

NOTICE:

The copyright law of the United States (Title 17, United States Code) governs the making of reproductions of copyrighted material. One specified condition is that the reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a user makes a request for, or later uses a reproduction for purposes in excess of "fair use," that user may be liable for copyright infringement.

RESTRICTIONS:

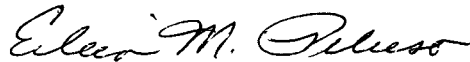
This student work may be read, quoted from, cited, and reproduced for purposes of research. It may not be published in full except by permission of the author.

Interactive Environments: The Design and Implementation of a
Terrain Simulator and Development Toolkit

Presented to the faculty of Lycoming College in partial fulfillment
Of the requirements for Departmental Honors in
Computer Science

By
Jason M. Black
Lycoming College
April 27th, 2005

Approved by:



(Dr. Eileen M. Peluso)



(Dr. Santhusht S. deSilva)



(Dr. David G. Fisher)



(David Heffner, Associate Dean of
Information Technology)

This project is dedicated to my parents, for supporting me in everything that I do.

Acknowledgements

A project of this size, whether it is created to have widespread use in the corporate world or created to be used by a few in academia, takes an enormous amount of planning, effort, and perseverance. No matter how dedicated an individual is to their work, their efforts will almost always grind to a halt without the support of other individuals. I had the support of many such individuals on this project, and I would like to thank them for their contributions.

First of all, I would like to thank Dr. Eileen Peluso for advising me for the year and a half from this project's conception to its conclusion. Whether it was the weekly reviews, getting me in touch with the right contacts for help, or the unscheduled emergency meetings, she was always there for me. I would never have been able to do this if you hadn't reined me in when I was overly zealous in my plans and supported me when I was experiencing a slump.

I would also like to thank my friend Jeremy Lothian for many late evenings of discussion on my project's contents. He practically taught me how to use XML, and saved me countless hours with his wisdom and insight.

I am thankful for the encouragement and motivation given to me by someone who is dear to me, Nicole Gugliucci. She has been a great muse.

This list would not be complete without expressing thanks to my parents, Clair and Karen Black, who have supported and believed in everything that I have done and hope to do.

Finally, I would like to give a thankful nod to the people at Microsoft, Google and APOD. I am thankful to Microsoft for a well documented graphics API (DirectX), an easy to use programming language (VB.NET), and a superb development environment (Visual Studio). As many others have been before me, I am thankful to Google for providing an efficient search technology that saved me in times of need and confusion. The pictures used in the Table of Contents and in the Introduction are courtesy of APOD (<http://antwrp.gsfc.nasa.gov/apod/>), and I thank them for their wonderful service.

About the Author

Jason Moses Black is an undergraduate senior at Lycoming College in Williamsport, Pennsylvania, where he is pursuing a Bachelor of Science degree in Computer Science. Jason intends to pursue a career in simulation design, either for serious uses by the government and the medical community, or for recreational and educational uses through the video game development industry. During his summers Jason has researched artificial intelligence at the University of Oklahoma and he intends to focus on this discipline in his future endeavors.

Table of Contents

Introduction	vii
1 Original Honors Project Proposal	3
2 Software Project Management Plan	15
3 Detailed Design Document	37
4 Material Editor Code and Screenshots	137
5 EDGE Tool Code and Screenshots	159
6 WIM Tool Code and Screenshots	179
7 Interactive Terrain Simulator Code and Screenshots	209
Index	325



Above: A young star forming, sending dust and gas clouds into neighboring space.

“In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move.”

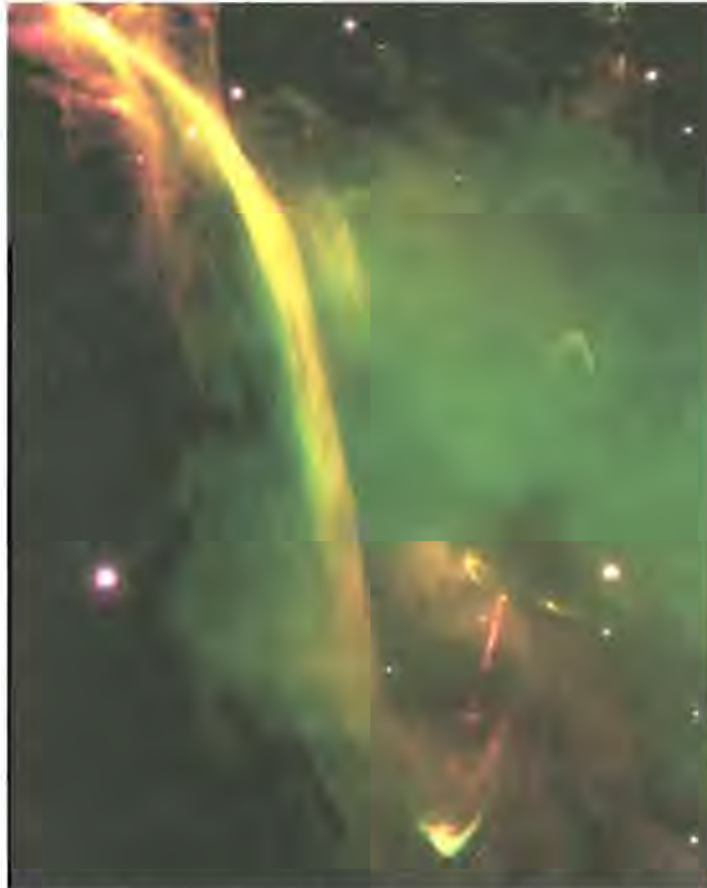
- Douglas Adams

Introduction

Everything has a source, a beginning. This is one such beginning.

Welcome to my honors project, codenamed Project Origins. The next few hundred pages of charts, documentation, and code are not here as a conclusion to this project, but as a record of my experiences. Looking back, I recall a wide variety of experiences. There were joyous moments of accomplishment, periods of frustration and despair, and many, many long hours of hard work. However, out of all of these experiences, I am most excited about the experiences and accomplishments that have yet to come.

The name of this project sums up this attitude: Origins. All journeys have destinations, and they also have origins. Often the origin of a journey is lost to time, forgotten by the time that the destination is finally reached. However, once in a while it is possible to glimpse the future in the present. I see such a future in the accomplishment of this honors project. Let this serve as a guide to those who come after me: a person can rarely predict where their current direction will lead them, but the greatest accomplishments are only attained through hard work and persistence. There is no better way to reach the end of a journey.



The image on this page was taken by the European Southern Observatory. In the bottom right is a young star, centered between two bright green caps of gas that mark the ends of two gas jets shooting outward from either side. The reason for the inclusion of this image is twofold. First, to remind us that everything has an origin, even something as massive and powerful as a star. Secondly, in this image there appears to be a waterfall of cosmic proportions pouring forth from an opening in a bank of sunlit clouds. Of course, what is seen here is neither water nor clouds, but giant arcs of gas that have not yet been explained. Mysterious and beautiful, the "waterfall" of gas reminds us that even after witnessing the birth of a star, there are still wonders even more spectacular to be seen. Therefore, as we each decide on a journey to traverse and a destination to strive for, we should remember the young star and the infinite wonders that lie beyond its light.

**Interactive Environments: The Design and Implementation of a Terrain
Simulator and Development Toolkit**

Section 1

Original Honors Project Proposal

Software Engineering of a Graphical Engine

Honors Project Proposal - Jason M. Black

Project Summary

The design and development of graphical applications, such as environmental simulations, games, and interactive training programs are active areas of research in the field of Computer Science. In this Honors Project, I will design and develop an Integrated Development Environment (IDE) that will support libraries of graphical objects that can be integrated into complex 3D scenes through which the user will be able to navigate. With this final product, users will be able to take individual objects created by existing image-creation applications and combine them, resulting in a virtual world simulation in which the user becomes a participant.

Project Outline

The Honors Project shall consist of the design and development of software, along with full documentation and a users' manual designed to accompany the software that explains all of its features. The software will be comprised of three components. This first fundamental component allows the user to incorporate graphical objects created by existing image-creation applications, e.g. Maya or Blender, into libraries containing information that describes the attributes of each object, e.g. height and weight. The second component will allow the user to create simulated terrains that will serve as the backdrop onto which objects created by the first component can be placed. The central component will translate the information created by the other two components into the actual simulated environment and will allow the user to navigate through it. A novel

feature of this software is that the attributes stored with each object can be used to add realism to the simulation. For example, the weight of an object could determine if the user could move it. As part of the IDE, during the simulation, the user will be able to click on any image in the simulation to obtain a detailed description of an object's attributes.

Preparation

Prior to undertaking this Honors Project, I will have completed the following courses needed to prepare me for such an undertaking:

CPTR 247 – Data Structures
CPTR 448 – Advanced Development & Design
PHYS 225 – Introductory Mechanics
PHYS 226 – Introductory Electricity & Magnetism
IIS 800 – Independent Study in Computer Graphics

The Data Structures and Advanced Development & Design classes have given me the background necessary to understand the type of structures needed to support the large amount of interactive data necessary for this project, as well as a clear understanding of the software engineering process required for the development of a complete product. My background in fundamental physics is necessary because the system will allow for the realistic representation of real-world objects. I am currently enrolled in *PHYS 331 – Classical Mechanics* for the Fall of 2004. This course will add to my understanding of the properties that need to be incorporated into the libraries developed in this project.

What will I gain from this project? How will I accomplish these goals?

When I continue with my education and/or begin my career once I graduate from

Lycoming College, I plan to be involved in the design and/or research of digital systems that represent realistic environments. I am particularly interested in systems that are as flexible as possible so that a wider array of environments and projects can be created using a single base engine. This Honors Project is basically the core of such an engine. It will strengthen and advance my following skills:

- Project Management
- Computer Graphic Design and Manipulation
- Advanced Data Structure Management
- Analysis of Algorithm Complexity
- Implementation of Physics through Software Engineering

The final product of this project will also give me a demo program which will showcase my software engineering abilities. So, while the saying “A journey is made to reach a goal, but it is the journey that matters, in the end” is true in this case, since I will be gaining advanced skills in the process, the “end of the journey” is also important. The final product will be an important stepping stone in starting my career or when applying to graduate school.

Environment / Availability

This project will work on a standard *win32 (Microsoft Windows)* operating system, and will be developed using *Microsoft Visual C++* and the *DirectX 9 SDK*. The final product will be a windows application and thus will be compatible with any Microsoft Windows operating system that supports *DirectX 9* and has it installed.

Technical Summary:

This project involves the creation of an XML data collection (known as the “Entity System”) that is able to represent any real-world entity, living or non-living, as a set of data objects and attributes. This data is to be interpreted and implemented in a real-time three-dimensional environment with all entities graphically represented in said environment. Limited user interaction will be provided in the final product, which shall be implemented using Microsoft Visual C++ and the DirectX 9 SDK on Windows XP. The last part of the project, the “demo program,” should be viewable on any windows system that supports DirectX 9.

The Entity System will be designed to hold information about a digital environment and to interpret and implement what is necessary for the system's current environment dynamically, using user-created rule-sets. In this way, the entity system is meant to be a complete environment generation toolkit, providing the data structure framework for any three-dimensional environment.

- Proposed Project Stages:**
- 1) Entity System (XML Data Collection)
 - 2) E.D.G.E Tool – Entity Template and Entity Manager
 - 3) Rule-set Format and Intermediate Functionality
 - 4) W.I.M. Tool – Environment and Entity Instance Manager
 - 5) Entity Instance and Environment Test Data Set
 - 6) User Interaction Module
 - First-Person (user) movement.
 - Identification of Entities selected using a mouse.
 - 7) Unit Demo

Proposed Project Stage Summaries:

Entity System

This set of XML data collections will be able to hold data suitable for any entity, graphical or non-graphical, physical or abstract. This Entity System's purpose is to be a system dynamic enough that any conceivable object, force, or environmental component can be suitably represented while keeping order and structure to the overall system. This second purpose will make sure the system is easy to understand so that users will be able to create their own environments and entities as easily as possible.

Stage Sub-Components

- outline basic set of XML Entity Templates
- compile lists of fundamental Entity attributes
- compile lists of more specific Entity attributes
- encode all XML Entity Templates into an XML file

E.D.G.E Tool – Entity Manager

The *Entity Manager* is to be a windowed application that allows the user to interface with the *Entity System* itself and to create, edit, and manage .ent files which contain collections of Entity data. Basically, this application will be used in order to create and manage the set of Entities created from the Entity Template collection.

Tasks and Subsections:

- design and create the basic windowed application
- allow a user to Add/Edit/Delete Entity Templates
- allow a user to Create/Edit/Delete Entity libraries (.elb files)
- allow a user to Add/Edit/Remove Entities from Entity libraries (.elb files)
- add a viewer window to the application so the user can preview the graphical representation of an Entity, if available.

Rule-set Format and Intermediate Functionality

This stage of the project involves using the *E.D.G.E. Tool* to add all of the Entity Templates created in Stage 1 to *E.D.G.E.* and also to add functionality to *E.D.G.E.* so that some form of 'rule-sets' can be created where a user can easily mass add/edit/remove attributes from sets of Entities and alter the function libraries the Entities point to.

The premise here is that not all virtual environments require the same functionality and/or entity attributes. Therefore, this tool will allow a user to easily convert Entities made for one environment into a format that is acceptable in a second environment that operates under different rules (i.e., calls a separate library of functions used for interaction). This setup means that once a single environment and its Entities are designed, the Entities will be reusable in a second environment with minimal effort.

Tasks and Subsections:

- design user control of rule-sets
- add Entity conversion functionality to *E.D.G.E.*
- allow the user to turn individual attributes on/off

W.I.M. Tool – Environment Manager

The *W.I.M. Tool* is much like the first tool: it allows the user to create, edit and manage data files. This tool allows the manipulation of .env files that determine environment data, such as landscape, viewing options, etc. Environment files will also contain information about the location and placement of Entities Instances, based off of Entities created using the *E.D.G.E. Tool*. These Entity Instances can be further adjusted inside of the *W.I.M. Tool*.

Tasks and Subsections:

- design environment data file format
- allow for the creation and saving of new environments (.wid files)
- allow for the editing and saving of environments (.wid files)
- allow for the management (delete, move, copy) of environments (.wid files)
- allow a user to add/edit/delete Entity Instances using .elb libraries

Entity Instance and Environment Test Data Set

This portion of the project is simply the use of the E.D.G.E. and W.I.M. tools to create a set of Entity Instances and an environment to use for testing purposes. This will be the first set of data used to create a digital environment using the *Entity System*.

Tasks and Subsections:

- create ten Entity Instances using the W.I.M. Tool
- create one environment using the W.I.M. Tool
- place all ten Entity Instances in the environment

User Interaction Module

This is the final module to be implemented in this graphical engine. This is where the core functionality is stored that allows a user to not only view an environment and its contents, but also to move about the environment and bring up data screens on individual entities.

Tasks and Subsections:

- display the environment and its contents properly
- allow the user to move about the environment
- allow the user to bring up data screens about an individual entity when clicked on using a mouse.

Unit Demo

This is the demonstration of the finished product once all of the other stages are complete. Library entries will be pulled into a scene where the user can move around and interact.

Proposed Project Schedule:

<u>June 2004</u>	Learn XML Requirements & Specification Documentation
<u>July 2004</u>	Learn DirectX SDK 9
<u>August 2004</u>	Learn DirectX SDK 9, continued Map out Entity Templates (Stage 1 Completed)
<u>September 2004</u>	Overall Design & Integration Documentation
<u>October 2004</u>	Detailed Design Documentation Research and Completion of General Project Documentation
<u>November 2004</u>	Develop E.D.G.E. Tool (Stage 2 Completed) User-Controlled Rule-Set implemented (Stage 3 Completed)
<u>December 2004</u>	Develop W.I.M. Tool (Stage 4 Completed)
<u>January 2005</u>	Revise Core Engine Documentation Create Test Data (Stage 5 Completed)
<u>February 2005</u>	Code Final Component of Graphical Engine
<u>March 2005</u>	Refine and document all Code and Documentation (Stage 6 Completed) Write User Manual
<u>April 2005</u>	Create Unit Demo (Stage 7 Completed)

Bibliography

Angel, Edward. Interactive Computer Graphics: A Top-Down Approach with OpenGL (3rd Edition). Boston: Addison-Wesley, 2002.

Bourg, David M. Physics for Game Developers. Sebastopol, CA: O'Reilly & Associates Inc., 2002.

Dalmau, Daniel Sanchez-Crespo. Core Techniques and Algorithms in Game Programming. New Riders, 2003.

Engel, Wolfgang, Andre' Lamothe, and Amir Geva. Beginning Direct3D Game Programming 2nd Edition. Premier Press, 2003.

Hansen, Henning. Nitty Gritty Windows Programming with C++. Addison-Wesley, 2001.

Harold, Elliotte Rusty. XML Bible 2nd Edition. New York, NY: Hungry Minds Inc., 2001.

Josuttis, Nicolai M. The C++ Standard Library. Boston: Addison-Wesley, 1999.

LaMothe, Andre. Tricks of the Windows Game Programming Gurus 2nd Edition. SAMS, 2002.

McShaffry, Mike. Game Coding Complete. Paraglyph Publishing, 2003.

Snook, Gregory. Real-Time 3D Terrain Engines Using C++ and DirectX 9. Charles River Media, 2003.

Stroustrup, Bjarne. The C++ Programming Language Special Edition. Florham Park, NJ: Addison-Wesley, 2000.

Walsh, Peter. The Zen of Direct3D Game Programming. Premier Press, 2002.

Section 2

Software Project Management Plan

IEEE Software Project Management Plan
Requirements & Specification Document

For

An Honors Project in Computer Science

Honors Project Developer:
Jason M. Black

Honors Committee Chair:
Dr. Eileen M. Peluso

Honors Committee Members:
Dr. Santhusht S. deSilva
Dr. David G. Fisher
David Heffner, Associate Dean of
Information Technology

Table of Contents

- 1. Introduction**
 - 1.1. Project Overview**
 - 1.1.1. Honors Project Summary
 - 1.1.2. Honors Project Outline
 - 1.1.3. Preparation for the Honors Project
 - 1.1.4. Honors Project Goal List
 - 1.2. Product Deliverables**
 - 1.2.1. Documentation
 - 1.2.2. Software
 - 1.3. Evolution of the SPMP**
 - 1.4. Reference Materials**
 - 1.4.1. Books
 - 1.4.2. Articles
 - 1.5. Definitions and Acronyms**
 - 1.5.1. Term Definitions
 - 1.5.2. Acronym Definitions
- 2. Project Organization**
 - 2.1. Process Model**
 - 2.1.1. Research Outline
 - 2.1.2. List of Project Milestones
 - 2.1.3. Format of Milestone Entries
 - 2.2. Organizational Structure**
 - 2.2.1. Contributors to the Honors Project
 - 2.2.2. Communication Diagram
- 3. Managerial Process**
 - 3.1. Management Objectives and Priorities**
 - 3.1.1. Purpose of the Honors Project
 - 3.1.2. Core Sentence
 - 3.1.3. Detailed Goal List
 - 3.2. Risk Management**
 - 3.3. Honors Project Review Process**
- 4. Technical Process**
 - 4.1. Methods, Tools, and Techniques**
 - 4.1.1. Hardware
 - 4.1.2. Software
 - 4.2. Software Documentation**
 - 4.2.1. Detailed Project Milestone List
 - 4.2.2. Data Dictionary
- 5. Schedule**
 - 5.1. Estimate of Time Commitment**
 - 5.2. Schedule**

1. Introduction

1.1. Project Overview

1.1.1. Honors Project Summary

The design and development of graphical applications, specifically realistic simulations such as environmental simulators, games, and interactive training programs, is an active area of research in the field of Computer Science. In this Honors Project, I will design and develop an Integrated Development Environment (IDE) that will support libraries of graphical objects that can be integrated into complex 3D scenes through which the user will be able to navigate. In the final product, users will be able to view individual objects created by existing image-creation applications and combine them, resulting in a virtual world simulation in which the user becomes a participant.

1.1.2. Honors Project Outline

The Honors Project shall consist of the design and development of software, along with full documentation and user manuals designed to accompany the software that explains all of its features. The software will be comprised of four components. The first piece of software will allow the user to specify physical attributes of materials and store that information in libraries. The second software component allows the user to incorporate graphical objects created by existing image-creation applications, e.g. Maya, 3DS Max and Blender, into libraries containing information that describes the attributes of each object, e.g. height and weight. The third component will allow the user to create simulated terrains that will serve as the backdrop onto which objects created by the second component can be placed. The final component, the simulation component, will translate the information created by the other three components into the actual simulated environment and will allow the user to navigate through it. A novel feature of this software is that the attributes stored with each object can be used to add realism to the simulation. For example, the weight of an object could determine if the user could move it with a push. As part of the IDE, during the simulation, the user will be able to click on any image in the simulation to obtain a detailed description of an object's attributes.

1.1.3. Preparation for the Honors Project

Prior to undertaking this Honors Project, I will have completed the following courses needed to prepare me for such an undertaking:

CPTR 247 – Data Structures
CPTR 448 – Advanced Development & Design
PHYS 225 – Introductory Mechanics
PHYS 226 – Introductory Electricity & Magnetism
IIS 800 – Independent Study in Computer Graphics

The Data Structures and Advanced Development & Design classes have given me the background necessary to understand the type of structures needed to support the large amount of interactive data necessary for this project, as well as a clear understanding of the software engineering process required for the development of a complete product. My background in fundamental physics is necessary because the system will allow for the realistic representation of real-world objects. I was enrolled in *PHYS 331 – Classical Mechanics* for the Fall of 2004. This course added to my understanding of the properties that need to be incorporated into the libraries developed in this project.

1.1.4. Honors Project Goal List

- Create and demonstrate a material-based object system
 - Load multiple objects into an environment
 - Ability to view an object's material properties
- Allow a user to interact with the 3D environment
 - Move around the environment
 - Push objects
 - Jumping
- Demonstrate the use of simulated physics in a 3D environment
 - Realistic movement of objects when pushed

1.2. Product Deliverables

1.2.1. Documentation

- Requirements and Specification Document (IEEE SPMP)
- Detailed Design Document
- Developer Tools User Manuals
- Simulation User Manual
- Commented Simulation Source Code

1.2.2. Software

- Material Editor
- Entity Dynamic Generation Environment (E.D.G.E.) Tool
- World Instance Manager (W.I.M.) Tool
- Functional Simulation Executable

1.3. Evolution of the SPMP

When it is necessary for this document to be updated, the update may be done immediately. Whenever such an update occurs, a copy of the updated document with all recent changes highlighted will be delivered to the chair of the Honors Project Committee within three (3) days.

1.4. Reference Materials

1.4.1. Books

Angel, Edward. Interactive Computer Graphics: A Top-Down Approach with OpenGL (3rd Edition). Boston: Addison-Wesley, 2002.

Bourg, David M. Physics for Game Developers. Sebastopol, CA: O'Reilly & Associates Inc., 2002.

Dalmau, Daniel Sanchez-Crespo. Core Techniques and Algorithms in Game Programming. New Riders, 2003.

Engel, Wolfgang, Andre' Lamothe, and Amir Geva. Beginning Direct3D Game Programming 2nd Edition. Premier Press, 2003.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Flynt, John P. Software Engineering for Game Developers. Thomson Course Technology, 2005.

Hansen, Henning. Nitty Gritty Windows Programming with C++. Addison-Wesley, 2001.

Harold, Elliotte Rusty. XML Bible 2nd Edition. New York, NY: Hungry Minds Inc., 2001.

Josuttis, Nicolai M. The C++ Standard Library. Boston: Addison-Wesley, 1999.

LaMothe, Andre. Tricks of the Windows Game Programming Gurus 2nd Edition. SAMS, 2002.

McShaffry, Mike. Game Coding Complete. Paraglyph Publishing, 2003.

Snook, Gregory. Real-Time 3D Terrain Engines Using C++ and DirectX 9. Charles River Media, 2003.

Stroustrup, Bjarne. The C++ Programming Language Special Edition. Florham Park, NJ: Addison-Wesley, 2000.

Walsh, Peter. The Zen of Direct3D Game Programming. Premier Press, 2002.

1.4.2. Articles

Various websites were referenced, but not significantly. Unfortunately no useful journal articles were found during my period of research.

1.4. Reference Materials

1.4.1. Books

Angel, Edward. Interactive Computer Graphics: A Top-Down Approach with OpenGL (3rd Edition). Boston: Addison-Wesley, 2002.

Bourg, David M. Physics for Game Developers. Sebastopol, CA: O'Reilly & Associates Inc., 2002.

Dalmau, Daniel Sanchez-Crespo. Core Techniques and Algorithms in Game Programming. New Riders, 2003.

Engel, Wolfgang, Andre' Lamothe, and Amir Geva. Beginning Direct3D Game Programming 2nd Edition. Premier Press, 2003.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Flynt, John P. Software Engineering for Game Developers. Thomson Course Technology, 2005.

Hansen, Henning. Nitty Gritty Windows Programming with C++. Addison-Wesley, 2001.

Harold, Elliotte Rusty. XML Bible 2nd Edition. New York, NY: Hungry Minds Inc., 2001.

Josuttis, Nicolai M. The C++ Standard Library. Boston: Addison-Wesley, 1999.

LaMothe, Andre. Tricks of the Windows Game Programming Gurus 2nd Edition. SAMS, 2002.

McShaffry, Mike. Game Coding Complete. Paraglyph Publishing, 2003.

Snook, Gregory. Real-Time 3D Terrain Engines Using C++ and DirectX 9. Charles River Media, 2003.

Stroustrup, Bjarne. The C++ Programming Language Special Edition. Florham Park, NJ: Addison-Wesley, 2000.

Walsh, Peter. The Zen of Direct3D Game Programming. Premier Press, 2002.

1.4.2. Articles

Various websites were referenced, but not significantly. Unfortunately no useful journal articles were found during my period of research.

1.5. Definitions and Acronyms

1.5.1. Term Definitions

Agent	An entity that acts in an environment with some degree of autonomy. Therefore, an agent is a self-controlled component of an environment.
Character	Short for ‘player character’; this is the representation of the user in the simulation.
Entity	Any object or software controlled agent in the environment. Subsequently, entities are any non-terrain, non-user atomic structures in the environment.
Environment	A finite simulation of a three-dimensional space, specifically referring to the most static elements of that space. Example: ground and sky in an outdoor simulation.
First-Person View	This is a method of viewing an environment where the user sees what the PC (see below) would actually see if the user were looking through its eyes.
Material	Referring to an elemental substance or a combination of such substances. Iron and oxygen are elemental substances, wood and water are combinations, but all are materials. All objects in an environment are composed of materials, and derive some of their properties from their material composition.
Non Player Character	Any agent in an environment that isn’t controlled by the user.
Object	These are non-intelligent, individual components of a simulation. Example: furniture, buildings and plants.
Player Character	This is the agent that is directly controlled by the user and is the user’s primary method of interacting with the environment.
User	This is the person who is using the simulation.
World	Another name for an Environment.

1.5.2. Acronym Definitions

PC	Player Character
SPMP	Software Project Management Plan

2. Project Organization

2.1. Process Model

2.1.1. Research Outline

A general outline of the stages that this honors project can be broken down into follows. The first stage consists of planning, scheduling, and brainstorming, culminating in the creation of this document. The second stage is a period of research done in preparation for the design document. This research includes reading primary texts, skimming reference texts, searching for useful journal articles, and examining the software architecture of two computer games (one published, one open source) whose source code is free to browse. Once the research stage is completed, the detailed design document will be constructed. This will lay out the software component of the honors project in detail. This stage of the project should be completed by late November 2004. After the documentation is completed the actual software development stages will commence with the development of design tools. Once the tools are finished, work will begin on the simulation software itself. This software will be constructed in builds listed in section 2.1.2 of this document and detailed in section 4.2.1 of this document. These builds will be worked through until April 2005 when the debugging, testing, and user documentation stages will be performed. The honors project will conclude with a demonstration and defense of the project to the honors committee.

2.1.2. List of Project Milestones

- 1) Specification Documentation
- 2) Literature Research
- 3) Detailed Design Documentation
- 4) Tool Creation
- 5) First Production
- 6) Second Production
- 7) Third Production
- 8) Fourth Production
- 9) Content Complete
- 10) Debugging
- 11) User Manuals
- 12) Honors Project Defense

2.1.3. Format of Milestone Entries

The following is a list of fields and explanations of what information is in said fields for the detailed milestone descriptions given in section 4.2.1.

Milestone Title

This is the name of the milestone.

Short Description

The goals of the milestone are laid out in paragraph form. Goals are to be specific, avoiding any vague references such as 'nearly' or 'optimal'.

Due Date

This is the date when the milestone is to have been fully completed and having passed the acceptance criteria.

Acceptance Criteria

A list of tests that must be passed before the milestone can be said to have been completed.

Risk Assessment

This is a description of what could go wrong with this milestone build and what can be done to prevent such a situation from occurring. Details of how to deal with an already occurred setback are also listed here.

Deliverables

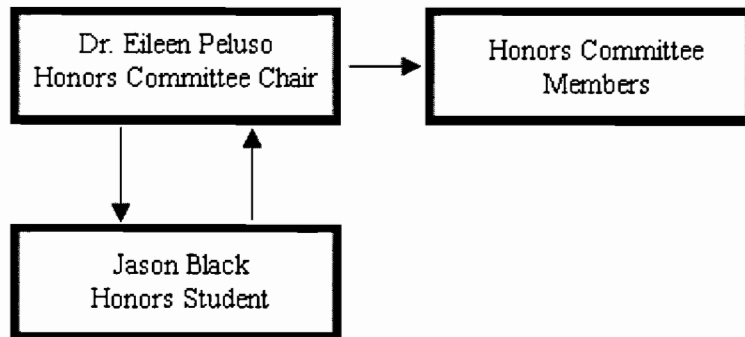
This is a bulleted list of the milestone's results. These results can include anything from physical documentation or code segments to accomplishing a specific type of research.

2.2. Organizational Structure

2.2.1. Contributors to the Honors Project

The basic organization of this honors project is that I, Jason Black, will be conducting the research and implementing the project in its various stages. Direct advising will be done through meetings with Dr. Eileen Peluso, the honors committee chair, once a week. Additional meetings will be scheduled as needed. If necessary, members of the honors committee will be called upon for assistance, though these instances should be rare. The honors committee will be updated on the state of the honors project by the honors committee chair as needed.

