTENTIONALLY LEFT BLANK

**APPENDIX E. DISKIT SENSOR AND MOVER DYNAMICS:
LOGARITHMICALLY RANGED ATTENUATED TRANSMISSION LOSS
SONAR**

**by
Rick Goldberg, Aniviza Inc.**

This document is intended to serve as background reading for implementers of Diskit Sensors, along with example material to demonstrate application of physically based models with Diskit, Viskit and Simkit

E.1 DISKIT SENSOR AND MOVER DYNAMICS

E.1.1 Overview

Diskit provides a set of base classes for defining 3D **Simkit**¹ Discrete Event Simulations (DES). This includes definitions for 3D entities, collision and sensor detection, weapons, target and munition adjudication, scenario management, and DIS protocol communications. **Diskit** is mostly based on 2D **Simkit** classes for the same, either directly by subclassing where possible, or in some cases by evolving **Simkit** utility classes, with some added features to enable networking and more rapid prototyping using **Viskit**. **Diskit** is part of the **Viskit** visual editor for **Simkit** distribution.

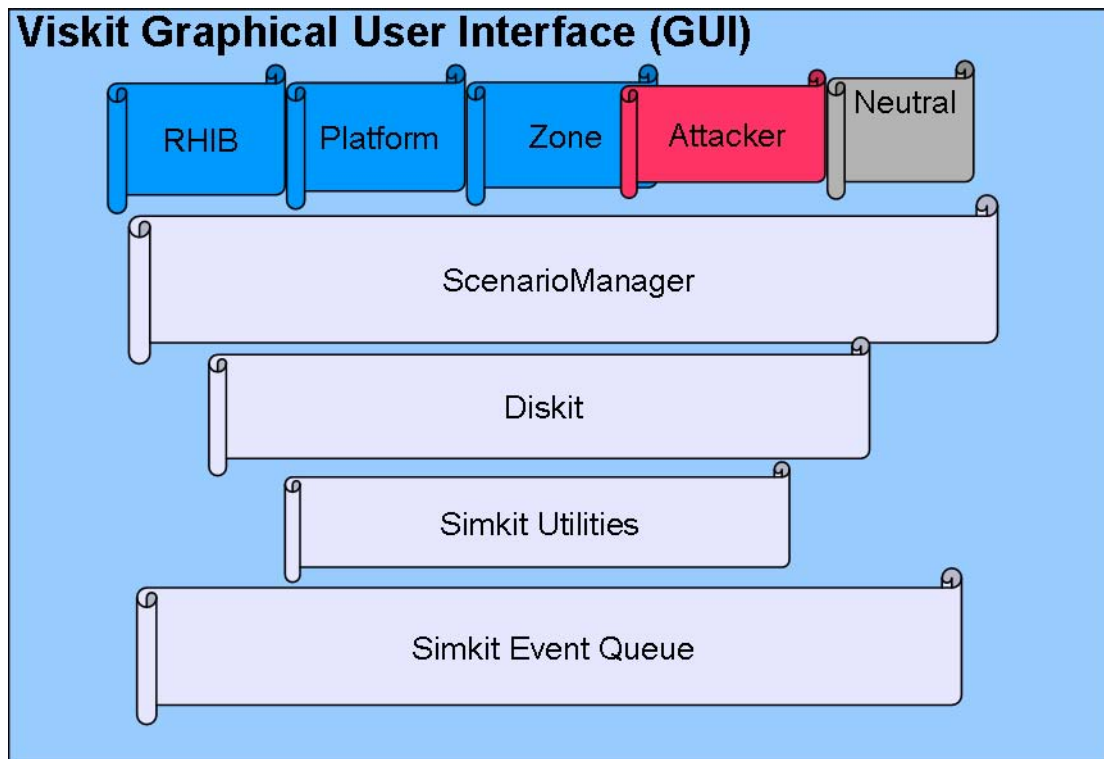


Figure 5. Simkit, Viskit, Diskit Platform Relationships

On top of **Diskit's** own base entity classes, sample classes for simulation of derivative behavior types such as patrol craft, bridge communications, neutrals, and terrorists are included.

¹ Simkit: see home page, <http://diana.gl.nps.navy.mil/Simkit/>

E.1.2 DISMover3D

The *DISMover3D* entity shown below listens for and generates a number of events that determine the 3D position and velocity for the point center of a moving object in space. More complex behaviors are based upon interposing filters, either by listening for these events or upon monitoring changes to the underlying state variables.

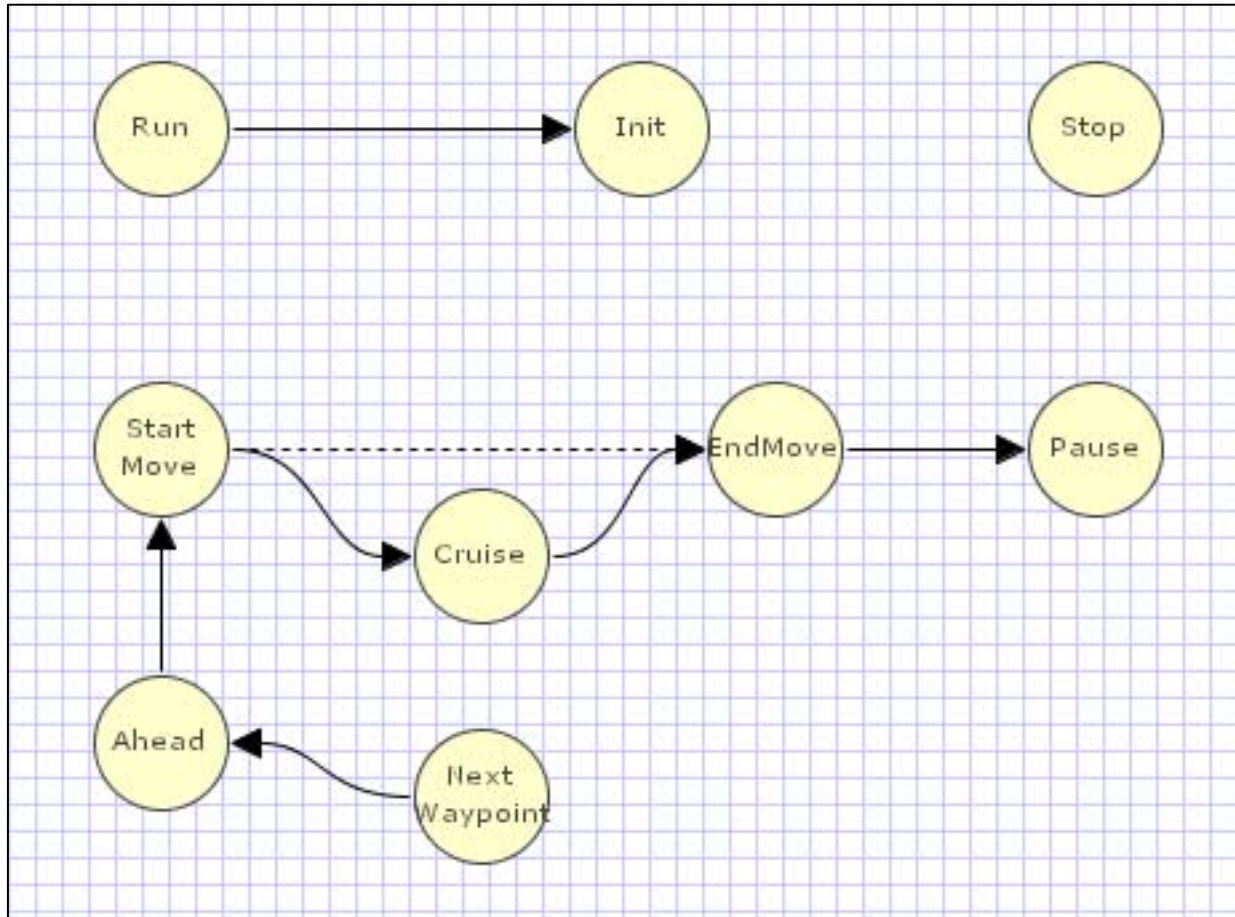


Figure 6. DISMover3D Entity in Event Graph Form

The base *DISMover3D* and all subclasses take at minimum the following parameter map:

Event graph parameters		
Double click a row to edit.		
name	type	description
originalStartPosition	diskit.Vec3d	Must be anything but the first waypoint.
maxSpeed	double	
moverID	int	
<div> <div>+</div> <div>-</div> </div>		

Figure 7. DISMover3D Parameters

Events ultimately cause some state to become altered. State variables for the *DISMover3D* cover speed, direction, start position, destination, movement type, and time of movement.

State Variables		
Double click a row to edit.		
name	type	description
velocity	diskit.Vec3d	
destination	diskit.Vec3d	
duration	double	
movementState	simkit.smdx.MovementState	
startPosition	diskit.Vec3d	
lastVelocity	diskit.Vec3d	
deltaTime	double	last checked time at start ...
cruisingSpeed	double	cruise speed
<div> <div>+</div> <div>-</div> </div>		

Figure 8. State Variables for the DISMover3D Entity

In the Event-Graph diagram above, in most cases it is sufficient to send a *StartMove* event to cause motion for the *DISMover3D*, which can also be reached by way of the *NextWaypoint* event. The *NextWaypoint* event itself is loaded with a *diskit.Vec3d* (Vector 3D) as the position value for the actual waypoint to go to next, as well as a double-precision valued *cruiseSpeed* for how fast to get there. Once sent and received, this event causes calculation of updated velocity information for the entity.

This eventually leads to the fundamental question, how is time represented and managed in a simulation? For the *DISMover3D*, time is a variable that can proceed in discrete arbitrary increments. It doesn't require passage of time duration in the real sense as it is just a calculation; when run, no more real time is needed to go from point A to B whether the distance great and speed small, or the other way around. This is great for analysis, for example, one would not want to wait a year to study a simulated year. For visualization and for DIS communication however, real time must be injected into the simulation run at regular intervals independently of all other operations.

This is accomplished by way of **Diskit's** *DISTimer* (formerly *DISPinger*). The *DISTimer* calculates the ratio of a given simulation time unit to real time delay, and causes the entire simulation to simply wait and pause the simulation thread of execution for the desired interval of real time, wake up long enough to send DIS network communication packets updating DIS listeners with current positions for all registered movers, fire any pending events, and go back to sleep for the next round. More about how entities are registered is outlined in section 4, **Scenario Manager**.

E.1.3 Sensors, Targets and Mediators

Diskit uses Simkit's base *Sensor* and *Target Mediator* architecture to schedule various type detection events in the queue stream. A *Sensor* generally consumes some volume of space and is located by the position of a *DISMover3D* that owns it. Conceptually, whenever any of the registered *Targets* changes velocity vector states, a calculation is done for each *Sensor* and *Target* to solve for any pending penetration and exit points, at some time in “the future” of the event queue, and at once canceling any such already-pending events that become invalid, thereby removing them from the queue. This is in contrast to fixed-timestep frame-by-frame collision

detection architectures, and allows for more complex behavior systems such as planned obstacle avoidance, for example.

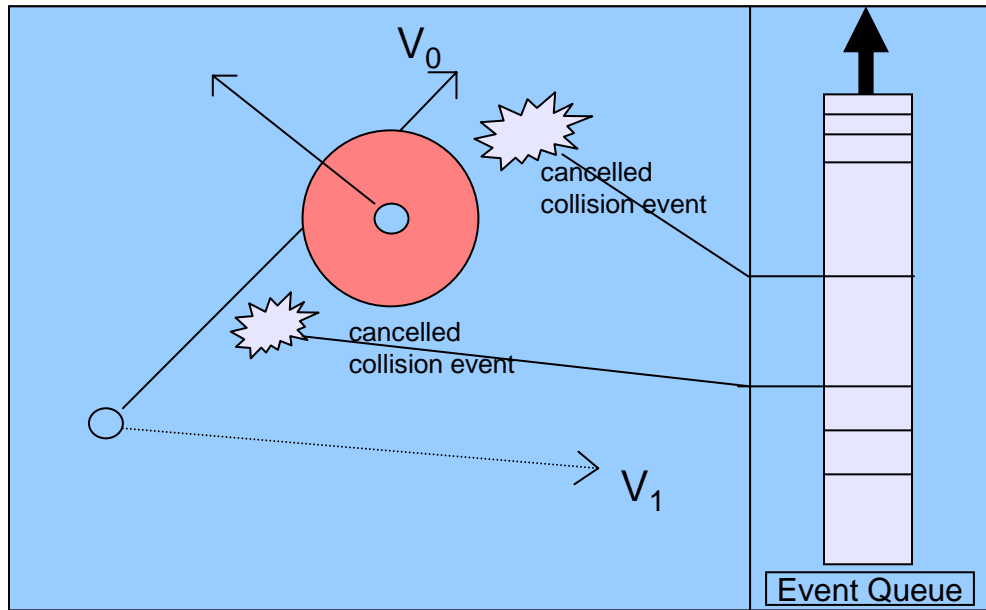


Figure 9. Diagram of Cancelling Invalid Pending Events Due to Change in Target Velocity Vector State

The base classes provided by **Diskit** that implement the *Sensor* interfaces are simplified moving sphere and ray intersections. However the architecture is intended to be extendable in such a way as to enable more complex detection algorithms and geometries. Once a *Target* enters or exits the range of a *Sensor* for instance, there may be additional logic to describe whether or not the mere *EnterRange* event is sufficient to cause a *Detection*, whereas the default *SphereCutterSensor* is always true for *EnterRange*. An alternate *Sensor* type can be registered with the *ScenarioManager* that may or may not schedule a *Detection* based upon probability parameters, or as well may check some other geometry.

The default intersection test for *SphereCutterSensor* only takes into account the point-ray intersection of a *DISMover3D*'s velocity vector versus the moving sphere boundary of a sensor, and does not factor in the *DISMover3D*'s own geometric bounds. By default, the *DISMover3D* has zero spatial bounds. Subclasses of course do usually know about their dimensions, and more advanced applications are required to specify higher fidelity models. But how is this accomplished?

First let's examine the algorithmic representation of simplified intersection math given by Diskit's *Intersector3D* class.

```
1  /*
2  * Intersector3D.java
3  *
4  * Created on November 18, 2004, 11:17 AM
5  * @author: Rick Goldberg
6  */
7
8  package diskit;
9
10 public class Intersector3D {
11
12     private Intersector3D() {
13     }
14
15     /** Solve moving sphere initersection */
16
17     public static double[] solve( Vec3d sensorLocation, Vec3d sensorVelocity,
18         double sensorRange, Vec3d targetLocation, Vec3d targetVelocity ) {
19
20         double px, py, pz, qx, qy, qz, vx, vy, vz, ux, uy, uz;
21         double[] times = new double[2];
22
23         //System.out.print("targetLocation :");
24         //targetLocation.print();
25
26         /* Let P = target position */
27         px = targetLocation.get(0);
28         py = targetLocation.get(1);
29         pz = targetLocation.get(2);
30
31         //System.out.print("sensorLocation :");
32         //sensorLocation.print();
33
34         /* Let Q = sensor position */
35         qx = sensorLocation.get(0);
36         qy = sensorLocation.get(1);
37         qz = sensorLocation.get(2);
38
39         //System.out.print("targetVelocity :");
40         //targetVelocity.print();
41
42         /* Let V = target velocity */
43         vx = targetVelocity.get(0);
44         vy = targetVelocity.get(1);
45         vz = targetVelocity.get(2);
46
47         //System.out.print("sensorVelocity :");
48         //sensorVelocity.print();
49
50         /* Let U = sensor velocity */
51         ux = sensorVelocity.get(0);
52         uy = sensorVelocity.get(1);
53         uz = sensorVelocity.get(2);
54
55         //System.out.println("sensorRange :"+sensorRange);
56
57         /* Solve the intersection of the ray from P through a sphere about Q */
58         /* First  $x^2 + y^2 + z^2 = R^2$ , but in cartesean coordinates, Q is */
59         /* also moving. Transformation of the coordinates to Q's own space */
60         /* can be simplified since there is no scale or rotation or skew or */
61         /* perspective, then Q is not moving and the values can be solved for*/
62         /* t by the quadratic equation, giving relative entry and exit times.*/
63
64         /* Step 1. Transform V to Q's coordinate system */
65
```

```

66     vx -= ux;
67     vy -= uy;
68     vz -= uz;
69
70     /* Step 2. Transofrom P to Q's coordinate system */
71
72     px -= qx;
73     py -= qy;
74     pz -= qz;
75
76     //System.out.println("px: "+px+" py: "+py+" pz: "+pz);
77     //System.out.println("vx: "+vx+" vy: "+vy+" vz: "+vz);
78
79     /* For the point S in Q space now represented by P, parametrically is */
80     /*  $Vt + P = S(t)$  , note V is also now in Q space but in practice save */
81     /* some runtime memory by reusing the variables but keeping the name. */
82     /* eg:
83     /*  $x(t) = (vx * t) + px$ ;
84     /*  $y(t) = (vy * t) + py$ ;
85     /*  $z(t) = (vz * t) + pz$ ; */
86     /* then  $x^2 + y^2 + z^2 = r^2$ 
87     /* expanding out, we see that we get something of the form
88     /*  $At^2 + Bt + C = 0$  and solve quadratically for time0 and time1 */
89
90
91     double a = (vx*vx + vy*vy + vz*vz);
92     double b = (2*(vx*px+vy*py+vz*pz));
93     double c = (px*px + py*py + pz*pz) - sensorRange*sensorRange;
94
95     //System.out.println("a: "+a+" b: "+b+" c: "+c);
96
97     double root = Math.sqrt(b*b - 4*a*c);
98
99     //System.out.println( "b^2 - 4ac : " + (b*b - 4*a*c));
100    //System.out.println( "root : " + root);
101
102    if ( root == Double.NaN || root == Double.POSITIVE_INFINITY || root ==
        Double.NEGATIVE_INFINITY ) {
103        times[0] = times[1] = root;
104    } else {
105        times[0] = (-b - root)/(2*a);
106        times[1] = (-b + root)/(2*a);
107    }
108
109    //System.out.println( "times: "+times[0]+" "+times[1] );
110
111    return times;
112 }
113 }

```

The results from the above code represent the relative times of penetration and exit, if they exist, and further show that if the *DISMover3D* was already inside the *Sensor* range, *times[0]* is negative, while *times[1]* being negative in this case would never happen since the ray does not terminate.

This is fine if the entity in question is small in comparison to the *Sensor* range. However if the *Sensor* range is small in comparison to the *Target*, for example if a visual contact during maneuvers, or if the bounds need to reflect a more accurate outline of collision between objects, some refinement may be required. **Diskit** optimizes the detection by breaking the problem down

into *EnterRange/ExitRange* events which are generalized by computing low-cost sphere intersection regions. From there a higher-fidelity model may be used to further see if an actual *Detection* happened. This factor is accounted for in the *Mediator* class for the particular *Sensor* type, as shown below in *SphereCutterMediator's EnterRange* event handler for example:

```

1 public void doEnterRange(Sensor sensor, Mover3D target) {
2     Mover3D contact = (Mover3D) contacts.get(target);
3     if (contact == null) {
4         contact = new Contact(target);
5         contacts.put(target, contact);
6     }
7     sensor.waitDelay("Detection", 0.0, new Object[] { sensor, contact } );
8 }

```

Two things are worth noting about the above code. A list of *Contacts* is maintained for all *Mover3D's* in range, which represent distinct positions for their *Mover3D's* which helps keep the entities' internal operations insulated from each other. Also note the *SphereCutterSensor* has a 0.0 time delay until it gets a *Detection* event; clearly, any algorithm for computing time delay can be inserted instead.

To answer the question at the beginning of this section, a more complex *Sensor* can be supplied by the user along with a *Mediator* for that *Sensor* that can calculate a *Detection/UnDetection* time between *EnterRange* and *ExitRange* events, given enough information from the *Sensor* and *Mover3D* of the target. Note that *DISMover3D* is an implementation of the **Diskit** *Mover3D* interface.

Below are the *Sensor* and *Mover3D* interfaces which can be used to build complex detection algorithms.

```

1 package diskit;
2
3 import java.util.Collection;
4 import simkit.SimEntity;
5 import simkit.smdx.MovementState;
6
7 /**
8  *
9  * @author ahbuss
10 */
11 public interface Sensor extends SimEntity {
12
13     public Vec3d getLocation();
14
15     public Vec3d getVelocity();
16
17     public MovementState getMovementState();
18
19     public double getMaxRange();
20 }

```

```

21     public void setMaxRange(double range);
22
23     public void doDetection(Sensor sensor, Mover3D contact);
24
25     public void doUnDetection(Sensor sensor, Mover3D contact);
26
27     public Collection getContacts();
28
29     public void setMover(Mover3D mover);
30
31     public Mover3D getMover();
32
33 }

```

```

1  /*
2  * Mover3D.java
3  *
4  * Created on October 6, 2004, 9:04 AM
5  */
6
7  package diskkit;
8
9  import simkit.SimEntity;
10 import simkit.smdx.MovementState;
11
12
13 public interface Mover3D extends SimEntity, Locatable3D {
14
15     public Vec3d getVelocity(); // dx,dy,dz
16
17     public double getCruiseSpeed();
18
19
20     public void setMaximumSpeed(double maxSpeed);
21     public double getMaximumSpeed();
22
23     public void setStartPosition(Vec3d sp); // start at xyz
24     public Vec3d getStartPosition();
25
26     // these two do basically the above two
27     public void setDestination(Vec3d d, double cs); // get there this fast
28     public void setDestination(Vec3d d); // get there max speed
29
30     public Vec3d getDestination();
31     // gets the location from the currentPosition,
32     public Vec3d getLocation();
33
34
35     public MovementState getMovementState();
36
37     public TacticalMode getTacticalMode();
38
39     public String getEntityType();
40
41     public void stop();
42
43     public void setMoverID(int id);
44
45     public int getMoverID();
46
47     public void setForceID(ForceID forceID);
48
49     public int getForceID();
50
51     public String getColor();
52
53     public diskkit.SMAL.SMAL getSMAL();
54 }

```

E.1.4 ScenarioManager

Putting it all together is the *ScenarioManager*, which handles registration of *Sensors* and *Targets* (*Locatable3D* components of *Mover3D*'s.) Registration is simply connecting *propertyChangeListener*s and *simEventListener*s, adding *Mediators* in an automated way. Entities are connected to the *ScenarioManager* so that the manager can send and receive events to each, and upon startup, anything that can be a *Target* or *Sensor* reports in. The *ScenarioManager* is a subclass of the *SensorTargetReferee*, which is where the actual *Intersector3D* is used from section 3. The *ScenarioManager* also handles any other kind of contact between arbitrary parties, such as *Munitions*, *Weapons*, *Impact*, *Escort* compliance, and synchronization with DIS packets.

The current implementation enables quick connection of *SphereCutterSensor*'s and available target types, however, it is not required to use the *ScenarioManager*'s interface to register a *Sensor*, *Target*, and *Mediator*, which can be done by calling upon static methods of the base *SensorTargetMediatorFactory*.