2.2.2. Communication Diagram



3. Managerial Process

3.1. Management Objectives and Priorities

3.1.1. Purpose of the Honors Project

When I continue with my education through graduate school and later begin my career once I graduate from Lycoming College, I plan to be involved in the design and research of digital systems that represent realistic environments. I am particularly interested in systems that are created with maximum flexibility so that a wider array of environments and projects can be created using a single base engine. This Honors Project is basically a fundamental implementation of such an engine. It will strengthen and advance my following skills:

- Project Management
- Computers Graphic Design and Manipulation
- Advanced Data Structure Management
- Analysis of Algorithm Complexity
- Implementation of Physics through Software Engineering

The final product of this project will also give me a working program which will showcase my software engineering abilities. So, while the saying “A journey is made to reach a goal, but it is the journey that matters, in the end” is true in this case, since I will be gaining advanced skills in the process, the “end of the journey” is also important. The final product will be an important stepping stone in starting my career and when applying to graduate school.

3.1.2. Core Sentence

This sentence is meant to represent the essence of the simulation module’s structure without going into heavy detail. The core sentence doesn’t limit a simulation implementation, but is created in order to remind the designer of the fundamental ideas behind the simulation’s design.

“This simulation will be a first-person outdoor simulation where the user takes on the role of a person who experiments with objects found in the environment.”

3.1.3. Detailed Goal List

Create and demonstrate a material-based object system

- Ability to view an object’s material properties

When viewing the environment the user will be able to click on any object in the environment, such as a box, a tree, or a rock. This action will bring up a list of the properties that this object has based on the materials that constitute it. The purpose of this goal is to allow the user to understand the material-based object system visually.

Allow a user to interact with the 3D environment

- Move around the environment

The PC will be able to move in all four cardinal directions as well as combinations thereof. The PC will also be able to smoothly turn in order to face any part of the environment. This goal is necessary for a user to be able to view and interact with the environment through his or her character.

- Push objects

Moveable objects will be able to be pushed when the PC moves into them. The objects will then follow physical laws and move, skid, and possibly roll based on the force and direction of impact. The purpose of this goal is to demonstrate the addition of physics to the simulated environment.

- Jump

The PC will be able to jump off of the ground into the air when this goal is implemented. The purpose here is to show that gravity works realistically and also to add another dimension of control to the user's character.

Demonstrate the use of simulated physics in a 3D environment

- Realistic movement of objects when pushed

Not only will objects be moveable, but if they are moved they will slide and roll across the terrain realistically. Also, if objects are loaded in the air or fall off of an edifice, they will be subject to gravity and will interact with the terrain accordingly.

3.2. Risk Management

There are two levels of general risk management of the honors project in use. The first is the continuous keeping of a project log where I keep my hours for the week. By doing this I am able to make sure that I put in an appropriate number of hours into the honors project each week. An estimation of these hours is specified in section 5.1 of this document. The second risk prevention is the weekly review meetings described in section 3.3 of this document. Milestone specific risk management and prevention is listed in the detailed project milestone list in section 4.2.1.

3.3. Honors Project Review Process

The general process for reviewing the progress of the honors project is that every Tuesday afternoon from 1:30-2:30 I will meet with my honors committee chair, Dr. Eileen Peluso, and we will review everything that was accomplished in the previous week. Updates to the schedule and future directions of the project will also be touched on at each meeting. Whenever this or another important document is updated in a given week, an updated copy will be submitted to the honors committee chair by 2:00 on Monday so that there is time for review before the Tuesday meeting.

4. Technical Process

4.1. Methods, Tools, and Techniques

4.1.1. Hardware

This project was designed to work on an IBM PC with Windows XP installed and with the DirectX 9.0 drivers installed. Other than these basic requirements, there are no hardware requirements.

4.1.2. Software

Software required for this honors project includes:

- Microsoft Visual Studio .NET 2003
 - VC++ .NET
 - VB .NET
 - MSXML v4.0
- DirectX 9.0 SDK
- A 3D Mesh Editor (DeleD)

4.2. Software Documentation

4.2.1. Detailed Project Milestone List

It is important to note that this section was continually updated as the project became more refined during the Detailed Design Documentation milestone.

Milestone #1: Specification Documentation Milestone

Short Description:

The goal of this milestone is to complete the Specification Documentation in the form of an IEEE SPMP document.

Due Date:

Saturday, September 18, 2004

Acceptance Criteria:

All subsections of the IEEE SPMP Specification Document must be filled out.

Risk Assessment:

The only risks for this milestone are that it could be incomplete or not completed on time. To avoid this it will be reviewed on Tuesday, September 21st at the weekly review meeting and approved by the honors committee chair.

Deliverables:

- IEEE SPMP Specification Document

Milestone #2: Literature Research Milestone

Short Description:

This milestone is meant to provide me with all of the information I will need for the design and development of the honors project. During this period I will read several books, pursue literature reviews of journals, and study preexisting free-to-view code that is similar to what I will be writing. The titles of the books to be read in this phase are available in the project calendar in section 5.4.

Due Date:

Saturday, October 30, 2004

Acceptance Criteria:

By the end of this milestone I should have a set of notes that will allow for the creation of the Detailed Design Document. A meeting with the honors committee chair on Tuesday, November 2nd will confirm that this criterion has been met and that I am prepared to work on the detailed design milestone.

Risk Assessment:

The major risk for this milestone is that I will have an incomplete view of what to detail in the detailed design documentation. In order to prevent this I need to read all of my materials and take thorough notes. This performance task will be confirmed at weekly review meetings.

Deliverables:

- Research Notes

Milestone #3: Detailed Design Documentation Milestone

Short Description:

The Detailed Design Document is the product of this milestone. This document will intricately detail all of the modules, major functions and algorithms that will be used in the development of the simulation program. Detailed information on data types and function parameters will also be included.

Due Date:

Friday, December 10, 2004

Acceptance Criteria:

The Detailed Design Document must completely outline the entire simulation program. All defined modules must be completely defined in terms of general internal structure, input, and output. This will be tested through the use of diagrams and weekly review sessions with the honors committee chair.

Risk Assessment:

The risks for this milestone are the loss of time to work on the milestone and the possibility that the resulting document will be incomplete. Weekly review

meetings will be crucial during this milestone. Most likely this milestone will be broken down into smaller segments when this point in the honors project is reached.

Deliverables:

- Detailed Design Document

Milestone #4: Tool Creation Milestone

Short Description:

Once the design for the simulation is completed in the Detailed Design Document, there will be three tools to assist in the creation of data that will allow the simulation to perform. These three tools are the Material Editor Tool, the E.D.G.E. Tool and the W.I.M. Tool. The Material Editor Tool allows for the maintenance of a database containing Material datatype data. The E.D.G.E. Tool allows for the creation and editing of Entity datatype data. The W.I.M. Tool allows for the creation and maintenance of Environment datatype data. These three datatypes are detailed in section 4.2.2 of this document.

Due Date:

Monday, January 31, 2005

Acceptance Criteria:

The acceptance criteria for this milestone is that each of the tools will create and edit their respective datatypes according to the general specification in section 4.2.2 of this document as well as the detailed datatype specifications available in (section pending) of the Detailed Design Document.

Risk Assessment:

The risks for this milestone are that it will not be accomplished in its short timeframe and that the output data will not meet specifications. The first risk is alleviated by the fact that this milestone takes place early in the spring semester when workload with other responsibilities is light, so that more than the average weekly time commitment can be spent on this milestone. As for not meeting datatype specifications, the Detailed Design Document should be complete enough that this will not be a problem. Also, the weekly review sessions with the honors committee chair should prevent this setback from occurring.

Deliverables:

- Material Editor Tool
- E.D.G.E. Tool
- W.I.M. Tool

Milestone #5: First Production Milestone

Short Description:

This milestone will be the first of four that deal with the simulation code. Only the core necessities for the simulation to run are included in this milestone. This milestone can be broken down into three tasks. The first task includes the

creation of the menus to enter and exit the program, as well as the ability for the user to pause and resume the simulation without exiting. Secondly, the loading of XML data created from the developer tools into the simulation must be accomplished. Finally, the rendering of the environment from this data (elevation only), without texture, needs to perform without any errors.

Due Date:

Friday, February 25, 2005

Acceptance Criteria:

Before this milestone is accepted it must pass the appropriate tests detailed in section 4.3 of this document.

Risk Assessment:

The primary risk for this milestone is the chance of going over the set time period for passing the acceptance criteria. In order to prevent this from happening the average work hours per week will be increased to 15, and secondary review meetings will be held if necessary.

Deliverables:

- Start Menu with 'Start' and 'Exit' options
- Pause Menu with 'Continue' and 'Exit' options
- Rendering of non-textured elevation map

Milestone #6: Second Production Milestone

Short Description:

This is the second of four milestones dealing with the simulation code. The first task in this milestone focuses on the PC. The user must be able to move the PC in all four directions, turn the PC smoothly in either direction, and also must be able to click on objects and view their properties. The second task in this milestone concentrates on incorporating collision detection between the PC and the environment.

Due Date:

Friday, March 4, 2005

Acceptance Criteria:

Before this milestone is accepted it must pass the appropriate tests detailed in section 4.3 of this document.

Risk Assessment:

The primary risk for this milestone is the chance of going over the set time period for passing the acceptance criteria. In order to prevent this from happening the average work hours per week will be increased to 15, and secondary review meetings will be held if necessary.

Deliverables:

- User control of the PC through movement and turning
- Working physics simulation for collision

Milestone #7: Third Production Milestone

Short Description:

The third production milestone is concerned with the loading and display of objects that will respond to physics in the environment. These moveable objects will take the form of boxes and crates that the user can push with the PC. These boxes and crates will not only move due to force applied by the PC, but will also slide, fall and roll according to physical laws

Due Date:

Friday, March 18, 2005

Acceptance Criteria:

Before this milestone is accepted it must pass the appropriate tests detailed in section 4.3 of this document.

Risk Assessment:

The primary risk for this milestone is the chance of going over the set time period for passing the acceptance criteria. In order to prevent this from happening the average work hours per week will be increased to 15, and secondary review meetings will be held if necessary.

Deliverables:

- Loading of box and crate objects
- Object collision detection

Milestone #8: Fourth Production Milestone

Short Description:

The fourth production milestone focuses on allowing the user to view the properties of an object by highlighting it. The secondary goal of this milestone is the addition of immobile objects to the environment. The completion of the secondary goal will depend on the remaining time.

Due Date:

Saturday, April 2, 2005

Acceptance Criteria:

Before this milestone is accepted it must pass the appropriate tests detailed in section 4.3 of this document.

Risk Assessment:

The primary risk for this milestone is the chance of going over the set time period for passing the acceptance criteria. In order to prevent this from happening the average work hours per week will be increased to 15, and secondary review meetings will be held if necessary.

Deliverables:

- Ability to examine object properties
- Loading of rock, plant, and bush environmental objects

Milestone #9: Content Complete Milestone

Short Description:

This brief milestone allots time for the creation of any additional graphics that have not been created up to this point. The texturing of the ground and objects is included in this milestone. Also, any object information that is incomplete or missing will be finished in this milestone.

Due Date:

Tuesday, April 5, 2005

Acceptance Criteria:

The acceptance criterion for this milestone is that no non-code data is missing from the honors project. This includes graphics, sound, and database information.

Risk Assessment:

There is no real risk in this milestone since this is a cleanup milestone.

Deliverables:

- No missing no-code content for the honors project
- Texture ground and objects

Milestone #10 Debugging Milestone

Short Description:

This milestone will consist of using the finished simulation and trying all of the options repeatedly. Any bugs or deviations from documentation will be recorded and fixed. Also, if time and resources permit, beta testers may be 'hired' in order to find bugs in the simulation.

Due Date:

Tuesday, April 12, 2005

Acceptance Criteria:

There are no known bugs in the simulation software, and any known bugs that have not been fixed have been written off as acceptable by both me and the honors committee chair.

Risk Assessment:

The primary risk for this milestone is the inability to fix all bugs with the simulation. The best way to prevent this situation is to have testers perform testing on the simulation in order to free up time for myself in order to fix already known bugs.

Deliverables:

- There are no bugs left in the simulation code.

Milestone #11: User Manuals Milestone

Short Description:

User manuals will be written for all three development tools as well as for the simulation itself. The tool manuals will detail how to make all of the data in order to create a simulation, and the simulation manual will teach a user how to use the simulation properly.

Due Date:

Tuesday, April 19, 2005

Acceptance Criteria:

Manuals are approved by the honors committee chair.

Risk Assessment:

There are no major risks involved in this project milestone.

Deliverables:

- User Manual for the Material Editor Tool
- User Manual for the E.D.G.E. Tool
- User Manual for the W.I.M. Tool
- User Manual for the Simulation Software

Milestone #12: Honors Project Defense Milestone

Short Description:

This is the actual presentation of the honors project, including but not limited to: all documentation, source code, user manuals, and a working version of the simulation itself.

Due Date:

Tuesday, April 26, 2005

Acceptance Criteria:

The honors committee approves of my honors project after I present all of the necessary information.

Risk Assessment:

At this point, there is little that can be done in order to assure success other than to practice the presentation and to organize all honors project materials.

Deliverables:

- Honors Presentation

4.2.2. Data Dictionary

This section has been moved to the Detailed Design Document.

5. Work Packages, Schedule, and Budget

5.1. Estimate of Time Commitment

The amount of time to be spent on the honors project in the period of a week is no less than 12 hours with an average of 12-15 hours per week, including weekly review meetings.

5.2. Schedule

Fall 2004

September 2004

Week 1:	Requirements Document
Week 2:	Specification Document
Week 3:	Specification Document <i>End of Specification Documentation Milestone</i>
Week 4:	Read Book: Game Coding Complete
Week 5:	Read Book: Core Techniques and Algorithms

October 2004

Week 1:	Read Book: Tricks of Windows Gurus
Week 2:	Read Book: 3D Terrain Engines
Week 3:	Review one MUD Codebase (DoT)
Week 4:	Review one 3D Engine (Quake 2) <i>End of Literature Research Milestone</i>

November 2004

Week 1:	Design Documentation
Week 2:	Design Documentation
Week 3:	Design Documentation – First Complete Draft
Week 4:	THANKSGIVING BREAK

December 2004

Week 1:	Design Documentation
Week 2:	Design Documentation <i>End of Detailed Design Documentation Milestone</i>
Week 3:	FINALS WEEK
Week 4:	VACATION
Week 5:	VACATION

Spring 2004

January 2004

Week 1: VACATION
Week 2: Material Editor
Week 3: E.D.G.E. Tool
Week 4: E.D.G.E. Tool
End of Tool Creation Milestone

February 2004

Week 1: W.I.M. Tool and Rendering Environment
Week 2: W.I.M. Tool and Rendering Environment
Week 3: W.I.M. Tool and Rendering Environment
Week 4: W.I.M. Tool and Rendering Environment
End of First Production Milestone

March 2004

Week 1: SPRING BREAK (PC Movement and Collision)
End of Second Production Milestone
Week 2: Display Objects and Object Physics
Week 3: Display Objects and Object Physics
End of Third Production Milestone
Week 4: Display of Object Properties
Week 5: Display of Object Properties
End of Fourth Production Milestone

April 2004

Week 1: *Content Complete Milestone*
Week 2: *Zero Bugs Milestone*
Week 3: *User Manuals Milestone*
Week 4: *Honors Defense Milestone* (Finals Week)

Section 3

Detailed Design Document

Detailed Design Document

Design & Testing Information

For

An Honors Project in Computer Science

Honors Project Developer:
Jason M. Black

Honors Committee Chair:
Dr. Eileen M. Peluso

Honors Committee Members:
Dr. Santhusht S. deSilva
Dr. David G. Fisher
David Heffner, Associate Dean of
Information Technology

Table of Contents

- 1. Project Overview**
 - 1.1. Relation to the SPMP**
 - 1.2. Work Packages**
 - 1.2.1. Honors Project Module List
 - 1.2.2. Detailed Module Descriptions Format
 - 1.2.3. Detailed Module Descriptions
 - 1.3. Module Dependencies (Data Flow Diagram)**
- 2. Detailed Module Information**
 - 2.1. Detailed Class Diagrams (by Module)**
 - 2.1.1. Material Editor
 - 2.1.2. E.D.G.E. Tool (Entity Dynamic Generation Environment)
 - 2.1.3. W.I.M. Tool (World Instance Manager)
 - 2.1.4. Window and State Management Framework
 - 2.1.5. Debugging Console
 - 2.1.6. Data Loading
 - 2.1.7. User Input
 - 2.1.8. Text Manipulation and Display
 - 2.1.9. Screen Management
 - 2.1.10. Camera
 - 2.1.11. Terrain Rendering
 - 2.1.12. Graphics / Rendering Pipeline
 - 2.1.13. Collision Detection
 - 2.2. Detailed Data Dictionary**
 - 2.3. Detailed Function Library**
 - 2.3.1. Material Editor
 - 2.3.2. E.D.G.E. Tool (Entity Dynamic Generation Environment)
 - 2.3.3. W.I.M. Tool (World Instance Manager)
 - 2.3.4. Window and State Management Framework
 - 2.3.5. Debugging Console
 - 2.3.6. Data Loading
 - 2.3.7. User Input
 - 2.3.8. Text Manipulation and Display
 - 2.3.9. Screen Management
 - 2.3.10. Camera
 - 2.3.11. Terrain Rendering
 - 2.3.12. Graphics / Rendering Pipeline
 - 2.3.13. Collision Detection
- 3. Acceptance Testing**
 - 3.1. Milestone Test List**
 - 3.2. Acceptance Test Details**
- 4. Appendix**
 - A. Test Driver for Module #6 (Data Loading)**
 - A.1. Driver Code
 - A.2. .wid Test File
 - A.3. .elb Test File
 - A.4. .mlb Test File
 - A.5. Expected Driver Output

1. Project Overview

1.1. Relation to the SPMP

The Detailed Design Document, while a separate document from the SPMP, is in many ways an extension of that document. The goals described in the SPMP are broken down further into modules here, and these modules are described at the function and data level. Also, the flow of data between these modules and data sources is described in the Data Flow Diagram. The data dictionary from the SPMP will be expanded and more detailed in this document. Finally, the exact nature of any tests used during the production milestones are listed and detailed. The Detailed Design Document is primarily the technical aspect of the information presented in the SPMP.

1.2. Work Packages

1.2.1. Honors Project Module List

The honors project code can be broken down into modules as follows:

Developer Modules

1. Material Editor
2. E.D.G.E. Tool
3. W.I.M. Tool

Simulation Modules

4. Window and State Management Framework
5. Debugging Console
6. Data Loading
7. User Input
8. Text Manipulation and Display
9. Screen Management
10. Camera
11. Terrain Rendering
12. Graphics / Rendering Pipeline
13. Collision Detection

1.2.2. Detailed Module Descriptions Format

Module Name

Type: The type of module. (Stand-Alone Program || C++ Utility Functions || C++ Class(es) and Functions || C++ Framework)

Input: This is the input the program takes, and where it comes from.

Output: What is the final product of this module?

Method: How was this coded? (Coded [in Visual Basic.NET / in C++])

Description: This is a full description of the module, its components, its uses, and also any technical aspects (algorithms chosen, etc.) that are currently known.

1.2.3. Detailed Module Descriptions

Material Editor : Module #1

<u>Type:</u>	Stand-Alone Program.
<u>Input:</u>	user input
<u>Output:</u>	Material database .XML file
<u>Method:</u>	Coded in Visual Basic.NET.
<u>Description:</u>	This will be a single page form used to browse through an .XML file, add entries, delete entries, and edit existing entries. There will also be an option of choosing which .XML database file to be used / edited, including the option of creating a brand new database. Each entry in the database will consist of the following fields: Name, Mass, ID, and Friction Rating (scale to be determined). Name is the name of the material. Mass is the mass of an object given an atomic piece of it (this is calculated using unit mass and AMUs). ID is an identification number for each material which must be unique across all libraries. Friction Rating (coefficient of friction?) will help determine how rough the surface of an object is, but the method of creating such a scale is to be determined.

E.D.G.E. Tool : Module #2

<u>Type:</u>	Stand-Alone Program.
<u>Input:</u>	User input, material database .XML file
<u>Output:</u>	Entity database .XML file
<u>Method:</u>	Coded in Visual Basic.NET.
<u>Description:</u>	The Entity Dynamic Generation Environment Tool will be composed of the following subcomponents: entity library manager, entity editor, and the material library loader.

The entity library manager will allow the loading of .XML files containing complete information on multiple entities. New libraries may be created at the user's discretion, thereby allowing entities to be sorted in multiple files. Any entity from a loaded library may be opened up in the entity editor portion of this program to be updated and saved, or deleted.

The entity editor will allow the developer to edit the following fields of entity information: Name, ID, Material List, X-File, Height, Width, Depth, and an Immobile flag. Name is a string that identifies an entity, but which does not need to be unique to that entity. ID is a user-assigned identification number for the entity that must be unique among all entities in a library (and other entity libraries as well, if they are to be used together). Material List is a list of all materials the entity is composed of as well as a percentage breakdown of that composition. These materials may come from multiple libraries. X-File is a reference to an .X file mesh that is the graphical representation of the entity in the simulation. Height, Depth and Width determine the dimensions of the entity in the simulation. Finally, the Immobile flag determines whether an entity can be moved due to force or whether it will always remain in the position that it loaded in.

The material library loader will allow the loading of different .XML material libraries that may be used to fill out the Material List in the above entity editor.

W.I.M. Tool : Module #3

<u>Type:</u>	Stand-Alone Program.
<u>Input:</u>	User input, entity database .XML file
<u>Output:</u>	World file (format TBA)
<u>Method:</u>	Coded in Visual Basic.NET.
<u>Description:</u>	The World Instance Manager Tool will be composed of the following subcomponents: bitmap editor, entity library loader, world file manager, and the world file editor.

The bitmap editor will consist of a display region for the bitmap, controls to edit the bitmap, and buttons to save and load bitmaps. The display region will be X by Y pixels in size (TBA). The editing controls will be a wand and two or three circles, all of which will allow the user to either increase or decrease the color in a bitmap as the control is drug around using the mouse. The bitmap itself will be in black and white and so each pixel will have a numerical value from 0 to 255 where 0 is the lowest elevation and 255 is the highest.

The entity library loader functions much like the material library loader in the E.D.G.E. Tool. It will allow the developer to switch between different .XML libraries of entities as well as to create new libraries if so desired.

The world file manager will allow the developer to load previous world file data in order to update it in the world file editor, and to save current world file data in the editor over an old world file or as a new world file.

The fourth subcomponent is the world file editor, which is basically the form that world file data is loaded into or typed into in order to set all of the environment parameters for the simulation. Fields in this form are: Name, ID, Bitmap Filepath, and the Entity List and Instance Data. Name is the non-unique string that references the world and ID is the unique identifier, which both work exactly as they do for entities and materials. Bitmap Filepath is a string that points to the location of the bitmap to be used for the world's height map.

The complicated part of this structure is the Entity List and Instance Data. There will be a list containing an entry for each entity to be loaded into the simulation. If there are five instances of a single entity known as "Rock" then there will be five entries labeled "Rock" in the entity list. Each of these instances contains a reference to the entity data as well as position information. Position information for a given entity is set as follows: the developer chooses an entity from a list created by the entity library loader. This entity is then added to the entity list. The developer may then either enter position information directly into the entity list or may drag the entry over to the bitmap editor. At this point the mouse cursor will be a small point. When the mouse button is released the entity's position information will be updated to the bitmap position the mouse is located at. The vertical position of the entity will be set to ground level by default, but the developer may change this at his or her discretion. The most likely GUI structure for the entity list will be an advanced list box that allows entries to have subentries and that will allow editing of both entries and subentries.

Window and State Management Framework : Module #4

Type: C++ Framework
Input: User input
Output: Changes state switch
Method: Coded in C++.
Description: The framework for this simulation is an adaptation of the Direct3D Application Framework class (CD3DApplication) combined with a small amount of functionality from the GDI portion of the Win32 API. The rest of the framework is composed of switch statements that allow easy state control using a global variable and housekeeping DirectX function calls.

Debugging Console : Module #5

Type: C++ Class and Functions
Input: User input, global world data
Output: Debugging data to screen
Method: Coded in C++.
Description: The debugging console is a tool primarily used by the programmer of a project, but which has its uses once the project is released. For this project, the debugging console will be accessible at runtime by a user by pressing a key (most likely '~') and then typing in commands at the prompt that appears. The debugging console will be a semi-transparent screen that will take up the top quarter of the screen. Commands that will be programmed into the console will most likely be commands that imitate normal user actions as well as commands that output object data so that it may be examined at runtime. There may be some useful end-user functionality that can be added to the console at a later date, but there are no such plans for such features in the foreseeable future.

Data Loading : Module #6

Type: C++ Class and Functions
Input: .XML Files
Output: Global world data
Method: Coded in C++.
Description: This series of functions will load the data from the material and entity .XML files, as well as the data from the world file, into memory. This does not include any data that may be directly accessed from the .XML databases during runtime (specific data TBA). Most, if not all, of this data will be stored in the world class and its substructures.

User Input : Module #7

Type: C++ Classes and Functions
Input: User input
Output: Control for state switching, camera position, and console commands
Method: Coded in C++.
Description: While some of the user's input may be handled through Win32 API functionality, it will be more centralized if this functionality is encapsulated in keyboard and mouse classes. Both of these classes will deal with DirectInput in order to adapt to whatever peripherals the user may use. The sole purpose of this module is the aforementioned encapsulation and centralization.

Text Manipulation and Display : Module #8

Type: C++ Utility Functions
Input: Strings from screen classes
Output: Text to screen
Method: Coded in C++.
Description: These independent utility functions will encapsulate DirectX's text drawing functions in order to make the placement of text easier. The including of text positioning based on the size of the rendered text is the primary concern of this module.

Screen Management : Module #9

Type: C++ Classes and Functions
Input: State control switch
Output: Screens and menus displayed to screen, calls to text manipulation
Method: Coded in C++.
Description: This set of classes will systematically create and destroy screens as needed. A screen consists of a background as well as menu options consisting of text displayed through the screen (courtesy of the Text Manipulation module) and connections to functionality elsewhere in the program. Therefore a screen is anything from a plain screen that displays text until it is clicked upon to a menu listing several options of how the user can proceed. This module receives its instructions from the Framework Module which has to handle pausing and resuming the simulation during screen display.

Camera : Module #10

Type: C++ Class and Functions
Input: User input, collision detection
Output: Updates the region of terrain and entities rendered to the screen
Method: Coded in C++.
Description: The camera class represents the point at which the user is looking into the simulated world. This means the camera has to know its angle (both vertical and horizontal), its position in three dimensions, as well as its viewing distance. The camera will move as the user directs it since the camera is for all intents and purposes the eyes of the character the simulation which the user is looking through. This class will use collision detection functions in order to prevent the character from existing in the same location as an entity or the terrain itself.

Terrain Rendering : Module #11

Type: C++ Class and Functions
Input: Global world data (geometry), camera information
Output: Render terrain to screen
Method: Coded in C++.
Description: The terrain renderer is responsible for converting the bitmap height map referenced in the world file into a three dimensional terrain. The two aspects of designing this module is the algorithm used in breaking the terrain into segments for efficient creation and display, and the actual rendering of the terrain. For this project, the terrain will be broken into simple strips unless time permits a more complex technique to be used, and then each of these strips will be rendered separately.

Graphics / Rendering Pipeline : Module #12

Type: C++ Classes and Functions
Input: Global world data (geometry), camera information, collision detection
Output: Render entities to screen
Method: Coded in C++.
Description: The graphics pipeline is responsible for storing the geometry of the entities to be rendered and actually rendering the information to the display. There are many classes that represent different geometrical data for rendering ranging in complexity from a single pixel point to a three dimensional mesh. These classes are detailed in section 2.1.12. Most of the classes and functionality for this module is adapted from preexisting code created by Peter Walsh, author of The Zen of Direct3D Game Programming.

Collision Detection : Module #13

Type: C++ Utility Functions
Input: Global world data (geometry)
Output: Boolean tests on collision sent to the graphics pipeline
Method: Coded in C++.
Description: The collision detection module will be a series of functions that will accept geometry data and determine whether various objects have collided. The collisions to detect are PC to terrain, entity to terrain, entity to entity, and PC to entity. The types of collisions are face to face collisions, point to point collisions, and face to point collisions. Due to time constraints the only collisions that are to be implemented are PC-to-terrain and Entity-to-Terrain, with the latter being of a limited nature.

1.3. Module Dependencies (Data Flow Diagram)

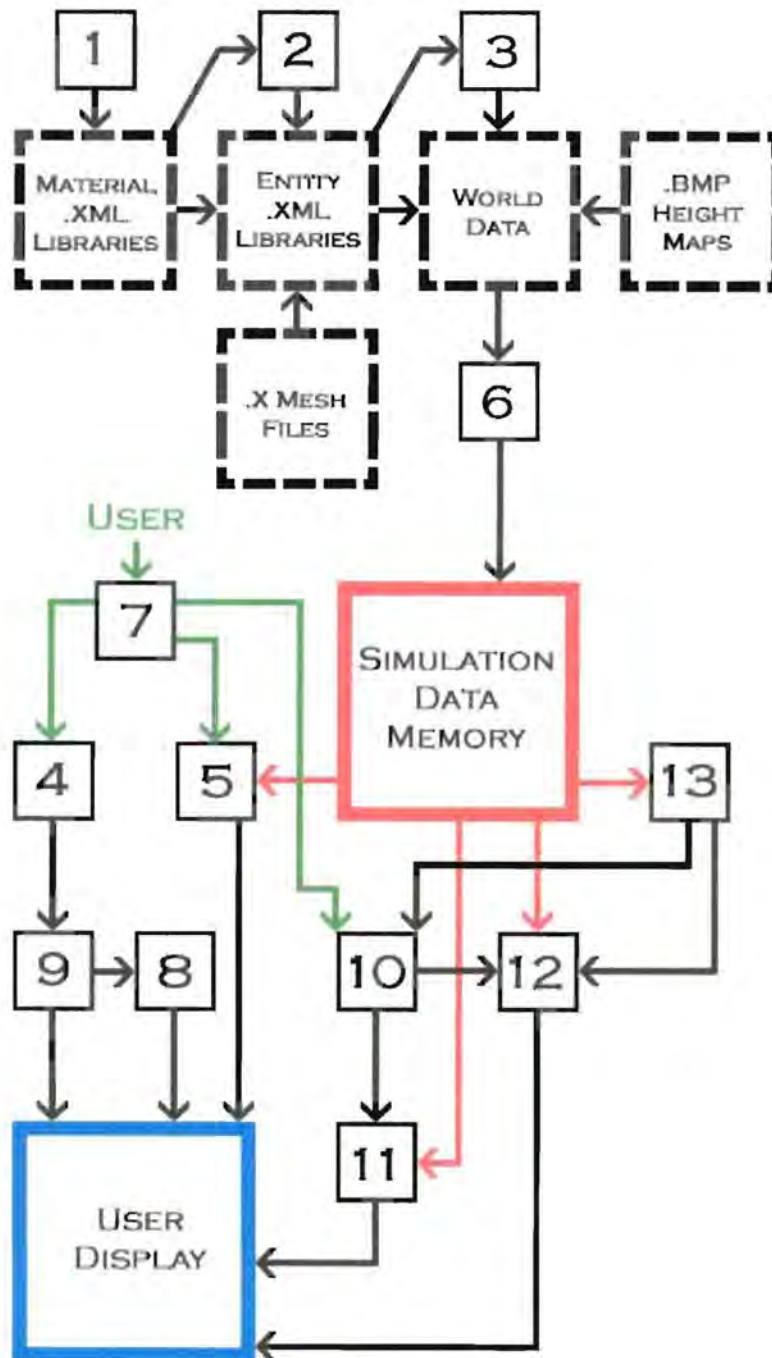


Diagram d-1: Solid boxes with numbers are project modules. Boxes with dashed borders are data repositories. To reference project module numbers with the module names please see section 1.2.1. The 'user' referenced in this diagram is the simulation user. While not labeled, modules one, two, and three take input from the developer user.

2. Detailed Module Information

2.1. Detailed Class Diagrams (by Module)

The following diagrams are broken up as follows: the class name is in its own cell, in bold; the class variables are listed in the middle row to the format “name : type”; and all function names are listed in the last row of the diagram (sans any parameters).

2.1.1. Material Editor

The Material Editor consists of a single class that represents the GUI form and all of its functionality, which is used to make .mlb XML files. Detailed information on all listed functions can be found in section 2.3.1 of this document.

frmMain	
stFilePathAndName : String	
stFileNameOnly : String	
xDoc : DOMDocument	
Nodes : IXMLDOMNodeList	
NewMatType : Integer	
NewComType : Boolean	
LoadXMLFile()	CheckProperSyntaxtandard()
NewXMLFile()	CheckProperSyntaxCombo()
DeleteXMLFile()	DeleteMaterial()
OnMaterialSelect()	DeleteNodeByID()
OnComponentSelect()	RefreshMaterialListBox()
CalcComMass()	RefreshComponentListBox()
CalcComFriction()	GetDependencies()
GetMaterialNameFromID()	SetMaterialListboxFocus()
GetNodeFromID()	ClearComponentListSelections()
ParseNameFromString()	New Component()
ParseIDFromString()	SaveComponent()
ExitProgram()	DeleteComponent()
NewMaterial()	ListInvalidCombinationMaterials()
ClearForm()	DisableObjectsBeforeLoad()
ClearMaterialListSelections()	EnableObjectsAfterLoad()
SaveMaterial()	ClearAll()
IsFileLoaded()	SetComponentListboxFocus()
RecalcMaxID()	

2.1.2. E.D.G.E. Tool (Entity Dynamic Generation Environment)

The EDGE Tool consists of a single class that represents the GUI form and all of its functionality, which is used to make .elb XML files. Detailed information on all listed functions can be found in section 2.3.2 of this document.

frmMain	
stFilePathAndName : String stFileNameOnly : String stMatFilePathAndName : String stMatFileNameOnly : String xDoc : DOMDocument xDocMat : DOMDocument Nodes : IXMLDOMNodeList MatNodes : IXMLDOMNodeList bNewEntity : Boolean	dxD3DX : Direct3D.D3DX dxDevice : Direct3D.Device dxMesh : Direct3D.Mesh oWidth : Double oHeight : Double oDepth : Double bFreezeAdjust : Boolean bFreezeAdjustAll : Boolean
frmMain_Load() ExitProgram() LoadXMLFile() NewXMLFile() DeleteXMLFile() ClearAll() RefreshEntityListBox() RefreshMaterialListBox() ChangeObjectsAfterLoad() DisableCommandsOnDelete() LoadMaterialXMLFile() RecalcMaxID() ClearEntityListSelections() ClearMaterialListSelections() GetNodeFromID() DeleteNodeByID()	GetEntityNameFromID() ParseNameFromString() ParseIDFromString() NewEntity() SaveEntity() DeleteEntity() OnMatSelect() OnEntitySelect() LoadMesh() LoadMeshValues() SetOriginalDimensionsForMesh() SetMaterial() SetEntityListboxFocus() Height_TextChanged() Width_TextChanged() Depth_TextChanged()

2.1.3. W.I.M. Tool (World Instance Manager)

The WIM Tool is composed of multiple GUI forms and several classes containing their functionality. The WIM Tool is used to produce .wid XML files as well as height-map bitmaps. Detailed information on all listed functions can be found in section 2.3.3 of this document.

frmMain	
stFilePathAndName : String	
stFileNameOnly : String	
stEntFilePathAndName : String	
stEntFileNameOnly : String	
xDoc : DOMDocument	
xDocEntity : DOMDocument	
Nodes : IXMLDOMNodeList	
EntityNodes : IXMLDOMNodeList	
LocalEntityNodes : IXMLDOMNodeList	
UserNode : IXMLDOMNode	
BitmapNode : IXMLDOMNode	
bNewLocalEntity : Boolean	
F2 : frmFilename	
F3 : frmBitmap	
objBitmap : Public Bitmap	
GBitmapFilename : Public String	
ExitProgram()	DeleteNodeByCoor()
LoadXMLFile()	GetEntityNameFromCoor()
NewXMLFile()	ParseNameFromString()
DeleteXMLFile()	ParseIDFromString()
SaveXMLFile()	ParseCoorFromString()
ClearAll()	LoadEntityLibrary()
ClearLocalEntity()	OnEntitySelect()
DisplayWorldData()	UseEntity()
ChangeObjectsAfterLoad()	OnLocalEntitySelect()
DisableCommandsOnWorldDelete()	NewLocalEntity()
RefreshEntityListBox()	SaveLocalEntity()
RefreshLocalEntityListBox()	DeleteLocalEntity()
ClearEntityListSelections()	NewBitmap()
ClearLocalEntityListSelections()	LoadBitmap()
SetLocalEntityListboxFocus()	OpenBitmapEditor()
GetNodeFromCoor()	SaveBitmap()

W.I.M. Tool (World Instance Manager) DCD (Continued)

frmFilename	CursorFactory	Coor
GFilename : Textbox	<i>None</i>	x : Integer y : Integer
Accept() Cancel()	LoadCursorFromFile() Create()	<i>None</i>

frmBitmap	BitmapManipStruct	bitmap_manip
OffsetX : Integer OffsetY : Integer X : Integer Y : Integer	BitmapBytes : Byte nStride : Integer TheBitmap : Bitmap BitmapData : BitmapData nTotalSize : Integer	<i>None</i>
OnFormLoad() CloseBitmapEditor() MouseMoveOverBitmap() MouseEntersBitmap() MouseExitsBitmap() MouseClickedOnBitmap() ChangeSensitivity() PerlinNoise() SubdivideDisplace() SDHelper()	Lock() Unlock()	TFInvertBitmap() TFWhitePixel() TFWriteNoisePixel() TFWritePixel() TFCircleTool()

2.1.4. Window and State Management Framework

This is the central portion of code for the entire simulator. The global data members and functions that belong to this module are used to control most of the interactions between the other modules, as well as directly controlling the actual flow of information both in memory and visually to the user.

Independent Variable and Function List	
g_bActive : Boolean	g_SavedPresParams :
g_DeviceHeight : Integer	D3DPRESENT_PARAMETERS
g_DeviceWidth : Integer	g_hWndMain : HWND
g_nStateFlag : Integer	g_hInstMain : HINSTANCE
g_bShowFPS : Boolean	g_pDI : LPDIRECTINPUT8
g_bShowCameraLoc : Boolean	g_dwTerrainColor : DWORD
g_LightCounter : static UINT	g_dwTerrainWireColor : DWORD
g_bConsoleOn : Boolean	g_pBackground :
g_bPauseLock : Boolean	LPDIRECT3DSURFACE9
g_fCameraSpeed : Float	g_pBackSurface :
g_fCameraYaw : Float	LPDIRECT3DSURFACE9
g_bCameraLocked : Boolean	g_pDefaultTexture :
g_bTerrainLoaded : Boolean	LPDIRECT3DTEXTURE9
g_pD3D : LPDIRECT3D9	g_pCursorSurf :
g_pDevice : LPDIRECT3DDEVICE9	LPDIRECT3DSURFACE9
WndProc()	SimRender()
WinMain()	SimCleanup()
SimInit()	ConsoleParser()
InitScene()	InitializeInput()
DestroyScene()	ShutdownInput()
SimLoop()	GetWIDFileNames()
HandleInput()	

2.1.5. Debugging Console

The console consists of a primary class (CConsole), classes representing lines of text in the console and parsed commands (CEntry and CCommand respectively), and global functions that allow the console to display text to the screen.

CConsole	
m_bInitialized : Boolean m_Width : Integer m_Height : Integer m_pConsoleSurface : LPDIRECT3DSURFACE9 m_pConsoleBackgroundSurf : LPDIRECT3DSURFACE9 m_pTargetSurface : LPDIRECT3DSURFACE9	m_pDevice : LPDIRECT3DDEVICE9 m_bVisible : Bool m_pActiveEntry : CEntry * m_pEntryList : CEntry * m_pfnCallback : Function Pointer m_bParserCallback : Boolean _instance : static CConsole *
Instance() CConsole() ~CConsole() Shutdown() Initialize() Render() GetVisibility() SetVisibility()	OutputString() Clear() OnChar() OnKeyDown() SetParserCallback() PreParse() ParseStringForNumber() RotateEntries()

Independent Variable and Function List (FontEngine)	CEntry	CCommand
g_AlphabetWidth : Integer g_AlphabetHeight : Integer g_AlphabetLetterWidth : Integer g_AlphabetLetterHeight : Integer g_AlphabetLettersPerRow : Integer g_pAlphabetSurface : LPDIRECT3DSURFACE9 g_bAlphabetLoaded : Boolean	m_pstrText : char * m_pNext : CEntry * m_nVerticalPos : Integer	pstrCommand : char * NumParams : Integer pstrParams : char *
LoadAlphabet() UnloadAlphabet() PrintChar() PrintString()	CEntry() ~Centry() RenderText() GetNext() SetNext() OnChar()	GetText() SetText() GetTextLength() SetVerticalPos() GetVerticalPos() CCommand() ~CCommand()

2.1.6. Data Loading

The WorldSingleton class is the container for all data created using the three Visual Basic development tools. All of this information, stored in XML files, is pulled into the WorldSingleton class using its member functions and assistant structures.

CWorldSingleton	
_instance : static WorldSingleton *	
sWorldName : String	
sBitmapFilename : String	
TheUser : User	
lstLocalEntities : list<LocalEntity *>	
HeightMap : BYTE *	
ByteRowWidth : Long Integer	
WorldSingleton()	LoadBitmap()
~ WorldSingleton()	BTS()
Instance()	STB()
LoadWIDFile()	StringToInt()
LoadEntityData()	StringToDouble()
LoadMaterialData()	

LocalEntity Struct	User Struct
name : String	x : Integer
x, y, z : Integer	y : Integer
roll : Double	z : Integer
pitch : Double	roll : Double
yaw : Double	pitch : Double
eid, mid : Integer	yaw : Double
elib, mlib : String	
xfile : String	
immobile : Boolean	
height : Double	
width : Double	
depth : Double	
mass : Double	
friction : Double	
xmesh : CZenMesh	

2.1.7. User Input

There are two classes used for user input, one for each of the primary devices. The keyboard class allows the program to test for key presses while the mouse class not only tests for mouse activity but also allows the display of a custom mouse cursor.

CZenMouse	CZenKeyboard
m_pMouseDev : LPDIRECTINPUTDEVICE8 m_bInitialized : Boolean m_bShowCursor : Boolean m_MouseData : DIMOUSESTATE m_position : POINT _instance : static CZenMouse *	m_pKeyDev : LPDIRECTINPUTDEVICE8 m_KeyBuffer[256] : char m_bInitialized : Boolean _instance : static CZenKeyboard *
CZenMouse() ~ CZenMouse() Initialize() Poll() GetMousePos() IsButtonDown() HandleSetCursor() ShowCursor() GetCursorPosition() SetCursorPosition() MoveCursor() UpdateCursorPos() Instance()	CZenKeyboard() ~ CZenKeyboard() Initialize() IsKeyDown() Instance()

2.1.8. Text Manipulation and Display

The sole class used in text manipulation and display is CZenFont. This class doesn't deal with the placement of text on the screen, but the proper display of it. Text placement is handed within module #9, Screen Management.

CZenFont	
m_FontColor : D3DCOLOR	
m_OrigColor : D3DCOLOR	
m_Align : Integer	
m_pFont : LPD3DXFONT	
m_bInitialized : Boolean	
CZenFont()	RestoreColor()
~CZenFont()	OutputText()
Initialize()	GetBoundingBox()
SetColor()	GetPtrToSelf()

2.1.9. Screen Management

The primary screen management class, Screen, represents a single screen visible to the user. The Text class creates objects that represent a single line of text and its formatting. The final class, fontbank, is used to store font formats so that they do not have to be destroyed and recreated every time the text is displayed to the screen.

Text	Screen	Fontbank
m_nID : Integer m_Font : CZenFont m_pTextString : char * m_x : Integer m_y : Integer m_pfnFuncPtr : VoidFuncPtr m_pfnWorldFuncPtr : WorldFuncPtr sWorldFilename : String	m_lstScreenText : list<Text> _instance : static Screen *	m_Fonts : vector<CZenFont> _instance : static Fontbank *
Text() Text(5 params) ~Text() Text(copy constructor) operator = () SetAttributes() GetID() SetFuncPtr() GetFuncPtr() SetWorldFuncPtr() GetWorldFuncPtr() SetWorldFile() GetWorldFile() GetFontPtr() GetTextPtr() GetX() GetY() Render()	Screen() ~Screen() Instance() Clear() SetText() SetFunc() GetTextList() SetWorldFunc() SetWorldFile()	Fontbank() ~Fontbank() Instance() AddFont() GetFont()

2.1.10. Camera

This relatively simply utility class is a wrapper for manipulations of the Direct3D transformation matrix, which controls where in simulated space the ‘camera’ or ‘user’ is seeing from.

CZenCamera	
m_Roll : Float m_Pitch : Float m_Yaw : Float m_position : D3DXVECTOR3 m_LookAt : D3DXVECTOR3 m_Up : D3DXVECTOR3 m_Right : D3DXVECTOR3 m_Velocity : D3DXVECTOR3 instance : static CZenCamera *	
CZenCamera() CZenCamera(copy) ~CZenCamera() SetUp() GetUp() SetRight() GetRight() SetVelocity() GetVelocity() SetPosition() GetPosition() SetLookPoint() GetLookPoint()	Update() Move() SetRoll() GetRoll() SetPitch() GetPitch() SetYaw() GetYaw() Reset() Render() GetSize() Instance()

2.1.11. Terrain Rendering

There are few functions in TerrainSingleton, but each of them are critical to the simulation as a whole. This class contains the transformed .wid data that represents the terrain, and is responsible for rendering and allowing access to information about the terrain.

TerrainSingleton
zvVertex : CZenVertex [500][500]
blsEmpty : Boolean
pVB : LPDIRECT3DVERTEXBUFFER9 [499]
_instance : static TerrainSingleton *
TerrainSingleton()
~ TerrainSingleton()
CreateVertexBuffer()
Render()
GetHeight()
Instance()

2.1.12. Graphics / Rendering Pipeline

The classes used to store graphical information and to render them represent the various geometries that are necessary: vertices, individual faces, and meshes among others. This module also contains functions that deal with timing, 2D graphics, and lighting.

Global Graphic and Timing Functions and Variables	CZenVertex	CZenObject
g_Frequency : Integer g_FrameCount : Integer g_FrameRate : Integer g_FrameDeviance : Float	m_Position : D3DVECTOR m_Normal : D3DVECTOR m_DiffuseColor : D3DCOLOR m_SpecularColor : D3DCOLOR m_tu : Float m_tv : Float	m_strName : char * m_pParentFrame : void * m_pNext : CZenObject *
LoadBitmapToSurface() InitTiming() Pause() GetNumTicksPerMs() FrameCount() SetAmbientLight()	CZenVertex() CZenVertex(copy) ~CZenVertex() Set()	CZenObject() CZenObject(copy) ~ CZenObject() Render() SetNext() GetNext() GetParentFrame() SetParentFrame() GetSize()

CZenFace	CZenMaterial	CZenMesh
m_Vertices : CZenVertex[3] m_pTexture : LPDIRECT3DTEXTURE9 m_bTextureSet : Boolean	m_Material : D3DMATERIAL9	m_NumMats : Integer m_pMesh : LPD3DXMESH m_pTextures : LPDIRECT3DTEXTURE9 * m_pMaterials : CZenMaterial *
CZenFace() CZenFace (copy) ~ CZenFace () SetProps() SetTexture() Render() GetSize()	CZenMaterial() ~ CZenMaterial() SetDiffuse() SetSpecular() SetAmbient() SetEmissive() Update()	CZenMesh() CZenMesh(copy) ~ CZenMesh() LoadXFile() Render() SetMaterial() GetSize() GetMesh()

Graphics / Rendering Pipeline Module DCD (Continued)

CZenFrame	CZenLight
m_pParameter : void * m_mLocal : D3DXMATRIX m_vPosition : D3DXVECTOR3 m_vVelocity : D3DXVECTOR3 m_Yaw : Float m_Pitch : Float m_Roll : Float m_pObjectList : CZenObject * m_pNext : CZenFrame * m_pChildFrameList : CZenFrame * m_pParentFrame : CZenFrame * m_pfnCallback : FRAME_MOVEMENT_CALLBACK m_bCallback : Boolean	m_Light : D3DLIGHT9 m_ID : Integer m_bIsOn : Boolean
CZenFrame() ~ CZenFrame() SetCallback() GetVelocity() SetVelocity() GetPosition() SetPosition() GetLocal() GetYaw() SetYaw() GetPitch() SetPitch() GetRoll() SetRoll() Update() AddObject() Render() SetNext() GetNext() AddFrame() SetParent() GetParent()	CZenLight() CZenLight(copy) ~ CZenLight() SetDiffuse() SetSpecular() SetAmbient() Enable() IsOn() Render() GetSize()

2.1.13. Collision Detection

The collision detection module contains all of the functions necessary to add in realistic physics to the simulation. Due to time restrictions, there is not a lot of content in this module. The functions that are here relate to the implementation of simple gravity for both the user and entities in the environment.

Module Functions and Variables
g_bCameraHitGround : Boolean
g_dJumpVelocity : Double
g_dGravityFactor : Double
g_dGravity : Double
CameraJump()
CameraGravity()
FindHighestTerrainVertex()
EntityGravity()

2.2. Detailed Data Dictionary

The data dictionary has been moved here from the SPMP. This dictionary lists formats and data types. Of particular interest, this section fully details the file data formats, examples of which can be found in Appendix A of this document. (Note: ellipses in the 'format' section denote that more than one instance of the previous tag may follow.)

Material Library XML File (.mlb)

Format:

```
<?xml version="" encoding=""?>
<materiallist maxID="">
  <material ID="" name="" mass="" friction=""/>
  ...
  <combo ID="" name="">
    <component ID="" percent=""/>
    ...
  </combo>
  ...
</materiallist>
```

Tags:

<u>Name</u>	<u>Description</u>
• xml	Holds information about the format of the data file.
• materiallist	Container flag denoting the beginning and end of the material nodes. Also keeps track of the identification numbers for these data nodes.
• material	Contains information about a single material.
• combo	Container flag denoting a material derived from existing materials.
• component	Contains information about one component of the parent combination material.

Attributes:

<u>Name</u>	<u>Description</u>
• xml:version	The version of XML being used.
• xml:encoding	The specific formatting of the XML document.
• materiallist:maxID	This represents the last integer used to identify a material. Used to choose identification numbers for new materials.
• material:ID	This unique identification number is used for reference.
• material:name	A material name is not unique and is used for convenience.
• material:mass	A real number representing the mass of a single unit of this material.
• material:friction	A real number representing the coefficient of friction for this material. While this isn't the actual coefficient of friction, this number allows a scale of frictionless to highly frictional to be used.
• combo:ID	This unique identification number is used for reference.
• combo:name	A material name is not unique and is used for convenience.
• component:ID	This references the material being included as a component.
• component:percent	From 1 to 100, this integer is how much of the combination material is composed of the included material.

Entity Library XML File (.elb file)**Format:**

```

<?xml version="" encoding=""?>
<entitylist maxID="">
  <entity ID="" name="">
    <mlib></mlib>
    <mID></mID>
    <xfile></xfile>
    <immobile></immobile>
    <size height="" width="" depth="" keepratio="" />
  </entity>
  ...
</entitylist>

```

Tags:

<u>Name</u>	<u>Description</u>
• xml	Holds information about the format of the data file.
• entitylist	Container flag denoting the beginning and end of the entity nodes. Also keeps track of the identification numbers for these data nodes.
• entity	Contains information about a single entity.
• mlib	The name of the .mlb file containing the entity's material.
• mID	The entity's material ID.
• xfile	The name of the .x file, where the 3D mesh is stored.
• immobile	This flag denotes whether an entity can move or not. (0 or 1)
• size	Render size attributes of the entity.

Attributes:

<u>Name</u>	<u>Description</u>
• xml:version	The version of XML being used.
• xml:encoding	The specific formatting of the XML document.
• entitylist:maxID	This represents the last integer used to identify an entity. Used to choose identification numbers for new entities.
• entity:ID	This unique identification number is used for reference.
• entity:name	An entity name is not unique and is used for convenience.
• mlib:TEXT	See tag description.
• mID:TEXT	See tag description.
• xfile:TEXT	See tag description.
• immobile:TEXT	See tag description.
• size:height	A double representing the height of the entity's 3D mesh.
• size:width	A double representing the width of the entity's 3D mesh.
• size:depth	A double representing the depth of the entity's 3D mesh.
• size:keepratio	Used in the EDGE tool in order to equalize the dimensions as the user changes them.

World Instance Data XML File (.wid file)**Format:**

```
<?xml version="" encoding=""?>
<world name="">
  <locals>
    <entity x="" y="" z="" roll="" pitch="" yaw=""
      name="" eID="" elib=""/>
    ...
  </locals>
  <bitmap filename=""/>
  <user x="" y="" z="" roll="" pitch="" yaw=""/>
</world>
```

Tags:

<u>Name</u>	<u>Description</u>
• xml	Holds information about the format of the data file.
• world	Container flag denoting the beginning and end of the world data.
• locals	Container flag for the list of local entity instances.
• entity	Information about a single local entity instance.
• bitmap	Contains information about the terrain file.
• user	The user's initial position is stored here. (camera position)

Attributes:

<u>Name</u>	<u>Description</u>
• xml:version	The version of XML being used.
• xml:encoding	The specific formatting of the XML document.
• world:name	Used as a convenience reference for a world. Non unique.
• entity:x	The x coordinate that the entity loads to.
• entity:y	The y coordinate that the entity loads to.
• entity:z	The z coordinate that the entity loads to.
• entity:roll	This integer is the angle that the entity is rotated around the y-axis (north-south direction). A change in roll causes the entity to tilt to either side, as if leaning.
• entity:pitch	This integer is the angle that the entity is rotated around the x-axis (east-west direction). A change in pitch causes the entity to face upwards or downwards instead of straight ahead.
• entity:yaw	This integer is the angle that the entity is rotated around the z axis (up-down direction). A change in yaw causes an entity to turn left or right.
• entity:eID	The identification number of the base entity.
• entity:elib	The filename of the .elb library containing the base entity data.
• bitmap:filename	The filename of the bitmap used for the terrain's heightmap.
• user:x	The x coordinate that the user loads to.
• user:y	The y coordinate that the user loads to.
• user:z	The camera's (user's) height above the terrain at all times.
• user:roll	The 'roll' here is the same as 'roll' for entity.
• user:pitch	The 'pitch' here is the same as 'pitch' for entity.
• user:yaw	The 'yaw' here is the same as 'yaw' for entity.

2.3. Detailed Function Library

Before browsing through the hundreds of function definitions on the following pages there are a few important notes. The ‘Notes’ field is used to describe what a function does when the function name and other information doesn’t make this obvious. Underlined headers precede each class or group of functions. Constructors, copy constructors, and destructors are not listed.

2.3.1. Material Editor

frmMain Class Functions

LoadXMLFile	
<i>Input Parameters</i>	User selects an .mlb library in a dialog box.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	RefreshMaterialListBox, EnableObjectsAfterLoad, ClearAll, RecalcMaxID
<i>Notes</i>	XML data from an .mlb library is loaded and the editor is set up for use.

NewXMLFile	
<i>Input Parameters</i>	User enters the filename for a new .mlb library.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	EnableObjectsAfterLoad, ClearAll
<i>Notes</i>	An empty .mlb library is created.

DeleteXMLFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DisableObjectsBeforeLoad, ClearAll
<i>Notes</i>	The currently loaded .mlb library is deleted from memory.

OnMaterialSelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearComponentListSelections, ClearMaterialListSelections, ParseIDFromString, GetMaterialNameFromID, CalcComMass, CalcComFriction
<i>Notes</i>	Handles the display of material information and the state of text boxes based on what material is currently selected.

OnComponentSelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearComponentListSelections, ParseIDFromString, ParseNameFromString
<i>Notes</i>	Handles the display of component information and the state of text boxes based on what component is selected.

CalcComMass	
<i>Input Parameters</i>	An IXMLDOMNode object.
<i>Function Output</i>	A double, representing the mass.
<i>Functions Referenced</i>	CalcComMass
<i>Notes</i>	Calculates the mass of a material, recursively checking the properties of any and all components.

CalcComFriction	
<i>Input Parameters</i>	An IXMLDOMNode object.
<i>Function Output</i>	A double, representing the friction.
<i>Functions Referenced</i>	CalcComFriction
<i>Notes</i>	Calculates the friction of a material, recursively checking the properties of any and all components.

GetMaterialNameFromID	
<i>Input Parameters</i>	A string containing a material ID.
<i>Function Output</i>	A string containing a material name.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetNodeFromID	
<i>Input Parameters</i>	A string containing a material ID.
<i>Function Output</i>	An IXMLDOMNode object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseNameFromString	
<i>Input Parameters</i>	A string in the format “name ID”.
<i>Function Output</i>	A string containing the name only.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseIDFromString	
<i>Input Parameters</i>	A string in the format “name ID”.
<i>Function Output</i>	A string containing the ID only.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ExitProgram	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Exits the program.

NewMaterial	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, RecalcMaxID
<i>Notes</i>	Sets up the editor for the entry of a new material.

ClearForm	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears all text boxes and list boxes on the form.

ClearMaterialListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselects any selection made in the material listbox.

SaveMaterial	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, RefreshMaterialListBox, SetMaterialListboxFocus, CheckProperSyntaxStandard, DeleteNodeByID, RefreshMaterialListBox, CheckProperSyntaxCombo
<i>Notes</i>	Updates a currently selected material if changes have been made in the editor, or saves a new material created in the editor.

IsFileLoaded	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Boolean flag based on the function name.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CheckProperSyntaxStandard	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Boolean flag based on the function name.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CheckProperSyntaxCombo	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Boolean flag based on the function name.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

DeleteMaterial	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, GetDependencies, DeleteNodeByID, RefreshMaterialListBox
<i>Notes</i>	Deletes the currently selected material from the form and from the library.

DeleteNodeByID	
<i>Input Parameters</i>	String containing a material ID.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetNodeByID
<i>Notes</i>	N/A

RefreshMaterialListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Refresh the contents of the material listbox.

RefreshComponentListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetNodeFromID, GetMaterialNameFromID
<i>Notes</i>	Refresh the contents of the component listbox.

GetDependencies	
<i>Input Parameters</i>	An integer containing a material ID.
<i>Function Output</i>	An array passed by reference containing a list of all materials that are dependant on the material referenced by the ID.
<i>Functions Referenced</i>	GetMaterialNameFromID
<i>Notes</i>	N/A

SetMaterialListboxFocus	
<i>Input Parameters</i>	A string containing a material ID.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetMaterialNameFromID
<i>Notes</i>	N/A

ClearComponentListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselects all component in the component listbox.

NewComponent	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, ClearComponentListSelections
<i>Notes</i>	Sets up the editor for the entry of a new component.

SaveComponent	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, GetNodeFromID, ParseNameFromString, ParseIDFromString, RefreshComponentListBox, SetComponentListboxFocus
<i>Notes</i>	Updates an existing component that had been changed in the editor or saves a new component created in the editor.

DeleteComponent	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	IsFileLoaded, GetNodeFromID, ParseIDFromString, RefreshComponentListBox
<i>Notes</i>	Deletes the currently selected component.

ListInvalidCombinationMaterials	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	A message box appears and lists any combination material with components that do not add up to 100%.

DisableObjectsBeforeLoad	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Disables almost all fields and buttons when there's no library present.

EnableObjectsAfterLoad	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Enables a couple of buttons after a library is loaded.

ClearAll	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears all of the editor's fields at once.

SetComponentListboxFocus	
<i>Input Parameters</i>	A string containing a material name and ID.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Selects the appropriate component in the component listbox.

RecalcMaxID	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Resets the maxID attribute of the loaded material library to the smallest possible correct value to prevent it from ballooning out of control from deleted and discarded materials.

2.3.2. E.D.G.E. Tool (Entity Dynamic Generation Environment)

frmMain Class Functions

frmMain Load	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets up the Direct3D Device when the form loads.

ExitProgram	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Exits the program.

LoadXMLFile	
<i>Input Parameters</i>	User selects an .elb library in a dialog box.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, RefreshEntityListBox, ChangeObjectsAfterLoad, RecalcMaxID
<i>Notes</i>	XML data from an .elb library is loaded and the editor is set up for use.

NewXMLFile	
<i>Input Parameters</i>	User enters the filename for a new .elb library.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, RefreshEntityListBox, ChangeObjectsAfterLoad
<i>Notes</i>	An empty .elb is set up for use.

DeleteXMLFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, DisableCommandsOnDelete
<i>Notes</i>	Deletes the currently loaded .elb library from memory.

ClearAll	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Resets all of the text boxes and lists in the form.

RefreshEntityListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Refresh the contents of the Entity list box.

RefreshMaterialListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Refresh the contents of the Material list box.

ChangeObjectsAfterLoad	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Disables and enables commands once a .elb is loaded.

DisableCommandsOnDelete	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Resets the form to its initial state when an entity library is unloaded and deleted.

LoadMaterialXMLFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	RefreshMaterialListBox
<i>Notes</i>	Loads a material library into the editor and lists its contents in a list box.

RecalcMaxID	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Recalculates the highest ID used so that the smallest free ID is used on the next entity created.

ClearEntityListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselects all entries on the entity list.

ClearMaterialListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselects all entries on the material list.

GetNodeFromID	
<i>Input Parameters</i>	A string containing an entity ID.
<i>Function Output</i>	An IXMLDOMNode object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

DeleteNodeByID	
<i>Input Parameters</i>	A string containing an entity ID.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetNodeByID
<i>Notes</i>	N/A

GetEntityNameFromID	
<i>Input Parameters</i>	A string containing an entity ID.
<i>Function Output</i>	A string containing the name of the entity.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseNameFromString	
<i>Input Parameters</i>	A string in the format “name ID”.
<i>Function Output</i>	A string containing the name only.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseIDFromString	
<i>Input Parameters</i>	A string in the format “name ID”.
<i>Function Output</i>	A string containing the ID only.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

NewEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearEntityListSelections
<i>Notes</i>	The form is set up for the entry of a new entity.

SaveEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DeleteNodeByID, RefreshEntityListBox, SetEntityListboxFocus,
<i>Notes</i>	Saves the entity to file if it is a new entity, otherwise it updates an existing entity entry with new information.

DeleteEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DeleteNodeByID, RefreshEntityListBox, ClearAll
<i>Notes</i>	Deletes the currently selected entity and clears the form.

OnMatSelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Enables and disables the ‘Set Material’ function.

OnEntitySelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearEntityListSelections, ParseIDFromString, SetOriginalDimensionsForMesh
<i>Notes</i>	Displays the appropriate information when an entity is selected in the entity list box.

LoadMesh	
<i>Input Parameters</i>	The user selects an .x file from memory.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	LoadMeshValues
<i>Notes</i>	This is a wrapper function for the core functionality stored in LoadMeshValues().

LoadMeshValues	
<i>Input Parameters</i>	A string with the .x mesh's filename.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	SetOriginalDimensionsForMesh
<i>Notes</i>	Loads an .x mesh into memory in order to calculate its bounding box.

SetOriginalDimensionsForMesh	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Stores a copy of an .x file's original dimensions for aspect ratio calculations.

SetMaterial	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ParseIDFromString
<i>Notes</i>	Sets an entity's material.

SetEntityListboxFocus	
<i>Input Parameters</i>	A string containing an entity ID.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetEntityNameFromID
<i>Notes</i>	Focuses on the given entity in the entity list box.

Height_TextChanged	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Aspect ratio code for one of the mesh's dimensions.

Width_TextChanged	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Aspect ratio code for one of the mesh's dimensions.

Depth_TextChanged	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Aspect ratio code for one of the mesh's dimensions.

2.3.3. W.I.M. Tool (World Instance Manager)

frmMain Class Functions

ExitProgram	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Closes down the WIM Tool.

LoadXMLFile	
<i>Input Parameters</i>	User selects a .wid file in a dialog box.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, DisplayWorldData, ChangeObjectsAfterLoad
<i>Notes</i>	The information from a .wid file is loaded into the editor.

NewXMLFile	
<i>Input Parameters</i>	User enters the filename for a new .wid file.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, DisplayWorldData, ChangeObjectsAfterLoad
<i>Notes</i>	A blank .wid file is created and loaded into the editor.

SaveXMLFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Changes to the current .wid file are saved.

DeleteXMLFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearAll, DisableCommandsOnWorldDelete
<i>Notes</i>	The currently loaded .wid file is deleted from memory.

ClearAll	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearLocalEntity
<i>Notes</i>	Clear the WIM Tool of all displayed content.

ClearLocalEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This clears the LocalEntity listbox.

DisplayWorldData	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Display all of the data from a recently loaded .wid file to the screen.

ChangeObjectsAfterLoad	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Reset the state of the Tool's commands when a file is loaded.

DisableCommandsOnWorldDelete	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Reset the state of the Tool's commands when a file is deleted.

RefreshEntityListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Refresh the entity listbox.

RefreshLocalEntityListBox	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Refresh the local entity listbox.

ClearEntityListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselect any selections on the entity listbox.

ClearLocalEntityListSelections	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Deselect any selections on the local entity listbox.

SetLocalEntityListboxFocus	
<i>Input Parameters</i>	A Coor object containing the coordinates of a local entity.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetEntityNameFromCoor
<i>Notes</i>	Select the specified object in the entity listbox.

GetNodeFromCoor	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An IXMLDOMNode containing local entity information.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

DeleteNodeByCoor	
<i>Input Parameters</i>	A Coor object containing the coordinates of a local entity.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetNodeFromCoor
<i>Notes</i>	N/A

GetEntityNameFromCoor	
<i>Input Parameters</i>	A Coor object containing the coordinates of a local entity.
<i>Function Output</i>	A string containing the name of a local entity.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseNameFromString	
<i>Input Parameters</i>	A string from the entity listbox.
<i>Function Output</i>	A string containing the name of an entity.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseIDFromString	
<i>Input Parameters</i>	A string from the entity listbox.
<i>Function Output</i>	A string containing the ID of an entity.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

ParseCoorFromString	
<i>Input Parameters</i>	A string from the local entity listbox.
<i>Function Output</i>	A Coor containing the location of a local entity.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

LoadEntityLibrary	
<i>Input Parameters</i>	User selects an .elb file in a dialog box.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	RefreshEntityListBox
<i>Notes</i>	The entity listbox is filled with the contents of the .elb file.

OnEntitySelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Enable the Use Entity command.

UseEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ParseNameFromString, ParseIDFromString
<i>Notes</i>	Load the information from the entity selected in the entity listbox into the appropriate local entity fields.

OnLocalEntitySelect	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearLocalEntityListSelections, ParseCoorFromString
<i>Notes</i>	Display the local entity's information in the appropriate fields.

NewLocalEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	ClearLocalEntity, ClearLocalEntityListSelections
<i>Notes</i>	Sets up the editor to accept a new local entity.

SaveLocalEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DeleteNodeByCoor, RefreshLocalEntityListBox, SetLocalEntityListboxFocus, ParseCoorFromString
<i>Notes</i>	Saves a new local entity, or updates an existing local entity that has had changes made.

DeleteLocalEntity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DeleteNodeByCoor, RefreshLocalEntityListBox, ClearLocalEntity
<i>Notes</i>	Deletes the currently selected local entity from the .wid file.

NewBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a new blank bitmap and loads it into the editor.

LoadBitmap	
<i>Input Parameters</i>	The user selects a .bmp file from a dialog box.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Loads an existing bitmap into the editor.

SaveBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Saves the current bitmap to file.

OpenBitmapEditor	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Opens the bitmap editing form.

frmFilename Class Functions

Accept	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A string containing the filename for a new bitmap.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the string that the user entered into the form.

Cancel	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns an empty string, ignoring any user input.

frmBitmap Class Functions

OnFormLoad	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Reset all of the form's controls.

CloseBitmapEditor	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Close the bitmap editing form and returns to the main WIM Tool screen.

MouseMoveOverBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	MouseClickedOnBitmap
<i>Notes</i>	Updates coordinate information as the mouse moves over the bitmap. Also edits the bitmap if the appropriate tool is selected and the proper mouse button is held down as the mouse moves.

MouseEntersBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Changes the mouse cursor when it moves over the bitmap to the appropriate tooltip.

MouseExitsBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Changes the mouse cursor when it leaves the bitmap to the appropriate tooltip.

MouseClicksOnBitmap	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	BitmapManipStruct: Unlock
<i>Notes</i>	Performs editing operations on the bitmap based on what tool is currently selected.

ChangeSensitivity	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This changes the strength of the editing tools, making them increase and decrease the height map more drastically when the sensitivity value is higher.

PerlinNoise	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	BitmapManipStruct: Lock, Unlock
<i>Notes</i>	This function was not implemented in the WIM Tool.

SubdivideDisplace	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	SDHelper, BitmapManipStruct: Lock, Unlock
<i>Notes</i>	This function generates random terrain using the Subdivide and Displace method.

SDHelper	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	SDHelper
<i>Notes</i>	This function performs much of the actual Subdivide and Displace algorithm.

CursorFactory Class Functions

LoadCursorFromFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Loads a mouse cursor from a file into memory.

Create	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates the actual mouse cursor seen by the user.

BitmapManipStruct Class Functions

Lock	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Locks the bitmap for editing.

Unlock	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Unlocks the bitmap. Used when the bitmap is no longer being edited.

Bitmap_manip Module Functions

TFInvertBitmap	
<i>Input Parameters</i>	A BitmapManipStruct and two integers containing the size of the bitmap.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	BitmapManipStruct: Lock
<i>Notes</i>	This function inverts the color values of the entire bitmap in a single click.

TFWhitePixel	
<i>Input Parameters</i>	A BitmapManipStruct, two integers containing the size of the bitmap, and two integers containing the local of the mouse click on the bitmap.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	BitmapManipStruct: Lock
<i>Notes</i>	A white pixel is drawn to the bitmap at the specified location.

TFWriteNoisePixel	
<i>Input Parameters</i>	A BitmapManipStruct, two integers containing the local of the mouse click on the bitmap, and a noise value stored in a double.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function is not used anywhere since Perlin Noise was left unimplemented.

TFWritePixel	
<i>Input Parameters</i>	A BitmapManipStruct, two integers containing the local of the mouse click on the bitmap, and the value to be written stored in an integer.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The value passed to the function is written to the bitmap at the specified location.

TFCircleTool	
<i>Input Parameters</i>	A BitmapManipStruct, two integers containing the size of the bitmap, two integers containing the local of the mouse click on the bitmap, a Boolean indicating the direction of the height change, and two integers representing the strength of the height change and the radius of effect.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	BitmapManipStruct: Lock
<i>Notes</i>	This function determines which circle editing tool is selected and performs adjustments to the bitmap based on sensitivity settings as well as what mouse button is being pushed.

2.3.4. Window and State Management Framework

Global Functions

WndProc	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #5 (CConsole) OnChar, OnKeyDown, Module #7 (CZenMouse) Instance, HandleSetCursor
<i>Notes</i>	This is an overload to the standard Windows function that handles various types of input to the program. This overload intercepts the user's input to the console and the use of the mouse.

WinMain	
<i>Input Parameters</i>	Four standard parameters that are always passed to WinMain.
<i>Function Output</i>	An integer indicating whether the program has exited with or without error.
<i>Functions Referenced</i>	SimInit, SimLoop, SimCleanup
<i>Notes</i>	This is the main function for a windows-based program and is similar to the standard main() function in C++. This function is used to call the various simulator functions when and where necessary.

SimInit	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	InitScene, Module #5: LoadAlphabet, (CConsole) Initialize, SetParserCallback, Module #7: (CZenKeyboard) Instance, (CZenMouse) Instance, ShowCursor, HandleSetCursor, Module #9: (FontBank) Instance, AddFont, Module #10: Instance, SetPosition, Module #12: InitTiming,
<i>Notes</i>	N/A

InitScene	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	GetWIDFileNames, Module #6: (WorldSingleton) Instance, Module #8: (CZenFont) GetBoundingBox, Module #9: (FontBank) Instance, GetFont, (Screen) Instance, Clear, SetText, SetFunc, SetWorldFunc, SetWorldFile, Module #10: Instance, SetPosition, Module #11: Instance, CreateVertexBuffer, CreateElevatedVertexBuffer, Module #12: LoadBitmapToSurface
<i>Notes</i>	N/A

DestroyScene	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears the background image loaded for a given scene when the scene changes.

SimLoop	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	HandleInput, SimRender, Module #12: FrameCount
<i>Notes</i>	Loops through the other core functions of the simulator.

HandleInput	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #7: (CZenKeyboard) Instance, IsKeyDown (CZenMouse) Instance, Poll, UpdateCursorPos, GetCursorPosition, IsButtonDown, Module #8: GetBoundingBox, SetColor, RestoreColor, Module #9: (Screen) Instance, GetTextList, (Text) GetFuncPtr, GetFontPtr, GetTextPtr, GetWorldFuncPtr, GetWorldFile, GetX, GetY, Module #10: Instance, GetPosition, SetPosition, GetVelocity, GetRight, Move, GetLookPoint, SetYaw, Update, Module #11: Instance, GetHeight, Module #13: CameraJump, CameraGravity
<i>Notes</i>	Controls what happens when input is received from the mouse or the keyboard.

SimRender	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #5: Instance, PrintString, Render, Module #6: Instance, Module #9: (Screen) Instance, GetTextList, Render, Module #10: Instance, Module #11: Instance, Render, GetHeight, Module #12: (CZenMesh) Render
<i>Notes</i>	Renders all 2D and 3D objects that need to be rendered based on the current state.

SimCleanup	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	DestroyScene, ShutdownInput, Module #5: Instance, Shutdown, UnloadAlphabet
<i>Notes</i>	Cleans up memory when the simulator is shutting down.

ConsoleParser	
<i>Input Parameters</i>	A CCommand object containing the line to be parsed.
<i>Function Output</i>	An integer indicating success.
<i>Functions Referenced</i>	Module #5: (Console) Instance, Clear, OutputString, Module #6: Instance, Module #10: Instance, GetPosition, Module #11: Instance
<i>Notes</i>	This function breaks apart the line input into the console and performs the appropriate command based on the input.

InitializeInput	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #7: (CZenKeyboard) Instance, Initialize, (CZenMouse) Instance, Initialize
<i>Notes</i>	Initialize the variables used by the input procedures.

ShutdownInput	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears variables related to the input objects.

GetWIDFileNames	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A vector containing string objects.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function reads the filenames of all .wid files stored in the local xml directory and stores them in a vector. This text is retrieved in order to display it on a menu to the user.

2.3.5. Debugging Console

Font Engine Global Functions

LoadAlphabet	
<i>Input Parameters</i>	A character string containing the filename of the alphabet image, and two integers representing the width and height of each alphabet character.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #12: LoadBitmapToSurface
<i>Notes</i>	Loads the alphabet bitmap into memory.

UnloadAlphabet	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Releases the surface the alphabet was loaded onto.

PrintChar	
<i>Input Parameters</i>	Two integers representing the location the character should be displayed at, the character itself, a Boolean value indicating whether the character is transparent or not, the color of the character, a pointer to the memory the character is being written to, and an integer containing the destination surface's pitch value.
<i>Function Output</i>	A character is written to a surface.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Used to display a single bitmap character to the screen.

PrintString	
<i>Input Parameters</i>	Two integers representing the location the string should be displayed at, the string of characters, a Boolean value indicating whether the text is transparent or not, the color of the text, a pointer to the memory the character is being written to, and an integer containing the destination surface's pitch value.
<i>Function Output</i>	A string of characters is written to a surface.
<i>Functions Referenced</i>	PrintChar
<i>Notes</i>	Used to display a series of bitmap characters to the screen.

CEntry Class Functions

RenderText	
<i>Input Parameters</i>	An integer for the maximum string length, a pointer to the memory to render the text to, and an integer containing the destination surface's pitch value.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	FontEngine: PrintString
<i>Notes</i>	Renders a line of console text to the display.

GetNext	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the next CEntry.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetNext	
<i>Input Parameters</i>	CEntry pointer to the next row of text.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the CEntry text that follows the current CEntry.

GetText	
<i>Input Parameters</i>	A char* that is filled up with the CEntry text and an integer of how many characters to copy.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Copies the CEntry text to the char*.

SetText	
<i>Input Parameters</i>	A char* that is used to set the value of the CEntry text.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

OnChar	
<i>Input Parameters</i>	A character representing what key was just pressed.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Adds appropriate characters to the buffer (CEntry text) when a key is pressed.

GetTextLength	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	The number of characters in the string as an integer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetVerticalPos	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	The vertical position of the row of text as an integer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetVerticalPos	
<i>Input Parameters</i>	An integer containing the y position for rendering.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CConsole Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton CConsole object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a CConsole object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

Shutdown	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	CConsole destructor
<i>Notes</i>	Empties the CConsole of all data.

SetParserCallback	
<i>Input Parameters</i>	A function pointer to a parsing function.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Clear	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	CEntry: GetNext, SetNext
<i>Notes</i>	Clears the contents of all console CEntry text.

OnChar	
<i>Input Parameters</i>	A character representing the key that was pressed.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	CEntry: OnChar
<i>Notes</i>	Interprets key presses as commands or as text entered into the buffer.

OnKeyDown	
<i>Input Parameters</i>	A representation of the key that was pressed.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	SetVisibility, OnChar, PreParse, OutputString, RotateEntries. CEntry: GetText
<i>Notes</i>	This function handles non-character keyboard input for special commands.

PreParse	
<i>Input Parameters</i>	A character string from a command entered into the console and a pointer to a CCommand object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The character string is split apart into the command name and parameters and is then stored in the CCommand object.

OutputString	
<i>Input Parameters</i>	A character string and a Boolean.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	RotateEntries. CEntry: SetText
<i>Notes</i>	The character string is sent to the console. The Boolean determines whether the string is simple output or a special console message, and formats the output accordingly.

RotateEntries	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	CEntry: GetNext, GetVerticalPos, SetVerticalPos, SetNext
<i>Notes</i>	The lines of console text are all moved up a position to make room for a new line at the bottom. If a line is moved beyond the upper limit it is erased.

Initialize	
<i>Input Parameters</i>	A D3D device pointer and a pointer to the surface the console is to be rendered to.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Shutdown. Module #12: LoadBitmapToSurface. CEntry: GetNext, SetNext, SetVerticalPosition
<i>Notes</i>	The console is initialized with starting values.

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	FontEngine: PrintString. CEntry: GetNext, RenderText.
<i>Notes</i>	Renders the console and its text to the display.

ParseStringForNumber	
<i>Input Parameters</i>	A character string.
<i>Function Output</i>	An integer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function searches the string for words that can be translated into numerical values and then returns the values.

GetVisibility	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A Boolean concerning the console's visibility.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetVisibility	
<i>Input Parameters</i>	A Boolean.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Set the console's visibility.

2.3.6. Data Loading

WorldSingleton Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Returns a pointer to the WorldSingleton object if it exists, otherwise, it creates a WorldSingleton object and returns its pointer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

LoadWIDFile	
<i>Input Parameters</i>	A string containing the filename of the .wid file to load.
<i>Function Output</i>	A Boolean value indicating success.
<i>Functions Referenced</i>	LoadEntityData, LoadBitmap, BTS, STB, StringToInt, StringToDouble
<i>Notes</i>	Data from the .wid file is loaded into a new LocalEntity structure to be added to the WorldSingleton.

LoadEntityData	
<i>Input Parameters</i>	A pointer to a LocalEntity structure, to be filled.
<i>Function Output</i>	A Boolean value indicating success.
<i>Functions Referenced</i>	LoadMaterialData, BTS, STB, StringToInt
<i>Notes</i>	Data from the .elb file (including .x meshes) is loaded into the above LocalEntity structure.

LoadMaterialData	
<i>Input Parameters</i>	A pointer to a LocalEntity structure, to be filled.
<i>Function Output</i>	A Boolean value indicating success.
<i>Functions Referenced</i>	BTS, STB, StringToInt, StringToDouble
<i>Notes</i>	Data from the .mlb file is loaded into the above LocalEntity structure.

LoadBitmap	
<i>Input Parameters</i>	A character pointer referencing the filename of the bitmap to be loaded.
<i>Function Output</i>	A BYTE pointer, referencing the array of BYTE values culled from the bitmap.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function reads the color values of the bitmap into memory and stores them in a BYTE * structure to be later stored in the WorldSingleton.

BTS	
<i>Input Parameters</i>	A <code>_bstr_t</code> string.
<i>Function Output</i>	A STL string.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function performs a conversion between string types.

STB	
<i>Input Parameters</i>	A char * string.
<i>Function Output</i>	A BSTR string.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function performs a conversion between string types.

StringToInt	
<i>Input Parameters</i>	A STL string.
<i>Function Output</i>	An integer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This is a simple type conversion function.

IntToString	
<i>Input Parameters</i>	A STL string.
<i>Function Output</i>	A double.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This is a simple type conversion function.

2.3.7. User Input

CZenKeyboard Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton CZenKeyboard object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a CZenKeyboard object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

Initialize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Initialize the keyboard.

IsKeyDown	
<i>Input Parameters</i>	An integer representing the key pressed.
<i>Function Output</i>	A Boolean value representing whether a key is down.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CZenMouse Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton CZenMouse object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a CZenMouse object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

Initialize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Initialize the mouse object and load an image to represent the cursor.

Poll	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Attempts to keep the mouse's focus on the current window and updated current mouse information for the class.

GetMousePos	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A POINT object representing the current location of the mouse.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

IsButtonDown	
<i>Input Parameters</i>	An integer marking which button is to be tested, where 0 is the primary button, 1 is the secondary button, and 2 is the middle button.
<i>Function Output</i>	A Boolean value representing whether the indicated button is up or down.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

HandleSetCursor	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function tells Windows not to do anything to the mouse cursor since the simulation will be taking care of it.

ShowCursor	
<i>Input Parameters</i>	A Boolean on whether the cursor is currently visible.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetCursorPosition	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Two integers passed by reference return the current cursor position.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetCursorPosition	
<i>Input Parameters</i>	Two integers representing the cursor's position.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Moves the cursor to the indicated position.

MoveCursor	
<i>Input Parameters</i>	Two integers representing the cursor's position.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Moves the cursor from its original position by the indicated distances.

UpdateCursorPos	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Updates the cursor's position based on mouse polling data.

2.3.8. Text Manipulation and Display

CZenFont Class Functions

Initialize	
<i>Input Parameters</i>	An HFONT format object and the color of the font.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	A new font is created.

SetColor	
<i>Input Parameters</i>	A new font color.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the text to a new, temporary color.

RestoreColor	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Resets the font to its original color from initialization.

OutputText	
<i>Input Parameters</i>	A char* containing the text to be rendered, and the coordinates it is to be rendered at.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetBoundingBox	
<i>Input Parameters</i>	A char* containing the text that will be rendered later.
<i>Function Output</i>	Two integers passed by reference.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The integers are filled with the width and height of the font once it is rendered.

GetPtrToSelf	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the parent CZenFont object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

2.3.9. Screen Management

Fontbank Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton Fontbank object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a Fontbank object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

AddFont	
<i>Input Parameters</i>	An integer ID and a CZenFont object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Adds the font to the fontbank under the indicated ID.

GetFont	
<i>Input Parameters</i>	An integer ID.
<i>Function Output</i>	A CZenFont pointer to the font with the matching ID.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Screen Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton Screen object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a Screen object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

Clear	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears the screen object of all text entries.

SetText	
<i>Input Parameters</i>	An integer ID, a CZenFont pointer, a char* containing a line of text, and two integers for the position the text is to be displayed at.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Adds a Text objects to the screen's list.

SetFunc	
<i>Input Parameters</i>	An integer ID and a void function pointer.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Associates a void function with the line of text with the given ID.

GetTextList	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the list of Text objects to be displayed.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetWorldFunc	
<i>Input Parameters</i>	An integer ID and a function pointer.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Associates a function with the line of text with the given ID.

SetWorldFile	
<i>Input Parameters</i>	An integer ID and a string of text.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Associates a filename with the line of text with the given ID. This is used to associate filenames with their names rendered to the display.

Text Class Functions

SetAttribute	
<i>Input Parameters</i>	An integer ID, a CZenFont pointer, a char* containing a line of text, and two integers for the position the text is to be displayed at.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Initializes all of the object's properties.

GetID	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer ID of the object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetFuncPtr	
<i>Input Parameters</i>	A void function pointer.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Associates a pointer with the Text object.

GetFuncPtr	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A void function pointer.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetWorldFuncPtr	
<i>Input Parameters</i>	A function pointer.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Associates a function with the Text object for use in associating a filename with its name rendered to the display.

GetWorldFuncPtr	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A function pointer to the filename world function.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetWorldFile	
<i>Input Parameters</i>	A string object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the filename to the value of the string object.

GetWorldFile	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A string object containing the filename value.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetFontPtr	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetTextPtr	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Text object's font object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetX	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer of the Text object's future X position.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetY	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer of the Text object's future Y position.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	CZenFont: OutputText
<i>Notes</i>	Renders the Text object to the display.

2.3.10. Camera

CZenCamera Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the Singleton CZenCamera object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a CZenCamera object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

Update	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets Direct3D's transformation matrix based on all of the values currently in the camera.

Move	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The camera's position is adjusted by the three coordinated passed to it as floating point numbers. This function prevents the camera from leaving its appropriate zone in the coordinate space.

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An HRESULT value, indicating whether the rendering was successful or not.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function does nothing right now, but if there was a model to be rendered where the camera exists (a model of the user's character, for example) then that code would go in this function.

Reset	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Clears and resets all vectors and values in the object.

GetSize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer representing the size of the camera object in memory.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetUp	
<i>Input Parameters</i>	Three floating point numbers, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the values of the ‘up vector’ through the input parameters. This is one of three vectors that orient the camera in the 3D coordinate space.

SetUp	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the values of the ‘up vector’ to the values of the input parameters.

GetRight	
<i>Input Parameters</i>	Three floating point numbers, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the values of the ‘right vector’ through the input parameters. This is one of three vectors that orient the camera in the 3D coordinate space.

SetRight	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the values of the ‘right vector’ to the values of the input parameters.

GetLookPoint	
<i>Input Parameters</i>	Three floating point numbers, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the values of the ‘look vector’ through the input parameters. This is one of three vectors that orient the camera in the 3D coordinate space.

SetLookPoint	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the values of the ‘look vector’ to the values of the input parameters.

GetPosition	
<i>Input Parameters</i>	Three floating point numbers, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the camera’s position through the input parameters.

SetPosition	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the camera's position to the values of the input parameters.

GetVelocity	
<i>Input Parameters</i>	Three floating point numbers, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the camera's velocity through the input parameters.

SetVelocity	
<i>Input Parameters</i>	Three floating point numbers.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the camera's velocity to the values of the input parameters.

GetRoll	
<i>Input Parameters</i>	A floating point number, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the camera's Roll value.

SetRoll	
<i>Input Parameters</i>	A floating point number.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the camera's Roll value.

GetYaw	
<i>Input Parameters</i>	A floating point number, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the camera's Yaw value.

SetYaw	
<i>Input Parameters</i>	A floating point number.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the camera's Yaw value.

GetPitch	
<i>Input Parameters</i>	A floating point number, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the camera's Pitch value.

SetPitch	
<i>Input Parameters</i>	A floating point number.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the camera's Pitch value.

2.3.11. Terrain Rendering

TerrainSingleton Class Functions

Instance	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to the TerrainSingleton object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Creates a TerrainSingleton object if it hasn't been created yet and returns a pointer, otherwise this function just returns the pointer.

CreateVertexBuffer	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A Boolean value indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Fills the TerrainSingleton's list of vertices with information from the WorldSingleton (heights) along with additional information specified here (x,z position, normal vectors, colors). This function then takes the above vertices and stores them in the appropriate vertex buffers, ending up with 499 horizontal triangle strips of vertices.

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A Boolean value indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The 499 vertex buffers containing horizontal triangle strips are rendered to the screen.

GetHeight	
<i>Input Parameters</i>	A pair of floating point numbers representing a location on the 2D terrain field.
<i>Function Output</i>	A floating point number containing the terrain height at the location specified in the input parameters.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	The height of the terrain at a specific point is returned. There are 30 cases that must be considered, since each terrain cell (area between four vertices) is composed of two triangles. These thirty cases are: on a vertex, on a horizontal line between vertices, on a vertical line between vertices, on a diagonal line between vertices (between the two triangles in a cell), and 13 separate tilts for each of the two triangles in the cell. These 13 tilts for the three vertices are: one possibility where all three vertices are equal, three possibilities where two vertices are equal and the third is smaller, three possibilities where two vertices are equal and the third is larger, and six possibilities where all three vertices are of different sizes.

2.3.12. Graphics / Rendering Pipeline

Global Module Functions

LoadBitmapToSurface	
<i>Input Parameters</i>	A character string containing the filepath of the bitmap to be loaded, a pointer to a LPDIRECT3DSURFACE9, and a pointer to the LPDIRECT3DDEVICE9.
<i>Function Output</i>	An integer indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Load a bitmap from file to a 2D drawing surface.

InitTiming	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A HRESULT indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Initializes a global timing mechanism, storing the number of ticks per minute.

Pause	
<i>Input Parameters</i>	An integer whose value is the number of milliseconds that the system should pause for.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Initializes a global timing mechanism.

GetNumTicksPerMs	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A float, containing the number of ticks per millisecond.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

FrameCount	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Stores the number of frames per second in a global value.

SetAmbientLight	
<i>Input Parameters</i>	A D3DCOLOR object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This sets the ambient light color for the Direct3D Device.

CZenVertex Class Functions

Set	
<i>Input Parameters</i>	Six floats for 3D coordinates and a normal vector, two D3DCOLOR objects containing diffuse and specular colors, and two more floats containing texture coordinates.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the values of the CZenVertex object.

CZenObject Class Functions

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Does nothing, as this is a base class for other classes.

GetNext	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Returns a pointer to the next object. CZenObjects and its derivative classes contain pointer in order to create one-way linked lists of objects.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetNext	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a pointer to the next object. CZenObjects and its derivative classes contain pointer in order to create one-way linked lists of objects.

GetParentFrame	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Returns a pointer to a CZenFrame object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetParentFrame	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a pointer to a CZenFrame object.

GetSize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer containing the size of the data structure.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	This function is a virtual function.

CZenFace Class Functions

SetProps	
<i>Input Parameters</i>	An integer to specify which vertex is being set (0-2), six floats for 3D coordinates and a normal vector, two D3DCOLOR objects containing diffuse and specular colors, and two more floats containing texture coordinates.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the properties of the CZenFace object.

SetTexture	
<i>Input Parameters</i>	A LPDIRECT3DTEXTURE9 object.
<i>Function Output</i>	An HRESULT value indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a texture to be loaded to the CZenFace object.

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Renders the CZenFace object to the environment.

GetSize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer containing the size of the data structure.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CZenMaterial Class Functions

SetDiffuse	
<i>Input Parameters</i>	Three floats representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetSpecular	
<i>Input Parameters</i>	Three floats representing the color of the light and a fourth float containing the power of the specular light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetAmbient	
<i>Input Parameters</i>	Three floats representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetEmissive	
<i>Input Parameters</i>	Three floats representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Update	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Resends the object's material to the Direct3D Device.

CZenMesh Class Functions

LoadXFile	
<i>Input Parameters</i>	A character pointer to the filepath of the xfile to be loaded.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Loads a .x file into the CZenMesh object.

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Renders the 3D mesh to the environment.