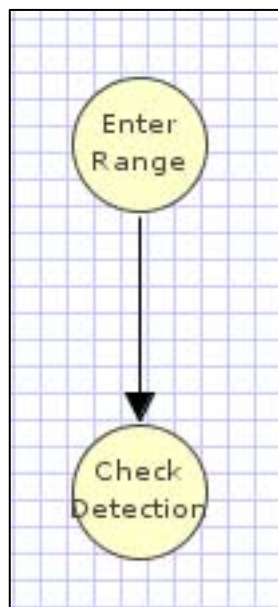


Figure 10. Simple *SonarMediator* Event Graph

The above example from *diskit.SonarMediator.xml* generates the glue code between **Diskit's** default *Sensor/Target/Mediator* pattern, and our customized *Sonar* sensor (see section 1.3). The main difference is that instead of an *EnterRange* event immediately scheduling a *Detection* event, the *Sonar Sensor* receives notification that it is time to start checking higher-fidelity logic encapsulated within the *Sonar's* event graph.

E.1.5 Example Multisectioned Log Range Attenuated Transmission Loss Sonar (MiltiLRATL)

With the above interfaces, we can now construct a general purpose sensor that simulates attenuation of a source signal as it propagates and calculates a *Figure of Merit (FOM)* for the *Detection* and *UnDetection* events. The sensor starts checking versus a *FOM* once the extreme boundary sphere has been penetrated, but only if the *Target* is within a visible section of the sensor. The example below shows a baffle zone or blind spot where no *FOM*-based detection can be checked. Other shapes and configurations are possible, for example a side mounted or an omni-directional sensor.

Baffled Sweep Approximation:

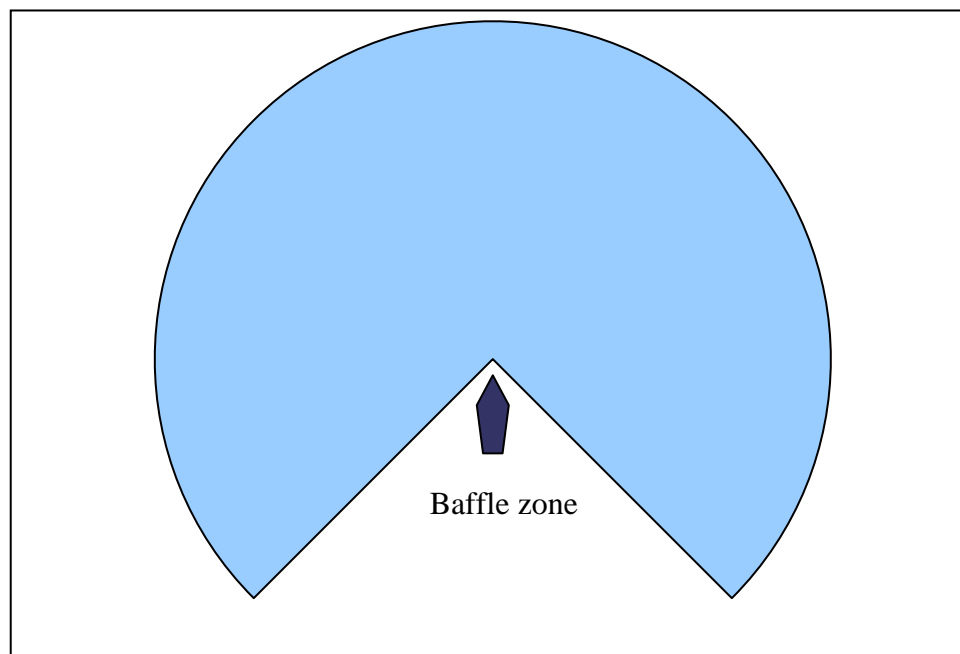


Figure 11. Looking Down on “Sweep”

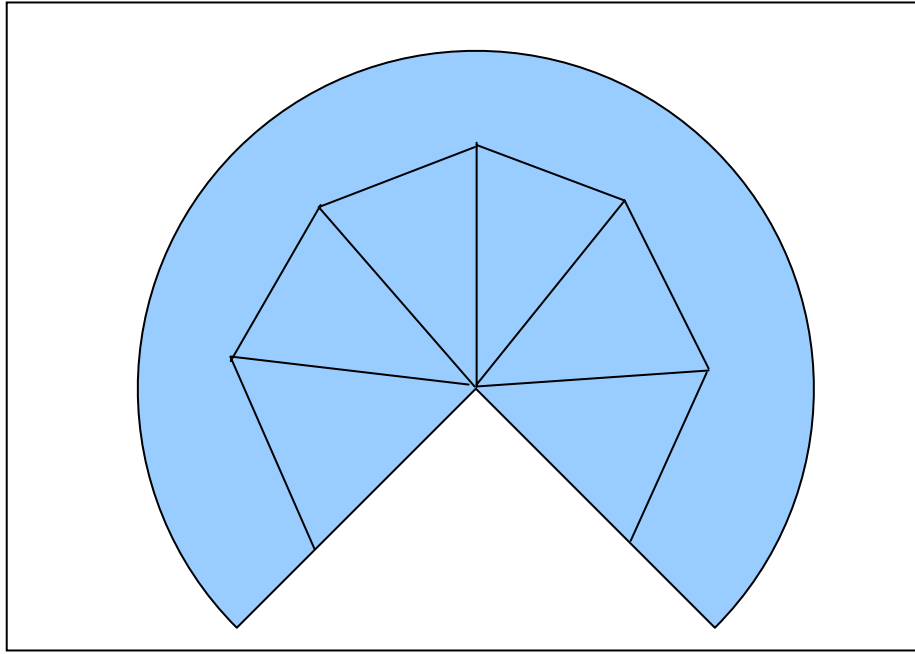


Figure 12. Baffle Geometry divided into triangular sections, viewed from above

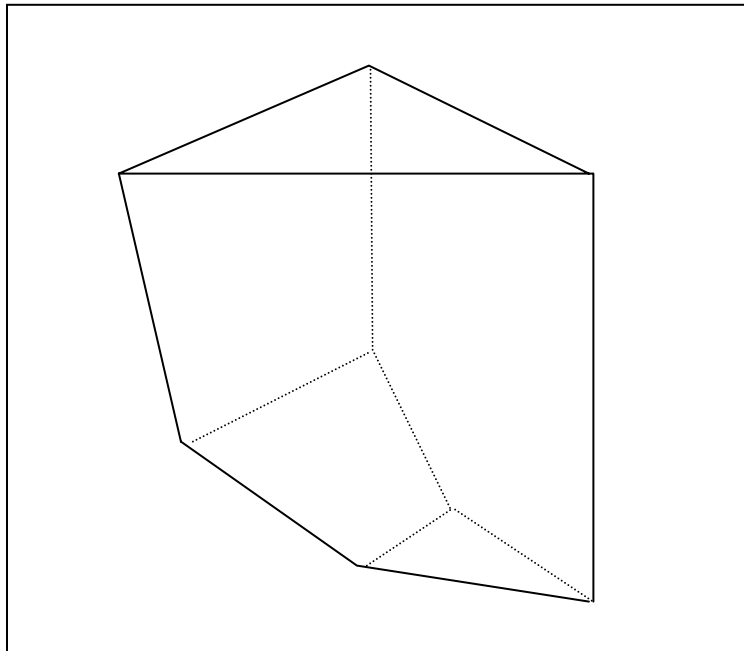


Figure 13. Side View of Approximation Geometry. First cut, "watermelon" slices.

At this point, we can generalize to more simplified forms knowing the potential intersections at each of the far corners.

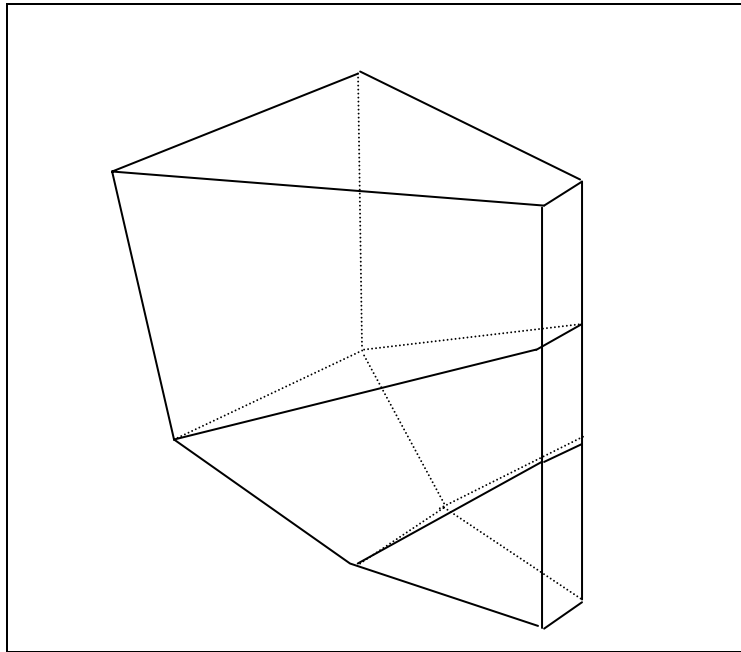


Figure 14. Further Simplification of Volume Geometry

After removing the bottom subsection, two six-sided volumes for this slice:

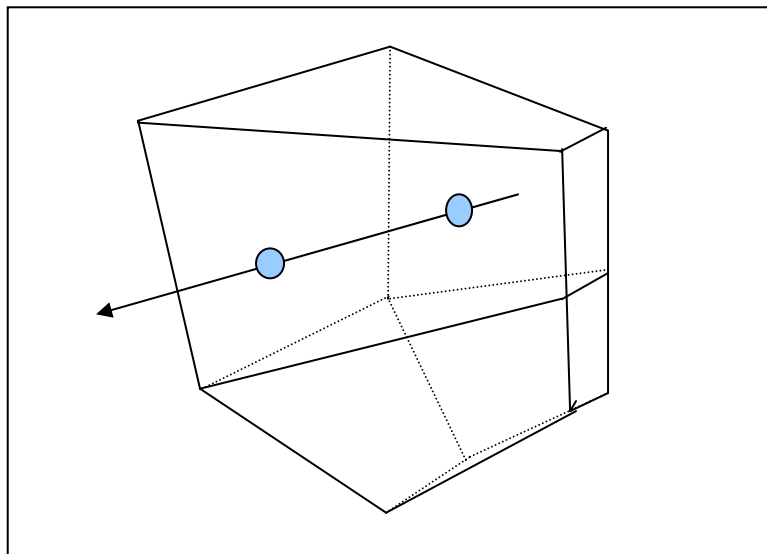


Figure 15. Final Geometry of Volume Space

The cross-section shown above represents only one of several that get checked; the sum total make up a complete sensor footprint for the sake of a single *Target's Detection/UnDetection* criteria.

A sensor composing these six-sided volumes can check each to see if it gets penetrated by the velocity vector of the *Target*. If each side of a volume is defined counter-clockwise, then the dot product is always negative for each normal vector taken as a dot product with a vector going from each vertex to the point in question (if that point is inside the volume). A computational optimization might be to calculate an interior point of the facet rather than check each vertex.

That's fine for seeing if a point is inside, but having detected a ray intersection in time, the probability of detection should be proportional to the time in the volume, and inversely proportional to the square of the distance of the points along the ray to the center of the beam.

Another technique for containment is to use energy field equations; conceptually taking the line integral around a function on a plane that contains a singularity yields a non-zero number. A Cauchy generating function for this in complex coordinates might be $\int_c 1/(z - z_0)^2 dz$.

The nice property is that the anti-derivatives in this case are bounded by simple line segments over a few additions and subtractions if numerically integrated. This generalizes to 3D using Greens and Stokes Theorems with a similar generator.

Once the ray projected from the moving target box vertex is determined to be within a scan volume, probability of detection could be approximated by $\int_{t_0}^{t_1} K \frac{1}{(d(t))^2} dt$ where K is some constant determined by parameters, d is the distance to the point at time t , t is time in volume, and where $d(t) = |\overline{P}(t) - \overline{C}| = \sqrt{((P_x(t) - C_x)^2 + (P_y(t) - C_y)^2 + (P_z(t) - C_z)^2)}$.

For simplicity, K could be assumed to be constant throughout the ray, so volumes should be selected to represent constant regions.

This could be computationally expensive, another simplification may be to state a “lock-in” period for the sensor, inversely proportionate to the profile area of the target, and proportionate to the square of the average distance and some constant. Upon

EnterRange/ExitRange, the time in the range is calculated, the probability being the proportion of the time-in to the lock-in time; greater than 1.0 means certain detection.

A *Detection* event would then be scheduled at the first certain lock-in time, or not if this time is after the *ExitRange*, in which case a random number is chosen 0.0-1.0 that if below the time proportion value a *Detection* is scheduled during the lock-in phase depending on the proportion of the random variable to the threshold. *ExitRange* for simplicity in this case would schedule the *UnDetection* of the Sensor, since it is “locked-in”. Of course, if a target was in sufficiently long for certain lock in, there could be the same possibility that a detection occurred during the lock-in phase, in which case the Detection event would be advanced similarly.

Then it actually does remain to determine the enter and exit points and times for a six-sided volume against a ray, instead of calculating a probability integral directly through the volume as paragraph prior to prior. Fortunately, the ray is infinite at one end. Given a normal to a plane and a center point, it is easy to see where a ray intersects it, solving for 0, however, the intersection point still needs to be checked vs. the facet edges to see if it is inside the facet.

Again Cauchy's integral looks interesting, since the bounds are 4 parameterized vectors the computation is relatively cheap, or could be solved by residue calculus. Unfortunately, the coordinates are not transformed to the Complex-Z plane.

Experimental evidence shows that generalization to 3D line integrals yields reasonably good results, some noise near very sharp corners may give false readings depending on the integration approximation used. Since the volumes are somewhat regular, very sharp corners should not be a concern, and furthermore it may be possible to solve exactly without numerical integration.

Even so, going back to the top, it might be just as fast to use the containment test by vector and dot products, divide up the ray into the least reasonable number of samples between *EnterRange* and *ExitRange* and see if any of the samples are captured, then take the amount of time between captured samples for the ratio test in each volume.

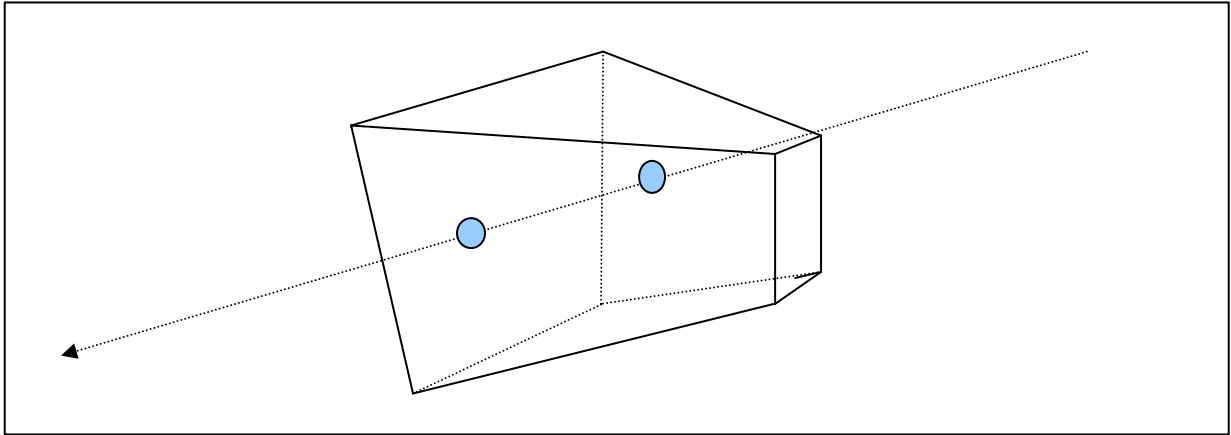


Figure 16. EnterRange and ExitRange Point Depicted

At this stage of the analysis, one of two paths should be chosen, subsample the ray or check for bounds intersection on the facets. Each has benefits and drawbacks.

Back to the “energy potential” calculus, it should be a simple calculation provided the force function is selected as such. The idea is to place a source or sink at the test point of intersection and see if any work is done by going around the facet edges. This technique has the benefit that the winding order is irrelevant, anything significantly different from 0 in either the positive or negative direction indicates the point was circumnavigated. Another benefit to this technique is it instantly generalizes to more complex regions with more edges, curves, non-convex contours, 3D surfaces, even bow-ties and other strange shapes, simply by supplying a longer list of vertices.

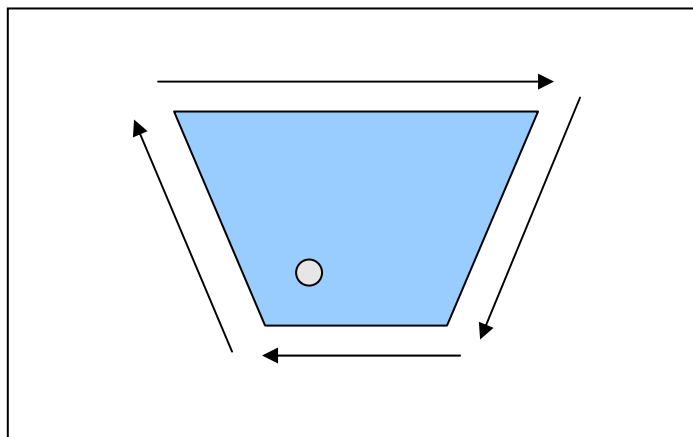


Figure 17. Energy Potential Calculus

One such force function could be $F(x, y, z) = \bar{i}/(x - x_0) + \bar{j}/(y - y_0) + \bar{k}/(z - z_0)$

whose partial derivatives are easy to see, and will blow up as $\lim_{r \rightarrow r_0} \bar{F}(\bar{r})$.

Taking the line

integral $\int_c \bar{F}(\bar{r}) \cdot d\bar{r} = \int_c (\bar{i} dx/(x - x_0) + \bar{j} dy/(y - y_0) + \bar{k} dz/(z - z_0))$ parameterized over s there are 4 line segments, as s goes from 0 to 1 $[(s_0, s_1), (s_1, s_2), (s_2, s_3), (s_3, s_0)]$ where each interval represents the parameterization of the line segment from each vertex to the next.

A line segment between vertices \bar{V}_m and \bar{V}_n from $s_m < s < s_n$ is represented by

$\bar{r}_{mn}(s) = \bar{V}_m(s - s_n / s_m - s_n) + \bar{V}_n(s_m - s / s_m - s_n)$. Then

$\bar{r}_{mn}(s) = 1/(s_m / s_n) + [\bar{V}_m - \bar{V}_n]s + \bar{V}_n s_m - \bar{V}_m s_n$, or expanded out

$x_{mn}(s) = 1/(s_m - s_n)[(v_{mx} - v_{nx})s + v_{nx}s_m - v_{mx}s_n]$

$y_{mn}(s) = 1/(s_m - s_n)[(v_{my} - v_{ny})s + v_{ny}s_m - v_{my}s_n]$

$z_{mn}(s) = 1/(s_m - s_n)[(v_{mz} - v_{nz})s + v_{nz}s_m - v_{mz}s_n]$

Since $d\bar{r}(s) = dx\bar{i} + dy\bar{j} + dz\bar{k}$, or expanding from above,

$dx_{mn}/ds = (v_{mx} - v_{nx})/(s_m - s_n)$

$dy_{mn}/ds = (v_{my} - v_{ny})/(s_m - s_n)$

$dz_{mn}/ds = (v_{mz} - v_{nz})/(s_m - s_n)$

The integral becomes the following, where n is consecutive to m except at the last edge of the facet, then n is 0:

$$\begin{aligned} & \sum_0^3 \int_m^n [\bar{i}/(1/(s_m - s_n))[(v_{mx} - v_{nx})s + v_{nx}s_m - v_{mx}s_n] - x_0] \\ & + [\bar{j}/(1/(s_m - s_n))[(v_{my} - v_{ny})s + v_{ny}s_m - v_{my}s_n] - y_0] \\ & + [\bar{k}/(1/(s_m - s_n))[(v_{mz} - v_{nz})s + v_{nz}s_m - v_{mz}s_n] - z_0] \\ & \cdot [\bar{i}((v_{mx} - v_{nx})/(s_m - s_n)) + \bar{j}((v_{my} - v_{ny})/(s_m - s_n)) + \bar{k}((v_{mz} - v_{nz})/(s_m - s_n))] ds \end{aligned}$$

which can now be simplified for s , carrying out the dot product.

First let $a_{xmn} = \frac{(v_{mx} - v_{nx})}{(s_m - s_n)}$ and similarly for a_{ymn} and a_{zmn} noting they represent the derivatives above, and in the algebraic simplification of F (omitted for brevity) they also appear as coefficients.

Then let $b_{xmn} = v_{mx} s_m - v_{nx} s_n$ and similarly for b_{ymn}, b_{zmn} . This makes each term of the dot product in the integral take the form $\frac{a}{(as+b)-x_0}$, or with a factored out, $\frac{1}{\{s + \frac{(b-x_0)}{a}\}}$ and

similarly for y and z. Finally this yields a simple integral solution in terms of a sum of natural logs, carefully noting that a_α 's and b_α 's change between m 's and n 's throughout the facet edges as above.

Expanding the integral in terms of ds gives

$$\sum_{m=0}^M \int_{s_m}^{s_n} \left\{ \frac{1}{(s + (b_x - x_0)/a_x)} + \frac{1}{(s + (b_y - y_0)/a_y)} + \frac{1}{(s + (b_z - z_0)/a_z)} \right\} ds$$

which conveniently solves to

$$\sum_{m=0}^M \ln(|(s + (b_x - x_0)/a_x)|) \Big|_{(s_m)}^{(s_n)} + \ln(|(s + (b_y - y_0)/a_y)|) \Big|_{(s_m)}^{(s_n)} + \ln(|(s + (b_z - z_0)/a_z)|) \Big|_{(s_m)}^{(s_n)}.$$

An optimization in the algorithmic expression would be to save the n 's as the next m 's. Again note above $M == 3$ for the facet shown, and n is consecutive to m except at the last edge of the facet where n is 0.

With that we now have an easy-to-solve equation that should be close to 0 if the point in question isn't contained, or significantly different than 0 otherwise. Now an algorithmic representation can be defined. While possible, it isn't important to describe the mathematical solution in terms of event graphs, but clearly the graphs should be able to implement the math as a utility.

The above treatment should also be considered for implicit surfaces, replacing the vertex calculations for the equation for the surface. For example the cardioid

$$x = r/(z+a)^2 \cos \Theta, y = r/(z+a)^2 \sin \Theta \text{ would trace out a forward facing cardioid volume.}$$

However, using the 6-sided polygon enables more varied shapes to easily be constructed, if only at the cost of more intersection tests.

In practice however, the above treatment is rather complex and more research examination would be needed to test the equations, and ultimately a simplified geometry may be easier to implement; instead of trying to solve the generalized case of intersection against a possibly concave or bow-tied perimeter, establishing a prerequisite that all facets are convex greatly simplifies the problem to just a few vector cross-products.

The following code section makes the convex assumption about the *Facet's* shape, then each ray extending from each vertex through any sample point's cross-product with the *Facet's* normal will all be in the same direction if the sample point is inside the facet, or not, if it is outside.

```

1  /*
2  * Facet.java
3  *
4  * Created on May 20, 2006, 8:45 PM
5  *
6  */
7  package diskkit;
8
9  import diskkit.util.Transform;
10
11 /**
12 *
13 * @author Rick Goldberg
14 */
15 public class Facet {
16     Vec3d[] vertices;
17     public static final double epsilon = .1;
18     private double tInt = 0.0;
19
20     public static String[][] parameterMap = new String[][] {
21         {
22             "diskkit.Vec3d[]", "vertices"
23         }
24     };
25
26     private static final boolean debug = false;
27
28
29     /** Creates a new instance of Facet
30     * Assumes vertices are in ccw order about the perimeter,
31     * looking down Z+ and all affine rotations, and that there are N>2 of them, and
32     * no vertices are sequentially duplicated. The Facet is convex!
33     */
34     public Facet(Vec3d[] vertices) {
35         this.vertices = new Vec3d[vertices.length + 1];
36         for ( int i = 0; i < vertices.length; i ++ ) {
37             this.vertices[i] = new Vec3d(vertices[i]);
38             if (debug) System.out.println("Vertex ["+i+"] "+vertices[i]);
39         }
40         // add an extra vertex at the end to simplify loop around
41         this.vertices[vertices.length] = new Vec3d(vertices[0]);
42     }
43
44     /** Given a location and a direction, calculate
45     * intersection point, or null if none
46     */
47     public Vec3d intersect(Vec3d point, Vec3d direction) {
48         Vec3d pt = new Vec3d();

```

```

49
50 // create normal to plane for plane vs. ray intersect
51 Vec3d v0 = new Vec3d(vertices[1]);
52 Vec3d v1 = new Vec3d(vertices[1]);
53 v0.sub(vertices[0]);
54 v1.sub(vertices[2]);
55
56 // calculate normal to plane
57 Vec3d normal = new Vec3d();
58 normal.cross(v0,v1);
59 normal.normalize();
60
61 // calculate constant D for plane eqn
62 // with normal N = (A,B,C)
63 // Ax + By + Cz = D
64 // for some/any vertex point
65 double d = normal.get(0)*vertices[0].get(0) + normal.get(1)*vertices[0].get(1) +
        normal.get(2)*vertices[0].get(2);
66
67 if (debug) System.out.println("Normal to plane is "+normal);
68 if (debug) System.out.println("Plane Constant D is "+d);
69
70 // find intersection from point along direction to plane
71 // solve for t parametrically, ray becomes
72 // point + direction * t as 0 < t < inf
73 // or
74 // x(t) = p[0] + d[0] * t
75 // y(t) = p[1] + d[1] * t
76 // z(t) = p[2] + d[2] * t
77 // then for t
78 // t = { D - [ N[0]*P[0] + N[1]*P[1] + N[2]*P[2] } / { N[0]*V[0] + N[1]*V[1] +
        N[2]*V[2] }
79 double t;
80 double nx, ny, nz, px, py, pz, dx, dy, dz;
81 nx = normal.get(0);
82 ny = normal.get(1);
83 nz = normal.get(2);
84 px = point.get(0);
85 py = point.get(1);
86 pz = point.get(2);
87 dx = direction.get(0);
88 dy = direction.get(1);
89 dz = direction.get(2);
90 try {
91     t = ( d - ( nx*px + ny*py + nz*pz ) ) / ( nx*dx + ny*dy + nz*dz);
92 } catch (java.lang.Exception e) {
93     // divide by zero means parallel
94     return null;
95 }
96 if (debug) System.out.print("t intersect parameterized at "+t+" ");
97 if (debug) if (t<=0.0) System.out.println("Never intercepts, going backwards then...");
98 // then substitute back into line eqn to get point from t
99 //
100 pt.set(0,px+dx*t);
101 pt.set(1,py+dy*t);
102 pt.set(2,pz+dz*t);
103 if (debug) System.out.println(pt);
104
105 // check containment of perimeter defined by vertices, or return null if not
    contained
106 // see: "Diskit Sensor and Mover Dynamics" section 5
107 // In terms of DIS coordinates, looking down, in the +z direction,
108 // a non-concave facet is wound ccw iff every surface normal
109 // points up, -z direction, as calculated by drawing a vector from each
110 // vertex to the previous and the next in the order given.
111 // ie
112 // N(Vm) = (Vm-1 - Vm) X ( Vm+1 - Vm)
113 // We've already calculated a normal, and it adheres to this
114 // convention
115 // For a point anywhere on the plane P will be inside the
116 // facet if for each Vm

```

```

117         // C(Vm) = (P- Vm) x ( Vm+1 - Vm)
118         // C(Vm) is in same direction as N
119         // or
120         // C(Vm) . N > 0
121         // and if C(Vm) is normalized by its length
122         // C(Vm) . N ==~ 1.0
123         // so that Sum ( C ( Vm), m=0,M ) ==~ M if P is inside the facet.
124
125
126         // recall 0th is copied to vertices[vertices.length]
127         // numEdges is vertices.length-1
128         double nE = (double)vertices.length-1.0;
129         if (debug) System.out.println("Edge determinator nE "+nE);
130         Vec3d p0 = new Vec3d(pt);
131
132         for ( int i = 0; i < vertices.length-1; i++) {
133             Vec3d V1 = new Vec3d(vertices[i+1]);
134             V1.sub(vertices[i]);
135             Vec3d VP = new Vec3d(p0);
136             VP.sub(vertices[i]);
137             VP.cross(V1);
138             VP.normalize();
139             nE -= VP.dot(normal);
140             if (debug) System.out.println("nE => "+nE);
141         }
142
143         if ( Math.abs( nE ) < epsilon ) {
144             // bingo !!
145             if (debug) System.out.println(pt+ " is inside facet");
146             this.tInt = t;
147             return pt;
148         }
149         return null;
150     }
151
152     /**
153     * same as intersect() except returns time of intersection
154     * in vec[3] as part of a Vec4d
155     */
156     public Vec4d intercept(Vec3d point, Vec3d velocity) {
157         Vec3d intersection = intersect(point, velocity);
158         if ( intersection != null) {
159             Vec4d interception = new Vec4d(intersection.get(0), intersection.get(1),
160                 intersection.get(2), tInt);
161             return interception;
162         }
163         return null;
164     }
165
166     /**
167     * returns a copy of the Facet as transformed
168     */
169     public Facet transform(Transform t) {
170         Vec3d[] verts = new Vec3d[vertices.length-1];
171         for ( int i = 0; i < verts.length; i++) {
172             verts[i] = new Vec3d( vertices[i] );
173             t.transform(verts[i]);
174         }
175         return new Facet(verts);
176     }
177 }

```

With the above transformable *Facet*, now a solid can be assembled as per the prior diagrams. This will be the basic building block for the sensor's capture volumes, the *QuadVolume* provides the same simple intersect method as the *Facet*, a call to intersect causes *QuadVolume* to call intersect on all its *Facets*.

```

1  /*
2  * QuadVolume.java
3  *
4  * Created on June 15, 2006, 5:00 PM
5  *
6  */
7
8  package diskkit;
9  import diskkit.util.Transform;
10
11  /**
12   *
13   * @author Rick Goldberg
14   */
15  public class QuadVolume {
16      // assumptions: this is a volume with 6 x 4 sided facets
17      protected Facet[] facets;
18      protected Transform transform;
19      public static String[][] parameterMap = new String[][] {
20          {
21              "diskit.util.Transform", "transform",
22              "diskit.Facet", "top",
23              "diskit.Facet", "bottom",
24              "diskit.Facet", "front",
25              "diskit.Facet", "back",
26              "diskit.Facet", "left",
27              "diskit.Facet", "right"
28          },
29          {
30              "diskit.util.Transform", "transform",
31              "diskit.Facet[]", "facets"
32          }
33      };
34      // in no particular reason of order:
35      // 0    top
36      // 1    bottom
37      // 2    front
38      // 3    back
39      // 4    left
40      // 5    right
41
42      public QuadVolume(Transform transform, Facet top, Facet bottom, Facet front, Facet back,
43          Facet left, Facet right) {
44          this.transform = transform;
45          this.facets = new Facet[6];
46          this.facets[0] = top.transform(transform);
47          this.facets[1] = bottom.transform(transform);
48          this.facets[2] = front.transform(transform);
49          this.facets[3] = back.transform(transform);
50          this.facets[4] = left.transform(transform);
51          this.facets[5] = right.transform(transform);
52      }
53
54      public QuadVolume(Transform transform, Facet[] facets) {
55          this.transform = transform;
56          this.facets = new Facet[6];
57          for ( int i = 0; i < facets.length; i++ ) {
58              this.facets[i] = facets[i].transform(transform);
59          }
60      }

```

```

60
61     protected QuadVolume() {
62         this(new Transform(), new Facet[0]);
63     }
64
65     // find interception points in time ( v0,v1,v2,t0 )
66     // against the 6 sided volume
67     // should return 2 points, or null
68     public Vec4d[] intercept(Vec3d point, Vec3d velocity) {
69         Vec4d[] interceptions = new Vec4d[6];
70         Vec4d[] intercepts = new Vec4d[2];
71         int c = 0;
72         for ( int i = 0; i < 6; i ++ ) {
73             Vec4d v = facets[i].intercept(point,velocity);
74             if (v != null) {
75                 interceptions[c++] = v;
76             }
77         }
78         // there should be either 2 times or none
79         // could be edge or vertex
80         if (c>1) {
81             intercepts[0] = interceptions[0];
82             final double eps = .01;
83             for ( int i = 0; i < c; i++) {
84                 if ( Math.abs(interceptions[0].get(3) - interceptions[i].get(3)) > eps) {
85                     intercepts[1] = interceptions[i];
86                     break;
87                 }
88             }
89             // check possible edge/corner condition
90             if (intercepts[1] == null) intercepts[1] = interceptions[0];
91             // return them sorted in time
92             if ( intercepts[0].get(3) > intercepts[1].get(3)) {
93                 Vec4d tmp = interceptions[0];
94                 interceptions[0] = interceptions[1];
95                 interceptions[1] = tmp;
96             }
97             return intercepts;
98         }
99         else return null;
100     }
101 }
102
103 }

```

Finally a *SonarScan* can be assembled from an array of *QuadVolumes*. In this case, *SonarScan* will create the “pie-sliced inwardly-squashed semi-cylinder” as depicted earlier, however any grouping of arbitrary 6-sided shapes can be similarly constructed. Furthermore, in the *SonarScan* object, 16 such volumes are created omni-directionally and stacked in 2 layers of 8, each of which is only checked if it is marked “on”, and by default they are all on.

A correction factor is used to adjust the radial endpoints to just beyond the outer bounding sphere, such that the far edges only touch the sphere at one point each. This is done because it is better to check and detect nothing than not to check when there could be something detectable.

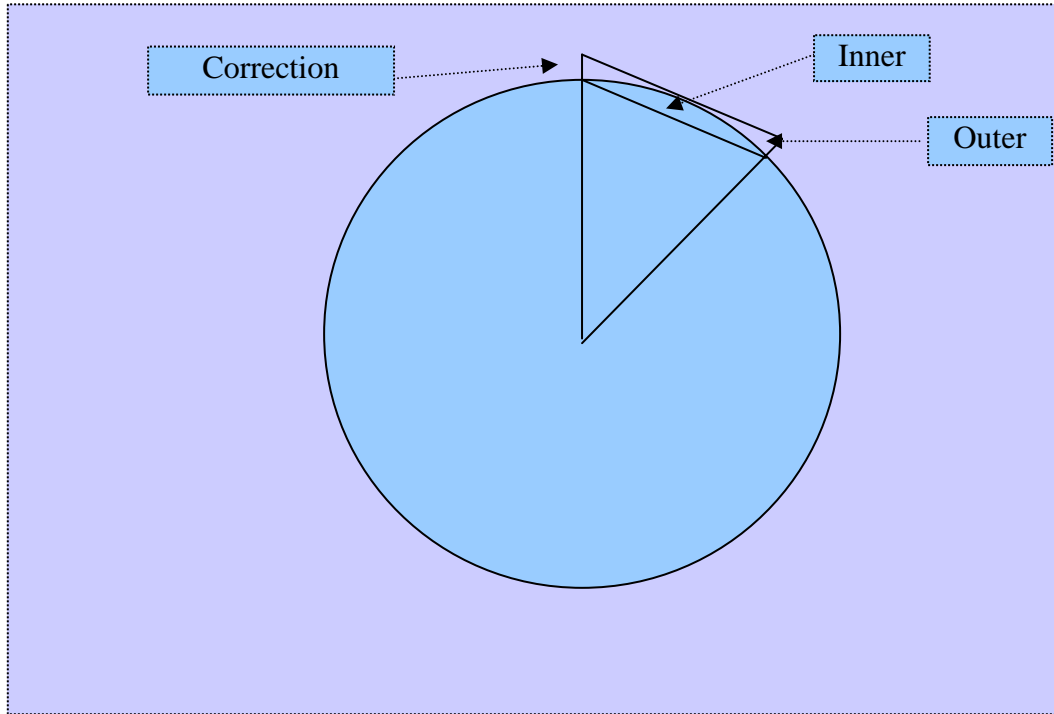


Figure 18. Showing Comparison between an Inner and Outer Approximation to the Sphere by a Facet.

Since these wedges are $\pi/4$ radians, the radial correction factor is 1.0/0.707.

```

1  /*
2  * SonarScan.java
3  *
4  * Created on September 1, 2006, 10:20 AM
5  *
6  * Creates a set of QuadVolumes that form a sort of layered-pie.
7  *
8  */
9
10 package diskkit;
11 import diskkit.util.Transform;
12 import diskkit.Facet;
13 import java.util.Vector;
14
15 /**
16 *
17 * @author Rick Goldberg
18 */
19 public class SonarScan {
20
21     Transform transform;
22     double maxRange;
23     QuadVolume[][] scanVolumes = new QuadVolume[8][2];
24     // looking down z, with x in front, going the 8 xy slices
25     // three parallel polylines trace out the sphere latitudinally from the equator,
26     // fourth and pole not used
27     Vec3d[] topEdge = new Vec3d[8];
28     Vec3d[] midEdge = new Vec3d[8];

```

```

29 Vec3d[] botEdge = new Vec3d[8];
30
31 private boolean[][] activeVolumes = new boolean[8][2]; // check activeVolume[0][n]
    forward right, [7][n] forward left
32
33 /**
34  * Creates a new instance of SonarScan
35  * Simplifies creating scan wedges of QuadVolumes that contain portions of a spherical
    volume.
36  * Sets up 8 sectors, which are 2 layers deep, for a total 16 QuadVolumes.
37  *
38  */
39 public SonarScan(Transform transform, double maxRange) throws IllegalArgumentException {
40
41     this.transform = transform;
42     this.maxRange = maxRange;
43
44     // adjustment factor, to circumscribe the sphere with planars instead of other way
    around
45     // generate these radii
46     double topL = maxRange/Math.cos(Math.PI/4.0);
47     double topH = 0.0;
48     double midL = topL * Math.cos( ( 1.0/8.0 ) * Math.PI );
49     double midH = topL * Math.sin( ( 1.0/8.0 ) * Math.PI );
50     double botL = topL * Math.cos( ( 1.0/4.0 ) * Math.PI );
51     double botH = topL * Math.sin( ( 1.0/4.0 ) * Math.PI );
52     for ( int i = 0; i < 8; i++ ) {
53         double angle = ((double)i ) * Math.PI / 4.0;
54         topEdge[i] = new Vec3d(topL*Math.sin(angle),topL*Math.cos(angle),topH);
55         midEdge[i] = new Vec3d(midL*Math.sin(angle),midL*Math.cos(angle),midH);
56         botEdge[i] = new Vec3d(botL*Math.sin(angle),botL*Math.cos(angle),botH);
57     }
58
59     scanVolumes = createVolumes();
60
61 }
62
63 // see order in createVolumes ... [8][2]
64 public void setActiveVolumes(boolean[][] volumes) {
65     this.activeVolumes = volumes;
66 }
67
68 public QuadVolume[][] createVolumes() {
69
70     // create 2 layers
71     for ( int i = 0; i < 2; i++ ) {
72         // of 8 slices
73         Vec3d[] tops, bottoms;
74         if ( i == 0 ) {
75             tops = topEdge;
76             bottoms = midEdge;
77         } else { // i==1
78             tops = midEdge;
79             bottoms = botEdge;
80         }
81         for ( int j = 0; j < 8; j++ ) {
82             int k = ( j == 7 ) ? 0 : j + 1;
83             Facet top, bottom, front, back, left, right;
84             Vec3d scaledK, scaledJ;
85             scaledK = new Vec3d(tops[k]); scaledK.scale(.001);
86             scaledJ = new Vec3d(tops[j]); scaledJ.scale(.001);
87             top = new Facet(new Vec3d[] { new Vec3d(tops[j]), new Vec3d(tops[k]),
                scaledK, scaledJ } );
88             scaledK = new Vec3d(bottoms[k]); scaledK.scale(.001);
89             scaledJ = new Vec3d(bottoms[j]); scaledJ.scale(.001);
90             bottom = new Facet(new Vec3d[] { new Vec3d(bottoms[j]), new
                Vec3d(bottoms[k]), scaledK, scaledJ } );
91
92             front = new Facet(new Vec3d[] { new Vec3d(tops[j]), new Vec3d(tops[k]), new
                Vec3d(bottoms[k]), new Vec3d(bottoms[j]) } );
93

```

```

94         // technical note on right and left, these may actually be getting created
95         // reversed here (tbd test), however,
96         // mediator makes time calculations that make it irrelevant, including ccw
97         // order
98
99         // note k for both
100        scaledK = new Vec3d(tops[k]);
101        scaledK.scale(.001);
102        scaledJ = new Vec3d(bottoms[k]);
103        scaledJ.scale(.001);
104        left = new Facet( new Vec3d[] { new Vec3d(tops[k]), new Vec3d(bottoms[k]),
105            new Vec3d(scaledJ), new Vec3d(scaledK) });
106
107        // note j for both
108        scaledK = new Vec3d(tops[j]);
109        scaledK.scale(.001);
110        scaledJ = new Vec3d(bottoms[j]);
111        scaledJ.scale(.001);
112        right = new Facet( new Vec3d[] { new Vec3d(tops[j]), new Vec3d(bottoms[j]),
113            new Vec3d(scaledJ), new Vec3d(scaledK) });
114
115        // back
116        Vec3d scaledTK = new Vec3d(tops[k]);
117        scaledTK.scale(.001);
118        Vec3d scaledTJ = new Vec3d(tops[j]);
119        scaledTJ.scale(.001);
120        scaledK = new Vec3d(bottoms[k]);
121        scaledK.scale(.001);
122        scaledJ = new Vec3d(bottoms[j]);
123        scaledJ.scale(.001);
124        back = new Facet( new Vec3d[] { scaledTK, scaledTJ, scaledJ, scaledK });
125
126        scanVolumes[j][i] = new
127        QuadVolume(transform,top,bottom,front,back,left,right);
128        activeVolumes[j][i] = true;
129    }
130    }
131    return scanVolumes;
132}
133
134public Vector intercept(Vec3d point, Vec3d velocity) {
135    Vector v = new Vector();
136    for ( int i = 0; i < 2; i ++ ) {
137        for ( int j = 0; j < 8; j++ ) {
138            if (activeVolumes[j][i]) {
139                Vec4d[] intercepts = scanVolumes[j][i].intercept(point,velocity);
140                if (intercepts != null) {
141                    v.add(intercepts);
142                }
143            }
144        }
145    }
146    return v;
147}

```

Now that a contact location strategy has been determined, it remains to integrate the *FOM* of detection within the capture volume(s). There are a number of factors which can contribute to the quality of a signal, such as ambient noise, frequency, initial energy, and *Target* geometry.

In signal analysis it is customary to represent energy levels in terms of Decibels (dB), in large part because perception of loudness is logarithmic for both biological and electro-mechanical devices. Using the equations for sonar signaling as presented by **Sonalyts, Inc.**, a detection threshold can readily be implemented in terms of **Diskit's Mover** and *Sensor* dynamics.

The principle behind the **Sonalyts FOM** equation is that overall signal integrity is composed of several factors, and in terms of transcendental mathematics, logarithms add or subtract as factors multiply and divide. This greatly simplifies quantitative results. So for example one could say return on a signal is proportional to the inverse of the square of the distance (and other factors), in terms of logs that becomes $-2\log X$ (+/- other factors.)

Once sufficient factors can be approximated, an overall summation of the signal in terms of dB can be constructed, including the amount of raw signal required for an operator to positively differentiate between noise and target reflection.

Let:

SL = Signal Strength of Ping
 TL = Transmission Loss of signal spread, approximately $10 \log(\text{range})$ overall $20 \log(\text{range})$ to account for the loss of the signal on its return.
 TS = Target Strength, a figure dependent upon shape, orientation, and composition of the target
 NL = Noise Level of ambient sound
 DI = Directivity Index, noise filter capability of the sonar
 RD = Recognition Differential, can be operator skill level, or a.k.a. DT as Detection Threshold
 AT = Attenuation loss due to frequency of signal,
 $(12/11) * (0.003 + 0.1f^2 / (1 + f^2)) + (40f^2) / (4100 + f^2) + 0.000275f^2$

Then:

if $SL - 2(TL) + TS - (NL - DI) - RD - AT > 0$ a detection event happens and conversely if < 0 , an *UnDetection* may happen if already detected.

<i>Frequency</i>	<i>Attenuation</i>
3.5 kHz	.22 dB/kyd (.24 dB/km)
10 kHz	1.08 dB/kyd (1.19 dB/km)
30 kHz	7.55 dB/kyd (8.31 dB/km)
60 kHz	19.79 dB/kyd (21.77 dB/km)
100 kHz	31.22 dB/kyd (34.34 dB/km)

Table 2. Table showing sample values of *AT* used for various frequencies of interest.

Some of these factors can be considered constant inputs to the equation, such as *SL* or *AT*, whereas other factors can have some randomness, such as *RD* (if say the operator was distracted) or *NL* since noise levels are themselves “noisy”. The *MultiLRATL Sonar* model enables the analyst to set variously shaped random variates for input parameters, so for example if *RD* of a skilled operator is measured to be typically 10.0 dB, the operator may have a standard deviation perhaps by 1.0 in a Normal Gaussian distribution. Similarly *NL* might be also 60 dB with a standard deviation of 5.0 in a particular harbor.