SetMaterial	
<i>Input Parameters</i>	A pointer to a CZenMaterial object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the mesh's materials to the input material.

GetSize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer representing the size of the CZenMesh structure.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

GetMesh	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	Returns a pointer to the CZenMesh's LPD3DXMESH object
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

CZenFrame Class Functions

SetCallback	
<i>Input Parameters</i>	A pointer to a function.
<i>Function Output</i>	A HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the pointer to a function that deals with frame movement.

GetVelocity	
<i>Input Parameters</i>	Three floats representing a velocity vector, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the frame's current velocity by reference.

SetVelocity	
<i>Input Parameters</i>	Three floats representing a velocity vector.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets the frame's velocity to the new values specified.

GetPosition	
<i>Input Parameters</i>	Three floats representing a position, passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Return's the frame's position by reference.

SetPosition	
<i>Input Parameters</i>	Three floats representing a position.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Updates the frames current position to the new values specified.

GetLocal	
<i>Input Parameters</i>	A D3DXMATRIX passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Update
<i>Notes</i>	Returns the frame's transformation matrix by reference.

GetYaw	
<i>Input Parameters</i>	A float passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the frame's Yaw by reference.

SetYaw	
<i>Input Parameters</i>	A single float value.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a new Yaw value.

GetRoll	
<i>Input Parameters</i>	A float passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the frame's Roll by reference.

SetRoll	
<i>Input Parameters</i>	A single float value.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a new Roll value.

GetPitch	
<i>Input Parameters</i>	A float passed by reference.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns the frame's Pitch by reference.

SetPitch	
<i>Input Parameters</i>	A single float value.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a new Pitch value.

Update	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Updates the frame's transformation matrix based on all of the current position and movement values.

AddObject	
<i>Input Parameters</i>	A pointer to a CZenObject object.
<i>Function Output</i>	A HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Adds a child object to the current frame

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A HRESULT object indicating success.
<i>Functions Referenced</i>	Update
<i>Notes</i>	Renders all of the child objects of the frame to the environment.

SetNext	
<i>Input Parameters</i>	A pointer to a CZenFrame object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a pointer to another frame, allowing frames to form a one-way linked list.

GetNext	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to a CZenFrame object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Gets a pointer to another frame, allowing frames to form a one-way linked list.

AddFrame	
<i>Input Parameters</i>	A pointer to a CZenFrame object.
<i>Function Output</i>	A HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Adds a child frame to the current frame.

GetParent	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A pointer to a CZenFrame object.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Returns a pointer to a frame's parent frame.

SetParent	
<i>Input Parameters</i>	A pointer to a CZenFrame object.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Sets a parent frame for the current frame.

CZenLight Class Functions

SetDiffuse	
<i>Input Parameters</i>	Three float values representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetSpecular	
<i>Input Parameters</i>	Three float values representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

SetAmbient	
<i>Input Parameters</i>	Three float values representing the color of the light.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Enable	
<i>Input Parameters</i>	A Boolean value for whether the light should be on or off.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Turns the light on or off.

IsOn	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A Boolean value indicating whether the light is turned on or off.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

Render	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	A HRESULT object indicating success.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	Renders the light to the environment through the Direct3D Device.

GetSize	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	An integer representing the size of the CZenLight structure.
<i>Functions Referenced</i>	N/A
<i>Notes</i>	N/A

2.3.13. Collision Detection

Physics Module Functions

CameraJump	
<i>Input Parameters</i>	N/A
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #10: SetVelocity
<i>Notes</i>	Sets the camera's velocity, as if the camera were a person who just applied a force to initialize a jump.

CameraGravity	
<i>Input Parameters</i>	A boolean that tells the function whether the camera is on the ground or not.
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	Module #10: GetVelocity, SetVelocity, GetPosition, SetPosition
<i>Notes</i>	Adjusts the camera's velocity and position accordingly.

FindHighestTerrainVertex	
<i>Input Parameters</i>	Four floats, representing the center x,z coordinate for the object to be tested, as well we the width and depth of the object.
<i>Function Output</i>	A float representing the tallest vertex below the object.
<i>Functions Referenced</i>	Module #11: GetHeight
<i>Notes</i>	The height of the tallest vertex beneath an object is returned to prevent an object from falling through the terrain.

EntityGravity	
<i>Input Parameters</i>	A pointer to a LocalEntity
<i>Function Output</i>	N/A
<i>Functions Referenced</i>	FindHighestTerrainVertex, Module #11: GetHeight
<i>Notes</i>	Adjusts an entity's velocity and position accordingly.

3. Acceptance Testing

3.1. Milestone Test List

<u>Test Name</u>	<u>Modules Used</u>
Test 1 – Material Editor General Functionality	1
Test 2 – EDGE Tool General Functionality	2
Test 3 – WIM Tool General Functionality	3
Test 4 – WIM Tool Bitmap Manipulation	3
Test 5 – Console Functionality and Display of Entity Properties	5, 12
Test 6 – Data Loading Driver	6
Test 7 – State Switching	4, 7, 8, 9
Test 8 – Terrain, Camera, and Character/Ground Collision Detection	7, 10, 11, 13
Test 9 – Entity Display and Entity/Ground Collision Detection	12, 13

3.2. Acceptance Test Details

Before examining any of the following acceptance tests it is important to note the methodology used in designing these tests. Due to the extremely complex nature of the simulator and its related developer tools, any sort of comprehensive enumerative testing would be out of the question, as thousands of test cases would need to be created just to fulfill the loosest criteria. Therefore, each test is composed of a walkthrough of the most critical functions and most common errors that could be experienced while using the program in question. This will give us the maximum assurance of software correctness that we could get without using comprehensive enumerative testing.

Test 1 – Material Editor General Functionality

	User Action	Expected Results	Passes?
1)	Run Program	The Material Editor should appear on the screen with only the <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> commands available.	Y
2)	Press the <i>Load File</i> Button	A window should appear allowing the user to browse for .mlb files.	Y
3)	Find and select standard.mlb and press the OK button.	The file browser should close and the following commands should be enabled: <i>Delete File</i> , <i>New Material</i> , and <i>List Invalid</i> . The first of the two listboxes should now be filled with the materials from standard.mlb in the following format: "name ID". There are three materials in standard.mlb: 'silver 1', 'iron 2', and 'stainless_steel 3'. Finally, the .mlb's filename should display in the label marked "Using:"	Y
4)	Select 'silver 1' from the listbox.	The <i>Save Material</i> and <i>Delete Material</i> commands should be available. The text fields below the listbox should now be filled as follows: Name, silver; Mass, 5.6; ID, 1; and Friction, 0.07. All of these fields except for ID should also be editable.	Y
5)	Select 'stainless_steel 3' from the listbox.	The <i>New Component</i> command should now be available. The fields below the listbox should now read: Name, stainless_steel; Mass, 6.06; ID, 3; and Friction, 0.11. Name should be the only editable text field. Finally, the second listbox should have two items listed in it: 'silver 1' and 'iron 2'.	Y

6)	Change the text for the Name field to 'sterling_silver' and click the <i>Save Material</i> button.	The selected text in the primary listbox should now read 'sterling_silver 3'. A confirmation message should appear in the status bar at the bottom of the editor: "Status: Material updated successfully."	Y
7)	Select 'silver 1' from the second listbox.	The <i>Delete Component</i> and <i>Save Component</i> commands are now available. The fields below the secondary listbox should contain the following values: Name, silver 1; Percent, 92.5. The percent field should be user editable. Note the value of the percent for the next few steps.	Y
8)	Select 'iron 2' from the second listbox.	The fields below the secondary listbox should contain the following values: Name, iron 2; Percent, 7.5. Note that $92.5 + 7.5 = 100$.	Y
9)	Change the Percent field to '15' and click the <i>Save Material</i> button.	The status bar should read "Status: Component update successful."	Y
10)	Click the <i>List Invalid</i> button. Click OK to exit the message box once the steps to the right are confirmed.	A message box should appear titled "Invalid Combination Material List" with an OK button and one line of text: "sterling_silver 3".	Y
11)	Change the Percent field to '7.5' and click the <i>Save Material</i> button.	The status bar should read "Status: Component update successful."	Y
12)	Click the <i>List Invalid</i> button. Click OK to exit the message box once the steps to the right are confirmed.	A message box should appear titled "Invalid Combination Material List" with an OK button and one line of text: "All combination materials are valid!"	Y
13)	Click the <i>New Material</i> button.	A message box should appear titled "Create New Material" with three buttons: Yes, No, and Cancel. The text should read: "Standard material (Yes) or Combination material (No)?"	Y
14)	Click the Yes button on the message box.	The <i>New Material</i> and <i>Delete Material</i> buttons should be unavailable. All four text fields below the primary listbox should be editable, and the ID field should display an integer that is equivalent to the highest ID in the listbox plus one.	Y
15)	Enter the following information in to the text fields: Name, gold; Mass, 4.2; Friction, .37. Press the <i>Save Material</i> button.	The <i>New Material</i> and <i>Delete Material</i> buttons should be available again. The ID field should be non-editable. Also, 'gold 4' should now be listed and selected in the primary listbox. The status bar along the bottom should read "Status: New material added successfully."	Y
16)	Go to the <i>Materials</i> menu and select <i>Delete Material</i> .	A message box should appear titled "Material Deletion" with two buttons, Yes and No. The text should read "Delete currently selected material?"	Y
17)	Select the Yes option in the message box.	The <i>Save Material</i> , <i>Delete Material</i> , and all three component commands should now be disabled. No text field should be editable, and all text fields should be blank. 'gold 4' is also no longer listed in the primary listbox.	Y

18)	Select 'iron 2' in the primary listbox.	The <i>Save Material</i> and <i>Delete Material</i> commands should be available. The text fields below the listbox should now be filled as follows: Name, iron; Mass, 11.4; ID, 2; and Friction, 0.52. All of these fields except for ID should also be editable.	Y
19)	Press the <i>Delete Material</i> button.	The status bar should now read "Status: Cannot delete a material with dependencies: sterling_silver Cannot perform deletion."	Y
20)	Go to the <i>File</i> menu and select <i>Exit</i> .	The program should exit without any errors.	Y
21)	Run Program	The Material Editor should appear on the screen with only the <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> commands available.	Y
22)	Press the <i>New File</i> Button	A window should appear allowing the user to enter a filename for a new .mlb file.	Y
23)	Enter in "temp" and press the OK button.	The file browser should close and the following commands should be enabled: <i>Delete File</i> , <i>New Material</i> , and <i>List Invalid</i> . temp.mlb's filename should display in the label marked "Using:"	Y
24)	Click the <i>New Material</i> button.	A message box should appear titled "Create New Material" with three buttons: Yes, No, and Cancel. The text should read: "Standard material (Yes) or Combination material (No)?"	Y
25)	Click the Yes button on the message box.	The <i>New Material</i> and <i>Delete Material</i> buttons should be unavailable. All four text fields below the primary listbox should be editable, and the ID field should display "1".	Y
26)	Enter the following information in to the text fields: Name, gold; Mass, 4.2; Friction, .37. Press the <i>Save Material</i> button.	The <i>New Material</i> and <i>Delete Material</i> buttons should be available again. The ID field should be non-editable. Also, 'gold 1' should now be listed and selected in the primary listbox. The status bar along the bottom should read "Status: New material added successfully."	Y
27)	Click the <i>New Material</i> button.	A message box should appear titled "Create New Material" with three buttons: Yes, No, and Cancel. The text should read: "Standard material (Yes) or Combination material (No)?"	Y
28)	Click the No button on the message box.	The <i>New Material</i> and <i>Delete Material</i> buttons should be unavailable. The Name and ID text fields below the primary listbox should be editable, and the ID field should display "2".	Y
29)	Enter the following information in to the text fields: Name, metal05. Press the <i>Save Material</i> button.	The <i>New Material</i> and <i>Delete Material</i> buttons should be available again. The ID field should be non-editable. Also, 'metal05 2' should now be listed and selected in the primary listbox. The status bar along the bottom should read "Status: New material added successfully."	Y
30)	Click the <i>New Component</i> button.	The <i>New Component</i> command is unavailable and the <i>Save Component</i> command is available. Both the Name and Percent text fields are now editable.	Y
31)	Enter the following: Name, gold 1; Percent, 50. Click the <i>Save Component</i> button.	<i>New Component</i> and <i>Delete Component</i> are available now. The Name field is non-editable. Also, 'gold 1' is now listed in the secondary listbox. The status bar should read "Status: Component saved successfully."	Y

32)	Click the <i>Delete Component</i> Button.	A message box should appear titled “Material Deletion” with the text “Delete currently selected component?” The button choices are Yes and No.	Y
33)	Click Yes in the message box.	The secondary listbox is now empty as ‘gold 1’ has been removed. The Save Component and Delete Component commands are no longer available. Both the Name and Percent text fields under the second listbox are blank and non-editable.	Y
34)	Click the <i>Delete Material</i> button.	A message box should appear titled “Material Deletion” with two buttons, Yes and No. The text should read “Delete currently selected material?”	Y
35)	Select the Yes option in the message box.	The <i>Save Material</i> , <i>Delete Material</i> , and all three component commands should now be disabled. No text field should be editable, and all text fields should be blank. ‘metal05 2’ is also no longer listed in the primary listbox.	Y
36)	Go to the <i>File</i> menu and select <i>Delete File</i> .	A message box should appear titled “Delete Confirmation” with the text “Delete temp.mlb?” and the options of Yes or No.	Y
37)	Select Yes in the message box.	The status bar should read “Status: temp.mlb has been successfully deleted.” The “Using:” label is now empty and the only commands available are <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> . All text fields are non-editable and blank.	Y
38)	Press the <i>Exit Editor</i> button.	The program should exit without any errors.	Y

Test 2 – EDGE Tool General Functionality

	User Action	Expected Results	Passes?
1)	Run Program	The EDGE Tool should appear on the screen with only the <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> commands available.	Y
2)	Press the <i>Load File</i> Button	A window should appear allowing the user to browse for .elb files.	Y
3)	Find and select standard.elb and press the OK button.	The file browser should close and the following commands should be enabled: <i>Delete File</i> and <i>New Entity</i> . The first of the two listboxes should now be filled with the entities from standard.elb in the following format: “name ID”. There are two entities in standard.elb: ‘steel_box 1’ and ‘silver_ball 2’. Finally, the .elb’s filename should display in the label marked “Entity Library:” The status bar should read “Status: XML file <filename> loaded successfully.”	Y
4)	Select ‘steel_box 1’ from the listbox.	The <i>Save Entity</i> , <i>Delete Entity</i> and <i>Load XMesh</i> commands should be available. The text fields below the listbox should now be filled as follows: Name, steel_box; Material, 2; ID, 1; X-File, box.x; and M-lib, standard.mlb. The Immobile checkbox should be unchecked. The fields below the “Feature Removed” box should read as follows: Height, 256.00; Width, 384.00; Depth, 384.00; and	Y

		the checkbox should be checked. The Name and ID fields should be editable.	
5)	Change the text for the Name field to 'iron_box' and click the <i>Save Entity</i> button.	The selected text in the primary listbox should now read 'iron_box 1'. A confirmation message should appear in the status bar at the bottom of the editor: "Status: Entity successfully updated."	Y
6)	Change the Height field to 128.00.	Both the Width and Depth fields should now read 192.00.	Y
7)	Change the Depth field to 384.00.	The Width field should now read 384.00 and the Height field should now read 256.00.	Y
8)	Uncheck the 'Keep Aspect Ratio' box and change Height to 128.00.	Neither of the other field values will change, since the auto-adjust feature has been disabled.	Y
9)	Press the <i>Exit Editor</i> button.	The program should exit without any errors.	Y
10)	Run Program	The EDGE Tool should appear on the screen with only the <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> commands available.	Y
11)	Press the <i>New File</i> Button	A window should appear allowing the user to enter a filename for a new .elb file.	Y
12)	Enter in "temp" and press the OK button.	The file browser should close and the following commands should be enabled: <i>Delete File</i> and <i>New Entity</i> . temp.elb's filename should display in the label marked "Using:" The status bar should read "Status: New XML file <filename> created successfully."	Y
13)	Click the <i>New Entity</i> button.	The Name field is now editable. The <i>Save Entity</i> command is available, and the <i>New Entity</i> command is disabled.	Y
14)	Use the Load Material Library button to load standard.mlb.	The status bar should read "Status: Entity library standard.mlb loaded successfully." The material listbox should be filled with three entries: iron 2, silver 1, and sterling silver 3.	Y
15)	Select 'silver 1'.	The <i>Set Material</i> command is now available.	Y
16)	Click the <i>Set Material</i> button.	The Material field should now contain a '1' and the M-lib field should now read 'standard.mlb'.	Y
17)	Click the Load XMesh button and select box.x.	The X-File field should read 'box.x'. The Height, Width, and Depth fields should still be non-editable, but should read 256.00, 384.00, and 384.00 respectively. The checkbox for aspect ratio should be checked. The status bar should read "Status: X-Mesh loaded successfully."	Y
18)	Type 'new' in the Name field and use the <i>Save Entity</i> command.	'new 1' is now listed in the entity listbox, and the <i>New Entity</i> and <i>Delete Entity</i> commands are now available. The dimension fields on the right are now editable. The status bar should read "Status: New entity added successfully."	Y
19)	Use <i>Delete Entity</i> and select 'Yes'.	'new 1' should disappear from the listbox and all of the fields should clear and be non-editable.	Y
20)	Click the Delete File button and select 'Yes'.	The only commands available should be <i>Load File</i> , <i>New File</i> , and <i>Exit Editor</i> . The 'Entity Library:' label should now be empty.	Y
21)	Click the <i>Exit Program</i> button.	The program should exit without any errors.	Y

Test 3 – WIM Tool General Functionality

	User Action	Expected Results	Passes?
1)	Run Program	The WIM Tool should appear on the screen with only the <i>Load World</i> , <i>New World</i> , and <i>Exit Editor</i> commands available.	Y
2)	Press the <i>Load World</i> button	A window should appear allowing the user to browse for .wid files.	Y
3)	Find and select a.wid and press the OK button.	The file browser should close and the following commands should be enabled: <i>Save World</i> , <i>Delete World</i> , <i>Load Entity Library</i> , <i>New Local Entity</i> , <i>New Bitmap</i> , <i>Load Bitmap</i> , and <i>Save Bitmap</i> . All six User Data fields should be filled in, as should the World Name field. The Local Entities listbox should contain three entities: crate (181, 475), large_box (245, 276), and large_box (255, 271). A grayscale bitmap should be displayed in the 'Bitmap Viewer and Commands' section of the window. The status bar should read "Status: world file loaded successfully."	Y
4)	Press the <i>New World</i> button.	A window should appear allowing the user to enter a filename for a new .elb file.	Y
5)	Enter in "temp" and press the OK button.	The file browser should close and the same commands as before should be available. The status bar should read "Status: New file created and loaded successfully."	Y
6)	Enter integer values into the six User fields and enter the name "Hills" into the World Name field. Once this is complete, go to the World menu and click on the <i>Save World</i> option.	The status bar should read: "Status: One or more fields have been left blank. World not saved." This is due to the fact that a bitmap has not been selected yet.	Y
7)	Press the <i>Load Bitmap</i> button.	A dialog box should appear listing bitmap files to select.	Y
8)	Select a bitmap from the list presented and press the OK button.	The bitmap should appear in the previously gray box in the Bitmap Viewer and Commands section. The name of the bitmap file should now appear in the field 'BMP Filename'.	Y
9)	Press the <i>Save World</i> button.	The status bar should read: "Status: World file saved."	Y
10)	Press the <i>Load Entity Library</i> button.	A dialog box should appear listing several .elb files.	Y
11)	Select one of the .elb files and press the OK button.	The Entity Library listbox should now list all of the 'Name ID' pairs for all entities stored in that library.	Y
12)	Press the <i>New Local Entity</i> button.	The six position fields in the Entity Data subsection are now editable.	Y
13)	Select one of the entities listed in the Entity Library listbox.	The <i>Use Entity</i> command is now available.	Y

14)	Press the <i>Use Entity</i> button.	The three non-editable fields in the Entity Data subsection are now filled with data from the selected entity.	Y
15)	Fill in the remaining entity fields with integers and press the <i>Save Local Entity</i> button.	The Local Entities listbox should now contain an entry in the format “Name (X, Y)” Where Name, X, and Y are fields in the Entity Data subsection. This entity is now saved to the .wid file.	Y
16)	Press the <i>Exit Editor</i> button.	The program should exit without any errors.	Y

Test 4 – WIM Tool Bitmap Manipulation

	User Action	Expected Results	Passes?
1)	Run Program	The WIM Tool should appear on the screen with only the <i>Load World</i> , <i>New World</i> , and <i>Exit Editor</i> commands available.	Y
2)	Press the <i>Load World</i> button	A window should appear allowing the user to browse for .wid files.	Y
3)	Find and select a.wid and press the OK button.	The file browser should close and the following commands should be enabled: <i>Save World</i> , <i>Delete World</i> , <i>Load Entity Library</i> , <i>New Local Entity</i> , <i>New Bitmap</i> , <i>Load Bitmap</i> , and <i>Save Bitmap</i> . All six User Data fields should be filled in, as should the World Name field. The Local Entities listbox should contain three entities: <i>crate</i> (181, 475), <i>large_box</i> (245, 276), and <i>large_box</i> (255, 271). A grayscale bitmap should be displayed in the ‘Bitmap Viewer and Commands’ section of the window. The status bar should read “Status: world file loaded successfully.”	Y
4)	Click the <i>Open Bitmap Editor</i> button.	The bitmap editor form should appear on top of the main form. All options (except for the disabled Perlín Noise option) should be available.	Y
5)	Move the mouse pointer across the bitmap.	The X, Y, and Z values below the bitmap should change to match the coordinates of the mouse pointer on the bitmap. The Z value represents the height of the particular pixel the cursor is over, where white = 255 and black = 0.	Y
6)	Select ‘Tool 1’ and move it across the bitmap.	The mouse cursor should have changed to a small red circle when over the bitmap.	Y
7)	Select ‘Tool 3’ and use the left mouse button to drag it across the bitmap.	A much larger red circle should have replaced the small red mouse cursor. As the cursor is dragged across the bitmap, the color of the pixels below the cursor should move closer to black.	Y
8)	Drag the mouse across the bitmap using the right mouse button.	As the cursor is dragged across the bitmap, the color of the pixels below the cursor should lighten, moving closer to white.	Y
9)	Raise the sensitivity bar to 20 and perform both drag operations again.	The mouse buttons perform the same actions as before, but it should be obvious that the pixel values are changing at a faster rate.	Y

10)	Click on the <i>Generate Terrain (Subdivide & Displace)</i> button.	After a few seconds of waiting a new bitmap should have been generated. It should have a black center and edges, appear fractal-like with squares inside of squares, and have a couple of white peaks. This is the automatic generation of hilly terrain.	Y
11)	Click the <i>Close</i> button.	The bitmap editor form should disappear.	Y
12)	Click the <i>Exit Editor</i> button.	The program should exit without error.	Y

Test 5 – Console Functionality and Display of Entity Properties

	User Action	Expected Results	Passes?
1)	Run the Simulator.	The main screen of the ITS should appear. The menu options available should be 'Load a Simulation' and 'Exit the Simulator'.	Y
2)	Press F11.	A gray command console should appear.	Y
3)	Type 'help' and press enter.	The console should respond with a list of commands under the heading 'Clarity Console Help'.	Y
4)	Type 'garbage' and press enter.	The console should respond 'Unknown Command.'	Y
5)	Type 'SetCameraSpeed 10'.	The console should respond 'Command unavailable while the WorldSingleton is empty.'	Y
6)	Press F11.	The command console should disappear.	Y
7)	Hover over 'Load a Simulation'.	The text that is being hovered over should turn red, indicating it is a menu option.	Y
8)	Click on 'Load a Simulation'.	The 'Simulation World Loader' screen should appear. On the right is a list titled 'World Data Files' with numerous .wid files listed.	Y
9)	Hover over any .wid file.	The name of the .wid file should turn green.	Y
10)	Click on any .wid file.	After a short period of waiting terrain and possible entities should be visible.	Y
11)	Press F11.	The command console should appear.	Y
12)	Type 'LEC' and press enter.	The console should display a list of all entity-related commands. These are: EntityList, GetEntityPos, GetEntityTF, GetEntityPosTF, GetEntityAttr, GetEntityMeshInfo, and ToggleEntityRenderMode.	Y
13)	Type 'EntityList' and press enter.	A list of entities should be displayed in the format: "<ID> <Name>. XMesh: <file.x>". This is a list of all entities in the current environment.	Y
14)	Type 'GetEntityPos #' where '#' is a number of an existing entity and press enter.	The coordinates of the selected entity should be output as three numbers.	Y
15)	Type 'GetEntityAttr #' where '#' is a number of an existing entity and press enter.	The mass value and the friction value for the entity in question are displayed in the console.	Y

16)	Type 'LOC' and press enter.	A list of all console commands not involving entities or the camera is listed here. These are: ToggleFPS, GetWorldName, GetBitmapFilename, ToggleGravity, SetJumpVelocity, and SetGravity.	Y
17)	Type 'ToggleGravity' and press enter.	The values in the bottom-center of the screen displaying gravity-related information should now be hidden.	Y
18)	Press F11.	The command console should disappear.	Y
19)	Press the 'P' button.	The pause menu should appear again.	Y
20)	Click on 'Exit the Simulator'.	The credits screen should appear for several seconds before the program closes without error.	Y

Test 6 – Data Loading Driver

	User Action	Expected Results	Passes?
1)	Run Driver Program in a Command Prompt	After a short amount of processing time the driver should display output to the screen as seen in Appendix A.5.	Y

Test 7 – State Switching

	User Action	Expected Results	Passes?
1)	Run the Simulator.	The main screen of the ITS should appear. The menu options available should be 'Load a Simulation' and 'Exit the Simulator'.	Y
2)	Hover over 'Load a Simulation'.	The text that is being hovered over should turn red, indicating it is a menu option.	Y
3)	Click on 'Load a Simulation'.	The 'Simulation World Loader' screen should appear. On the right is a list titled 'World Data Files' with numerous .wid files listed.	Y
4)	Hover over any .wid file.	The name of the .wid file should turn green.	Y
5)	Click on any .wid file.	After a short period of waiting terrain and possible entities should be visible.	Y
6)	Press the 'P' button.	A menu will appear with 'Simulation Paused' at the top. Menu options should include 'Resume Simulation', 'Exit to World Loading Screen', and 'Exit the Simulator'.	Y
7)	Click on 'Resume Simulation'.	The simulation should reappear and the camera should be facing the same entities and section of terrain as when the simulation was paused.	Y
8)	Press the 'P' button.	The pause menu should appear again.	Y
9)	Click on 'Exit to World Loading Screen'.	The 'Simulation World Loader' screen should appear again, with the list of .wid files.	Y
10)	Click on any .wid file.	After a short period of time the selected environment should load, including terrain and entities.	Y
11)	Press the 'P' button.	The pause menu should appear again.	Y
12)	Click on 'Exit the Simulator'.	The credits screen should appear for several seconds before the program closes without error.	Y

Test 8 – Terrain, Camera, and Character/Ground Collision Detection

	User Action	Expected Results	Passes?
1)	Run the Simulator.	The main screen of the ITS should appear. The menu options available should be 'Load a Simulation' and 'Exit the Simulator'.	Y
2)	Hover over 'Load a Simulation'.	The text that is being hovered over should turn red, indicating it is a menu option.	Y
3)	Click on 'Load a Simulation'.	The 'Simulation World Loader' screen should appear. On the right is a list titled 'World Data Files' with numerous .wid files listed.	Y
4)	Hover over any .wid file.	The name of the .wid file should turn green.	Y
5)	Click on any .wid file.	After a short period of waiting terrain and possible entities should be visible.	Y
6)	Press the Space Bar.	The camera should arc straight upwards from its current position and then arc back down until the camera lands back on the ground. During the "jump" it should be obvious that the camera is moving up and down in an accelerated arc, as if the camera were a user that had jumped.	Y
7)	Press the 'W' key.	The camera should move forward a number of units equal to the camera speed (viewable in the console).	Y
8)	Press the 'S' key.	The camera should move backward a number of units equal to the camera speed (viewable in the console).	Y
9)	Press the 'D' key	The camera should move to the right a number of units equal to the camera speed (viewable in the console).	Y
10)	Press the 'A' key	The camera should move to the left a number of units equal to the camera speed (viewable in the console).	Y
11)	Press and hold the 'E' key for a few seconds.	The camera should stay in its initial position but should rotate clockwise (to the right) as long as this key is held down.	Y
12)	Press and hold the 'Q' key for a few seconds.	The camera should stay in its initial position but should rotate counter-clockwise (to the left) as long as this key is held down.	Y
13)	Use the movement keys to attempt to walk off of the edge of the terrain.	If the user attempts to walk the character/camera off of the terrain he or she will be unable to do so. The simulator should keep the camera within the bounds of the terrain.	Y
14)	Press the 'P' button.	A menu will appear with 'Simulation Paused' at the top. Menu options should include 'Resume Simulation', 'Exit to World Loading Screen', and 'Exit the Simulator'.	Y
15)	Click on 'Exit the Simulator'.	The credits screen should appear for several seconds before the program closes without error.	Y

Test 9 – Entity Display and Entity/Ground Collision Detection

	User Action	Expected Results	Passes?
1)	Run the Simulator.	The main screen of the ITS should appear. The menu options available should be 'Load a Simulation' and 'Exit the Simulator'.	Y
2)	Hover over 'Load a Simulation'.	The text that is being hovered over should turn red, indicating it is a menu option.	Y
3)	Click on 'Load a Simulation'.	The 'Simulation World Loader' screen should appear. On the right is a list titled 'World Data Files' with numerous .wid files listed.	Y
4)	Hover over objecttest.wid.	The name of the .wid file should turn green.	Y
5)	Click on objecttest.wid.	After a short period of waiting terrain and possible entities should be visible. Multiple entities, boxes in this case, should appear above the surface of the terrain and immediately begin to fall due to gravity.	Y
6)	Wait until the entities collide with the terrain and stop moving.	The boxes should stop as soon as an edge, vertex or surface collides with the terrain.	Y
7)	Walk around and observe the boxes.	Each box should be resting on some sort of terrain surface. Any box that is over a ledge should stop where their bottom faces collided with the terrain, either with the top of the ledge or on the slope of the ledge.	Y
8)	Press the 'P' button.	A menu will appear with 'Simulation Paused' at the top. Menu options should include 'Resume Simulation', 'Exit to World Loading Screen', and 'Exit the Simulator'.	Y
9)	Click on 'Exit the Simulator'.	The credits screen should appear for several seconds before the program closes without error.	Y

Appendix A Test Driver for Module #6 (Data Loading)

A.1 Driver Code

```

-----
// File:      WorldDriver.cpp
//
// Desc:      This is a driver file used to test the WorldSingleton class.
//
// First created on:  February 27th, 2005
// Last modification: February 28th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Notice:    This code would have been impossible without a great deal of
//            guidance and assistance from my dear friend Phoenix, particularly
//            in the areas of namespaces and string conversion.
//            "J Lothian - Remember, Lycoming rejected his application!"
// Notice:    LoadBitmap() function based off of code by Eric Carr.
//            Please see project credits for contact information.
-----
// Revision History:
//
// 02-27-05:  This driver file was created. LoadWIDFile() and all of the
//            string conversion functions coded in.
// 02-28-05:  Added in loading of referenced XML files, and .x meshes.
//            Can now load bitmap data into memory.
-----

#define WIN32_LEAN_AND_MEAN
// Include files.
#include <Windows.h>
#include <string>
#include <sstream>
#include <list>
#include <tchar.h>
#include <iostream>

// This imports the MSXML functionality used to read from the XML files.
#import "msxml4.dll"

// DX9 Include files.
#include <D3DX9.h>
#include "DXUtil.h"
#include "D3DEnumeration.h"
#include "D3DSettings.h"
#include "D3DApp.h"
#include "D3DFile.h"
#include "D3DFont.h"
#include "D3DUtil.h"
#include "dinput.h"

using namespace std;

// Two of my includes, needed to make the driver work.
#include "debug.h"
#include "zen.h"

/**/ The following structures are used to store data from the file. ***/

struct User
{
    int x, y, z;
    double roll, pitch, yaw;
};

struct LocalEntity
{
    string name;
    int x, y, z;
    double roll, pitch, yaw;
};

```

```

    int eid, mid;
    string elib, mlib;
    string xfile;
    bool immobile;
    double height, width, depth;
    double mass, friction;
    CZenMesh xmesh;
};

typedef class WorldSingleton
{
public:
    // Returns a pointer to the WorldSingleton.
    static WorldSingleton* Instance();

    // These functions load data to memory.
    bool LoadWIDFile(string filename);
    bool LoadEntityData(LocalEntity * LocalEntity);
    bool LoadMaterialData(LocalEntity * LocalEntity);
    BYTE* LoadBitmap(char * filename);

    // Data members.
    string sWorldName;
    string sBitmapFilename;
    User TheUser;
    list<LocalEntity *> lstLocalEntities;
    BYTE* HeightMap;
    long ByteRowWidth;    // A row offset for HeightMap.
protected:
    WorldSingleton();
    ~WorldSingleton();
private:
    static WorldSingleton* _instance;

    // String conversion functions.
    string BTS(_bstr_t bstrString);
    BSTR STB(const char * temp);
    int StringToInt(string temp);
    double StringToDouble(string temp);
} CWorldSingleton;

WorldSingleton* WorldSingleton::_instance = 0;

WorldSingleton* WorldSingleton::Instance()
{
    if(_instance == 0)
    {
        _instance = new WorldSingleton;
    }
    return _instance;
}

WorldSingleton::WorldSingleton()
{
    // Constructor!
}

WorldSingleton::~WorldSingleton()
{
    // Destructor!
}

/** String conversion functions. */

string WorldSingleton::BTS(_bstr_t bstrString)
{
    // Convert a BSTR to a string.
    return (LPCTSTR)bstrString;
}

BSTR WorldSingleton::STB(const char * temp)
{
    // Convert a string to a BSTR.