Perhaps the region has some areas that are noisier than others, and data are available at regular intervals for noise levels, then a map can be constructed using **Diskit's** *InterpolatedXYVariate*, which is a drop-in replacement for any abstract *simkit.random.RandomVariate* parameter, incidentally using the above-mentioned *Facet* to perform the interpolation.

```

1  /*
2  * InterpolatedXYVariate.java
3  *
4  * Created on July 9, 2006, 11:07 AM
5  *
6  * An M x N grid in X,Y, where
7  * Xm = m * Dx/Dm + X0
8  * Yn = n * Dy/Dn + Y0
9  */
10 package diskit;
11
12 import simkit.random.RandomVariateBase;
13
14 /**
15 *
16 * @author Rick Goldberg
17 */
18 public class InterpolatedXYVariate extends RandomVariateBase {
19     public static double MAX_Z = 1000.0;
20     double xScale; // Dx/Dm
21     double xShift; // X0
22     double yScale; // Dy/Dn
23     double yShift; // Y0

```

```

24 Double[][] zValues; // zValues[ y rows ][ x columns ]
25 double x,y; // sample point
26 Facet quad;
27
28 /** Creates a new instance of InterpolatedXYVariate */
29 public InterpolatedXYVariate() {
30     setXScale(1.0);
31     setXShift(0.0);
32     setYScale(1.0);
33     setYShift(0.0);
34     setX(0.0);
35     setY(0.0);
36     // initially make a flat 2x2 of z values, generally by row[0], row[1] ... row[n]
37     Object[] zGrid = new Object[] { new Double[] { new Double(0.0), new Double(0.0) } ,
38         new Double[] { new Double(0.0), new Double(0.0) } };
39     setParameters(zGrid);
40 }
41
42 // x,y should be set each generate(), otherwise this behaves like a constant variate,
43 // unless data was touched.
44 public double generate() {
45     int xI, xJ, yI, yJ; // index into zValues
46     double x0 = x/xScale - xShift;
47     double y0 = y/yScale - yShift;
48
49     // first xI, to be low x index, then xI+1 is hi
50     xI = (int)x0;
51     xJ = xI + 1;
52
53     // clamp to edge of map
54     if ( xJ > zValues[0].length ) {
55         xJ = zValues[0].length;
56         xI = xJ - 1;
57     }
58
59     if ( xI < 0 ) {
60         xI = 0;
61         xJ = 1;
62     }
63
64     yI = (int)(y0);
65     yJ = yI + 1;
66
67     if ( yJ > zValues.length ) {
68         yJ = zValues.length;
69         yI = yJ - 1;
70     }
71
72     if ( yJ < 0 ) {
73         yJ = 1;
74         yI = 0;
75     }
76
77     Vec3d[] verts = new Vec3d[4];
78     verts[0] = new Vec3d( (double)xI, (double)yI, (zValues[yI][xI]).doubleValue() );
79     verts[1] = new Vec3d( (double)xI, (double)yJ, (zValues[yJ][xI]).doubleValue() );
80     verts[2] = new Vec3d( (double)xJ, (double)yJ, (zValues[yJ][xJ]).doubleValue() );
81     verts[3] = new Vec3d( (double)xJ, (double)yI, (zValues[yI][xJ]).doubleValue() );
82
83     // can carry out the interp in normalized space, same result as full x,y coords
84     quad = new Facet(verts);
85
86     // using the 4d version don't really need a MAX_Z, can be backwards in 'time', but
87     // preserving sense of +z down could also be used for terrain; orig. intent was for
88     // noise
89     // intensity and other data however.
90
91     Vec4d intercept = quad.intercept( new Vec3d(x0,y0,-MAX_Z), new Vec3d(0.0,0.0,1.0) );
92     return intercept.get(2);
93 }

```

```

93
94  /* Parameters are object[N] = Double[M]
95   * of Z values.
96   */
97  public void setParameters(Object[] object) {
98      int N = object.length;
99      for ( int n = 0; n < N; n++ ) {
100          zValues[n] = (Double[])object[n];
101      }
102  }
103
104  /* returns actual data, not a copy
105   */
106  public Object[] getParameters() {
107      Object[] ret = new Object[zValues.length];
108      for ( int n = 0; n < zValues.length; n ++ ) {
109          ret[n]=zValues[n];
110      }
111      return ret;
112  }
113
114  public void setYShift(double d) {
115      yShift=d;
116  }
117
118  public void setXShift(double d) {
119      xShift=d;
120  }
121
122  public void setXScale(double d) {
123      xScale=d;
124  }
125
126  public void setYScale(double d) {
127      yScale=d;
128  }
129
130  public void setX(double d) {
131      x=d;
132  }
133
134  public void setY(double d) {
135      y=d;
136  }
137 }

```

A quote from (Urick 1986) and commentary courtesy Douglas Nelson, **Sonalysts, Inc.:**

No measurement work in the real ocean has been done in this frequency range, except for the measurements of Anderson And Gruber at 30, 90 and 150 kHz in the ports of San Diego, Long Beach in California, Balboa and Christobal in the Pacific Canal Zone, and Norfolk, Virginia. These locations were found to be extremely noisy and showed great variability from port to port. The average levels in these ports was some 20 dB higher than the Knudsen extrapolated levels for sea state 6. Surprisingly small differences were found between day and night; the lower levels due to industrial activity during the night were evidently compensated by higher noise due to snapping shrimp. Comparing the various ports, there was a general tendency for the noise levels to increase with decreasing latitude, as would be expected from greater abundance of shrimp in lower latitudes.

From the Knudsen curves, we should expect average noise levels to be 66 dB at 30 kHz, 57 dB at 90 kHz, and 53 dB at 150 kHz. Going by Urick's comment on latitudes, we should probably give Bremerton and Annapolis lower values and Pearl a higher value. Interesting points are the small differences noted between day and night, and that wind/sea state related noise is completely dominated by other sources.

These random factors generate discrete values each time they are sampled by a “Ping” from the *MultiLRATLSonar*. The net result is they define a probability of detection that tapers off at the furthest maximum range as denoted by the *Mediator's* detection sphere; consequently, a *Sensor* should be initialized with its maximum range parameter calculated to be that where factors such as *NL* or *RD* are at their best.

There are now sufficient components to assemble a working model within **Viskit**. The following shows the Event-Graph layout and generated code for the *MultiLRATLSonar*, at which point it will be dropped into an existing **SMAL**-based sample scenario from the **ATFP BehaviorLibraries**.

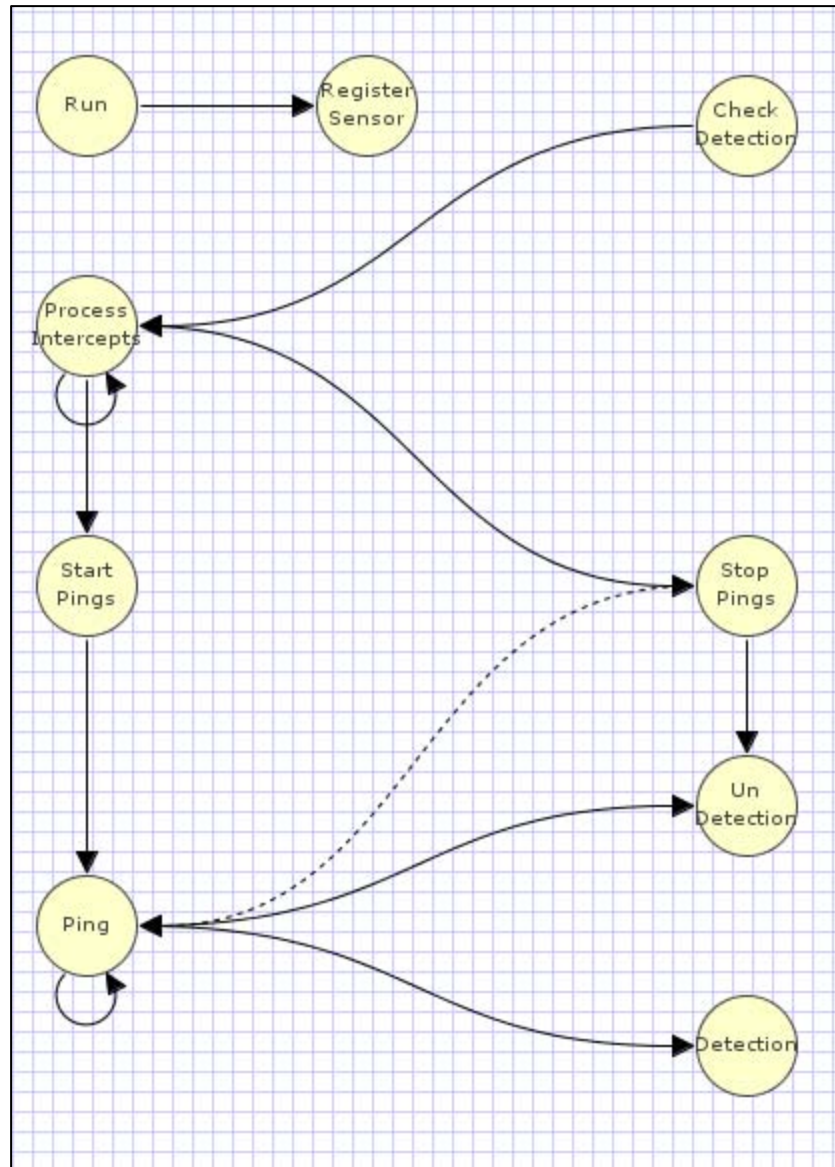


Figure 19. CheckDetection Event Graph

Once the *CheckDetection* event is heard from the *Mediator*, a number of *Pings* are scheduled while the potential *Target* is sampled throughout its traversal of the *SonarScan* volume. If the $FOM > 0.0$ a *Detection* occurs, otherwise *UnDetection* occurs if already detected.

This event graph generates the following runnable code:

```
1 package diskit;
2
3 import simkit.*;
4 import simkit.random.*;
5 import java.util.*;
6
7 public class MultiLRATLSonar extends diskit.SphereCutterSensor {
8
9     /* inherited parameter diskit.Mover3D mover */
10    /* inherited parameter double maxRange */
11    private double SL;
12    private double DI;
13    private simkit.random.RandomVariate RD;
14    private diskit.SonarScan scans;
15    private double pingInterval;
16    private simkit.random.RandomVariate noise;
17    private double frequency;
18
19    protected double TL;
20    protected double TS;
21    protected double DT;
22    protected double NL;
23    protected double AT;
24    protected java.util.Hashtable detections = new java.util.Hashtable();
25
26    /** Creates a new instance of MultiLRATLSonar */
27    public MultiLRATLSonar(diskit.Mover3D mover,
28        double maxRange,
29        double SL,
30        double DI,
31        simkit.random.RandomVariate RD,
32        diskit.SonarScan scans,
33        double pingInterval,
34        simkit.random.RandomVariate noise,
35        double frequency) {
36
37        super(mover,maxRange);
38        setSL(SL);
39        setDI(DI);
40        setRD(RD);
41        setScans(scans);
42        setPingInterval(pingInterval);
43        setNoise(noise);
44        setFrequency(frequency);
45    }
46
47    /** Set initial values of all state variables */
48    public void reset() {
49
50        super.reset();
51
52        /** StateTransitions for the Run Event */
53
54        DT = RD.generate();
55    }
56
57    public void doRun() {
58        super.doRun();
59        firePropertyChange("DT",DT);
60        if (true) {
61            waitDelay("RegisterSensor",0.0,new Object[]{this},0);
62        }
63    }
64 }
65
```

```

66     public void doCheckDetection(diskit.Mover3D contact) {
67         diskit.Vec3d relativeLocation = (diskit.Vec3d)new
            diskit.Vec3d(contact.getLocation());
68         relativeLocation.sub(getMover().getLocation());
69         diskit.Vec3d relativeVelocity = (diskit.Vec3d)new
            diskit.Vec3d(contact.getVelocity());
70         relativeVelocity.sub(getMover().getVelocity());
71         java.util.Vector intercepts =
            (java.util.Vector)scans.intercept(relativeLocation,relativeVelocity);

72
73         /* Code insertion for Event CheckDetection */
74         System.out.println(">>>>>>Checking detection of "+contact);
75         System.out.println(">>>>>>Intercepts at "+intercepts+" length
            "+intercepts.size());
76         /* End Code insertion */
77         /* StateTransition for detections */
78         java.util.Hashtable _old_Detections = getDetections();
79         detections.put(contact, new Boolean(false));
80         firePropertyChange("detections", _old_Detections, getDetections());
81
82
83         if (intercepts.size() > 0) {
84             waitDelay("ProcessIntercepts",0.0,new Object[]{intercepts,new
                Integer(0),contact},0);
85         }
86     }
87
88     public void doStartPings(diskit.Mover3D contact, diskit.Vec4d enterPoint,
        diskit.Vec4d exitPoint) {
89         /* Code insertion for Event StartPings */
90         System.out.println(">>>>>>Starting pings...");
91         /* End Code insertion */
92
93         if (true) {
94             waitDelay("Ping",0.0,new Object[]{contact},0);
95         }
96     }
97
98     public void doPing(diskit.Mover3D contact) {
99         double range =
            (double)Vec3d.distance(getMover().getLocation(),contact.getLocation());
100         double fSq = (double)frequency*frequency;
101
102         /* Code insertion for Event Ping */
103         System.out.println(">>>>>>Ping to range "+range);
104         /* End Code insertion */
105         /* StateTransition for TL */
106         double _old_TL = getTL();
107         TL = 10 * Math.log(range);
108         firePropertyChange("TL", _old_TL, getTL());
109
110         /* StateTransition for DT */
111         double _old_DT = getDT();
112         DT = getRD().generate();
113         firePropertyChange("DT", _old_DT, getDT());
114
115         /* StateTransition for TS */
116         double _old_TS = getTS();
117         TS = -15.0;
118         firePropertyChange("TS", _old_TS, getTS());
119
120         /* StateTransition for NL */
121         double _old_NL = getNL();
122         NL = noise.generate();
123         firePropertyChange("NL", _old_NL, getNL());
124
125         /* StateTransition for AT */
126         double _old_AT = getAT();
127         AT = (range/1000.0) * (.003 + (.1*fSq/(1+fSq)) + (40.0*fSq/(4100.0 + fSq))
            + .000275*fSq) * 12.0/11.0;

```

```

128      /* Code block for pre-transition */
129      System.out.println("SL: "+SL+" TL: "+TL+" TS: "+TS+" NL: "+NL+" DI "+DI+"
      DT: "+DT+" AT: "+AT+"\nSL-(2*TL)+TS-(NL-DI)-DT-AT = "+(SL-(2*TL) +TS-
      (NL-DI)-DT-AT));
130      firePropertyChange("AT", _old_AT, getAT());
131
132      if (true) {
133          waitDelay("Ping",getPingInterval(),new Object[]{contact},0);
134      }
135      if (((SL - (2 * TL) + TS - ( NL - DI ) - DT - AT) > 0.0) && (! (
      ((Boolean)(detections.get(contact)) ).booleanValue())) {
136          waitDelay("Detection",0.0,new Object[]{contact},0);
137      }
138      if (((SL - (2 * TL) + TS - ( NL - DI ) - DT - AT) <= 0.0) && ((
      ((Boolean)(detections.get(contact)) ).booleanValue())) {
139          waitDelay("UnDetection",0.0,new Object[]{contact},0);
140      }
141  }
142
143  public void doStopPings(diskit.Mover3D contact) {
144      /* Code insertion for Event StopPings */
145
146      /* End Code insertion */
147
148      if (true) {
149          interrupt("Ping",new Object[]{contact});
150      }
151      waitDelay("UnDetection",0.0,new Object[] {},0.0);
152  }
153
154  public void doDetection(diskit.Mover3D contact) {
155      /* Code insertion for Event Detection */
156      System.out.println("MultiLRATLSonar "+this+" Detected "+contact);
157      /* End Code insertion */
158      /* StateTransition for detections */
159      java.util.Hashtable _old_Detections = getDetections();
160      detections.put(contact,new Boolean(true));
161      firePropertyChange("detections", _old_Detections, getDetections());
162  }
163
164
165  public void doUnDetection(diskit.Mover3D contact) {
166      /* Code insertion for Event UnDetection */
167      System.out.println("UnDetection "+contact);
168      /* End Code insertion */
169      /* StateTransition for detections */
170      java.util.Hashtable _old_Detections = getDetections();
171      detections.put(contact,new Boolean(false));
172      firePropertyChange("detections", _old_Detections, getDetections());
173  }
174
175
176  public void doProcessIntercepts(java.util.Vector intercepts, int count,
      diskit.Mover3D contact) {
177      diskit.Vec4d[] intercept = (diskit.Vec4d[])(diskit.Vec4d[])
      (intercepts.get(count));
178
179      /* Code insertion for Event ProcessIntercepts */
180      System.out.println(">>>>>>>Intercept 0 "+intercept[0]);
181      System.out.println(">>>>>>>Intercept 1 "+intercept[1]);
182      /* End Code insertion */
183
184      if (count < intercepts.size() - 1) {
185          waitDelay("ProcessIntercepts",0.0,new Object[]{intercepts,new
      Integer(count+1),contact},0);
186      }
187      if (intercept[1].get(3) > 0.0) {
188          waitDelay("StopPings",intercept[1].get(3),new Object[]{contact},0);
189      }
190      if (intercept[1].get(3) > 0.0 ) {
191

```

```

waitDelay("StartPings",intercept[0].get(3)>0.0?intercept[0].get(3):0.0,new
Object[] {contact,intercept[0],intercept[1]},0);
192     }
193 }
194
195 public void doRegisterSensor(diskit.Sensor sensor) {
196     /* Code insertion for Event RegisterSensor */
197
198     /* End Code insertion */
199 }
200
201 public void setSL(double SL) {
202     this.SL = SL;
203 }
204
205 public double getSL() {
206     return SL;
207 }
208
209 public void setDI(double DI) {
210     this.DI = DI;
211 }
212
213 public double getDI() {
214     return DI;
215 }
216
217 public void setRD(simkit.random.RandomVariate RD) {
218     this.RD = RD;
219 }
220
221 public simkit.random.RandomVariate getRD() {
222     return RD;
223 }
224
225 public void setScans(diskit.SonarScan scans) {
226     this.scans = scans;
227 }
228
229 public diskit.SonarScan getScans() {
230     return scans;
231 }
232
233 public void setPingInterval(double pingInterval) {
234     this.pingInterval = pingInterval;
235 }
236
237 public double getPingInterval() {
238     return pingInterval;
239 }
240
241 public void setNoise(simkit.random.RandomVariate noise) {
242     this.noise = noise;
243 }
244
245 public simkit.random.RandomVariate getNoise() {
246     return noise;
247 }
248
249 public void setFrequency(double frequency) {
250     this.frequency = frequency;
251 }
252
253 public double getFrequency() {
254     return frequency;
255 }
256
257 public double getTL() {
258     return TL;
259 }
260 }

```

```

261
262
263     public double getTS() {
264         return TS;
265     }
266
267     public double getDT() {
268         return DT;
269     }
270
271     public double getNL() {
272         return NL;
273     }
274
275
276     public double getAT() {
277         return AT;
278     }
279
280     public java.util.Hashtable getDetections() {
281         return (java.util.Hashtable) detections.clone();
282     }
283
284     /* Inserted code for MultiLRATLSonar */
285     /*
286     Some Frequencies of interest
287     3.5 kHz    :   .22 db/kyd (.24 db/km)
288     10 kHz     :   1.08 db/kyd (1.19 db/km)
289     30 kHz     :   7.55 db/kyd (8.31 db/km)
290     60 kHz     :  19.79 db/kyd (21.77 db/km)
291     100 kHz    :  31.22 db/kyd (34.34 db/km)
292     */
293
294     public static double FREQ_3_5_Khz = 3.5000;
295     public static double FREQ_10_Khz = 10.0000;
296     public static double FREQ_30_Khz = 30.0000;
297     public static double FREQ_60_Khz = 60.0000;
298     public static double FREQ_100_Khz = 100.0000;
299     /* End inserted code */
300
301 }

```

As can be seen from the above code, the *Sensor* derives its sense of location from a *Mover* that it has been mounted on, and that it is irrelevant whether or not the mounting point is stationary or moving.

The following **Viskit Assembly** scenario, *IndianIslandSonarTest*, mounts an unmanned *MultiLRATLSonar* to a stationary *AmmoPier*. Since manned **SMAL** entities use the abstract *Sensor*, they easily mount with a *MultiLRATLSonar* and respond to obstacles and other objects in the same way, however this example for simplicity is stationary and causes no response within the scenario.

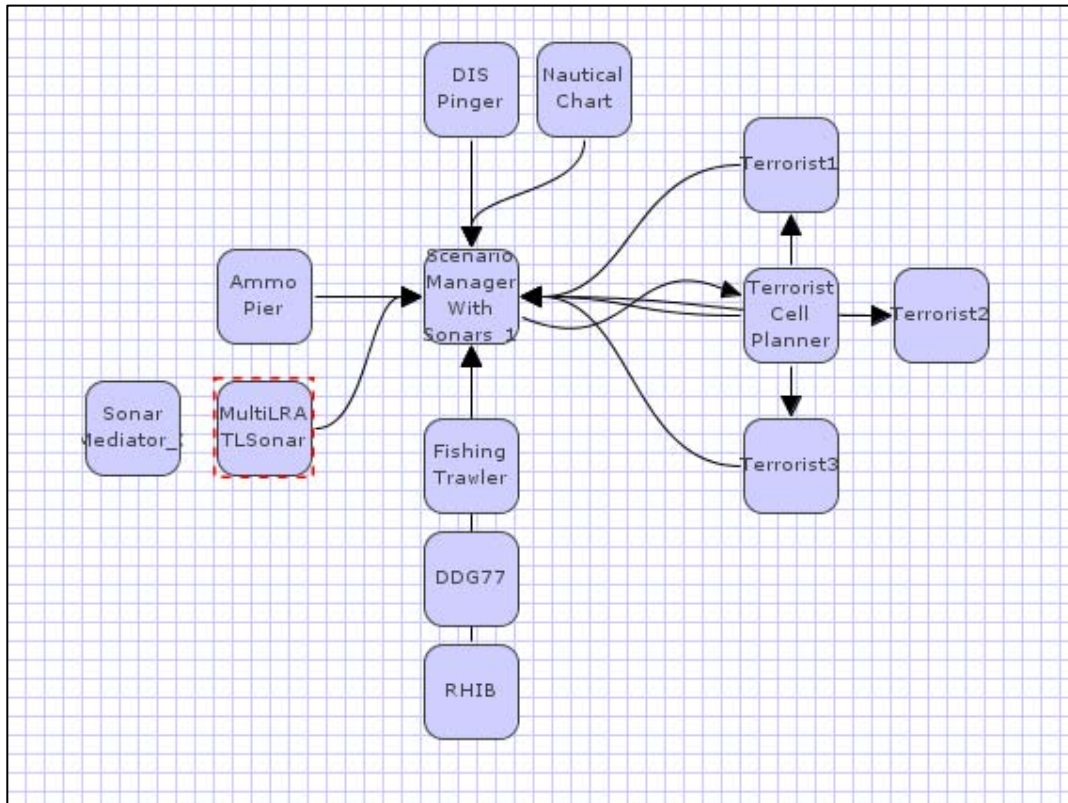


Figure 20. Adding the *MultiLRATLSonar* as a drop in component.

The scenario is now ready to run. Below shows a sample run of *IndianIslandSonarTest* within **Viskit's Assembly Runner**. The output debug messages shown are ordinarily disabled, but are demonstrative of the *FOM* determining *Detection* and *UnDetection* events, as entities traverse the *MultiLRATLSonar's* detection zone, and in particular, two such entities, one of which momentarily becomes detectable.

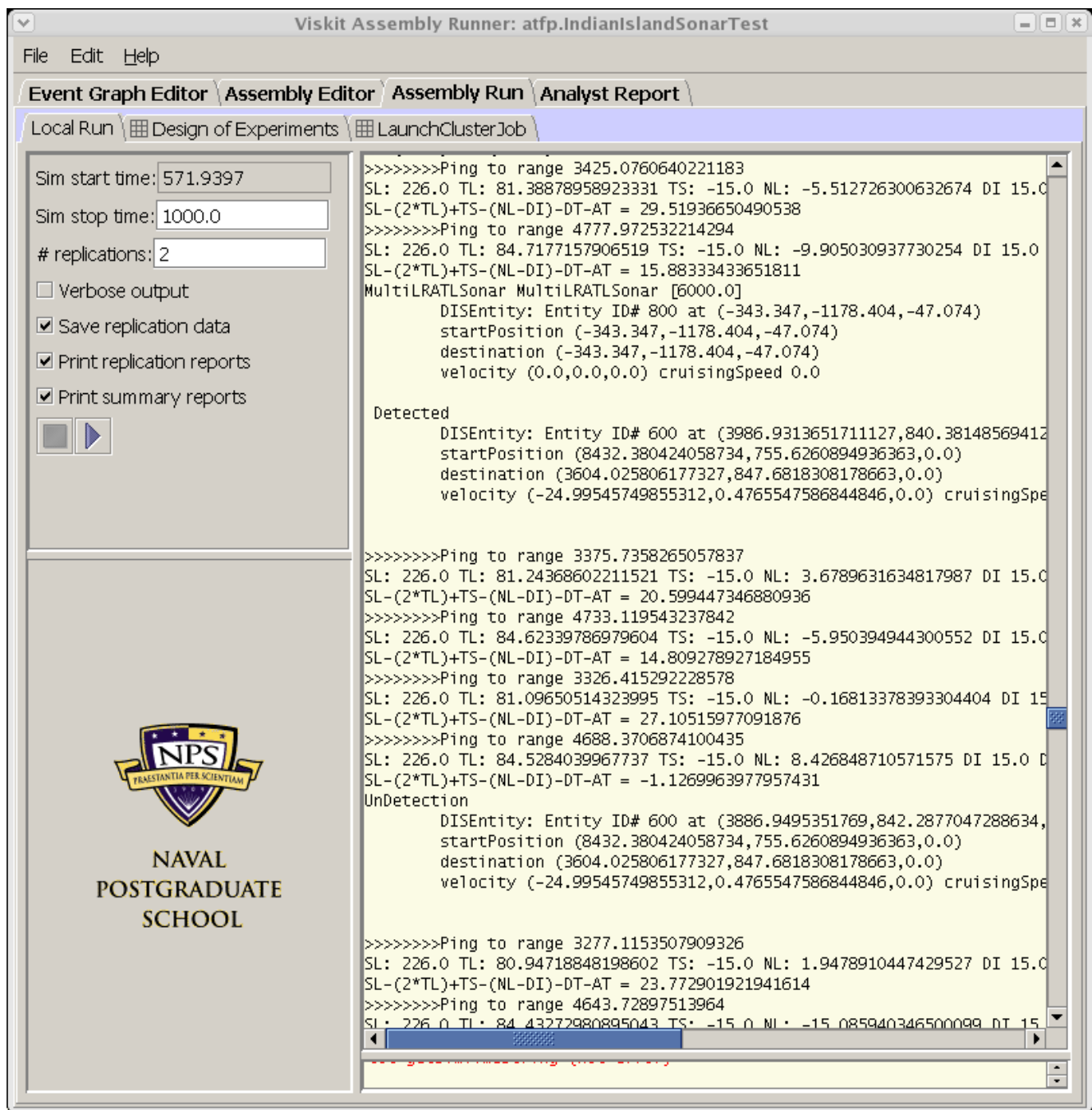


Figure 21. Debug Output from a Random Run of *IndianIslandSonarTest* in **Viskit**.

APPENDIX F: PLANET 9 PRESENTATION SLIDASETS

F.1 INTRODUCTION

These presentations were originally give during the May M&S Workshop here at NPS by Christian Greuel and David Colleen. They are reprinted here by permission of the Planet 9 Studios Art Team and are relevant to the processes required to building port and shore side installation models.

F.1 BUILDING GEO-REGISTERED X3D

Building Geo-Registered X3D

Port & Harbor Models... Accurately Located



Christian Greul
Director of Art & Production
Planet 9 Studios
www.planet9.com
cgreul@planet9.com

Unclassified Christian Greul - Port/Harbor Modeling planet

Goal of This Presentation



This presentation will provide a high-level overview of the production process used by Planet 9 Studios to create high-fidelity geo-referenced X3D models of waterside facilities for the WSS AT/FP Project.

Unclassified Christian Greul - Port/Harbor Modeling planet

Activities Modeled for US Navy

- Al-Basrah Oil Terminal (ABOT)
- Friday Harbor
- MCAS Miramar
- NAS North Island (rough)
- NAVMAG Indian Island
- NAVSTA Bremerton
- Pearl Harbor
- Port Hueneme (rough)
- SUBASE Bangor
- Washington Navy Yard
- Yokosuka, Japan

Unclassified Christian Greul - Port/Harbor Modeling planet

Activities Modeled for WSS AT/FP

- Al-Basrah Oil Terminal (ABOT)
- Friday Harbor
- MCAS Miramar
- NAS North Island (rough)
- NAVMAG Indian Island
- NAVSTA Bremerton
- Pearl Harbor
- Port Hueneme (rough)
- SUBASE Bangor
- Washington Navy Yard
- Yokosuka, Japan



Unclassified Christian Greul - Port/Harbor Modeling planet

Activities Modeled for Other Govt.

- Ft. Benning - McKenna MOUT
- Ft. Campbell - Cassidy MOUT
- Ft. Dix - Range 65
- Ft. Lewis - Leschi MOUT
- Ft. Polk - Self Airbase
- George AFB (now SCLA)
- MCAS Miramar
- Moffett Field (NASA)
- Nellis AFB
- Peterson AFB
- Puhakuloa Training Area

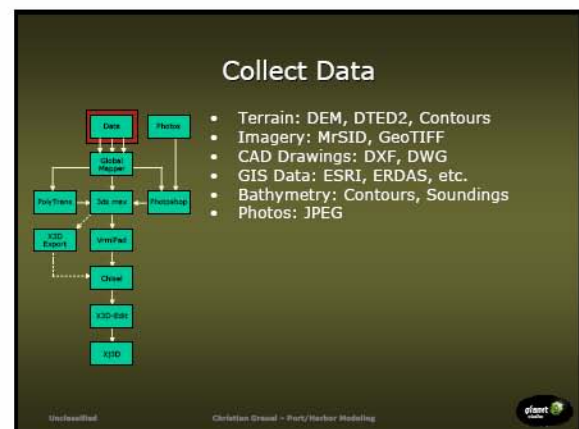
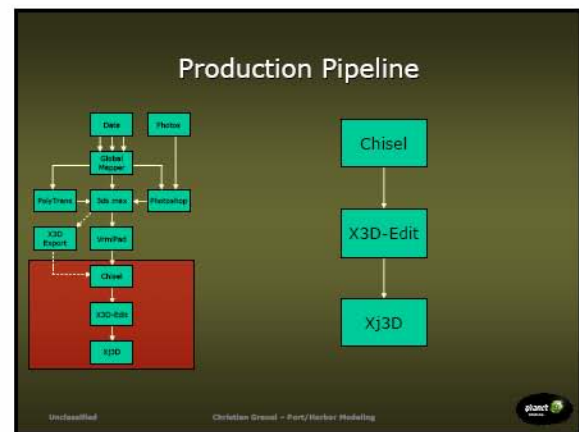
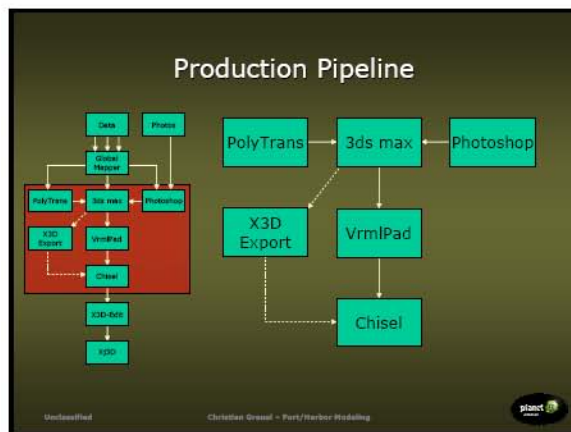
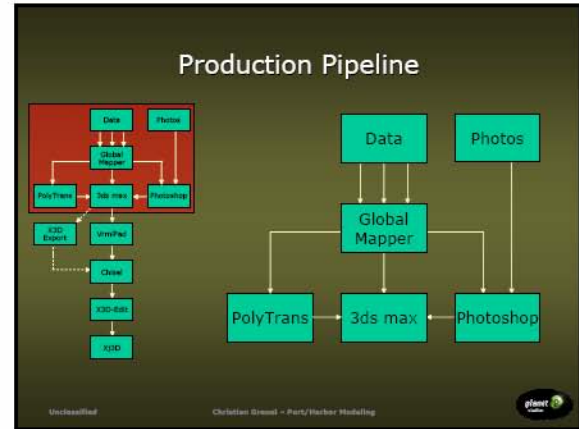
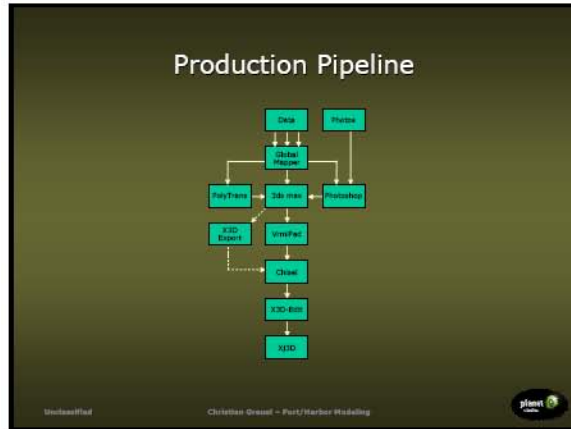
Unclassified Christian Greul - Port/Harbor Modeling planet

Tools Used

Many excellent tools exist. Our production pipeline makes use of the following:

- Photoshop - www.adobe.com
- Global Mapper - www.globalmapper.com
- 3D Studio Max - www.discreet.com
- PolyTrans - www.okino.com
- VrmIPad - www.parallelographics.com
- Chisel - <http://www2.hrp.no/vr/tools/chisel/install.htm>
- X3D-Edit - <http://www.web3d.org/x3d/content/README-X3D-Edit.html>
- Xj3D - <http://www.web3d.org/x3d/applications/xj3d/>

Unclassified Christian Greul - Port/Harbor Modeling planet



Data Sources



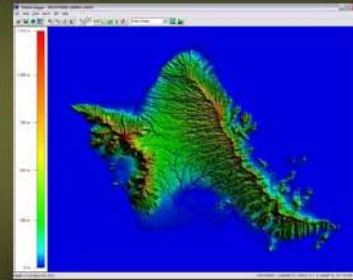
- USGS
- NGA
- NFESC
- NAVSEA Warfare Centers Division
- GeoReadiness Repository (GRR)
- Navy Region GIS Center of Excellence
- Local Base: Engineering Division
- Commercial (Space Imaging, etc.)
- Internet (beware copyright, Ref. only)

Unclassified

Christian Gressel - Port/Harbor Modeling



Import Terrain → Global Mapper



Unclassified

Christian Gressel - Port/Harbor Modeling



Geographic Projections



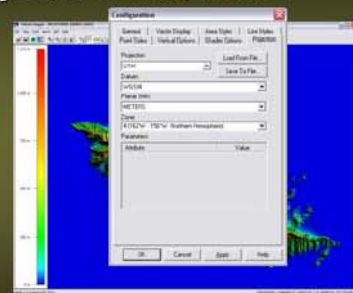
Projection - Latitude/Longitude vs. **UTM**
 Datum - NAD27, NAD83, **WGS84**
 Units - Inches, Feet, **Meters**

Unclassified

Christian Gressel - Port/Harbor Modeling



Set Projection to UTM/WGS84



Unclassified

Christian Gressel - Port/Harbor Modeling



Import Imagery

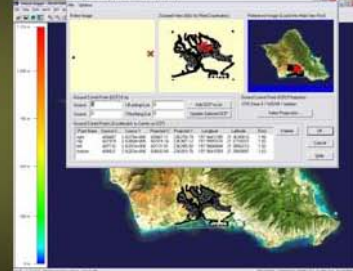


Unclassified

Christian Gressel - Port/Harbor Modeling



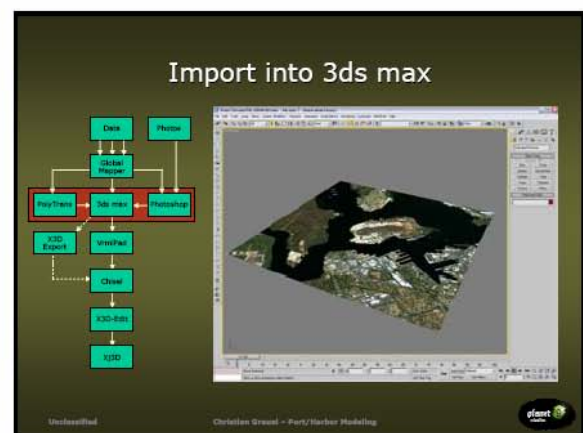
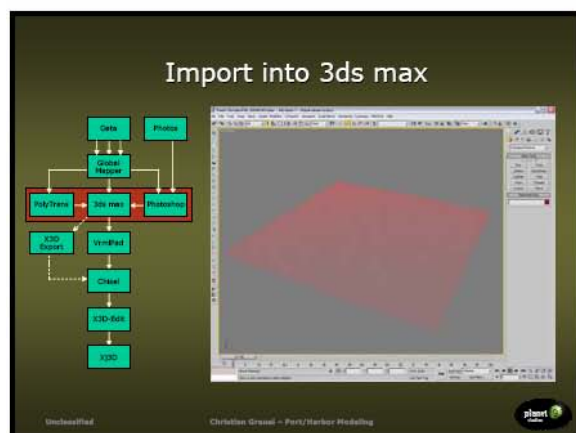
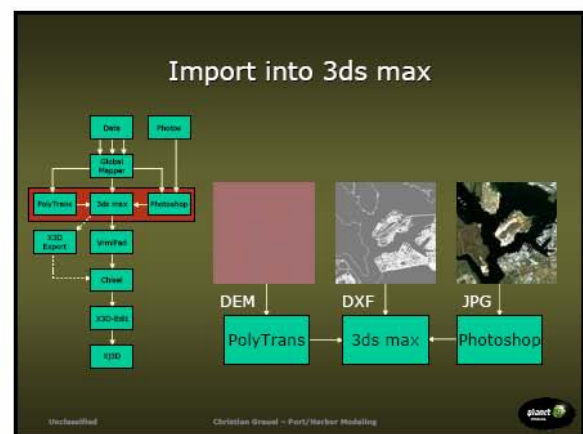
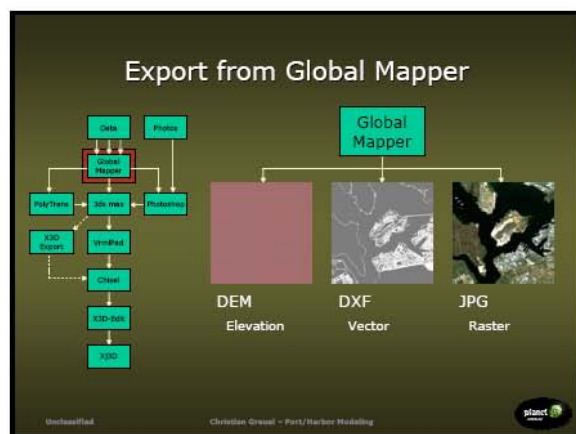
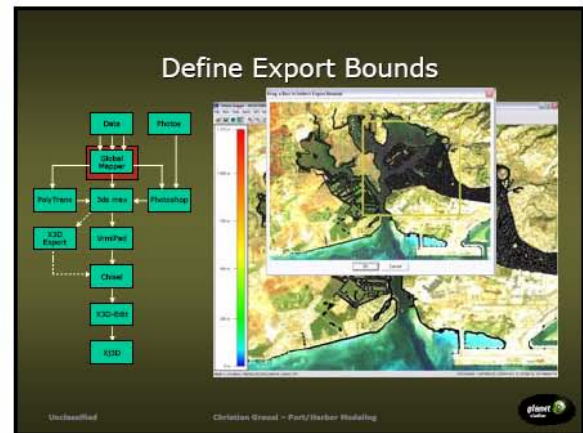
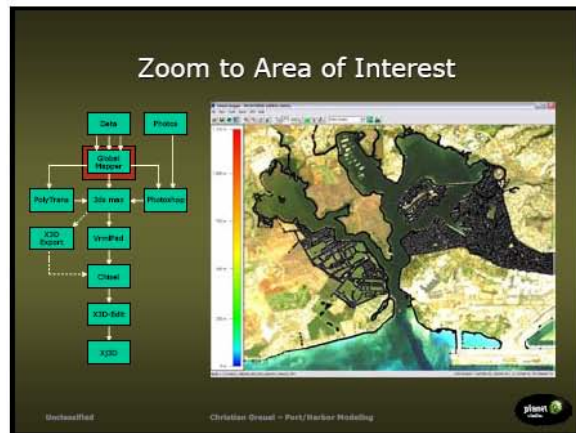
Import & Rectify CAD Drawings

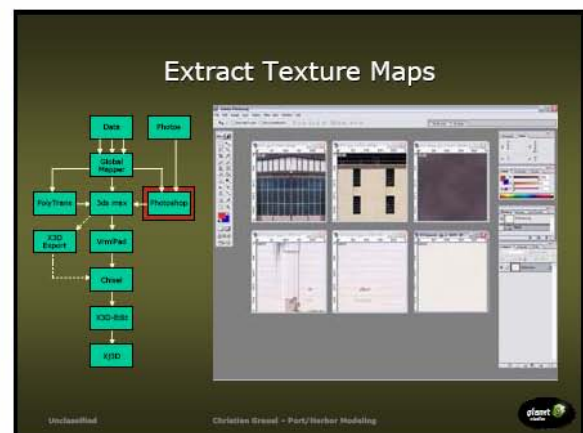
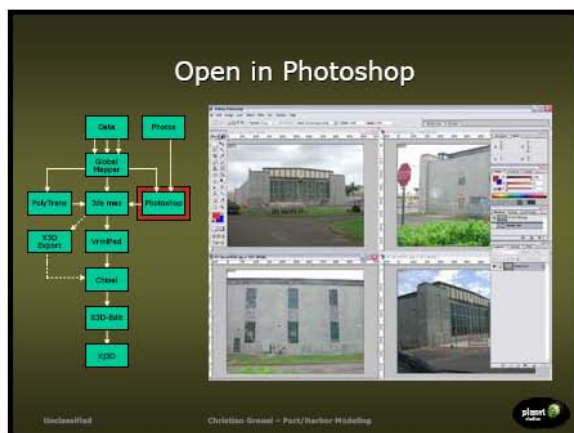
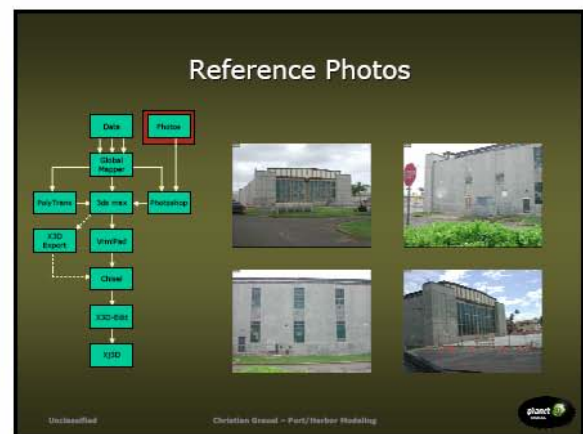
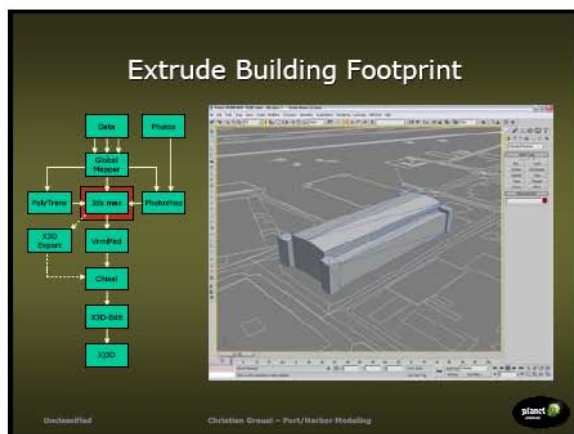
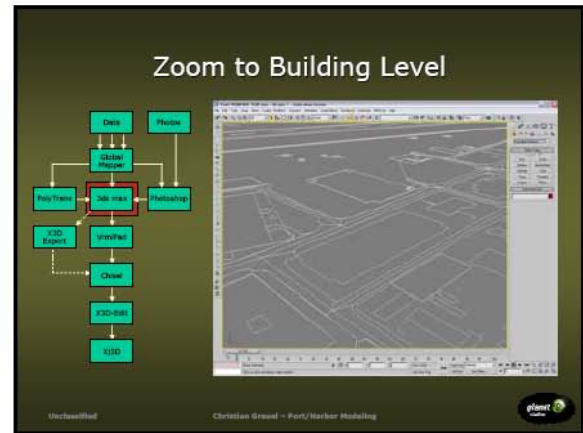
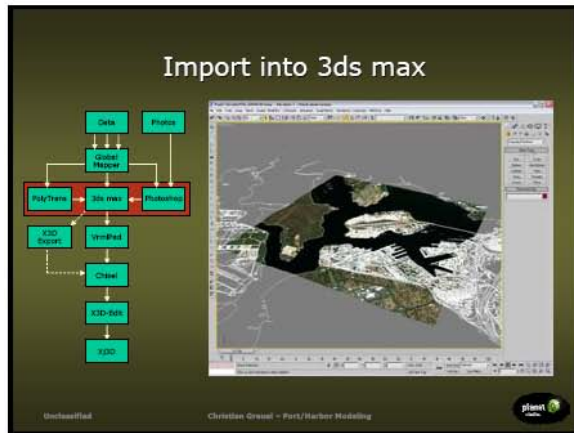