```

```

    _bstr_t bs1 = temp;
    return bs1.copy();
}

int WorldSingleton::StringToInt(string temp)
{
    int n;
    stringstream ssBuffer;
    ssBuffer << temp;
    ssBuffer >> n;
    return n;
}

double WorldSingleton::StringToDouble(string temp)
{
    double n;
    stringstream ssBuffer;
    ssBuffer << temp;
    ssBuffer >> n;
    return n;
}

bool WorldSingleton::LoadWIDFile(string filename)
{
    // Variable declarations.
    MSXML2::IXMLDOMNodePtr xNode, xLocalNode, xTemp;
    MSXML2::IXMLDOMNodeListPtr NodeList, EntityList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    LocalEntity * tempLocalEntity;
    _bstr_t bstrTemp;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    bool bLoadXML = xmlDoc->load(filename.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML WID file failed to load." );
        return false;
    }

    // Load 'world|name attribute.
    bstrTemp = xmlDoc->documentElement->attributes->getNamedItem(
        STB("name"))->nodeValue;
    sWorldName = BTS(bstrTemp);

    // Loop through world's data nodes.
    NodeList = xmlDoc->documentElement->childNodes;
    long lNodeCount;
    NodeList->get_length(&lNodeCount);
    for (int i = 0; i < lNodeCount; i++)
    {
        // Get next child node.
        NodeList->get_item(i, &xNode);
        sData = BTS(xNode->GetnodeName());

        if(sData == "locals")
        {
            EntityList = xNode->childNodes;
            long lEntityCount;
            EntityList->get_length(&lEntityCount);
            for (int j = 0; j < lEntityCount; j++)
            {
                // Get next child node.
                EntityList->get_item(j, &xLocalNode);

                // Point the temp pointer to a new struct.
                tempLocalEntity = new LocalEntity;
            }
        }
    }
}

```

```

// Load all of the values from file into the new structure.
tempLocalEntity->name = BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("name"))->nodeValue );
tempLocalEntity->x = StringToInt( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("x"))->nodeValue ) );
tempLocalEntity->y = StringToInt( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("y"))->nodeValue ) );
tempLocalEntity->z = StringToInt( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("z"))->nodeValue ) );
tempLocalEntity->roll = StringToDouble( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("roll"))->nodeValue ) );
tempLocalEntity->pitch = StringToDouble( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("pitch"))->nodeValue ) );
tempLocalEntity->yaw = StringToDouble( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("yaw"))->nodeValue ) );
tempLocalEntity->eid = StringToInt( BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("eID"))->nodeValue ) );
tempLocalEntity->elib = BTS( (_bstr_t)xLocalNode->attributes _
->getNamedItem(STB("elib"))->nodeValue );

// Load entity information into memory.
if( LoadEntityData(tempLocalEntity) == false )
{
    return false;
}

// Save the new structure to the list.
lstLocalEntities.push_back(tempLocalEntity);

// Clear the temp pointer.
tempLocalEntity = 0;
}
Debug( "Found locals node!" );
}
else if(sData == "bitmap")
{
    // Load bitmap data from file.
    bstrTemp = xNode->attributes->getNamedItem(STB("filename"))->nodeValue;
    sBitmapFilename = BTS(bstrTemp);
}
else if(sData == "user")
{
    // Load user data from file into the User structure.
    TheUser.x = StringToInt( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("x"))->nodeValue ) );
    TheUser.y = StringToInt( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("y"))->nodeValue ) );
    TheUser.z = StringToInt( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("z"))->nodeValue ) );
    TheUser.roll = StringToDouble( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("roll"))->nodeValue ) );
    TheUser.pitch = StringToDouble( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("pitch"))->nodeValue ) );
    TheUser.yaw = StringToDouble( BTS( (_bstr_t)xNode->attributes _
->getNamedItem(STB("yaw"))->nodeValue ) );
}
else
{
    Debug( "An invalid node has been found while loading the WID file." );
    return false;
}
}

Debug( "The world file has been loaded successfully." );

// Load the bitmap to memory.
HeightMap = LoadBitmap((char*)sBitmapFilename.c_str());

Debug( "The height map has been loaded successfully." );

```

```

    return true;
}

bool WorldSingleton::LoadEntityData(LocalEntity * LocalEntity)
{
    // Fill in mid, mlib, xfile, immobile, height, width, depth.
    // Variable declarations.
    MSXML2::IXMLDOMNodePtr xNode, xSubNode;
    MSXML2::IXMLDOMNodeListPtr EntityList, SubList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    int nID;
    bool bFound;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    bool bLoadXML = xmlDoc->load(LocalEntity->elib.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML ELB file failed to load." );
        return false;
    }

    // Loop through the <entity> objects.
    EntityList = xmlDoc->documentElement->childNodes;
    long lEntityCount;
    EntityList->get_length(&lEntityCount);
    for (int i = 0; i < lEntityCount; i++)
    {
        // Get next child node.
        EntityList->get_item(i, &xNode);
        nID = StringToInt( BTS( (_bstr_t)xNode->attributes _
            ->getNamedItem(STB("ID"))->nodeValue ) );
        if(nID == LocalEntity->eid)
        {
            // Load entity data from file into the structure.
            SubList = xNode->childNodes;
            long lSubCount;
            SubList->get_length(&lSubCount);
            for(int j = 0; j < lSubCount; j++)
            {
                // Get next child node.
                SubList->get_item(j, &xSubNode);
                sData = BTS(xSubNode->GetnodeName());

                if(sData == "mlib")
                {
                    LocalEntity->mlib = BTS( (_bstr_t)xSubNode->text );
                }
                else if(sData == "mID")
                {
                    LocalEntity->mid = StringToInt( _
                        BTS( (_bstr_t)xSubNode->text ) );
                }
                else if(sData == "xfile")
                {
                    LocalEntity->xfile = BTS( (_bstr_t)xSubNode->text );
                    LocalEntity->xmesh.LoadXFile((char *) (LocalEntity _
                        ->xfile).c_str()); // Load .x mesh
                }
                else if(sData == "immobile")
                {
                    int n = StringToInt(BTS((_bstr_t)xSubNode->text));
                    if(n == 1)
                    {
                        LocalEntity->immobile = true;
                    }
                    else
                    {

```



```

        LocalEntity->immobile = false;
    }
    }
    else if(sData == "size")
    {
LocalEntity->height = StringToDouble( BTS( (_bstr_t)xSubNode->attributes
->getNamedItem(STB("height"))->nodeValue ) );
LocalEntity->width = StringToDouble( BTS( (_bstr_t)xSubNode->attributes
->getNamedItem(STB("width"))->nodeValue ) );
LocalEntity->depth = StringToDouble( BTS( (_bstr_t)xSubNode->attributes
->getNamedItem(STB("depth"))->nodeValue ) );
    }
    else
    {
        Debug( "An invalid node has been found while
loading an ELB file." );
        return false;
    }
}

// Load material information into memory.
if( LoadMaterialData(LocalEntity) == false )
{
    return false;
}

// We found the entity ...
bFound = true;
break;
}
}

if(bFound == false) { return false; }

return true;
}

bool WorldSingleton::LoadMaterialData(LocalEntity * LocalEntity)
{
    // Fill in mass and friction.
    // Variable declarations.
    MSXML2::IXMLDOMNodePtr xNode;
    MSXML2::IXMLDOMNodeListPtr MaterialList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    int nID;
    bool bFound;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    bool bLoadXML = xmlDoc->load(LocalEntity->mLib.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML MLB file failed to load." );
        return false;
    }

    // Loop through the <entity> objects.
    MaterialList = xmlDoc->documentElement->childNodes;
    long lMatCount;
    MaterialList->get_length(&lMatCount);
    for (int i = 0; i < lMatCount; i++)
    {
        // Get next child node.
        MaterialList->get_item(i, &xNode);
        nID = StringToInt( BTS( (_bstr_t)xNode->attributes
->getNamedItem(STB("ID"))->nodeValue ) );

        if(nID == LocalEntity->mid)

```

```

        {
            LocalEntity->mass = StringToDouble( BTS( ( _bstr_t)xNode _
                ->attributes->getNamedItem(STB("mass"))->nodeValue ) );
            LocalEntity->friction = StringToDouble( BTS( ( _bstr_t)xNode _
                ->attributes->getNamedItem(STB("friction"))->nodeValue ) );
            bFound = true;
            break;
        }
    }

    if(bFound == false) { return false; }

    return true;
}

BYTE* WorldSingleton::LoadBitmap(char * filename)
// This function based off of code by Eric Carr.
{
    BITMAPINFOHEADER infoheader;
    BYTE *bitmapData;
    BYTE *bitmapDone;
    FILE *bitmapFile;
    BYTE red, blue, green;
    int padding;

    bitmapFile = fopen(filename, "rb");
    fseek(bitmapFile, sizeof(BITMAPFILEHEADER), SEEK_SET);
    fread(&infoheader, sizeof(BITMAPINFOHEADER), 1, bitmapFile);

    // Save infoheader.biWidth to structure.
    ByteRowWidth = infoheader.biWidth;

    // Get the padding at the end of the bitmap.
    padding = 4 - ((infoheader.biWidth * 3) % 4);
    if(padding == 4)
    {
        padding = 0;
    }

    bitmapData = new BYTE[infoheader.biWidth * infoheader.biHeight];
    bitmapDone = new BYTE[infoheader.biWidth * infoheader.biHeight];

    // Copy the bitmap data into bitmapData.
    for( int y = 0; y < infoheader.biHeight; ++y)
    {
        for( int x = 0; x < infoheader.biWidth; ++x)
        {
            fread(&blue, sizeof(BYTE), 1, bitmapFile);
            fread(&green, sizeof(BYTE), 1, bitmapFile);
            fread(&red, sizeof(BYTE), 1, bitmapFile);

            bitmapData[y*infoheader.biWidth + x] = red; // white = 255, black = 0
        }

        // Skip past the padding in the file.
        fseek(bitmapFile, padding, SEEK_CUR);
    }

    // Copy the bitmap data from bitmapData into bitmapDone, right-side up.
    int heightIndex = 0;
    for( y = infoheader.biHeight - 1; y >= 0; --y)
    {
        for( int x = 0; x < infoheader.biWidth; ++x)
        {
            bitmapDone[heightIndex*infoheader.biWidth + x] = _
                bitmapData[y*infoheader.biWidth + x];
        }
        ++heightIndex;
    }

    delete bitmapData;
}

```

```

fclose(bitmapFile);

return bitmapDone;
}

/** Output and Main Functions */

void PrintWorldStructureToScreen(WorldSingleton * World)
{
    cout << "World File: " << World->sWorldName << endl << endl;

    cout << "Bitmap Data:" << endl;
    cout << "  Filename: " << World->sBitmapFilename << endl << endl;

    cout << "User Data:" << endl;
    cout << "  X: " << World->TheUser.x << endl;
    cout << "  Y: " << World->TheUser.y << endl;
    cout << "  Z: " << World->TheUser.z << endl;
    cout << "  Roll: " << World->TheUser.roll << endl;
    cout << "  Pitch: " << World->TheUser.pitch << endl;
    cout << "  Yaw: " << World->TheUser.yaw << endl << endl;

    cout << "Local Entity Data:" << endl;
    int count = 1;
    for(list<LocalEntity *>::iterator i = World->lstLocalEntities.begin(); i _
        != World->lstLocalEntities.end(); i++)
    {
        cout << "  Entity " << count << ": " << endl;
        cout << "    (Primary)" << endl;
        cout << "    Name: " << (*i)->name << endl;
        cout << "    X: " << (*i)->x << endl;
        cout << "    Y: " << (*i)->y << endl;
        cout << "    Z: " << (*i)->z << endl;
        cout << "    Roll: " << (*i)->roll << endl;
        cout << "    Pitch: " << (*i)->pitch << endl;
        cout << "    Yaw: " << (*i)->yaw << endl;
        cout << "    eID: " << (*i)->eid << endl;
        cout << "    elib: " << (*i)->elib << endl;
        cout << "    (Entity)" << endl;
        cout << "    mlib: " << (*i)->mllib << endl;
        cout << "    mID: " << (*i)->mid << endl;
        cout << "    xfile: " << (*i)->xfile << endl;
        if((*i)->immobile)
        {
            cout << "    imm.: " << "true" << endl;
        }
        else
        {
            cout << "    imm.: " << "false" << endl;
        }
        cout << "    height:" << (*i)->height << endl;
        cout << "    width : " << (*i)->width << endl;
        cout << "    depth : " << (*i)->depth << endl;
        cout << "    (Material)" << endl;
        cout << "    mass  : " << (*i)->mass << endl;
        cout << "    fric. : " << (*i)->friction << endl;
        count++;
    }
    cout << endl;

    cout << "Bitmap (first row only):" << endl;
    for(int j = 0; j < 20; j++)
    {
        for(int k = 0; k < 25; k++)
        {
            cout << (int)World->HeightMap[j*25 + k] << " ";
        }
        cout << endl;
    }
    cout << endl;
    cout << "End of World File." << endl;
}

```

```
}  
  
int main(int argc, char* argv[])  
{  
    WorldSingleton * World = WorldSingleton::Instance();  
  
    // The following 'Co' functions are for purposes of handling COM.  
    CoInitialize(NULL);  
    { // Extra braces for scope only.  
        bool bTest = World->LoadWIDFile("a.wid");  
        if(!bTest)  
        {  
            cerr << "Loading this .wid file failed." << endl;  
            return 0;  
        }  
    }  
    CoUninitialize();  
  
    PrintWorldStructureToScreen(World);  
    return 0;  
}
```

Appendix A **Test Driver for Module #6 (Data Loading)**
A.2 **.wid Test File**

a.wid

```
<?xml version="1.0" encoding="UTF-8"?>
<world name="Asgard">
  <locals>
    <entity name="large_box" x="245" y="276" z="0" roll="0" pitch="0"
      yaw="0" eID="3" elib="sample_entity_xml.elb"/>
    <entity name="large_box" x="255" y="271" z="0" roll="0" pitch="0"
      yaw="0" eID="3" elib="sample_entity_xml.elb"/>
    <entity name="crate" x="180" y="475" z="25" roll="30.5"
      pitch="90" yaw="45" eID="1" elib="sample_entity_xml.elb"/>
  </locals>
  <bitmap filename="test.bmp"/>
  <user x="23" y="367" z="5" roll="45" pitch="45" yaw="90"/>
</world>
```

Appendix A Test Driver for Module #6 (Data Loading)

A.3 .elb Test File

sample_entity_xml.elb

```
<?xml version="1.0" encoding="UTF-8"?>
<entitylist maxID="3">
  <entity ID="1" name="crate">
    <mlib>sample_material_xml.mlb</mlib>
    <mID>3</mID>
    <xfile>box.x</xfile>
    <immobile>0</immobile>
    <size height="256.00" width="384.00" depth="384.00"
      kepratio="1"/>
  </entity>
  <entity ID="2" name="bush">
    <mlib>sample_material_xml.mlb</mlib>
    <mID>2</mID>
    <xfile>sphere.x</xfile>
    <immobile>1</immobile>
    <size height="330.00" width="330.00" depth="330.00"
      kepratio="1"/>
  </entity>
  <entity ID="3" name="large_box">
    <mlib>sample_material_xml.mlb</mlib>
    <mID>1</mID>
    <xfile>testbox.x</xfile>
    <immobile>1</immobile>
    <size height="128.00" width="252.00" depth="252.00"
      kepratio="1"/>
  </entity>
</entitylist>
```

Appendix A Test Driver for Module #6 (Data Loading)
A.4 .mlb Test File

sample material xml.mlb

```
<?xml version="1.0" encoding="UTF-8"?>
<materiallist maxID="3">
  <material ID="1" name="copper" mass="6.3" friction="2.1"/>
  <material ID="2" name="bronze" mass="2.2" friction="1.6"/>
  <combo ID="3" name="brass" mass="4.36" friction="2.75">
    <component ID="1" percent="30"/>
    <component ID="2" percent="70"/>
  </combo>
</materiallist>
```

Appendix A Test Driver for Module #6 (Data Loading)

A.5 Expected Driver Output

Run WorldDriver.exe. Expected Output:

World File: Asgard

Bitmap Data:

 Filename: test.bmp

User Data:

 X: 23

 Y: 367

 Z: 5

 Roll: 45

 Pitch: 45

 Yaw: 90

Local Entity Data:

 Entity 1:

 (Primary)

 Name: large_box

 X: 245

 Y: 276

 Z: 0

 Roll: 0

 Pitch: 0

 Yaw: 0

 eID: 3

 elib: sample_entity_xml.elb

 (Entity)

 mlib: sample_material_xml.mlb

 mID: 1

 xfile: testbox.x

 imm.: true

 height:128

 width :252

 depth :252

 (Material)

 mass :6.3

 fric. :2.1

 Entity 2:

 (Primary)

 Name: large_box

 X: 255

 Y: 271

 Z: 0

 Roll: 0

 Pitch: 0

 Yaw: 0

 eID: 3

 elib: sample_entity_xml.elb

 (Entity)

 mlib: sample_material_xml.mlb

 mID: 1

 xfile: testbox.x

 imm.: true

 height:128

 width :252

 depth :252

 (Material)

136 Section 3 - Detailed Design Document

```
mass :6.3
fric. :2.1
Entity 3:
(Primary)
Name: crate
X: 180
Y: 475
Z: 25
Roll: 30.5
Pitch: 90
Yaw: 45
eID: 1
elib: sample_entity_xml.elb
(Entity)
mlib: sample_material_xml.mlb
mID: 3
xfile: box.x
imm.: false
height:256
width :384
depth :384
(Material)
mass :4.36
fric. :2.75
```

Bitmap (first row only):

```
124 108 90 79 75 70 64 53 43 35 34 39 45 53 57 58 60 57 64 71 81 87 95 102 103
98 92 83 74 63 53 50 47 43 45 43 39 40 34 28 27 22 22 24 29 31 35 41 44 53
58 64 69 64 56 52 49 53 61 70 70 67 55 49 48 51 53 55 57 61 69 80 82 81 76
66 56 50 47 45 49 51 52 46 45 38 40 37 42 50 65 68 75 74 63 56 45 42 43 48
56 64 61 62 55 48 45 45 50 64 72 84 84 76 63 53 49 50 56 69 79 86 78 70 61
55 47 44 45 41 43 45 52 60 64 65 64 56 59 60 62 65 64 54 53 41 41 39 43 41
42 41 45 46 45 43 38 36 35 39 53 68 82 93 94 93 88 87 89 84 83 79 75 68 61
58 57 57 59 53 51 47 44 41 38 41 43 41 47 42 43 42 39 36 31 30 30 34 43 48
58 60 68 70 72 74 77 76 74 66 62 55 54 60 62 66 66 73 71 73 69 71 70 69 73
74 71 66 62 61 59 61 68 74 83 93 88 79 68 58 51 54 56 60 62 70 64 56 47 35
31 23 23 28 32 37 44 49 47 45 42 38 33 32 33 42 51 59 64 60 58 52 46 41 41
46 53 59 71 74 72 70 60 55 44 36 38 31 33 33 34 38 41 48 47 50 55 57 59 59
64 70 80 86 88 81 77 67 59 60 58 59 62 66 74 80 89 91 94 88 77 67 52 49 53
62 70 78 73 71 64 54 56 56 59 59 57 54 48 43 45 51 60 70 79 85 88 89 86 86
87 85 87 85 79 68 57 43 36 29 27 34 39 53 56 64 65 63 61 55 51 48 40 40 42
46 57 66 70 80 77 77 76 71 74 76 76 77 80 73 68 63 59 58 63 67 64 62 57 51
52 56 58 59 58 53 56 53 54 58 53 50 43 38 29 28 30 36 47 58 70 74 75 67 65
61 67 71 80 76 70 65 58 58 61 58 59 56 61 56 59 60 54 54 49 51 51 56 62 65
64 64 63 66 71 85 93 103 107 100 84 69 60 59 65 73 78 78 70 62 58 57 59 59 57
49 45 38 34 36 42 46 54 56 51 49 42 44 43 46 48 53 51 48 42 38 35 33 34 39
```

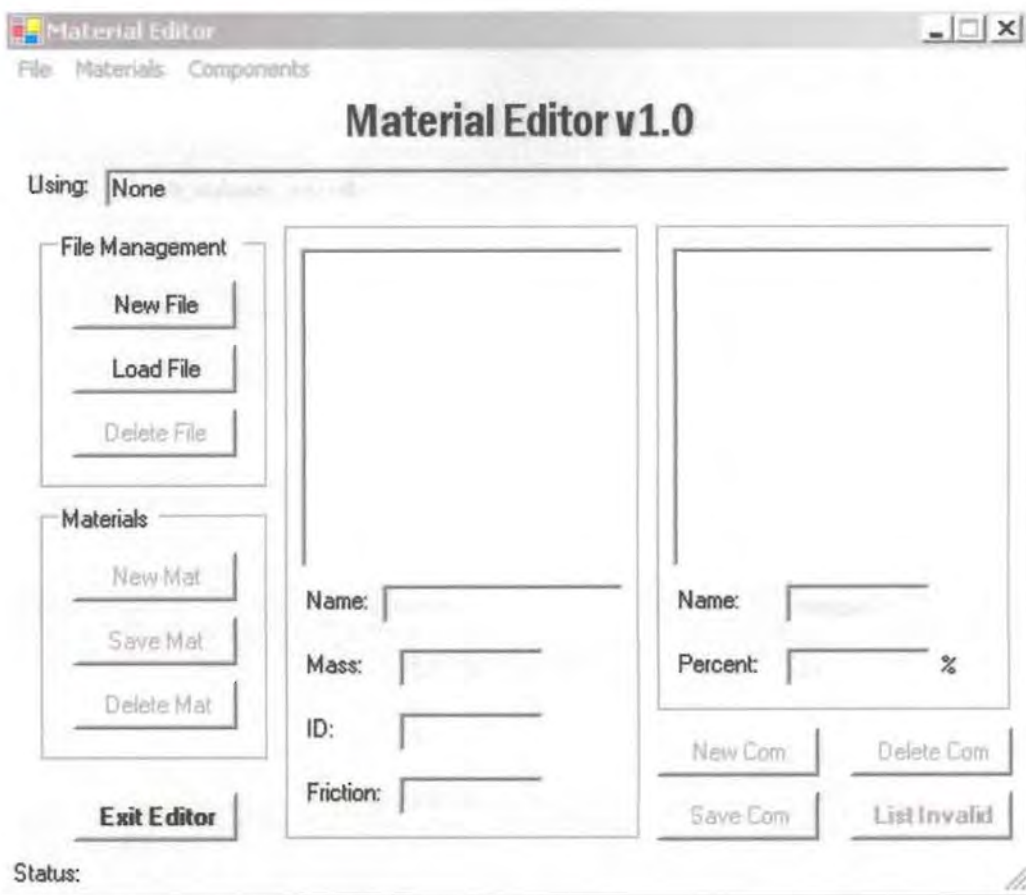
End of World File.

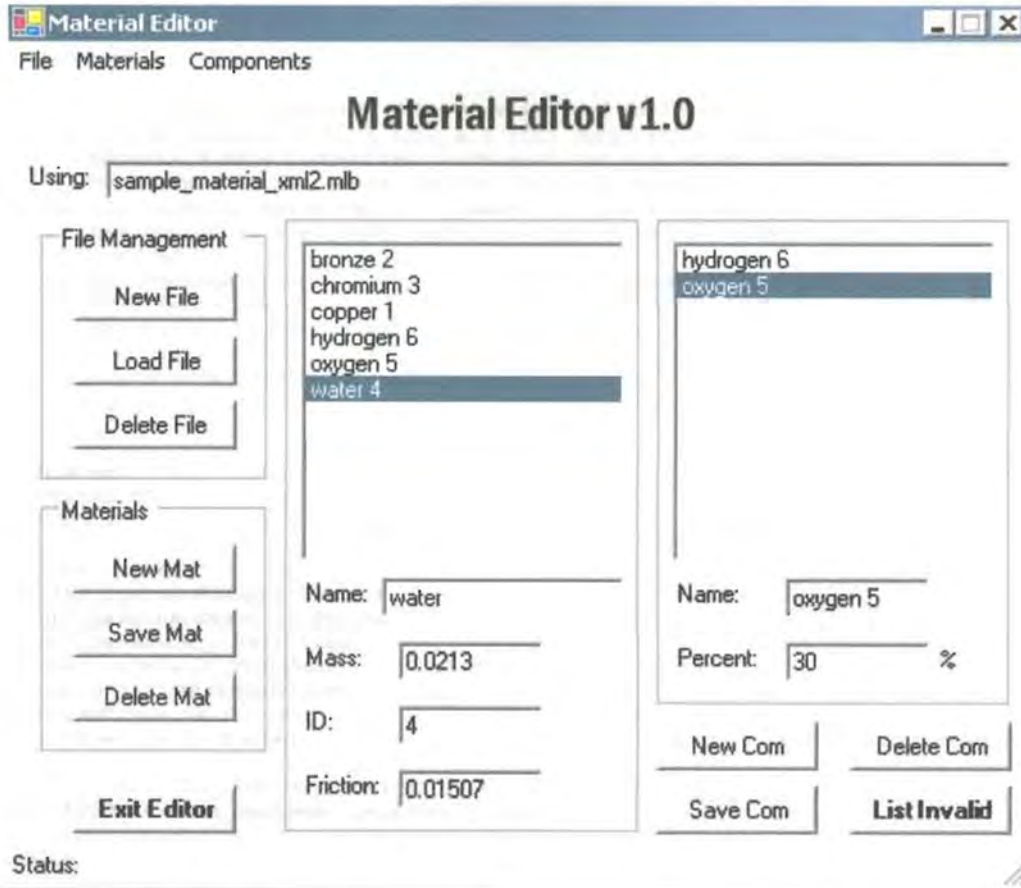
Section 4

Material Editor Code and Screenshots

Material Editor Screenshot

(1 of 2)





```

' File: frmMain.vb
'
' Desc: This is the primary and only file for the Material Editor.
'
' First created on:    December 5th, 2004
' Last modification:  March 2nd, 2005
'
' Copyright (c) Jason M. Black (donblas@donblas.org)
'-----
' Revision History:
'
' 12-05-04:  Interface designed. File creation and deletion implemented.
' 12-20-04:  XML loads from file into all form components. Combination
'            materials have properties calculated and displayed correctly.
'            Menus and buttons added. Interface design finalized.
' 12-21-04:  All material manipulation commands are finished and tested. Error
'            prevention is also done. Materials may be added, updated, and
'            removed.
' 12-22-04:  All component manipulation commands are finished and tested.
'            This marks the completion of the first build of this program!
' 01-21-05:  New files are now loaded into memory. Fixed a bunch of fields'
'            read-only attributes so everything is more logical.
' 01-22-05:  Added RecalcMaxID into the program to keep MaxID from growing.
' 03-02-05:  Adjusted read/write paths and recommended the code.
'-----

```

```

Imports System
Imports System.IO
Imports MSXML2

```

```

Public Class frmMain
    Inherits System.Windows.Forms.Form
    Dim stFilePathAndName As String
    Dim stFileNameOnly As String
    Dim xDoc As New XmlDocument 'The XML Library.
    Dim Nodes As IXMLDOMNodeList 'The nodes in the library.
    Dim NewMatType As Integer '0 is no new object, 1 is standard, 2 is combo.
    Dim NewComType As Boolean 'FALSE is no new component, TRUE is a new component.