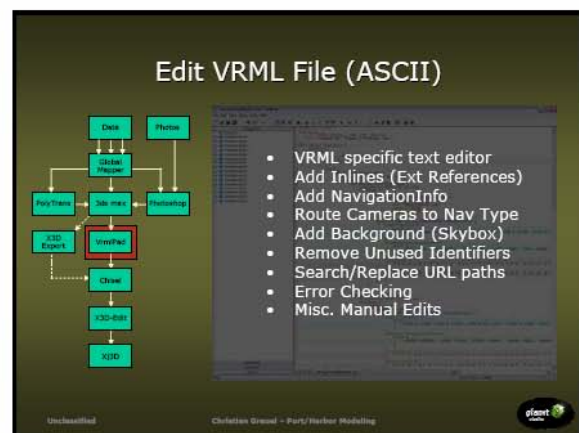
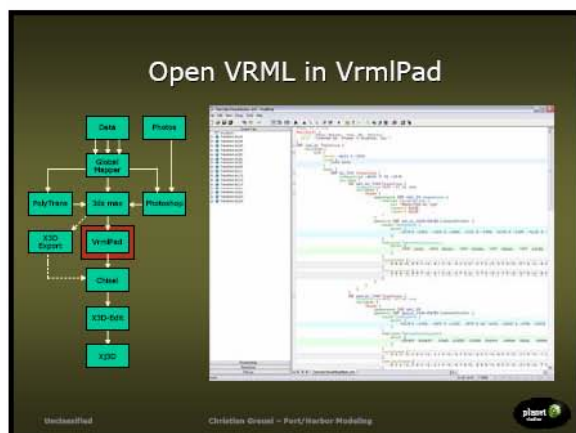
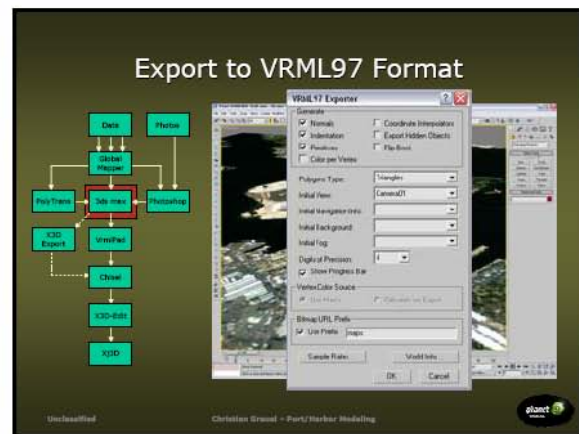
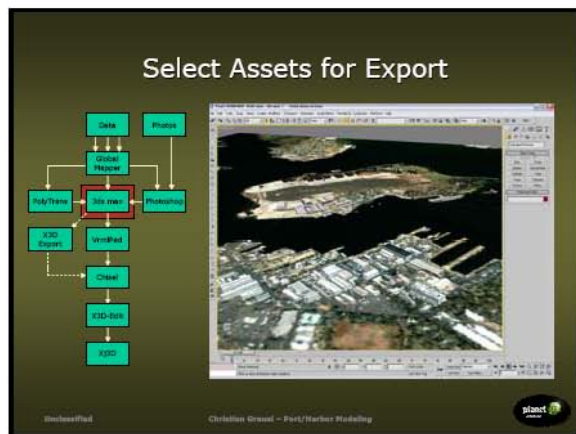
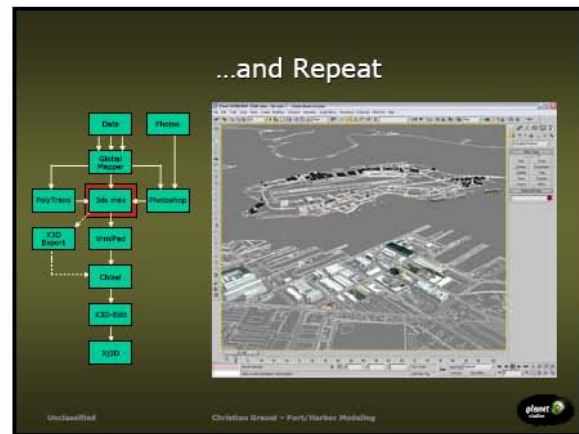
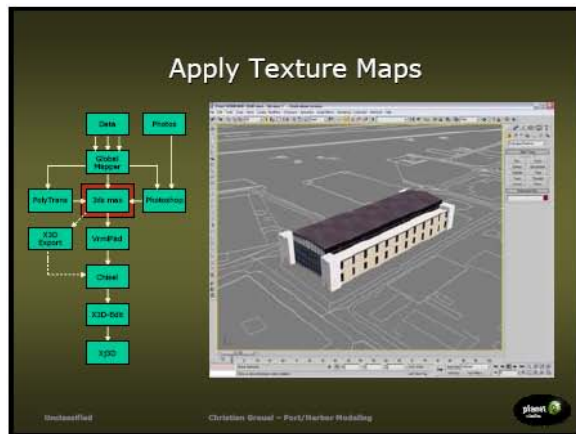
Unclassified

Christian Gressel - Port/Harbor Modeling









Load VRML in Chisel

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios

Edit VRML in Chisel

- Further optimization
- Create USE/DEF (instancing)
- Remove Material nodes
- Remove Useless nodes
- Remove Default fields

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios

Import VRML into X3D-Edit

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios

Edit X3D in X3D-Edit

- X3D specific XML editor
- Imports VRML as X3D
- Add Metadata
- Add GeoLocation
- Add GeoViewpoints
- Add GeoLODs
- etc.

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios

Finally: Load X3D into Xj3D

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios


Thank You!

Christian Greuel
Director of Art & Production
Planet 9 Studios
www.planet9.com
cgreuel@planet9.com

Unclassified Christian Greuel - Port/Harbor Modeling planet 9 studios

F.2 3D GEOSPATIAL DATA INTERFACES AND TOOLS

3D Geospatial Data Interfaces & Tools



David Colleen, CEO
Planet 9 Studios
San Francisco & Orlando
www.planet9.com

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

NAVFAC Project Goals

Alex Viana:

"The Naval Facilities Engineering Service Center (NFESC) began a pilot project in 1996 to evaluate if web based, virtual 3D geo-spatial waterfront facility models could be useful as training aids, and computer graphical user interfaces to facility information management databases. These efforts undertaken, in general, support the Office of the Secretary of Defense's strategic goals for the DoD's Infrastructure Information Environment:

- Improve awareness:** Inform real property professionals of the importance of accurate inventories to support planning and programming for future requirements.
- Enhance access:** Provide all potential Defense users real-time access to standardized real property information.
- Optimize resources:** Ensure that resources currently used to transfer and transform data are enhanced or reduced and redirected to improve maintenance and accessibility of data.

All work has been performed using open source, ISO based standards, and has been conducted in partnership with the private sector (Planet 9 Studios), academia (Naval Postgraduate School), and other Governmental Agencies (National Institutes of Standards and Technology). We believe that the results of our efforts may be a useful process for others."

NAVFAC

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

An Open Approach




Diagram illustrating the relationships between various ISO standards for 3D geospatial data:

- ISO 14772: vml, vmlr, VRML 97 Specification
- ISO 19776-1: x3dv Classic VRML Encoding
- ISO 19776-2: x3db Compressed Binary Encoding
- ISO 19776-3: x3db Compressed Binary Encoding (in progress)
- ISO 19777-1: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-2: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-3: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-4: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-5: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-6: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-7: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-8: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-9: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-10: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-11: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-12: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-13: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-14: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-15: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-16: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-17: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-18: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-19: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-20: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-21: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-22: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-23: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-24: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-25: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-26: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-27: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-28: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-29: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-30: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-31: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-32: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-33: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-34: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-35: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-36: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-37: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-38: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-39: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-40: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-41: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-42: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-43: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-44: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-45: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-46: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-47: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-48: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-49: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-50: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-51: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-52: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-53: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-54: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-55: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-56: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-57: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-58: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-59: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-60: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-61: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-62: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-63: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-64: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-65: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-66: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-67: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-68: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-69: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-70: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-71: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-72: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-73: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-74: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-75: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-76: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-77: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-78: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-79: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-80: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-81: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-82: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-83: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-84: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-85: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-86: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-87: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-88: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-89: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-90: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-91: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-92: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-93: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-94: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-95: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-96: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-97: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-98: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-99: Scene Access Interface (SAI) scripting API for Java
- ISO 19777-100: Scene Access Interface (SAI) scripting API for Java

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

Tools of the Trade

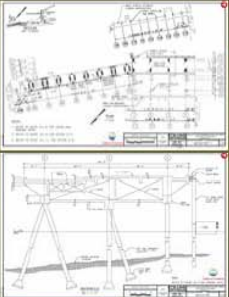
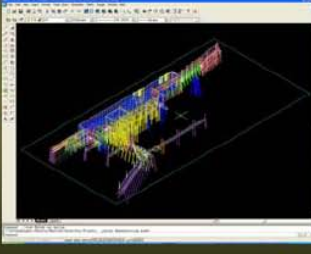
Tools That We Use:

- AutoDesk's AutoCAD
- 3D Studio Max
- Global Mapper
- Chisel
- Parallel Graphics' VRMLpad

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

Data Transformation Process

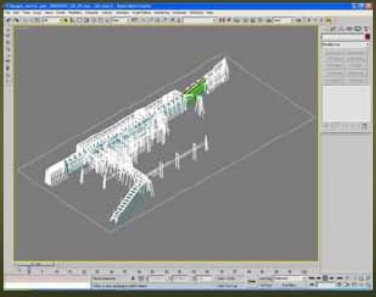
Step One – Transform existing 2D files (plan, elevation & sections obtained from design, as-built, and inspection files) into 3D AutoCAD models.

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

Data Transformation Process

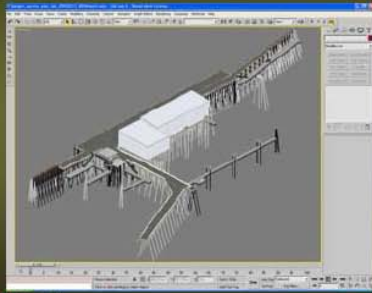
Step Two – Import 3D AutoCAD model into AutoDesk 3D Studio Max & Optimize Geometry.



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified

Data Transformation Process

Step Three – Within 3D Studio Max, apply material textures to the model surfaces, integrate geo-located data sets, preset facility views and create a Virtual Reality Modeling Language (VRML) file.

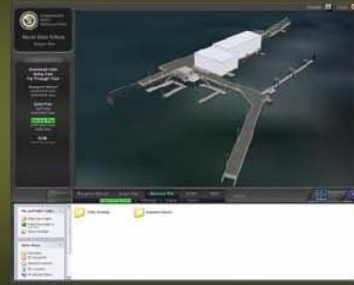


Collaborative 3D Geospatial Data Interfaces & Tools



Data Transformation Process

Step Four – Import waterfront facilities VRML file into Chisel for restructuring then translate to X3D. Customize HTML user interface with external data links.



Collaborative 3D Geospatial Data Interfaces & Tools



Data Access & Interoperability



Collaborative 3D Geospatial Data Interfaces & Tools



Case Studies



What's a BIM?
A 3D model linked to a database is now called a "Building Information Model"...or BIM for short.



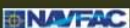
Collaborative 3D Geospatial Data Interfaces & Tools



Virtual Pearl Harbor



The Pearl Harbor BIM started as a base redevelopment planning tool. It was next used to develop renovation proposals for the Arizona Memorial. This was subsequently used on the National Parks Service public web site as a visitor orientation tool.....and so on. The current use of the BIM is for port security planning and training.



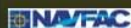
Collaborative 3D Geospatial Data Interfaces & Tools



Virtual Bangor



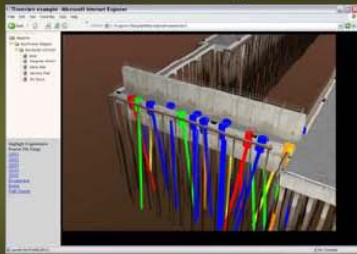
Our BIM has links to individual elements.



Collaborative 3D Geospatial Data Interfaces & Tools



Virtual Bangor



Piling maintenance scheduling is color coded.



Collaborative 3D Geospatial Data Interfaces & Tools



Virtual Bangor



Selecting another year populates the 3D scene with fresh data from the Access database..



Collaborative 3D Geospatial Data Interfaces & Tools



Port Management



Virtual Friday Harbor - Port Management Tool



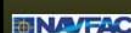
Collaborative 3D Geospatial Data Interfaces & Tools



ATFP - Port Protection



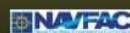
ABOT Iraqi Oil Platform - Anti-terrorist trainer



Collaborative 3D Geospatial Data Interfaces & Tools



Navy Virtual Earth



Collaborative 3D Geospatial Data Interfaces & Tools



Lessons Learned

- Existing AutoCAD data can be translated into open, interoperable formats..
- Evolution...not revolution.
- Project pre-planning saves time later.
- Work done in open standards still works.



Collaborative 3D Geospatial Data Interfaces & Tools



Other Related Projects



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



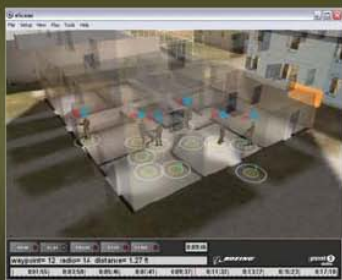
Army Landwarrior & Boeing IB-CSAS Programs



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Army Landwarrior & Boeing IB-CSAS Programs



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Army Landwarrior & Boeing IB-CSAS Programs



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Army FBCB2



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



DARPA HURT Program (Northrop Grumman/ SRI)



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



ARDA NIMD Program



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



CIA/NGA Traveler Program



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



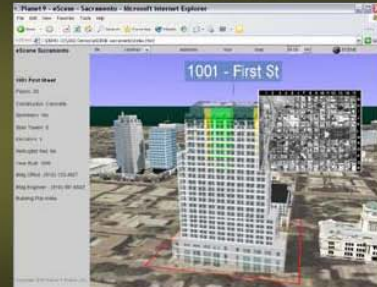
Emergency Response



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



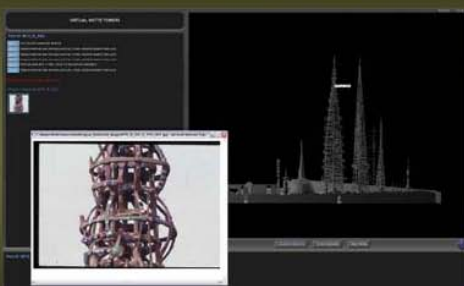
911 Call Centers



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



BIM's – Watts Towers



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Virtual Earth™



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Car Navigation



Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



Thank-You!



David Colleen
Planet 9 Studios
San Francisco & Orlando
www.planet9.com
dcolleen@planet9.com

Colleen-3D Geospatial Data Interfaces & Tools
Unclassified



APPENDIX G: PHASE II TECHNICAL SUMMARY OF X3D MODEL CONSTRUCTION

G.1 PLANET 9 STUDIOS ART TEAM

Overview

In February 2006, Planet 9 Studios received a scope of work (SOW) for Phase II of the Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection (AT/FP) Alternatives. The major portion of the SOW included the development of Extensible 3D (X3D) models of waterside buildings and underlying terrain at Pearl Harbor, Hawaii. This technical summary documents important attributes of these models, including naming conventions and hierarchical organization, which will be useful for understanding their structure and use.

Terrain

A geo-referenced X3D terrain model of Oahu was developed from 10-meter Spatial Data Transfer Standard (SDTS) Digital Elevation Model (DEM) files. This source data, originating from the U.S. Geological Survey (USGS), was obtained at no cost from the publicly available GeoCommunity™ website (www.geocomm.com). A total of sixteen individual quads comprising Honolulu County were required for complete coverage of the island. These quads were Haleiwa, Hauula, Honolulu, Kaena, Kaena OE W, Kahana, Kahuku, Kaneohe, Koko Head, Mokapu, Pearl Harbor, Scholfield Barracks, Waianae, Waimea, and Waipahu.

The individual quads were combined with Global Mapper software, a geographic data viewer and format converter. The dataset was then re-projected into the Universal Transverse Mercator (UTM) coordinate system with the following attributes:

Projection: UTM

Zone: 4

Datum: WGS84

Planar Units: Meters

This projection has two basic benefits. First, UTM is defined within Cartesian XYZ space, which is the same as that in which the 3D models are authored. Second, it uses meters as its planar unit, which is the same unit of measure used by X3D.

The combined elevation model of Oahu, measuring 75x60 km, was exported as a single DEM, re-sampled at a 500-meter resolution. The Pearl Harbor area (10x10 km), being the area of interest, was exported separately at a higher resolution of 50-meters. These new DEMs were imported into the 3D authoring tool, Autodesk 3dsmax, with the UTM coordinate (608354.00, 2362714.00, 0.00) being centered at its local XY origin (0, 0, 0). To the immediate area around Pearl Harbor, topographic data was integrated into the greater terrain model for further refinement.

The terrain models were each divided into separate sections along a regular grid. This allows for efficient view frustum culling as well as the implementation of Level of Detail (LOD) switching. The Oahu terrain was divided into a 5x4 grid, with each section measuring 15x15 km. The higher-resolution Pearl Harbor terrain was divided into an octagonal 4x4 grid, with each section measuring 2500x2500 meters. Each of these grid sections was then optimized by utilizing a polygon reduction modifier, automatically substituting larger triangles for continuous areas of the terrain mesh that were co-planar within an acceptable threshold.

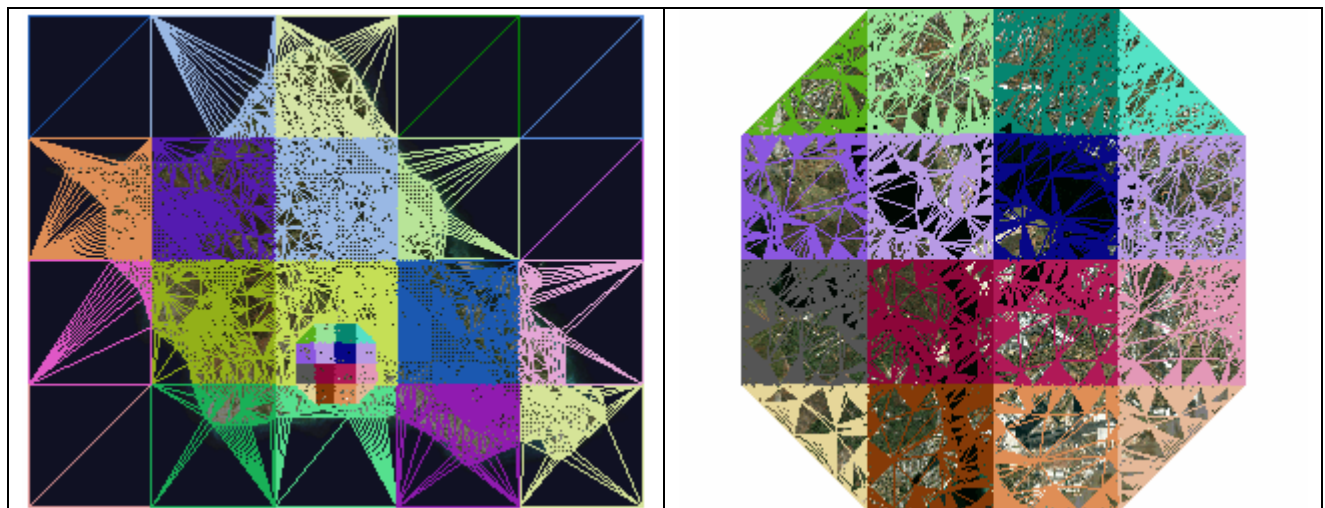


Figure 22. Oahu and Pearl Harbor Terrain Grids Optimized



Figure 23. Oahu and Pearl Harbor Terrains Drapped with Imagery

The Oahu terrain mesh was draped with color-corrected 30-meter Land Remote-Sensing Satellite (LANDSAT) imagery. This higher-resolution Pearl Harbor area was draped with 1-meter imagery originating from Space Imaging. This area included the Naval Station, Naval Shipyard, SUBASE, FISC, and Ford Island facilities. The resulting geometry was then optimized for real-time rendering, including the addition of Levels of Detail (LOD) for increased efficiency, and geo-referenced within the Universal Transverse Mercator (UTM) coordinate system as described in the X3D Geospatial specification.

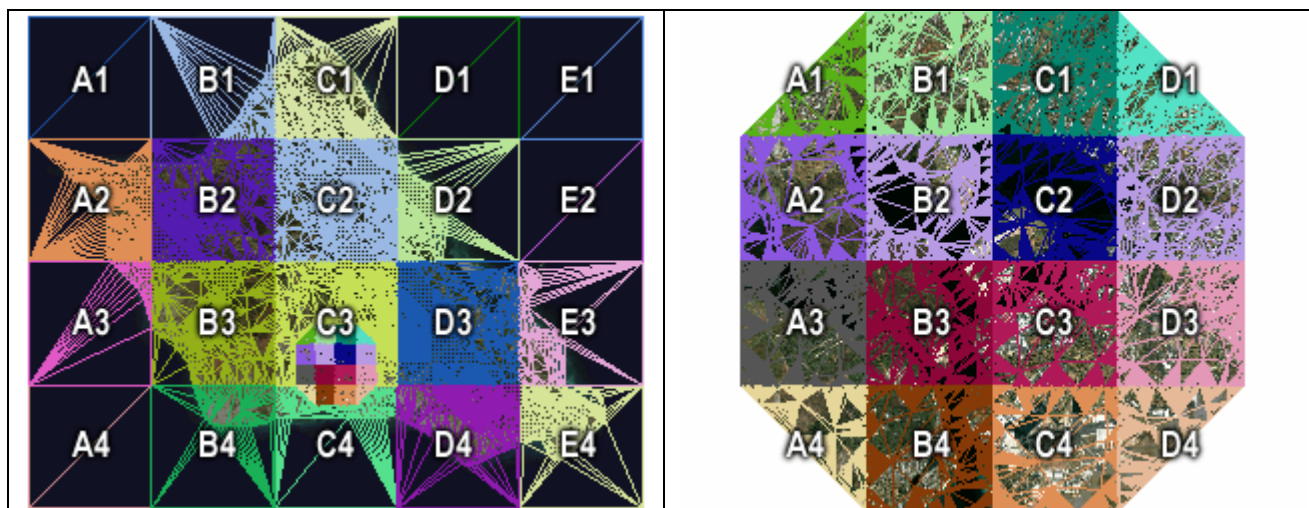


Figure 24. Oahu and Pearl Harbor Terrain Grids Designated

Given its low resolution, the entire Oahu terrain was able to be saved as a single X3D file, complete with the exception of the inlined octagonal Pearl Harbor area (see below). However, the Oahu terrain file does have five different versions, each named with a convention of “Terrain*.x3d”, where “*” identifies which version(s) of the Pearl Harbor terrain are to be inlined. Thus the file selected from one of the following five becomes the master terrain file:

Terrain.x3d	Oahu terrain; inlines 16 Pearl Harbor sections under GeoLODs: PearlHarborTerrain<col><row>High.x3d PearlHarborTerrain<col><row>Low.x3d
TerrainHigh.x3d	Oahu terrain; inlines 16 Pearl Harbor sections: PearlHarborTerrain<col><row>High.x3d
TerrainMed.x3d	Oahu terrain; inlines 16 Pearl Harbor sections: PearlHarborTerrain<col><row>Med.x3d
TerrainLowPiers.x3d	Oahu terrain; inlines 16 Pearl Harbor sections: PearlHarborTerrain<col><row>LowPiers.x3d
TerrainLow.x3d	Oahu terrain; inlines 16 Pearl Harbor sections: PearlHarborTerrain<col><row>Low.x3d

Table 3. Various Resolution Version of the Oahu Terrain

Within each version of the “Terrain*.x3d” files are two Group nodes. The group “OahuTerrain” contains each of the twenty grid sections of the greater Oahu terrain. Each grid section is individually named with a convention of “gnd_Oahu<col><row>”, where “gnd_” identifies the object as ground, “<col>” is a grid column letter from A-E starting from the left, and “<row>” is a grid row number from 1-4 starting from the top. Thus, the individual grid sections of Oahu are named as follow:

gnd_OahuA1	gnd_OahuB1	gnd_OahuC1	gnd_OahuD1	gnd_OahuE1
gnd_OahuA2	gnd_OahuB2	gnd_OahuC2	gnd_OahuD2	gnd_OahuE2
gnd_OahuA3	gnd_OahuB3	gnd_OahuC3	gnd_OahuD3	gnd_OahuE3
gnd_OahuA4	gnd_OahuB4	gnd_OahuC4	gnd_OahuD4	gnd_OahuE4

Table 4. Individual Grid Sections of the Oahu Terrain

Under the second group, “PearlHarborTerrain”, each of the external Pearl Harbor terrain grid section files is called. This is done via the X3D “GeoLOD” node, thereby allowing for LOD switching if desired.

The sixteen Pearl Harbor grid sections were each saved as individual X3D files named with a convention of “PearlHarborTerrain<col><row>*.x3d”, where “<col>” is a grid column letter from A-D starting from the left, “<row>” is a grid row number from 1-4 starting from the top, and “*” identifies the resolution of the terrain imagery. Note that the resolution component of these file names correspond to those of the master “Terrain*.x3d” files, which call them (see above). Example file names for the single Pearl Harbor grid section “A1” are as follow:

PearlHarborTerrainA1High.x3d	Pearl Harbor section A1, with High-Res 2048x2048 Terrain imagery. Includes Pier geometry.
PearlHarborTerrainA1Med.x3d	Pearl Harbor section A1, with Medium-Res 1024x1024 Terrain imagery. Includes Pier geometry.
PearlHarborTerrainA1LowPiers.x3d	Pearl Harbor section A1, with Low-Res 512x512 Terrain imagery. Includes Pier geometry.
PearlHarborTerrainA1Low.x3d	Pearl Harbor section A1, with Low-Res 512x512 Terrain imagery. DOES NOT include Pier geometry.

Table 5. Example File Names for the Single Pearl Harbor Grid Section A1

Each Pearl Harbor terrain grid section has four different LOD resolutions. These are differentiated not by geometry, but rather by the resolution of the texture map applied to each. The high resolution texture maps are each 2048x2048 pixels in size. The medium resolution texture maps are 1024x1024 pixels in size. The low resolution texture maps are 512x512 pixels in size. The one case of reduced geometric resolution is in those sections named “PearlHarborTerrain<col><row>Low.x3d”, in which the piers have been removed for increased performance when needed.

The texture maps for the terrain files are located in subdirectories of the Textures directory. The three numbered subdirectories contain the texture maps for the section grids of the various terrain resolutions: High (2048), Medium (1024), and Low (0512). Note the use of the leading zero for the three-digit number. The Oahu subdirectory contains the texture maps for the section grids of the greater Oahu terrain. Finally, the Wharfs subdirectory contains texture maps for the various piers and wharfs.

Buildings

Atop the terrain were constructed geo-referenced X3D models of the majority of Navy buildings visible from the water, as well as piers and wharves attached to the facilities. The piers and wharves are part of the terrain files (see above). The location and footprint of each structure was extracted from the provided computer-aided drafting (CAD) files, which had been manually rectified to the geo-referenced terrain. The footprints were then extruded and the resulting geometry modified to create a representational 3D model of each structure. To these were applied texture maps which were derived from the location photographs, thereby resulting in a photo-realistic model for use in the AT/FP simulation software.

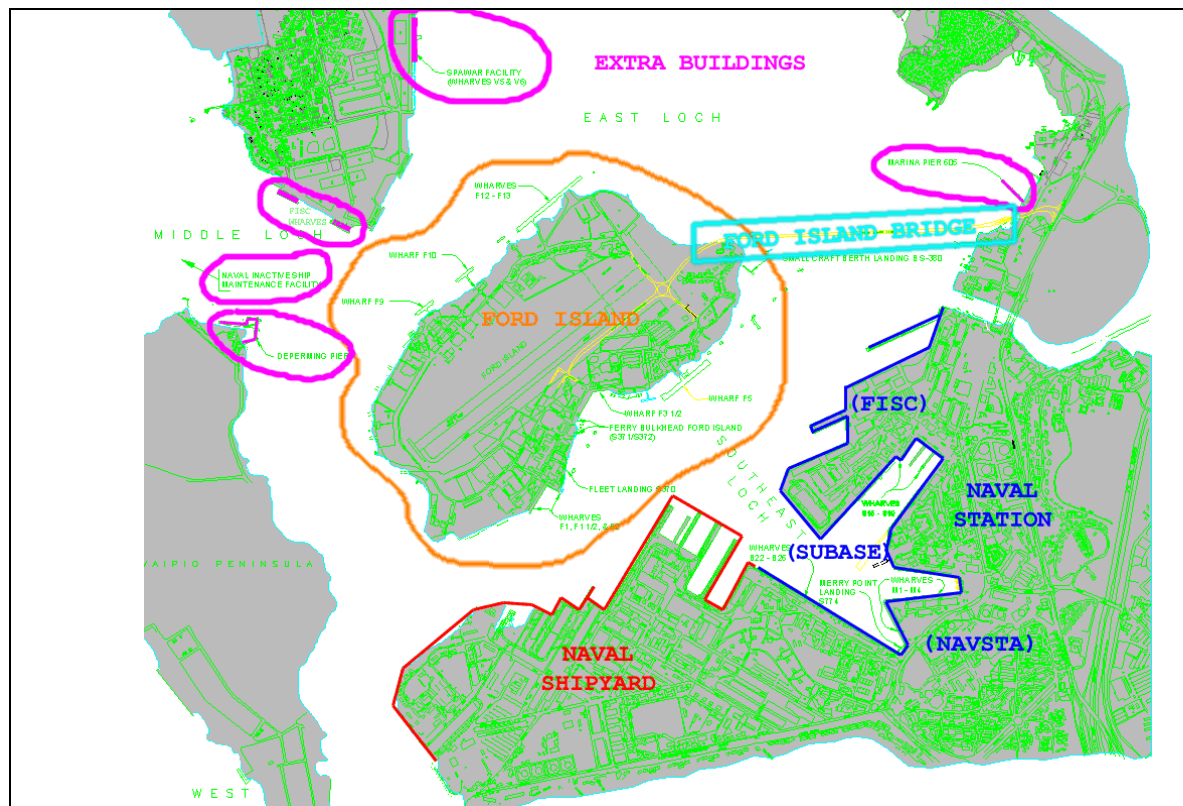


Figure 25. Pearl Harbor Buildings Grouped into Five Separate Files Determined by Location.

For ease of management, the buildings were grouped into five separate files determined by their location. These areas are Ford Island, Ford Island Bridge, Naval Shipyard, Naval Station (comprised of NAVSTA, SUBASE, and FISC), and Extra Buildings (outlying area amongst and beyond the lochs). Furthermore, each of these five files is available with two different resolutions of texture maps applied. The high resolution version is identified simply as the name of the

location (e.g. `FordIsland.x3d`). The low resolution version appends the word “Low” to the file base name (e.g. `FordIslandLow.x3d`)

In turn, the appropriate resolution of these five location files is called by one of five master building files. These go by the naming convention of “`Building*.x3d`”, where “*” identifies which version of the building locations are to be inlined. Additionally, the master building file will also call one of the previously discussed master terrain files. Thus the file selected from one of the following five becomes the overall master file.

<code>Buildings.x3d</code>	High-Res Building files inlined; Terrain inlined via: <code>Terrain.x3d</code>
<code>BuildingsHigh.x3d</code>	High-Res Building files inlined; Terrain inlined via: <code>TerrainHigh.x3d</code>
<code>BuildingsMed.x3d</code>	High-Res Building files inlined; Terrain inlined via: <code>TerrainMed.x3d</code>
<code>BuildingsLowPiers.x3d</code>	Low-Res Building files inlined; Terrain inlined via: <code>TerrainLowPiers.x3d</code>
<code>BuildingsLow.x3d</code>	Low-Res Building files inlined; Terrain inlined via: <code>TerrainLow.x3d</code>

Within each location file (e.g. `FordIsland.x3d`) reside the individual building models. Each building is situated underneath its own Transform node. Whenever possible, this Transform node is named for its designated building number as identified on the provided CAD drawings. The naming convention for the buildings then becomes “`bld_<num>`”, where “`bld_`” identifies the object as a building, and “`<num>`” is the designated number of the building. In cases where a building number was not to be found in the CAD drawings, a unique number was assigned to it.

Furthermore, clusters of buildings in close proximity to each other are grouped together as blocks for more efficient scene culling. It should be noted that these block groupings are not officially recognized blocks, but rather chosen with spatial considerations for best performance. The naming convention for the blocks is “`blk_<num>`”, where “`blk_`” identifies the grouping as a block, and “`<num>`” is a unique number assigned to the block.

The texture maps for the building files are located in subdirectories of the Textures directory. First, they are segregated into either the High or Low subdirectory. Below this there are subdirectories for each of the building locations. In general, the texture maps in the Low subdirectories are 50% of the pixel size of those in the High subdirectories.

```

[Locations]
|-- [PearlHarborBuildings]
|   |-- Buildings.x3d                                <-- OVERALL MASTER FILE
|       |-- FordIsland.x3d
|           |-- [Maps]
|               |-- [High]
|                   |-- [FordIsland]
|       |-- FordIslandBridge.x3d
|           |-- [Maps]
|               |-- [High]
|                   |-- [FordIslandBridge]
|       |-- NavalShipyard.x3d
|           |-- [Maps]
|               |-- [High]
|                   |-- [NavalShipyard]
|       |-- NavalStation.x3d
|           |-- [Maps]
|               |-- [High]
|                   |-- [NavalStation]
|       |-- ExtraBuildings.x3d
|           |-- [Maps]
|               |-- [High]
|                   |-- [ExtraBuildings]
|
[../PearlHarborTerrain]
|-- Terrain. x3d                                <-- MASTER TERRAIN FILE
|   |-- PearlHarborTerrainA1High.x3d
|       |-- [Maps]
|           |-- [2048]
|   |-- PearlHarborTerrainA1Low.x3d
|       |-- [Maps]
|           |-- [0512]
|   |-- ...
|   |-- PearlHarborTerrainD4High.x3d
|       |-- [Maps]
|           |-- [2048]
|   |-- PearlHarborTerrainD4Low.x3d
|       |-- [Maps]
|           |-- [0512]
|   |-- [Oahu]
|   |-- [Wharfs]

```

Figure 26. Example Building/Terrain File Dependency Chart

Aids to Navigation

A series of Aids to Navigation (ATON) X3D PROTO and X3D (non-PROTO) models was produced to allow the virtual waterways to be populated with charted marks as they are in the actual world. Each ATON model can be positioned and oriented at precise locations within a given scene. The selection includes the following models:

- Danger Daybeacon (non-PROTO)
- Daybeacon
- Light
- Lighted Buoy
- Light Post
- Marker Buoy (non-PROTO)
- Mooring Buoy (non-PROTO)
- Range Light

The ATON X3D PROTO models contain switches to allow assignment of a few specific attributes such as port (green) vs. starboard (red), light on vs. light off, and brightness of light glow. These attributes are based upon International Hydrographic Organization (IHO) S-57 (<http://www.caris.com/s-57>). This standard, prepared by the IHO Committee on Hydrographic Requirements for Information Systems (CHRIS), is for the coding and exchange of hydrographic digital data.

While only a few attributes are currently available, a comprehensive system of ATON X3D PROTO models with attributes adherent to S-57 is the subject of a future scope of work. These would include defined options such as numeric designation, types of sounds, and precise light flashing characteristics. The system would also include a more complete selection of ATON types (e.g. Cans/Nuns, Mileboards, Warning Markers, etc.)

Following is a list of the ATON X3D PROTO models that currently exist, along with their available attributes and options:

RangeLightPrototype.x3d		
Attribute	Default	Option
LightType	1	0=LightOff, 1=LightOn, 2=LightFlashing(NotImplemented)
LightGlow	1 1 1	XYZ Scale of Light Glow Effect

Table 6. Aids to Navigation X3D Proto Models - RangeLight

NOTE: Range Light model points due North (-Z) and its light glow effect is only visible from that direction. The Range Light should be rotated into its proper orientation.

DaybeaconPrototype.x3d		
Attribute	Default	Option
Catlam* *Category of Lateral Marker	1	0=None(Unlikely), 1=PortHand(GreenSquare), 2=StarboardHand(RedTriangle), 3=PreferredChannelToStarboard(TopmostBandGreen), 4=PreferredChannelToPort(TopmostBandRed)
Number	N/A	Not Implemented

Table 7. Aids to Navigation X3D Proto Models – Daybeacon

LightPrototype.x3d		
Attribute	Default	Option
Catlam* *Category of Lateral Marker	1	0=None(Unlikely), 1=PortHand(GreenSquare), 2=StarboardHand(RedTriangle), 3=PreferredChannelToStarboard(TopmostBandGreen), 4=PreferredChannelToPort(TopmostBandRed)
LightType	1	0=LightOff, 1=LightOn, 2=LightFlashing(NotImplemented)
LightGlow	1 1 1	XYZ Scale of Light Glow Effect
PileType	1	0=NoPile(Unlikely), 1=SinglePile, 2=MultiPile
Number	N/A	Not Implemented

Table 8. Aids to Navigation X3D Proto Models – LightPrototype

LightedBuoyPrototype.x3d		
Attribute	Default	Option
Catlam* *Category of Lateral Marker	1	0=None(Unlikely), 1=PortHand(GreenSquare), 2=StarboardHand(RedTriangle), 3=PreferredChannelToStarboard(TopmostBandGreen), 4=PreferredChannelToPort(TopmostBandRed)
LightType	1	0=LightOff, 1=LightOn, 2=LightFlashing(NotImplemented)
LightGlow	1 1 1	XYZ Scale of Light Glow Effect
Number	N/A	Not Implemented

Table 9. Aids to Navigation X3D Proto Models – LightedBouyPrototype

LightPostPrototype.x3d		
Attribute	Default	Option
LightType	1	0=LightOff, 1=LightOn, 2=LightFlashing(NotImplemented)
LightGlow	1 1 1	XYZ Scale of Light Glow Effect

Table 10. Aids to Navigation X3D Proto Models – LightPostPrototype

```

EXTERNPROTO RangeLight [
    exposedField SFInt32 LightType
    exposedField SFVec3f LightGlow
]
"RangeLightPrototype.wrl#RangeLight"

EXTERNPROTO Light [
    exposedField SFInt32 Catlam
    exposedField SFInt32 LightType
    exposedField SFVec3f LightGlow
    exposedField SFInt32 PileType
    exposedField SFInt32 Number
]
"LightPrototype.wrl#Light"

DEF RangeLightFront Transform {
    translation 0 0 -25
    rotation 0 1 0 3.14159      # Rotate to face South
    children [
        RangeLight {
            LightType 1          # Light On
            LightGlow 4 4 1      # Glow effect scaled 4x wide (XY only)
        }
    ]
}

DEF LightPort Transform {
    translation -10 0 25
    children [
        Light {
            Catlam 1              # Green (Port)
            LightType 1          # Light On
            LightGlow 2 2 2      # Glow effect scaled two times (XYZ)
            PileType 1           # Single Pile
        }
    ]
}

DEF LightStarboard Transform {
    translation 10 0 25
    children [
        Light {
            Catlam 2              # Red (Starboard)
            LightType 1          # Light On
            LightGlow 2 2 2      # Glow effect scaled two times (XYZ)
            PileType 2           # Multi Pile
        }
    ]
}

```

Figure 27. Example ATON X3D Code (VRML Syntax) with One Range Light and Two Lights

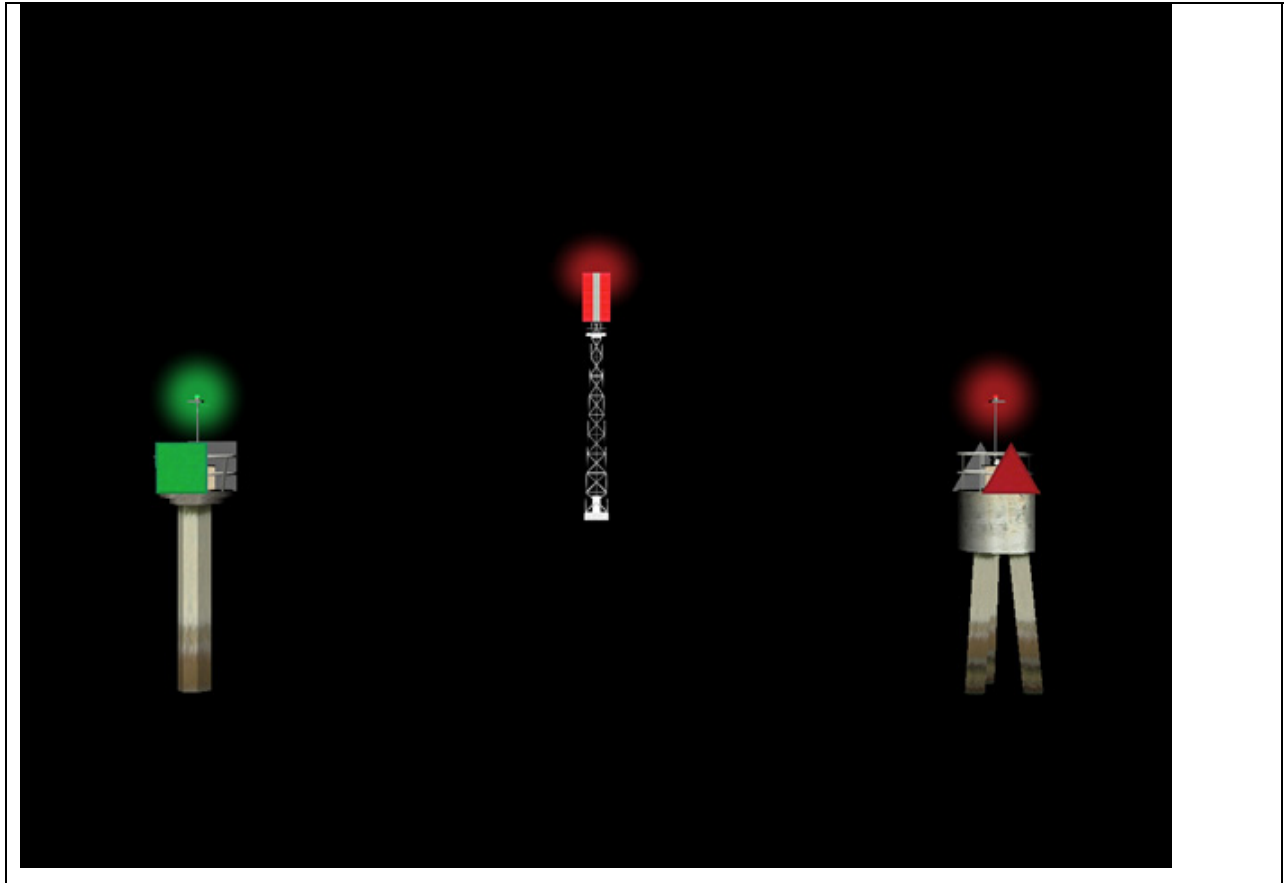


Figure 28. Example ATON X3D Scene with One Range Light (in distance) and Two Lights

Compass Rose

Planet 9 Studios was asked create a 2D Compass Rose X3D PROTO for general direction finding, to be displayed in a Heads-Up Display (HUD) manner over any given X3D scene. The resulting model consists of a texture mapped compass face which rotates in direct correlation with the orientation of the user's viewpoint.

While the visual components were complete with basic functionality in place, the file was not finished as of the final report. There remained an issue of gimbal lock in the compass rotation, due to the fact that it was tied to the viewpoint orientation via the X3D "ProximitySensor" node. The visual artifact is not noticeable when the viewpoint is parallel to the ground, but becomes apparent when viewpoint is pitched up or down. It has been suggested that a quaternion approach may need to be implemented in order to alleviate this issue. This has been documented as XMSF Issue Tracker Bug #1009.

The Compass Rose X3D PROTO includes two attributes which may be assigned values. The first is the location offset, which defines the position of the compass face relative to the center of the user's screen. The second attribute is the size, which is the XYZ scaling of the compass face default dimensions.

CompassRosePrototype.x3d		
Attribute	Default	Option
locationOffset	0 0 0	XYZ Modified screen location
size	1 1 1	XYZ Modified compass size

Table 11. Compass Rose X3D PROTO Attributes and Options

```

EXTERNPROTO CompassRose [
    field SFVec3f locationOffset
    field SFVec3f size
]
"CompassRosePrototype.x3d"

Inline {
    url    "CheckeredGround.x3d"
}

CompassRose {
    locationOffset -0.075 -0.045 0
    size 1 1 1
}

```

Figure 29. Example Compass Rose X3D Code (VRML Syntax)

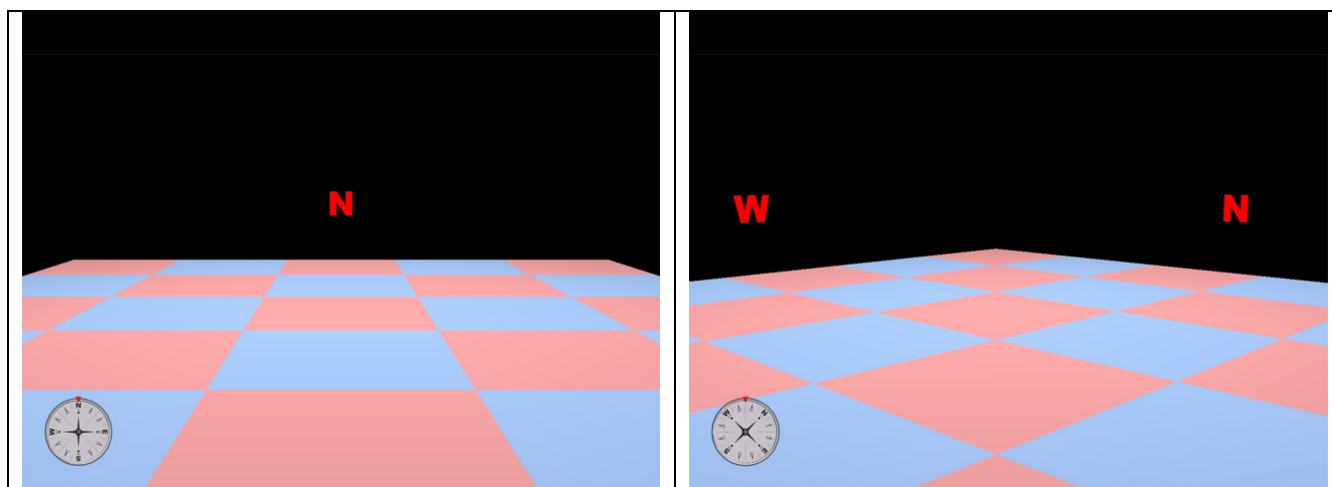


Figure 30. X3D Example Compass Rose Scene First Looking North, then Northwest

Port Security Barrier

Planet 9 Studios provided a copy of its previously existing Port Security Barrier (PSB) X3D PROTO of the for release into the open source FOUO SavageDefense archive, thereby allowing it to be freely used and modified accordingly.

This X3D PROTO defines a single section of a Port Security Barrier. A complete barrier system is created by stringing several PSB sections together in a continuous line. In most cases, one PSB section will be coupled to the next section via a normal hardware connection. However, in cases where the PSB section must be connected to a special float, as in the case of a gate opening, then it must have a special connector. The X3D PROTO allows the option to choose such a connector, either on the left or right side of the section, via the attribute “whichChoice”.

Other attributes include “translation” and “rotation”. These two attributes are not truly necessary, of course, as a transform could just as easily be applied to the X3D node from which the PROTO is called. Note that each PSB section is 15.3 meters in length, and so this would be the standard translation offset when there is a straight line of these running along a primary axis (see example code below).

The PSB model contains four levels of detail (LODs). The highest LOD consists of 2134 polygons, while the lowest is made up of only 60 polygons. Depending on the number of PSB sections included in a given scene, the “range” field of the X3D “LOD” node may need to be adjusted within the X3D PROTO code to enable the higher LODs to switch out sooner. In this way, the overall performance may be increased.