```

```

'All code in this "Region" was created by VS.NET as the graphical form was assembled.
#Region " Windows Form Designer generated code "
' Auto-generated code removed for clarity.
#End Region

```

```

Private Sub LoadXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnLoadFile.Click, MenuItem4.Click

```

```

    Dim openFileDialog1 As New OpenFileDialog
    Dim stFileName As String

```

```

    openFileDialog1.InitialDirectory = "xml\"
    openFileDialog1.Title = "Open Material Library"
    openFileDialog1.Filter = "Material Library (*.mlb)|*.mlb"
    openFileDialog1.FilterIndex = 1
    openFileDialog1.RestoreDirectory = True

```

```

If openFileDialog1.ShowDialog() = DialogResult.OK Then
' Extract file strings from the dialog.
stFilePathAndName = openFileDialog1.FileName
Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
stFileNameOnly = MyFile.Name ' Global data.
txtFile.Text = MyFile.Name ' Display to Screen.

' This loads the XML into xDoc.
xDoc.load(stFilePathAndName)

' This loads Node data into Nodes.
Nodes = xDoc.documentElement.childNodes

'Clear out any old data.

```

```

ClearAll()
' Loop through and display materials.
RefreshMaterialListBox()
' Enable 'Delete Library' and 'List Invalid'.
DisableObjectsBeforeLoad()
EnableObjectsAfterLoad()

' Reset maxID to its proper value.
RecalcMaxID()

' Set the status bar.
StatusBar1.Text = "Status: Material library " + stFileNameOnly + _
    " has been loaded."
End If
End Sub

Private Sub NewXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewFile.Click, MenuItem2.Click

Dim saveFileDialog1 As New SaveFileDialog

saveFileDialog1.InitialDirectory = "xml\"
saveFileDialog1.Filter = "Material Library (*.mlb)|*.mlb"
saveFileDialog1.FilterIndex = 1
saveFileDialog1.RestoreDirectory = True

If saveFileDialog1.ShowDialog() = DialogResult.OK Then
' Set up streams for writing.
Dim filename As String = saveFileDialog1.FileName
Dim myFileStream As New System.IO.FileStream(filename, _
    System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.Write, _
    System.IO.FileShare.None)
Dim XMLWriter As New System.IO.StreamWriter(myFileStream)

' Write XML data to file.
XMLWriter.WriteLine("<?xml version=""1.0"" encoding=""UTF-8""?>")
XMLWriter.WriteLine("<materiallist maxID=""0"">")
XMLWriter.WriteLine("</materiallist>")

' Close the streams.
XMLWriter.Close()
myFileStream.Close()

' Load this new, empty file into the program.
stFilePathAndName = saveFileDialog1.FileName
Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
stFileNameOnly = MyFile.Name ' Global data.
txtFile.Text = MyFile.Name ' Display to Screen.

' This loads the XML into xDoc.
xDoc.load(stFilePathAndName)

' This loads Node data into Nodes.
Nodes = xDoc.documentElement.childNodes

'Clear all fields.
ClearAll()
' Loop through and display materials.
RefreshMaterialListBox()
' Enable 'Delete Library' and 'List Invalid'.
DisableObjectsBeforeLoad()
EnableObjectsAfterLoad()

' Set the status bar.
StatusBar1.Text = "Status: New material library " + stFileNameOnly + _
    " has been created.."
End If

End Sub

Private Sub ClearAll()

```

```

' Clear all fields.
txtName.Clear()
txtMass.Clear()
txtID.Clear()
txtFriction.Clear()
txtComName.Clear()
txtComPercent.Clear()
lstMaterial.Items.Clear()
lstComponents.Items.Clear()
End Sub

```

```

Private Sub DisableObjectsBeforeLoad()
    btnDeleteFile.Enabled = False
    MenuItem5.Enabled = False

    btnNewMat.Enabled = False
    btnSaveMat.Enabled = False
    btnDeleteMat.Enabled = False
    MenuItem7.Enabled = False
    MenuItem8.Enabled = False
    MenuItem9.Enabled = False

    btnNewCom.Enabled = False
    btnSaveCom.Enabled = False
    btnDelCom.Enabled = False
    MenuItem11.Enabled = False
    MenuItem12.Enabled = False
    MenuItem13.Enabled = False

    txtName.ReadOnly = True
    txtMass.ReadOnly = True
    txtID.ReadOnly = True
    txtFriction.ReadOnly = True
    txtComName.ReadOnly = True
    txtComPercent.ReadOnly = True

    btnListInvalid.Enabled = False

    NewMatType = 0
    NewComType = False
End Sub

```

```

Private Sub EnableObjectsAfterLoad()
    btnDeleteFile.Enabled = True
    MenuItem5.Enabled = True

    btnNewMat.Enabled = True
    MenuItem7.Enabled = True

    btnListInvalid.Enabled = True
End Sub

```

```

Private Sub DeleteXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDeleteFile.Click, MenuItem5.Click
    If (stFilePathAndName <> "") Then
        If (MsgBox("Delete " + stFileNameOnly + "?", MsgBoxStyle.OKCancel, "Delete _
Confirmation") = MsgBoxResult.OK) Then
            ' Delete the file and clear the appropriate field.
            Kill(stFilePathAndName)
            txtFile.Text = ""

            ' Set the status bar.
            StatusBar1.Text = "Status: Material library " + stFileNameOnly + _
                " has been successfully deleted."

            ' Clear the editor and disable all commands that can't be used.
            ClearAll()
            DisableObjectsBeforeLoad()
        End If
    End If
End Sub

```



```

Private Sub OnMaterialSelect(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstMaterial.SelectedIndexChanged
    ' Search through the XML and fill lstMaterial. Format: "String ID".
    Dim xNode, xComNode As IXMLDOMNode
    Dim idString As String
    Dim mResult As MsgBoxResult

    ' Do not do anything in this subroutine if the materials are being deselected by
    NewMaterial().
    If ((NewComType = True)) Then
        If (lstComponents.SelectedIndices.Count() = 1) Then
            mResult = MsgBox("Discard new material?", MsgBoxStyle.YesNo, _
                "Discard new material?")
            If (mResult = MsgBoxResult.Yes) Then
                ' Continue.
            Else
                ClearComponentListSelections()
                Exit Sub
            End If
        Else
            Exit Sub
        End If
    End If

    NewComType = 0 ' Reset this global to 'existing material'.

    ' Do not do anything in this subroutine if the materials are being deselected by
    NewMaterial().
    If ((NewMatType <> 0)) Then
        If (lstMaterial.SelectedIndices.Count() = 1) Then
            mResult = MsgBox("Discard new material?", MsgBoxStyle.YesNo, _
                "Discard new material?")
            If (mResult = MsgBoxResult.Yes) Then
                ' Continue.
            Else
                ClearMaterialListSelections()
                Exit Sub
            End If
        Else
            Exit Sub
        End If
    End If

    NewMatType = 0 ' Reset this global to 'existing material'

    idString = ParseIDFromString(lstMaterial.SelectedItem())

    txtID.ReadOnly = True
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = idString) Then
            IF (xNode.nodeName.StartsWith("material")) Then
                ' If the node is a standard material ...
                txtName.ReadOnly = False
                txtName.Text = xNode.attributes.getNamedItem("name").nodeValue()
                txtMass.Text = Cdbl(xNode.attributes.getNamedItem( _
                    "mass").nodeValue()).ToString("F")
                txtID.Text = xNode.attributes.getNamedItem("ID").nodeValue()
                txtFriction.Text = Cdbl(xNode.attributes.getNamedItem( _
                    "friction").nodeValue()).ToString("F")
                txtFriction.ReadOnly = False
                txtMass.ReadOnly = False
                txtComName.Clear()
                txtComPercent.Clear()
                lstComponents.Items.Clear()
                txtComName.ReadOnly = True
                txtComPercent.ReadOnly = True
                lstComponents.Enabled = False

                ' Adjust component buttons.
                btnNewCom.Enabled = False
            End If
        End If
    End For
End Sub

```

```

    btnSaveCom.Enabled = False
    btnDelCom.Enabled = False
    MenuItem11.Enabled = False
    MenuItem12.Enabled = False
    MenuItem13.Enabled = False
ElseIf (xNode.nodeName.StartsWith("combo")) Then
' If the node is a combination material ...
    txtName.ReadOnly = False
    txtName.Text = xNode.attributes.getNamedItem("name").nodeValue()
    txtMass.Text = Cdbl( CalcComMass(xNode)).ToString("F")
    txtID.Text = xNode.attributes.getNamedItem("ID").nodeValue()
    txtFriction.Text = Cdbl( CalcComFriction(xNode)).ToString("F")
    txtFriction.ReadOnly = True
    txtMass.ReadOnly = True
    txtComName.Clear()
    txtComPercent.Clear()
    lstComponents.Items.Clear()
    txtComName.ReadOnly = True
    txtComPercent.ReadOnly = True
    lstComponents.Enabled = True

' Adjust component buttons.
    btnNewCom.Enabled = True
    btnSaveCom.Enabled = False
    btnDelCom.Enabled = False
    MenuItem11.Enabled = True
    MenuItem12.Enabled = False
    MenuItem13.Enabled = False

' Add components to the list.
    Dim ComNodes As IXMLDOMNodeList
    Dim newComMaterial As String
    ComNodes = xNode.childNodes
    For Each xComNode In ComNodes
        If (xComNode.nodeName.StartsWith("component")) Then
            newComMaterial = GetMaterialNameFromID( _
                xComNode.attributes.getNamedItem("ID").nodeValue() + " " _
                + xComNode.attributes.getNamedItem("ID").nodeValue() )
            lstComponents.Items.Add(newComMaterial)
        End If
    Next
End If
End If
Next xNode

' Enable buttons.
    btnNewMat.Enabled = True
    btnSaveMat.Enabled = True
    btnDeleteMat.Enabled = True
    MenuItem7.Enabled = True
    MenuItem8.Enabled = True
    MenuItem9.Enabled = True
End Sub

Private Sub OnComponentSelect(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstComponents.SelectedIndexChanged
' Update the text fields when a component object is selected.
    Dim ComNodes As IXMLDOMNodeList
    Dim xNode, xComNode As IXMLDOMNode
    Dim nString, idComString, nComString, cPercent As String
    Dim bExit As Boolean
    Dim mResult As MsgBoxResult

' Do not do anything in this subroutine if the materials are being deselected by
NewComponent().
    If ((NewComType = True)) Then
        If (lstComponents.SelectedIndices.Count() = 1) Then
            mResult = MsgBox("Discard new material?", MsgBoxStyle.YesNo, _
                "Discard new material?")
            If (mResult = MsgBoxResult.Yes) Then
                Continue.
            End If
        End If
    End If

```

```

        Else
            ClearComponentListSelections()
            Exit Sub
        End If
    Else
        Exit Sub
    End If
End If

NewComType = 0 ' Reset this global to 'existing material'.

'Clear the text fields.
txtComName.Clear()
txtComPercent.Clear()
txtComName.ReadOnly = True

' This string is the ID of the currently selected component.
idComString = ParseIDFromString(lstComponents.SelectedItem())
' This string is the Name of the currently selected component.
nComString = ParseNameFromString(lstComponents.SelectedItem()) + " " _
    + idComString
' This string is the material selected in the top listbox.
nString = ParseNameFromString(lstMaterial.SelectedItem())

' Search through the list of nodes in order to find the proper % to display.
For Each xNode In Nodes
    If (xNode.nodeName.StartsWith("combo") And _
        (xNode.attributes.getNamedItem("name").nodeValue() = nString)) Then
        ComNodes = xNode.childNodes
        For Each xComNode In ComNodes
            If (xComNode.attributes.getNamedItem("ID").nodeValue() = idComString)
                Then
                    cPercent = xComNode.attributes.getNamedItem( _
                        "percent").nodeValue()
                    bExit = True
                    Exit For
                End If
            Next
        End If
        If (bExit = True) Then
            Exit For
        End if
    Next xNode

' Display the proper text.
txtComName.Text = nComString
txtComPercent.Text = cPercent
txtComPercent.ReadOnly = False

' Adjust component buttons.
btnNewCom.Enabled = True
btnSaveCom.Enabled = True
btnDelCom.Enabled = True
MenuItem11.Enabled = True
MenuItem12.Enabled = True
MenuItem13.Enabled = True
End Sub

Private Function CalcComMass(ByVal xNode As IXMLDOMNode) As Double
' Recursively determine a combo material's mass.
Dim ChildNodes As IXMLDOMNodeList
Dim xChildNode As IXMLDOMNode
Dim NodeID, Percent As Integer
If (xNode.hasChildNodes()) Then
    ChildNodes = xNode.childNodes
    For Each xChildNode In ChildNodes
        Percent = xChildNode.attributes.getNamedItem("percent").nodeValue()
        xChildNode = GetNodeFromID(xChildNode.attributes.getNamedItem( _
            "ID").nodeValue())
        CalcComMass += CalcComMass(xChildNode) * (Percent / 100)
    Next
End Function

```

```

    Return CalcComMass
Else
    If (xNode.nodeName() = "material") Then
        Return xNode.attributes.getNamedItem("mass").nodeValue()
    Else
        Return 0
    End If
End If
End Function

Private Function CalcComFriction(ByVal xNode As IXMLDOMNode) As Double
    ' Recursively determine a combo material's friction.
    Dim ChildNodes As IXMLDOMNodeList
    Dim xChildNode As IXMLDOMNode
    Dim NodeID, Percent As Integer
    If (xNode.hasChildNodes()) Then
        ChildNodes = xNode.childNodes
        For Each xChildNode In ChildNodes
            Percent = xChildNode.attributes.getNamedItem("percent").nodeValue()
            xChildNode = GetNodeFromID(xChildNode.attributes.getNamedItem( _
                "ID").nodeValue())
            CalcComFriction += CalcComFriction(xChildNode) * (Percent / 100)
        Next
        Return CalcComFriction
    Else
        If (xNode.nodeName() = "material") Then
            Return xNode.attributes.getNamedItem("friction").nodeValue()
        Else
            Return 0
        End If
    End If
End Function

Private Function GetMaterialNameFromID(ByVal id As String) As String
    Dim xNode As IXMLDOMNode
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = id) Then
            Return xNode.attributes.getNamedItem("name").nodeValue()
        End If
    Next
End Function

Private Function GetNodeFromID(ByVal id As String) As IXMLDOMNode
    Dim xNode As IXMLDOMNode
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = id) Then
            Return xNode
        End If
    Next
End Function

Private Function ParseNameFromString(ByVal aString As String) As String
    ' The string is in the format 'Name ID'
    Dim nLength As Integer
    nLength = aString.IndexOf(" ")
    If (nLength < 0) Then
        ParseNameFromString = ""
        Exit Function
    End If
    ParseNameFromString = aString.Substring(0, nLength)
End Function

Private Function ParseIDFromString(ByVal aString As String) As String
    ' The string is in the format 'Name ID'
    Dim nLength As Integer
    nLength = (aString.Length() - aString.IndexOf(" ")) - 1
    If (nLength < 0) Then
        ParseIDFromString = ""
        Exit Function
    End If
    ParseIDFromString = aString.Substring((aString.IndexOf(" ") + 1), nLength)
End Function

```

```

End Function

Private Sub ExitProgram(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnExit.Click, MenuItem6.Click
    Close()
End Sub

Private Sub NewMaterial(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewMat.Click, MenuItem7.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    ' Choose the type of material to be created.
    Dim typeResult, nameResult As MsgBoxResult
    typeResult = MsgBox("Standard material (Yes) or Combination material (No)?", _
        MsgBoxStyle.YesNoCancel, "Create New Material")

    If (typeResult = MsgBoxResult.Yes) Then
        NewMatType = 1
        ClearForm()
        txtName.ReadOnly = False
        txtFriction.ReadOnly = False
        txtMass.ReadOnly = False
        txtComName.ReadOnly = True
        txtComPercent.ReadOnly = True
    ElseIf (typeResult = MsgBoxResult.No) Then
        NewMatType = 2
        ClearForm()
        txtName.ReadOnly = False
        txtFriction.ReadOnly = True
        txtMass.ReadOnly = True
        txtComName.ReadOnly = True
        txtComPercent.ReadOnly = True
    ElseIf (typeResult = MsgBoxResult.Cancel) Then
        Exit Sub
    End If

    ' Reset maxID to its proper value.
    RecalcMaxID()

    txtID.ReadOnly = False
    txtID.Text = (xDoc.documentElement.attributes.getNamedItem( _
        "maxID").nodeValue() + 1)
    txtName.Focus()

    ' Adjust commands available.
    btnNewMat.Enabled = False
    btnSaveMat.Enabled = True
    btnDeleteMat.Enabled = False
    MenuItem7.Enabled = False
    MenuItem8.Enabled = True
    MenuItem9.Enabled = False

    ' Set the status bar.
    StatusBar1.Text = "Status: " _
        + "The editor is ready for the entry of a new material."
End Sub

Private Sub ClearForm()
ClearMaterialListSelections()
txtName.Clear()
txtMass.Clear()
txtID.Clear()
txtFriction.Clear()
txtComName.Clear()
txtComPercent.Clear()
lstComponents.Items.Clear()
End Sub

```

```

Private Sub ClearMaterialListSelections()
    For nIndex As Integer = 0 To (lstMaterial.Items.Count() - 1)
        lstMaterial.SetSelected(nIndex, False)
    Next nIndex
End Sub

Private Sub SaveMaterial(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSaveMat.Click, MenuItem8.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    Dim xNode, newNode As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute
    Dim mResult As MsgBoxResult

    ' Check state - 0 means update, 1 or 2 means create new node of a specific type.
    If (NewMatType = 0) Then 'This is an existing material.
        ' Copy over old values.
        For Each xNode In Nodes
            If (txtID.Text() = xNode.attributes.getNamedItem("ID").nodeValue()) Then
                ' Update 'Name'.
                newAttr = xDoc.createAttribute("name")
                newAttr.nodeValue = txtName.Text()
                xNode.attributes.setNamedItem(newAttr)
                ' Update 'Mass'.
                newAttr = xDoc.createAttribute("mass")
                newAttr.nodeValue = txtMass.Text()
                xNode.attributes.setNamedItem(newAttr)
                ' Update 'Friction'.
                newAttr = xDoc.createAttribute("friction")
                newAttr.nodeValue = txtFriction.Text()
                xNode.attributes.setNamedItem(newAttr)
                ' Save XML data.
                xDoc.save(stFilePathAndName)
                RefreshMaterialListBox()
                SetMaterialListboxFocus(txtID.Text())

                ' Set the status bar.
                StatusBar1.Text = "Status: Material " + txtName.Text _
                    + " updated successfully."
            End If
        Next
    ElseIf (NewMatType = 1) Then 'This is a new Standard Material.
        ' Check to see if the ID already exists.
        Dim bIDExists As Boolean
        For Each xNode In Nodes
            If (txtID.Text() = xNode.attributes.getNamedItem("ID").nodeValue()) Then
                mResult = MsgBox("Copy over old material?", MsgBoxStyle.YesNo, _
                    "ID Already Exists")
                If (mResult = MsgBoxResult.Yes) Then
                    bIDExists = True
                    Exit For
                Else
                    Exit Sub
                End If
            End If
        Next
    End If

    ' Save data as <material>.

    ' Check to see that no fields are null or filled out incorrectly.
    If (CheckProperSyntaxStandard() = False) Then
        Exit Sub
    End If

    ' Delete old node.
    If (bIDExists) Then

```

```

DeleteNodeByID(txtID.Text)
End If

' Save data in a new node:
newNode = xDoc.createElement("material")
' ID
newAttr = xDoc.createAttribute("ID")
newAttr.nodeValue = txtID.Text()
newNode.attributes.setNamedItem(newAttr)
' Name
newAttr = xDoc.createAttribute("name")
newAttr.nodeValue = txtName.Text()
newNode.attributes.setNamedItem(newAttr)
' Mass
newAttr = xDoc.createAttribute("mass")
newAttr.nodeValue = txtMass.Text()
newNode.attributes.setNamedItem(newAttr)
' Friction
newAttr = xDoc.createAttribute("friction")
newAttr.nodeValue = txtFriction.Text()
newNode.attributes.setNamedItem(newAttr)
' Add in the new information as a new node.
xDoc.documentElement.appendChild(newNode)
' Update maxID.
newAttr = xDoc.createAttribute("maxID")
newAttr.nodeValue = (xDoc.documentElement.attributes.getNamedItem( _
    "maxID").nodeValue() + 1)
xDoc.documentElement.attributes.setNamedItem(newAttr)
' Save xml to file.
xDoc.save(stFilePathAndName)

' Reset state variable and refresh the Material List:
NewMatType = 0
RefreshMaterialListBox()
SetMaterialListboxFocus(txtID.Text())

' Set the status bar.
StatusBar1.Text = "Status: New standard material " + txtName.Text _
    + " added successfully."

ElseIF (NewMatType = 2) Then 'This is a new Combination Material
' Check to see if the ID already exists.
Dim bIDExists As Boolean
For Each xNode In Nodes
    IF (txtID.Text() = xNode.attributes.getNamedItem("ID").nodeValue()) Then
        mResult = MsgBox("Copy over old material?", MsgBoxStyle.YesNo, _
            "ID Already Exists")
        IF (mResult = MsgBoxResult.Yes) Then
            bIDExists = True
            Exit For
        Else
            Exit Sub
        End If
    End If
Next

' Save data as <combo>.

' Check to see that no fields are null or filled out incorrectly.
IF (CheckProperSyntaxCombo() = False) THEN
    Exit Sub
End IF

' Deletes old node.
IF (bIDExists) THEN
    DeleteNodeByID(txtID.Text)
End IF

' Save data in a new node.
newNode = xDoc.createElement("combo")
' ID

```

```

newAttr = xDoc.createAttribute("ID")
newAttr.nodeValue = txtID.Text()
newNode.attributes.setNamedItem(newAttr)
' Name
newAttr = xDoc.createAttribute("name")
newAttr.nodeValue = txtName.Text()
newNode.attributes.setNamedItem(newAttr)
' Add in the new information as a new node.
xDoc.documentElement.appendChild(newNode)
' Update maxID.
newAttr = xDoc.createAttribute("maxID")
newAttr.nodeValue = (xDoc.documentElement.attributes.getNamedItem( _
    "maxID").nodeValue() + 1)
xDoc.documentElement.attributes.setNamedItem(newAttr)
' Save xml to file.
xDoc.save(stFilePathAndName)

'Reset state variable and refresh the Material List.
NewMatType = 0
RefreshMaterialListBox()
SetMaterialListboxFocus(txtID.Text())

' Set the status bar.
StatusBar1.Text = "Status: New combination material " _
    + txtName.Text + " added successfully."
End If

' Adjust available commands.
btnNewMat.Enabled = True
btnSaveMat.Enabled = True
btnDeleteMat.Enabled = True
MenuItem7.Enabled = True
MenuItem8.Enabled = True
MenuItem9.Enabled = True
End Sub

Private Function IsFileLoaded() As Boolean
    If (txtFile.Text = "None") Then
        ' Set the status bar.
        StatusBar1.Text = _
            "Status: This function is unavailable as there is no library loaded!"
        Return False
    End If
    Return True
End Function

Private Function CheckProperSyntaxStandard() As Boolean
    If ((txtName.Text.Length() = 0) Or (txtName.Text.IndexOf(" ") > -1)) Then
        ' Set the status bar.
        StatusBar1.Text = _
            "Status: Syntax error. Invalid Name. Text missing or contains spaces."
        Return False
    End If
    If (txtMass.Text.Length() = 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Syntax error. Mass field is empty."
        Return False
    End If
    If (txtFriction.Text.Length() = 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Syntax error. Friction field is empty."
        Return False
    End If
    If (txtID.Text.Length() = 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Syntax error. ID field is empty."
        Return False
    End If
    Return True
End Function

```



```

Private Function CheckProperSyntaxCombo() As Boolean
    If ((txtName.Text.Length() = 0) Or (txtName.Text.IndexOf(" ") > -1)) Then
        ' Set the status bar.
        StatusBar1.Text =
            "Status: Syntax error. Invalid Name. Text missing or contains spaces."
        Return False
    End If
    If (txtID.Text.Length() = 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Syntax error. ID field is empty."
        Return False
    End If
    Return True
End Function

Private Sub DeleteMaterial(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDeleteMat.Click, MenuItem9.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    ' Can't delete if this is a new node.
    If (NewMatType <> 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Cannot delete an unsaved material."
        Exit Sub
    End If

    ' Can't delete a material if combo materials depend on it. Report dependencies to
the user.
    Dim nArray As Array = Array.CreateInstance(GetType(String), Nodes.Length())
    nArray.SetValue("0", 0)
    Dim eString As String
    GetDependencies(nArray, txtID.Text())
    If (nArray.GetValue(0) <> "0") Then
        eString = "Cannot delete a material with dependencies:"
        For nMember As Integer = 0 To (nArray.Length() - 1)
            eString = eString + " " + nArray.GetValue(nMember)
        Next nMember
        StatusBar1.Text = "Status: " + eString + ". Cannot perform deletion."
        Exit Sub
    End If

    ' Delete the current node.
    Dim dResult As MsgBoxResult
    dResult = MsgBox("Delete currently selected Material?", _
        MsgBoxStyle.YesNo, "Material Deletion")
    If (dResult = MsgBoxResult.Yes) Then
        DeleteNodeByID(txtID.Text())
        RefreshMaterialListBox()
        xDoc.save(stFilePathAndName)
    Else
        Exit Sub
    End If

    ' Set the status bar.
    StatusBar1.Text = "Status: Material " + txtName.Text + " deleted successfully."

    ' Clear the form since the material has been deleted.
    txtName.Clear()
    txtMass.Clear()
    txtID.Clear()
    txtFriction.Clear()
    txtName.ReadOnly = True
    txtMass.ReadOnly = True
    txtID.ReadOnly = True
    txtFriction.ReadOnly = True
    txtComName.Clear()
    txtComPercent.Clear()
    txtComName.ReadOnly = True

```

```

txtComPercent.ReadOnly = True
lstComponents.Items.Clear()
lstComponents.Enabled = True

btnNewMat.Enabled = True
btnSaveMat.Enabled = False
btnDeleteMat.Enabled = False
MenuItem7.Enabled = True
MenuItem8.Enabled = False
MenuItem9.Enabled = False
End Sub

Private Sub DeleteNodeByID(ByVal nID As String)
    Dim xNode As IXMLDOMNode
    xNode = GetNodeFromID(nID)
    xDoc.documentElement.removeChild(xNode)
End Sub

Private Sub RefreshMaterialListBox()
    Dim xNode As IXMLDOMNode
    Dim newMaterial As String
    lstMaterial.Items.Clear()
    For Each xNode In Nodes
        newMaterial = xNode.attributes.getNamedItem("name").nodeValue() _
            + " " + xNode.attributes.getNamedItem("ID").nodeValue()
        lstMaterial.Items.Add(newMaterial)
    Next xNode
End Sub

Private Sub RefreshComponentListBox()
    Dim xNode, cNode As IXMLDOMNode
    Dim ComNodes As IXMLDOMNodeList
    Dim newCom As String

    lstComponents.Items.Clear()
    cNode = GetNodeFromID(txtID.Text())
    ComNodes = cNode.childNodes()
    For Each xNode In ComNodes
        newCom = GetMaterialNameFromID(xNode.attributes.getNamedItem(_
            "ID").nodeValue()) + " " + xNode.attributes.getNamedItem("ID").nodeValue()
        lstComponents.Items.Add(newCom)
    Next xNode
End Sub

Private Function GetDependencies(ByRef nArray As Array, ByVal nID As Integer)
    Dim xNode, xComNode As IXMLDOMNode
    Dim ComNodes As IXMLDOMNodeList
    Dim nCount As Integer = 0
    For Each xNode In Nodes
        IF (xNode.nodeName.StartsWith("combo")) Then
            ComNodes = xNode.childNodes
            For Each xComNode In ComNodes
                IF (xComNode.nodeName.StartsWith("component")) Then
                    IF (xComNode.attributes.getNamedItem("ID").nodeValue() = nID)
                        Then
                            Dim sName As String = GetMaterialNameFromID(_
                                xNode.attributes.getNamedItem("ID").nodeValue())
                            nArray.SetValue(sName, nCount)
                            nCount += 1
                        End If
                    End If
                Next
            End If
        Next
    End If
    Next xNode
End Function

Private Sub SetMaterialListBoxFocus(ByVal nID As String)
    Dim mString As String = GetMaterialNameFromID(nID) + " " + nID
    lstMaterial.SetSelected(lstMaterial.Items.IndexOf(mString), True)
End Sub

```

```

Private Sub SetComponentListboxFocus(ByVal nString As String)
    lstComponents.SetSelected(lstComponents.Items.IndexOf(nString), True)
End Sub

Private Sub ClearComponentListSelections()
    For nIndex As Integer = 0 To (lstComponents.Items.Count() - 1)
        lstComponents.SetSelected(nIndex, False)
    Next nIndex
End Sub

Private Sub NewComponent(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewCom.Click, MenuItem11.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    ' Is there a combination material selected?
    Dim tNode As IXMLDOMNode
    If (lstMaterial.SelectedIndex() > -1) Then
        Dim ID As String = ParseIDFromString(lstMaterial.SelectedItem())
        tNode = GetNodeFromID(ID)
        If (tNode.baseName() <> "combo") Then
            ' Set the status bar.
            StatusBar1.Text = "Status: This function may only be used _
                if a Combination Material is selected!"
            Exit Sub
        End If
    Else
        ' Set the status bar.
        StatusBar1.Text = "Status: This function may only be used if _
            a Combination Material is selected!"
        Exit Sub
    End If

    ' Set the status bar.
    StatusBar1.Text = "Status: The editor is ready for the entry of a new component."

    ' Set up the fields in order to add a new component.
    NewComType = True
    ClearComponentListSelections()
    txtComName.Clear()
    txtComPercent.Clear()
    txtComPercent.ReadOnly = False
    txtComName.ReadOnly = False
    txtComName.Focus()

    ' Adjust component buttons.
    btnNewCom.Enabled = False
    btnSaveCom.Enabled = True
    btnDelCom.Enabled = False
    MenuItem11.Enabled = False
    MenuItem12.Enabled = True
    MenuItem13.Enabled = False
End Sub

Private Sub SaveComponent(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSaveCom.Click, MenuItem12.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    ' Is there a combo material selected?
    Dim tNode As IXMLDOMNode
    If (lstMaterial.SelectedIndex() > -1) Then
        Dim ID As String = ParseIDFromString(lstMaterial.SelectedItem())
        tNode = GetNodeFromID(ID)
        If (tNode.baseName() <> "combo") Then
            ' Set the status bar.
            StatusBar1.Text = "Status: This function may only be used if _
                if a Combination Material is selected!"
        End If
    End If
End Sub