PortSecurityBarrierPrototype.x3d		
Attribute	Default	Option
translation	0 0 0	XYZ Modified position of barrier section
rotation	0 1 0 0	Modified rotation of barrier section, in vector (XYZ) and rotation (radians)
whichChoice	0	0=Normal, 1=LeftConnector, 2=RightConnector

Table 12. Port Security Barrier X3D PROTO attributes and options:

```

EXTERNPROTO Barrier [
    exposedField SFVec3f translation
    exposedField SFRotation rotation
    exposedField SFInt32 whichChoice
]
"PortSecurityBarrierPrototype.x3d"

Barrier {
# normal barrier section, i.e. it only connects to other barriers.
    whichChoice 0
    translation 0 0 0
}
Barrier {
# special barrier section with connection hardware on its left side.
    whichChoice 1
    translation -15.3 0 0
}
Barrier {
# special barrier section with connection hardware on its right side.
    whichChoice 2
    translation 15.3 0 0
}

```

Figure 31. Example PSB X3D Code (VRML Syntax)

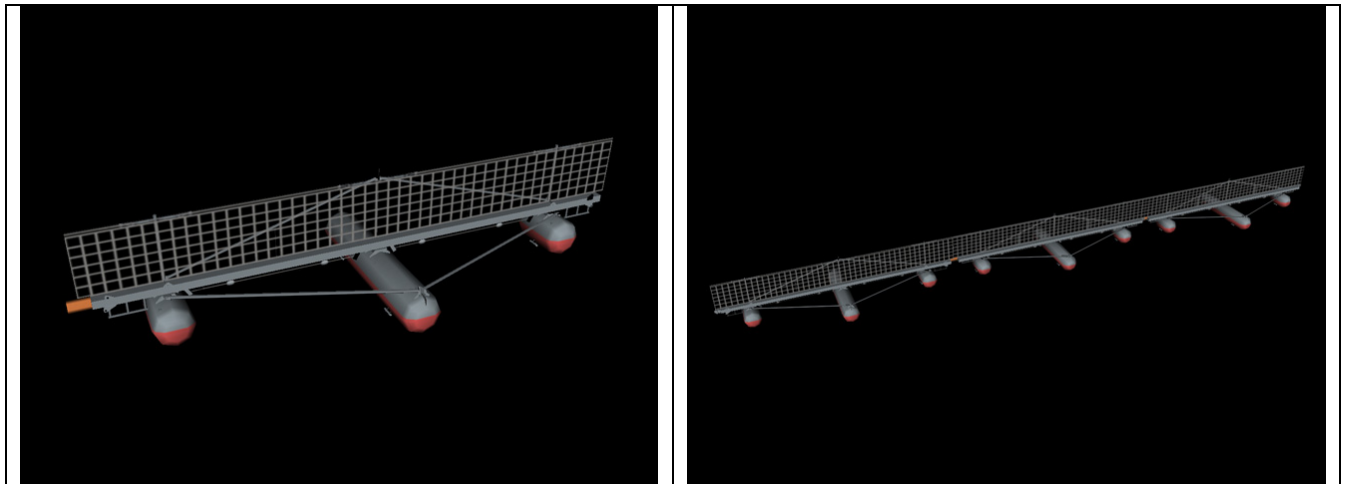


Figure 32. X3D Example PSB Scene First with One Section, then Three Sections.

X3D Model Locations

Terrain:

<https://savagedefense.nps.navy.mil/SavageDefense/Locations/PearlHarborTerrain/index.html>

Buildings:

<https://savagedefense.nps.navy.mil/SavageDefense/Locations/PearlHarborBuildings/index.html>

Aids to Navigation:

<https://savage.nps.edu/Savage/HarborEquipment/NavigationAids/index.html>

Compass Rose:

<https://savage.nps.edu/Savage/Tools/HeadsUpDisplays/index.html>

Port Security Barriers:

<https://savagedefense.nps.navy.mil/SavageDefense/HarborEquipment/FloatingBarriers/index.html>

Planet 9 Studios AT/FP Art Team

David Colleen, CEO

Christian Greuel, Director of Art & Production

Danny Lee, 3D Artist

Carlos Newcomb, 3D Artist

Ken Rhee, 3D Artist

THIS PAGE INTENTIONALLY LEFT BLANK

GLOSSARY OF TERMS AND ACRONYMS

2D	Two Dimensional
3D	Three Dimensional
ABOT	Al-Basrah Oil Terminal
ACTD	Advanced Concept Technology Demonstration
API	Application Program Interface
ARES	Applied Research and Engineering Sciences
ATD	Atmospheric Transport and Dispersion
AT/FP	Anti-Terrorism/Force Protection
ATON	Aids to Navigation
AUV	Autonomous Unmanned Vehicle
AVERT	Automated Vulnerability Evaluation for Risks of Terrorism
BAA	Broad Agency Announcement
C2	Command and Control
C4I	Command, Control, Communications, Computers, and Intelligence
CAC	Common Access Card
CAD	Computer Aided Design
CASS	Comprehensive Acoustic Simulation System
CAW	Center for Asymmetric Warfare
C-BML	Coalition Battle Management Language
CCRTS	Command and Control Research and Technology Symposium
CEO	Chief Executive Officer
CFFC	Commander, Fleet Forces Command
CFO	Chief Financial Officer

CHDS	Center for Homeland Defense and Security
CHRIS	IHO Committee on Hydrographic Requirements for Information Systems
CNI	Commander, Navy Installations
CNIC	Commander, Navy Installations Command
CNO	Chief of Naval Operations
CONOPS	Concept of Operations
CONUS	Continental United States
CVS	Concurrent Versioning System
DARPA	Defense Advanced Research Projects Agency
DARWARS	DARPA-funded program for achieving training superiority
DEM	Digital Elevation Model
DES	Discrete Event Simulation
DHS	Department of Homeland Security
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
DNC	Digital Nautical Chart
DoD	Department of Defense
DOE	Design of Experiments
DTED	Digital Terrain Elevation Data
EHSS	Electronic Harbor Security System
EPiCS	Emergency Preparedness Incident Command Simulation
EO	Electro-optical
EOD	Explosive Ordnance Disposal

ERDC	Engineer Research and Development Center
ESRI	Environmental Systems Research Institute
FAQ	Frequently Asked Questions
FIRST	Financial Institution Risk Strategy Tool
FISC	Fleet Industrial Supply Center
FOM	Figure of Merit
FOUO	FOR OFFICIAL USE ONLY
GIG	Global Information Grid
GIS	Geographic Information System
GOPLATS	Gas and Oil Platforms
GPS	Global Positioning System
GUI	Graphical User Interface
HiRSA	High-Resolution Situational Awareness
HLA	High Level Architecture
HPC	High Performance Computing
HUD	Heads up Display
HSDL	Homeland Security Digital Library
IEEE	Institute for Electronic and Electrical Engineers
IHO	International Hydrographic Organization
I/ITSEC	Interservice/Industry Training, Simulation, and Education Conference
Inc.	Incorporated
IR	Infra-red
IT/21	Information Technology for the 21 st Century

JCATS	Joint Conflict and Tactical Simulation
JMBL	Joint METOC Broker Language
ICA	Independent Computing Architecture
ISO	International Standards Organization
LANDSAT	Land Remote-Sensing Satellite
LIDAR	Light Detection and Ranging
LOD	Level of Detail
LT	Lieutenant
M&S	Modeling and Simulation
MCAS	Marine Corps Air Station
METOC	Meteorological and Oceanographic Center
MIL-STD	Military Standard
MOE	Measure of Effectiveness
MOP	Measure of Performance
MOVES	Modeling, Virtual Environments, and Simulation
MSDL	Military Scenario Definition Language
NAS	Naval Air Station
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organization
NAVFAC	Naval Facilities Command
NAVMAG	Naval Magazine
NAVSEA	Naval Sea Systems Command
NAVSTA	Naval Station
NFESC	Naval Facilities Engineering Service Center

NMCI	Navy-Marine Corps Internet
NOAA	National Oceanic and Atmospheric Administration
NPS	Naval Postgraduate School
NSWC	Naval Surface Weapons Center
NUWC	Naval Undersea Warfare Center
OCONUS	Outside Continental United States
ODE	Open Dynamics Engine
OPNAV	Office of the Chief of Naval Operations
ONR	Office of Naval Research
OR	Operations Research
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PEO	Program Executive Office
Ph.D.	Doctor of Philosophy
Pkill	Probability of Kill
PMS	Program Manager Surface
POA&M	Plan of Actions and Milestones
POC	Point of Contact
PSB	Port Security Barrier
R&D	Research and Development
RFID	Radio Frequency Identification
RHIB	Rigid Hull Inflatable Boat
RSIMS	Regional Shore Installation Management System
RTI	Run-Time Infrastructure

S&ST	Sound and Sea Technologies
SAIC	Science Applications International Corporation
SAVAGE	Scenario Authoring and Visualization for Advanced Graphical Environments
SBIR	Small Business Innovative Research
SDTS	Spatial Data Transfer System
SecForDMT	Security Forces Distributed Mission Training
SEDRIS	Synthetic Environment Data Representation Interchange Standard
SIPRNET	Secret Internet Protocol Router Network
SISO	Simulation Interoperability Standards Organization
SIW	Simulation Interoperability Workshop
SCORM	Shareable Content Object Reference Model
SMAL	Savage Modeling and Analysis Language
SOP	Standard Operating Procedures
SOW	Statement of Work
SPAWAR	Space and Naval Warfare Command
SQL	Standard Query Language
STRATA	Synthetic Teammates for Realtime Anywhere Training and Assessment
STRI	Simulation, Training, and Range Instrumentation
SUBASE	Submarine Base
SWAT	Special Weapons and Tactics
TENA	Test and Training Enabling Architecture
TRAC	TRADOC Analysis Center
TRADOC	Training and Doctrine Command

UI	User Interface
UML	Unified Modeling Language
UNO	University of Nebraska, Omaha
URL	Uniform Resource Locator
USGS	United States Geological Survey
USN	United States Navy
UTM	Universal Transverse Mercator
VC	Visual C++
V&V	Verification and Validation
VR	Virtual Reality
VRML	VR Modeling Language
VV&A	Verification, Validation, and Accreditation
WEAVER	Web-Enabled Architecture for Visualization, Evaluation and Research
WSMR	White Sands Missile Range
WSS	Waterside Security
X3D	Extensible 3D Graphics
Xj3D	Extensible Java™ API for X3D
XML	Extensible Markup Language
XMSF	Extensible Modeling and Simulation Framework
XSBC	XML Schema-based Binary Compression
XTC	XML-based Tactical Chat
ZLIB	An Open Source Compression Library

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

- “Building High Resolution City Models... Evolving Standards”, David Colleen, Christian Greuel, Charles Newcomb, Danny Lee; Planet 9 Studios; IMAGE 2005 Conference; July 2005; http://www.web3d2006.org/slides/Digital_Cities_whitepaper_P9.pdf
- “CARIS S-57 ENC Object Catalogue, Edition 3.1;” <http://www.caris.com/s-57>
- “Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives Phase I Statement of Work,” Naval Facilities Engineering Service Center tasking to the Naval Postgraduate School, August 2005.
- “Modeling and 3D Visualization for Evaluation of Anti-Terrorism/Force Protection Alternatives Phase II Statement of Work,” Naval Facilities Engineering Service Center tasking to the Naval Postgraduate School, March 2006.
- Blais, C., Brutzman, D., Drake, D., Moen, D., Morse, K., Pullen, M., and Tolk, A., “Extensible Modeling and Simulation Framework (XMSF) 2004 Project Summary Report,” Technical Report NPS-MV-05-002, Naval Postgraduate School, Monterey, California, 28 February 2005. Retrieved October 2006 from: <http://library.nps.navy.mil/uhtbin/cgiisirs/Tue+Oct+10+22:51:44+PDT+2006/SIRSI/0/520/NPS-MV-05-002.pdf>
- Brutzman, D., M. Zyda, J.M. Pullen, and K.L. Morse, “Extensible Modeling and Simulation Framework (XMSF): Challenges for Web-Based Modeling and Simulation,” Naval Postgraduate School, October 2002. Retrieved October 2006 from: <http://www.movesinstitute.org/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf>
- Brutzman, D., Buss, A., and Blais, C., “Connecting Simkit Discrete Event Simulation (DES) and the Naval Simulation System (NSS) via Web Services for Extensible Modeling & Simulation (XMSF)-Capable Analysis,” Naval Postgraduate School, 27 September 2004. Retrieved October 2006 from: <http://terra.cs.nps.navy.mil/www.movesinstitute.org/n81.pdf>
- Brutzman, D. P., Blais, C. L., and Norbraten, T. D., “Naval Installation Security Modeling and Simulation Workshop Summary Report,” Technical Report NPS-MV-06-001, Naval Postgraduate School, Monterey, CA, 12 May 2006. Designated For Official Use Only (FOUO). Available on CD-ROM to U.S. Government Agencies and their Contractors. See Appendix D.
- Buss, Arnold H. and Paul Sanchez, “[Simple Movement and Detection in Discrete Event Simulation](#),” *Proceedings of the 2005 Winter Simulation Conference*, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

Cioppa, T. M., “Efficient Nearly Orthogonal and Space-Filling Experimental Designs for High-Dimensional Complex Models,” Ph.D. Dissertation, Naval Postgraduate School, Monterey, CA, September 2002. Retrieved October 2006 from:
http://theses.nps.navy.mil/02sep_Cioppa_PhD.pdf

Garrood, D., “Review of Current Modeling and Simulation Software within the Department of Defense Community,” Sound and Sea Technology, February 10, 2006.

Garrood, D., Meggitt, D. and Wilson, J., “Feasibility Report of Modeling and Simulation Software for Anti-Terrorist/Force Protection Ashore Harbor Vulnerability Assessment,” January 20, 2006.

Harney, J. W., “Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships Using X3D Graphics and Agent-Based simulation,” Master’s Thesis, Naval Postgraduate School, Monterey CA, March 2003. Retrieved October 2006 from: http://theses.nps.navy.mil/03Mar_Harney.pdf

Homeland Security Affairs is the online journal of the Center for Homeland Defense and Security (<http://www.chds.us>) at the Naval Postgraduate School and is a peer-reviewed journal providing a forum to propose and debate strategies, policies, and organizational arrangements to strengthen U.S. Homeland Security. See <http://www.hsaj.org/hsa>.

Homeland Security Digital Library (HSDL) is a collection of homeland security policy and strategy related documents. See <https://www.hSDL.org>.

“Light List, Vol. 6, Pacific Coast and Pacific Islands”; United States Coast Guard; July 2006;
<http://www.navcen.uscg.gov/pubs/LightLists/V6COMPLETE.PDF>

Rauch, T. M., “Savage Modeling and Analysis Language (SMAL): Metadata for Tactical Simulations and X3D Visualizations,” Master’s Thesis, Naval Postgraduate School, Monterey, CA, March 2006. Retrieved October 2006 from:
http://theses.nps.navy.mil/06Mar_Rauch.pdf

Sullivan, P. J., “Evaluating the Effectiveness of Waterside Security Alternatives for Force Protection of Navy Ships and Installations Using X3D Graphics and Agent-Based Simulation,” Master’s Thesis, Naval Postgraduate School, Monterey, CA, September 2006. Retrieved October 2006 from: http://theses.nps.navy.mil/06Sep_Sullivan.pdf

Urick, R. J., *Ambient Noise in the Sea*. Los Altos, CA: Peninsula Publishing, 1986, p.p. 2-28.

“X3D Scene Authoring Hints”; Don Brutzman, Web3D Consortium;
<http://www.web3d.org/x3d/content/examples/X3dSceneAuthoringHints.html>

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California
3. President
Naval Postgraduate School
Monterey, CA
4. Provost
Naval Postgraduate School
Monterey, CA
5. Platt Brabner
21st Century Systems, Inc.
Omaha, NE
6. Doug Kozal
21st Century Systems, Inc.
Omaha, NE
7. Rick Goldberg
Aniviza, Inc.
Los Gatos, CA
8. Steve Kunkle
ARES Corporation
Arlington, VA
9. Chris Guryan
ARES Corporation
Arlington, VA
10. David Garvey
Boeing Phantom Works
Seal Beach, CA
11. Elizabeth Morin
Booz-Allen
McLean, VA

12. Joyce Borgen
Center for Asymmetric Warfare
Point Mugu, CA
13. Michael Ayling
Commander, Naval Installations Command
Anacostia Annex, DC
14. Doug Backes
Commander United States Pacific Fleet
Pearl Harbor, HI
15. CDR John E. Iman, USN
Commander United States Pacific Fleet
Pearl Harbor, HI
16. Doris Turnage
Engineer Research and Development Center (HQ)
Vicksburg, MI
17. J. D. Miller
John Hopkins University – Applied Physics Laboratory
Laurel, MD
18. Doug Weihnacht
Kinection
Santa Cruz, CA
19. Pete Swan
MAK Technologies
Cambridge, MA
20. Chris Carlson
Metron, Inc.
Reston, VA
21. Ray Jakobovits
Metron, Inc.
Reston, VA
22. Jeff Debrine
OPNAV N81
Washington, DC

23. Milon Essoglou
Naval Facilities Engineering Command (HQ)
Washington, DC
24. LT Norlando Antonio, USN
Naval Station Pearl Harbor
Pearl Harbor, HI
25. Robert Taylor
Naval Facilities Engineering Service Center
Port Hueneme, CA
26. Alexandria DeVisser
Naval Facilities Engineering Service Center
Port Hueneme, CA
27. Dr. Leonard Ferrari
Naval Postgraduate School
Monterey, CA
28. Dr. Paul Stockton
Naval Postgraduate School
Monterey, CA
29. Dr. Peter Purdue
Naval Postgraduate School
Monterey, CA
30. Prof. Dan Boger
Naval Postgraduate School
Monterey, CA
31. Prof. Rudy Darken
Naval Postgraduate School
Monterey, CA
32. Prof. Don Brutzman
Naval Postgraduate School
Monterey, CA
33. Curt Blais
Naval Postgraduate School
Monterey, CA

34. Jeff Weekley
Naval Postgraduate School
Monterey, CA
35. Terry Norbraten
Naval Postgraduate School
Monterey, CA
36. LT Patrick Sullivan, USN
Naval Postgraduate School
Monterey, CA
37. David Zeltzer
Northrop Grumman Corp. (HQ)
Los Angeles, CA
38. J. Riley Goodin
Northrop Grumman Corp. (HQ)
Los Angeles, CA
39. David Colleen
Planet 9 Studios
San Francisco, CA
40. Dan Ancona
Planet 9 Studios
San Francisco, CA
41. Chris Greuel
Planet 9 Studios
San Francisco, CA
42. Ayman El-Swaify
NAVFAC Information Technology Center
Port Hueneme, CA
43. Robert Seligman
Science Applications International Corporation
San Diego, CA
44. Margaret Bailey
Sonalysts, Inc.
Waterford, CT

45. Dallas Meggit
Sound and Sea Technology
Edmonds, WA
46. Dennis Garrood
Sound and Sea Technology
Edmonds, WA
47. Larry Ambruster
Sound and Sea Technology
Edmonds, WA
48. Mario Pozzo
Sound and Sea Technology
Edmonds, WA
49. Ron Brackett
Sound and Sea Technology
Edmonds, WA
50. MAJ Darryl Ahner, USA
Training and Doctrine Command
Monterey, CA
51. Bill Posage
USCG Research and Development Center
Groton, CT
52. LT Charles Adams, USN
USS Bonhomme Richard (LHD 6)
San Diego, CA
53. Alan Hudson
Yumetech, Inc.
Seattle, WA
54. Dr. Julie Seton
Advanced Systems Technology, Inc.
White Sands Missile Range, NM
55. Manoj K. Bhardwaj
Sandia National Laboratories
Albuquerque, NM

56. COL George Stone, USA
Battle Command, Simulation and Experimentation
Arlington, VA
57. William Duval
Army Modeling and Simulation Office
Arlington, VA
58. Robert Wiebe
Boeing Phantom Works
Seal Beach, CA
59. Skip Garrett
Center for Asymmetric Warfare
Point Mugu, CA
60. Frank Greitzer
Center for Asymmetric Warfare
Point Mugu, CA
61. Patrick Connor
Commander, Naval Installations Command
Anacostia Annex, DC
62. Adam Davidson
Commander United States Pacific Fleet
Pearl Harbor, HI
63. Shawn Hynes
Commander Operational Test and Evaluation Force
Norfolk, VA
64. CAPT David Yoshihara, USN (Ret.)
Commander United States Pacific Fleet
Pearl Harbor, HI
65. COL Jerry Glasow
Defense Modeling and Simulation Office
Alexandria, VA
66. Kenn Atkinson
Defense Modeling and Simulation Office
Alexandria, VA

67. David McDarby
Defense Threat Reduction Agency
Fort Belvoir, VA
68. Kenneth De Jong
George Mason University
Fairfax, VA
69. Dr. J. Mark Pullen
George Mason University
Fairfax, VA
70. Ron Hellbusch
L3/Titan
San Diego, CA
71. Dr. Michael Bailey
USMC Training and Education Command
Quantico, VA
72. MAJ J.P. McDonough, USMC
USMC Training and Education Command
Quantico, VA
73. Tom Stefanick
Metron, Inc.
Reston, VA
74. Tim Spivak
Navy Antiterrorism Technology Coordination Office
Washington, DC
75. Steven Iselin
Naval Facilities Engineering Command (HQ)
Washington, DC
76. CAPT Taylor Skardon, USN
NAVSTA Pearl Harbor
Pearl Harbor, HI
77. CDR Douglas Holderman
NAVSTA Pearl Harbor
Pearl Harbor, HI

78. LT Edward Twigg
NAVSTA Pearl Harbor
Pearl Harbor, HI
79. Bill Seelig
Naval Facilities Engineering Command (HQ)
Washington, DC
80. John Moore
Navy Modeling and Simulation Office
Washington, DC
81. James Ehlert
Naval Postgraduate School
Monterey, CA
82. Dr. John Sokowloski
Old Dominion University
Norfolk, VA
83. Dr. Andreas Tolk
Virginia Modeling and Analysis Center
Suffolk, VA
84. Erik Chaum
Naval Undersea Warfare Center
Newport, RI
85. Richard Lee
Office of the Secretary of Defense – Acquisition, Technology and Logistics
Washington, DC
86. Albert Giambalvo
Office of the Secretary of Defense – Acquisition, Technology and Logistics
Washington, DC
87. M. K. Tribble
Office of the Secretary of Defense – Acquisition, Technology and Logistics
Washington, DC
88. Dr. Jay Roland
Roland and Associates
Monterey, CA

89. Dr. Kathrine Morse
SAIC (HQ)
San Diego, CA
90. E. Belinger
SAIC (HQ)
San Diego, CA
91. Mike Irwin
Sandia National Laboratories
Albuquerque, NM
92. L.A. Cano
Sandia National Laboratories
Albuquerque, NM
93. Edward Bender
Space and Naval Warfare Systems Command
San Diego, CA
94. LTC Michael Kuchta, USAF
Department of the Air Force
Washington, DC
95. Jack Jackson
Training and Doctrine Command
Monterey, CA
96. Jeff Stahl
Research, Development and Education Command
Aberdeen Proving Ground, MD
97. G. Guy Thomas
Maritime Domain Awareness Program Integration Office
Washington, DC
98. Alex Vianna
Naval Facilities Engineering Command (HQ)
Washington, DC