```

```

        a Combination Material is selected!"
    Exit Sub
End If
Else
    ' Set the status bar.
    StatusBar1.Text = "Status: This function may only be used if a _
        Combination Material is selected!"
    Exit Sub
End If

' Validate that the ID exists -- also check to see if it's already a child.
otherwise, add it.
Dim xNode, newNode, cNode As IXMLDOMNode
Dim ComNodes As IXMLDOMNodeList
Dim newAttr As IXMLDOMAttribute
Dim bValid As Boolean

cNode = GetNodeFromID(txtID.Text())
ComNodes = cNode.childNodes()
If (NewComType = True) Then ' Save a new component.
    ' Validate the material is it a real material?
    For Each xNode In Nodes
        If ((ParseNameFromString(txtComName.Text()) = _
            xNode.attributes.getNamedItem("name").nodeValue()) And _
            (ParseIDFromString(txtComName.Text()) = _
            xNode.attributes.getNamedItem("ID").nodeValue())) Then
            bValid = True
        End If
    Next
    If (bValid = False) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: This is not a valid material."
        Exit Sub
    End If
    ' Check to make sure the new material isn't a duplicate.
    For Each xNode In ComNodes
        If (ParseIDFromString(txtComName.Text()) = _
            xNode.attributes.getNamedItem("ID").nodeValue()) Then
            ' This ID already exists.
            ' Set the status bar.
            StatusBar1.Text = "Status: This ID is already a component."
            Exit Sub
        End If
    Next
    ' Prevent a material from adding itself as a component.
    If (ParseIDFromString(txtComName.Text()) = _
        ParseIDFromString(lstMaterial.SelectedItem())) Then
        ' This ID already exists.
        ' Set the status bar.
        StatusBar1.Text = "Status: Cannot define self as a component."
        Exit Sub
    End If

    ' Save the component.

    newNode = xDoc.createElement("component")
    ' Name
    newAttr = xDoc.createAttribute("ID")
    newAttr.nodeValue = ParseIDFromString(txtComName.Text())
    newNode.attributes.setNamedItem(newAttr)
    ' Percent
    newAttr = xDoc.createAttribute("percent")
    newAttr.nodeValue = txtComPercent.Text()
    newNode.attributes.setNamedItem(newAttr)
    ' Save XML data.
    cNode.appendChild(newNode)
    xDoc.save(stFilePathAndName)
    RefreshComponentListBox()

    ' Reset permissions on txtComName.
    txtComName.ReadOnly = True

```

```

NewComType = False

SetComponentListboxFocus(txtComName.Text())

' Set the status bar.
StatusBar1.Text = "Status: New component " + txtComName.Text _
+ " saved successfully."
Else ' Update an old component.
For Each xNode In ComNodes
If (ParseIDFromString(txtComName.Text()) = _
xNode.attributes.getNamedItem("ID").nodeValue) Then
' Update 'Percent'.
newAttr = xDoc.createAttribute("percent")
newAttr.nodeValue = txtComPercent.Text()
xNode.attributes.setNamedItem(newAttr)
' Save XML data.
xDoc.save(stFilePathAndName)

' Set the status bar.
StatusBar1.Text = "Status: Component " + txtComName.Text _
+ " updated successfully."
Exit For
End If
Next
End If

' Adjust component buttons.
btnNewCom.Enabled = True
btnSaveCom.Enabled = True
btnDelCom.Enabled = True
MenuItem11.Enabled = True
MenuItem12.Enabled = True
MenuItem13.Enabled = True
End Sub

Private Sub DeleteComponent(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDelCom.Click, MenuItem13.Click
' Is there a library of data loaded?
If (IsFileLoaded() = False) Then
Exit Sub
End If

' Is there a combo material selected?
Dim tNode As IXMLDOMNode
If (lstMaterial.SelectedIndex() > -1) Then
Dim ID As String = ParseIDFromString(lstMaterial.SelectedItem())
tNode = GetNodeFromID(ID)
If (tNode.baseName() <> "combo") Then
' Set the status bar.
StatusBar1.Text = "Status: This function may only be used if a _
Combination Material is selected!"
Exit Sub
End If
Else
' Set the status bar.
StatusBar1.Text = "Status: This function may only be used if a _
Combination Material is selected!"
Exit Sub
End If

' Delete the current component.
Dim dResult As MsgBoxResult
dResult = MsgBox("Delete currently selected Component?", _
MsgBoxStyle.YesNo, "Material Deletion")
If (dResult = MsgBoxResult.Yes) Then
Dim xNode, cNode As IXMLDOMNode
Dim ComNodes As IXMLDOMNodeList

cNode = GetNodeFromID(txtID.Text())
ComNodes = cNode.childNodes()
For Each xNode In ComNodes

```

```

        If (ParseIDFromString(lstComponents.SelectedItem()) = _
            xNode.attributes.getNamedItem("ID").nodeValue()) Then
            cNode.removeChild(xNode)
            Exit For
        End If
    Next xNode
    RefreshComponentListBox()
    xDoc.save(stFilePathAndName)

    ' Set the status bar.
    StatusBar1.Text = "Status: The component " + txtComName.Text _
        + " has been removed from material " + txtName.Text + "."

    txtComName.Clear()
    txtComPercent.Clear()
    txtComName.ReadOnly = True
    txtComPercent.ReadOnly = True
Else
    Exit Sub
End If

' Adjust component buttons.
btnNewCom.Enabled = True
btnSaveCom.Enabled = False
btnDelCom.Enabled = False
MenuItem11.Enabled = True
MenuItem12.Enabled = False
MenuItem13.Enabled = False
End Sub

Private Sub ListInvalidCombinationMaterials(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListInvalid.Click
    ' Is there a library of data loaded?
    If (IsFileLoaded() = False) Then
        Exit Sub
    End If

    ' Use a MsgBox to display all invalid combination materials (components don't add
    up to 100%).
    Dim eString As String
    Dim xNode, cNode As IXMLDOMNode
    Dim ComNodes As IXMLDOMNodeList
    Dim nPercent As Integer

    For Each xNode In Nodes
        If (xNode.nodeName() = "combo") Then
            nPercent = 0
            ComNodes = xNode.childNodes()
            For Each cNode In ComNodes
                nPercent += cNode.attributes.getNamedItem("percent").nodeValue()
            Next
            If (nPercent <> 100) Then
                eString += xNode.attributes.getNamedItem("name").nodeValue() _
                    + " " + xNode.attributes.getNamedItem("ID").nodeValue() + vbCrLf
            End If
        End If
    Next

    ' Set this message if no combo materials are invalid.
    If (eString = "") Then
        eString = "All combination materials are valid!"
    End If

    MsgBox(eString, MsgBoxStyle.Information, "Invalid Combination Material List")
End Sub

Private Sub RecalcMaxID()
    Dim xNode As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute
    Dim ProperMax As Integer

    ProperMax = 0

```

```
For Each xNode In Nodes
  If (xNode.attributes.getNamedItem("ID").nodeValue() > ProperMax) Then
    ProperMax = xNode.attributes.getNamedItem("ID").nodeValue()
  End If
Next

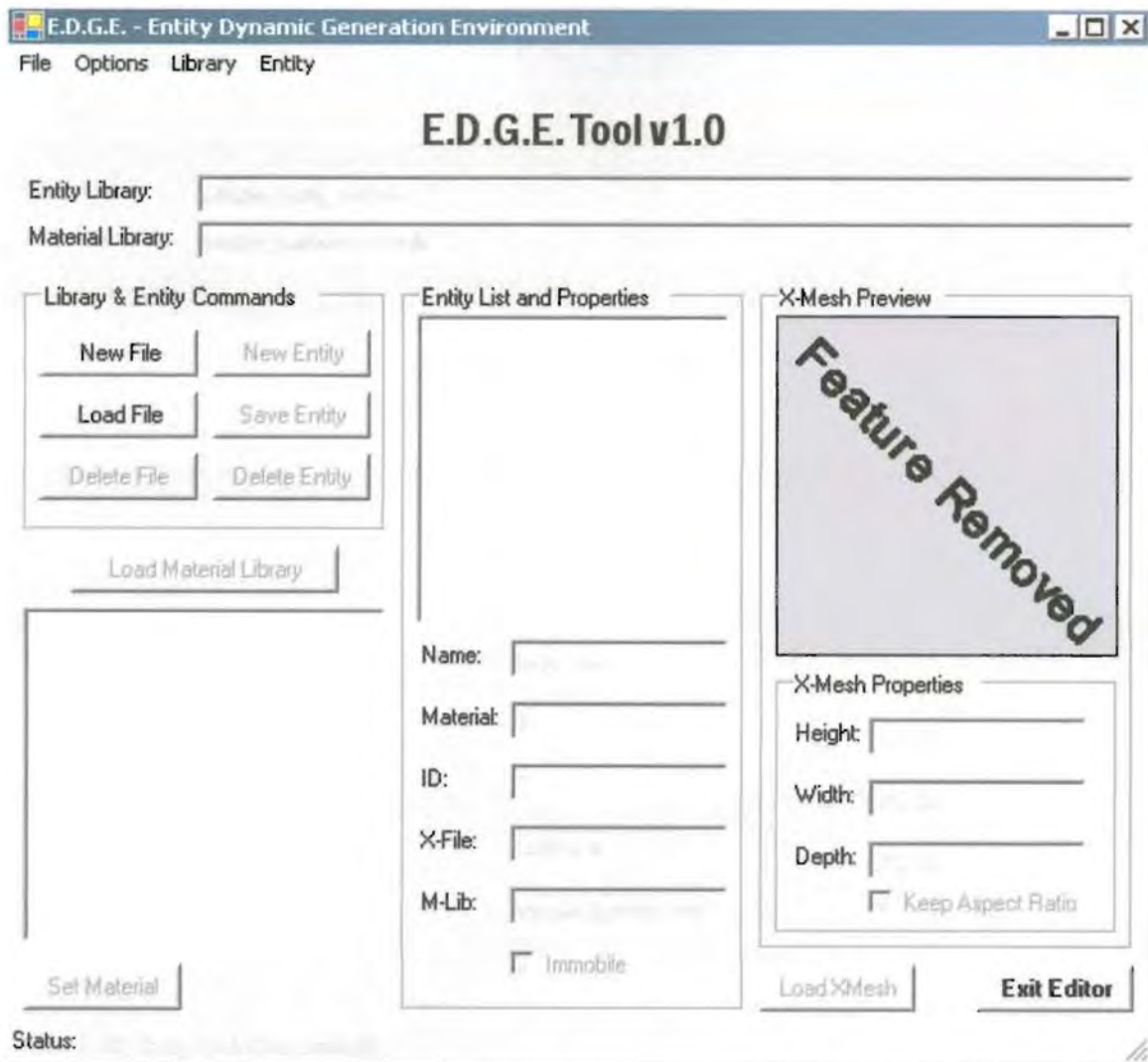
' Update 'maxID'.
newAttr = xDoc.createAttribute("maxID")
newAttr.nodeValue = ProperMax
xDoc.documentElement.attributes.setNamedItem(newAttr)
' Save XML data.
xDoc.save(stFilePathAndName)
End Sub
End Class
```

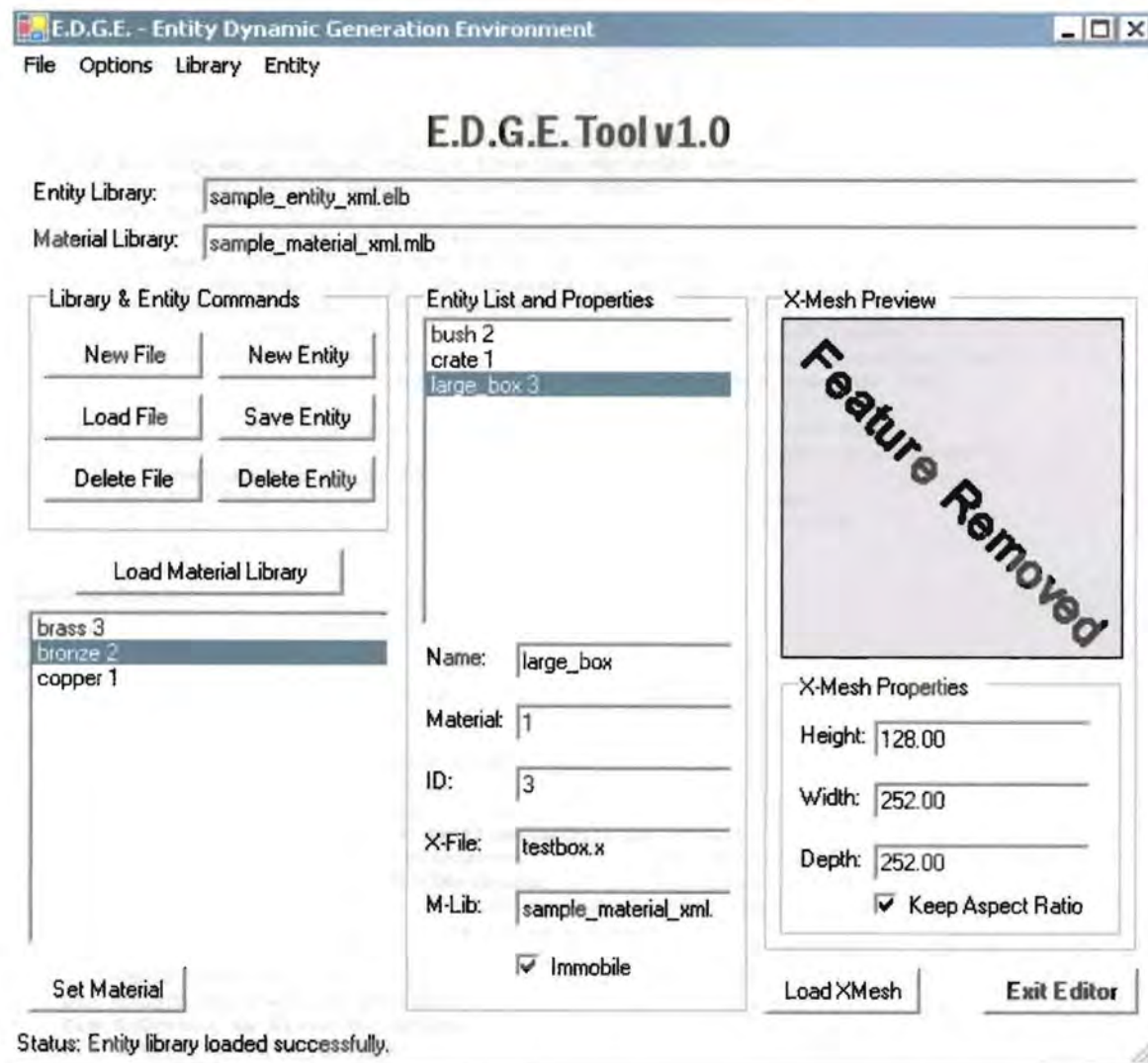
Section 5

E.D.G.E. Tool Code and Screenshots

E.D.G.E. Tool Screenshot

(1 of 2)





```

-----
' File: frmMain.vb
'
' Desc: This is the primary and only file for the EDGE Tool.
'
' First created on:    January 30th, 2005
' Last modification:  March 18th, 2005
'
' Copyright (c) Jason M. Black (donblas@donblas.org)
-----
' Revision History:
'
' 01-20-05: Started work on the interface.
' 01-21-05: Finished the interface. New files can be created.
' 01-22-05: Copied in 'RecalcMaxID' from the Material Editor.
'          NewEntity and OnEntityListSelect added.
' 01-27-05: SetMaterial implemented properly.
' 02-04-05: Attempting to get a Mesh rendered. Difficulties involving setup.
'          Most likely I will not render the mesh, but perhaps a 2D image
'          of the mesh instead. Unfortunately, getting the rendering to
'          synch with the application is difficult, especially with little to
'          no documentation on DX9 and VB.NET combined in this manner.
' 02-07-05: Abandoned the rendering of meshes. I do calculate dimensions from
'          the meshes now though. Added in Save and Delete commands for
'          entities. MaxID is also now recalculated properly.
' 02-08-05: Mesh size auto-adjust is added in and tested. v1.0 complete.
' 03-01-05: Fixed two bugs: ID is now non-editable, and fields are disabled
'          when an entity is deleted.
' 03-02-05: Adjusted read/write paths and recommented the code.
' 03-18-05: Added in the saving of original mesh dimensions to XML.
-----

```

```

Imports System
Imports System.IO
Imports MSXML2
Imports Microsoft.DirectX
Imports Microsoft.DirectX.Direct3D

```

```

Public Class frmMain
    Inherits System.Windows.Forms.Form
    Dim stFilePathAndName As String
    Dim stFileNameOnly As String
    Dim stMatFilePathAndName, stMatFileNameOnly As String
    Dim xDoc, xDocMat As New DOMDocument ' The XML Library.
    Dim Nodes, MatNodes As IXMLDOMNodeList ' The nodes in the Library.
    Dim bNewEntity As Boolean ' FALSE is normal, TRUE indicates a new entity
                              is being created.

    ' DirectX Objects.
    Dim dxD3DX As New DirectX3D.D3DX
    Dim dxDevice As DirectX3D.Device
    Dim dxMesh As DirectX3D.Mesh
    Dim oWidth, oHeight, oDepth As Double
    ' Used for exclusivity when adjusting 1 of 3 size values.
    Dim bFreezeAdjust As Boolean
    ' Used to halt all adjustments when loading initial values.
    Dim bFreezeAdjustAll As Boolean

    ' All code in this "Region" was created by VS.NET as the graphical form was assembled.
    #Region " Windows Form Designer generated code "
    ' Auto-generated code removed for clarity.
    #End Region

    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Me.Show()

        ' Initialize DirectX objects.
        Dim dxPP As New DirectX3D.PresentParameters
        dxPP.Windowed = True
        dxPP.SwapEffect = SwapEffect.Copy ' Discard?

```

```

dxPP.BackBufferFormat = Format.A8R8G8B8
dxDevice = New Direct3D.Device(0, DeviceType.Hardware, Me, _
    CreateFlags.SoftwareVertexProcessing, dxPP)
End Sub

Private Sub ExitProgram(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnExit.Click, MenuItem2.Click
    Close()
End Sub

Private Sub LoadXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnLoadFile.Click, MenuItem5.Click

    Dim openFileDialog1 As New OpenFileDialog
    Dim stFileName As String

    openFileDialog1.InitialDirectory = "xml\"
    openFileDialog1.Title = "Open Entity Library"
    openFileDialog1.Filter = "Entity Library (*.elb)|*.elb"
    openFileDialog1.FilterIndex = 1
    openFileDialog1.RestoreDirectory = True

    If openFileDialog1.ShowDialog() = DialogResult.OK Then
        ' Extract file strings from the dialog.
        stFilePathAndName = openFileDialog1.FileName
        Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
        stFileNameOnly = MyFile.Name ' Global data.
        txtEntityFile.Text = MyFile.Name ' Display to Screen.

        ' This loads the XML into xDoc.
        xDoc.load(stFilePathAndName)

        ' This loads Node data into Nodes.
        Nodes = xDoc.documentElement.childNodes

        ' Clear out any old data.
        ClearAll()
        ' Loop through and display entities.
        RefreshEntityListBox()
        ' Enable 'Delete Library' and 'Exit Invalid', disable everything else.
        ChangeObjectsAfterLoad()
        ' Correct the maxID attribute.
        RecalcMaxID()

        ' Set the status bar.
        StatusBar1.Text = "Status: XML file " + stFileNameOnly _
            + " loaded successfully."
    End If
End Sub

Private Sub NewXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewFile.Click, MenuItem4.Click

    Dim saveFileDialog1 As New SaveFileDialog

    saveFileDialog1.InitialDirectory = "xml\"
    saveFileDialog1.Filter = "Entity Library (*.elb)|*.elb"
    saveFileDialog1.FilterIndex = 1
    saveFileDialog1.RestoreDirectory = True

    If saveFileDialog1.ShowDialog() = DialogResult.OK Then
        ' Set up streams for writing.
        Dim filename As String = saveFileDialog1.FileName
        Dim myFileStream As New System.IO.FileStream(filename, _
            System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.Write, _
            System.IO.FileShare.None)
        Dim XMLWriter As New System.IO.StreamWriter(myFileStream)

        ' Write XML data to file.
        XMLWriter.WriteLine("<?xml version=""1.0"" encoding=""UTF-8""?>")
        XMLWriter.WriteLine("<entitylist maxID=""0"">")
    End If
End Sub

```

```

XMLWriter.WriteLine("</entitylist>")

' Close the streams.
XMLWriter.Close()
myFileStream.Close()

' Load this new, empty file into the program.
stFilePathAndName = saveFileDialog1.FileName
Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
stFileNameOnly = MyFile.Name      ' Global data.
txtEntityFile.Text = MyFile.Name  ' Display to Screen.

' This loads the XML into xDoc.
xDoc.load(stFilePathAndName)

' This loads Node data into Nodes.
Nodes = xDoc.documentElement.childNodes

' Clear all fields.
ClearAll()
' Loop through and display materials.
RefreshEntityListBox()
' Enable 'Delete Library' and 'Dist Invalid', disable everything else.
ChangeObjectsAfterLoad()

' Set the status bar.
StatusBar1.Text = "Status: New XML file " + stFileNameOnly _
+ " created successfully."
End If

End Sub

Private Sub DeleteXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDeleteFile.Click, MenuItem6.Click
If (stFilePathAndName <> "") Then
If (MsgBox("Delete " + stFileNameOnly + "?", MsgBoxStyle.OKCancel, _
"Delete Confirmation") = MsgBoxResult.OK) Then
' Delete the file and reset the appropriate field
Kill(stFilePathAndName)
txtEntityFile.Text = ""

' Set the status bar.
StatusBar1.Text = "Status: " + stFileNameOnly _
+ " has been successfully deleted."

' Clear the editor and disable all commands that can't be used.
ClearAll()
DisableCommandsOnDelete()
End If
End If
End Sub

Private Sub ClearAll()
txtMatFile.Text = ""
lstMaterials.Items.Clear()
lstEntity.Items.Clear()
txtName.Text = ""
txtMat.Text = ""
txtMatLib.Text = ""
txtID.Text = ""
txtXFile.Text = ""
chkImmobile.Checked = False
chkAspect.Checked = False
txtHeight.Text = ""
txtWidth.Text = ""
txtDepth.Text = ""
End Sub

Private Sub RefreshEntityListBox()
Dim xNode As IXMLDOMNode
Dim newEntity As String

```

```

lstEntity.Items.Clear()
For Each xNode In Nodes
    newEntity = xNode.attributes.getNamedItem("name").nodeValue() _
        + " " + xNode.attributes.getNamedItem("ID").nodeValue()
    lstEntity.Items.Add(newEntity)
Next xNode
End Sub

Private Sub RefreshMaterialListBox()
    Dim xNode As IXMLDOMNode
    Dim newMaterial As String
    lstMaterials.Items.Clear()
    For Each xNode In MatNodes
        newMaterial = xNode.attributes.getNamedItem("name").nodeValue() _
            + " " + xNode.attributes.getNamedItem("ID").nodeValue()
        lstMaterials.Items.Add(newMaterial)
    Next xNode
End Sub

Private Sub ChangeObjectsAfterLoad()
    MenuItem6.Enabled = True      ' Delete File.
    MenuItem8.Enabled = True      ' Create Entity.
    MenuItem12.Enabled = True     ' Load M-Lib.
    btnDeleteFile.Enabled = True
    btnNewEntity.Enabled = True
    btnLoadMatLib.Enabled = True

    MenuItem9.Enabled = False
    MenuItem10.Enabled = False
    MenuItem13.Enabled = False
    MenuItem15.Enabled = False

    btnSaveEntity.Enabled = False
    btnDeleteEntity.Enabled = False

    btnSetMaterial.Enabled = False
    btnLoadMesh.Enabled = False

    txtName.ReadOnly = True
    chkImmobile.Enabled = False

    txtHeight.ReadOnly = True
    txtWidth.ReadOnly = True
    txtDepth.ReadOnly = True
    chkAspect.Enabled = False
End Sub

Private Sub DisableCommandsOnDelete()
    MenuItem6.Enabled = False     ' Delete File.
    MenuItem8.Enabled = False     ' Create Entity.
    MenuItem9.Enabled = False
    MenuItem10.Enabled = False
    MenuItem12.Enabled = False    ' Load M-Lib.
    MenuItem13.Enabled = False
    MenuItem15.Enabled = False

    btnDeleteFile.Enabled = False
    btnNewEntity.Enabled = False
    btnSaveEntity.Enabled = False
    btnDeleteEntity.Enabled = False

    btnLoadMatLib.Enabled = False
    btnSetMaterial.Enabled = False
    btnLoadMesh.Enabled = False

    txtName.ReadOnly = True
    chkImmobile.Enabled = False

    txtHeight.ReadOnly = True
    txtWidth.ReadOnly = True
    txtDepth.ReadOnly = True

```

```

chkAspect.Enabled = False
End Sub

```

```

Private Sub LoadMaterialXMLFile(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLoadMatLib.Click, MenuItem12.Click

```

```

Dim openFileDialog1 As New OpenFileDialog
Dim stFileName As String

openFileDialog1.InitialDirectory = "xml\"
openFileDialog1.Title = "Open Material Library"
openFileDialog1.Filter = "Material Library (*.mlb)|*.mlb"
openFileDialog1.FilterIndex = 1
openFileDialog1.RestoreDirectory = True

```

```

If openFileDialog1.ShowDialog() = DialogResult.OK Then
    ' Extract file strings from the dialog.
    stMatFilePathAndName = openFileDialog1.FileName
    Dim MyFile As FileInfo = New FileInfo(stMatFilePathAndName)
    stMatFileNameOnly = MyFile.Name ' Global data.
    txtMatFile.Text = MyFile.Name ' Display to screen.

    ' This loads the XML into xDoc.
    xDocMat.load(stMatFilePathAndName)

    ' This loads Node data into Nodes.
    MatNodes = xDocMat.documentElement.childNodes

    ' Loop through and display materials.
    RefreshMaterialListBox()

    ' Set the status bar.
    StatusBar1.Text = "Status: Entity library " + MyFile.Name _
        + " loaded successfully."

    ' Disable the 'set material' command.
    btnSetMaterial.Enabled = False
    MenuItem15.Enabled = False

```

```

End If
End Sub

```

```

Private Sub RecalcMaxID()
    Dim xNode As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute
    Dim ProperMax As Integer

    ProperMax = 0
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() > ProperMax) Then
            ProperMax = xNode.attributes.getNamedItem("ID").nodeValue()
        End If
    Next

    ' Update 'maxID'.
    newAttr = xDoc.createAttribute("maxID")
    newAttr.nodeValue = ProperMax
    xDoc.documentElement.attributes.setNamedItem(newAttr)
    ' Save XML data.
    xDoc.save(stFilePathAndName)
End Sub

```

```

Private Sub ClearEntityListSelections()
    For nIndex As Integer = 0 To (lstEntity.Items.Count() - 1)
        lstEntity.SetSelected(nIndex, False)
    Next nIndex
End Sub

```

```

Private Sub ClearMaterialListSelections()
    For nIndex As Integer = 0 To (lstMaterials.Items.Count() - 1)
        lstMaterials.SetSelected(nIndex, False)
    Next nIndex

```



```

End Sub

Private Function GetNodeFromID(ByVal id As String) As IXMLDOMNode
    Dim xNode As IXMLDOMNode
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = id) Then
            Return xNode
        End If
    Next
End Function

Private Sub DeleteNodeByID(ByVal nID As String)
    Dim xNode As IXMLDOMNode
    xNode = GetNodeFromID(nID)
    xDoc.documentElement.removeChild(xNode)
End Sub

Private Function GetEntityNameFromID(ByVal id As String) As String
    Dim xNode As IXMLDOMNode
    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = id) Then
            Return xNode.attributes.getNamedItem("name").nodeValue()
        End If
    Next
End Function

Private Function ParseNameFromString(ByVal aString As String) As String
    Dim nLength As Integer
    nLength = aString.IndexOf(" ")
    If (nLength < 0) Then
        ParseNameFromString = ""
        Exit Function
    End If
    ParseNameFromString = aString.Substring(0, nLength)
End Function

Private Function ParseIDFromString(ByVal aString As String) As String
    Dim nLength As Integer
    nLength = (aString.Length() - aString.IndexOf(" ")) - 1
    If (nLength < 0) Then
        ParseIDFromString = ""
        Exit Function
    End If
    ParseIDFromString = aString.Substring((aString.IndexOf(" ") + 1), nLength)
End Function

Private Sub NewEntity(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewEntity.Click
    ' Reset this value to its proper #.
    RecalcMaxID()

    ' Clear the form.
    txtName.Text = ""
    txtMat.Text = ""
    txtMatLib.Text = ""
    txtID.Text = (xDoc.documentElement.attributes.getNamedItem( _
        "maxID").nodeValue() + 1)
    txtXFile.Text = ""
    chkImmobile.Checked = False

    chkAspect.Checked = False
    txtHeight.Text = ""
    txtWidth.Text = ""
    txtDepth.Text = ""

    ' Enable the fields.
    txtName.ReadOnly = False
    chkImmobile.Enabled = True

    txtHeight.ReadOnly = True
    txtWidth.ReadOnly = True

```

```

txtDepth.ReadOnly = True
chkAspect.Enabled = False

' Set the focus & change commands available.
txtName.Focus()

btnNewEntity.Enabled = False
MenuItem8.Enabled = False

btnSaveEntity.Enabled = True
MenuItem9.Enabled = True

btnDeleteEntity.Enabled = False
MenuItem10.Enabled = False

' Only enable this command if there is a material library loaded.
If (txtMatFile.Text.Length > 0) Then
    btnSetMaterial.Enabled = True
    MenuItem15.Enabled = True
Else
    btnSetMaterial.Enabled = False
    MenuItem15.Enabled = False
End If

btnLoadMesh.Enabled = True
MenuItem13.Enabled = True

' Clear selections.
ClearEntityListSelections()

' Set the global creation flag to true.
bNewEntity = True

' Set the status bar.
StatusBar1.Text = "Status: EDGE ready for new entity input."
End Sub

Private Sub SaveEntity(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSaveEntity.Click
    Dim xNode, subNode, newNode As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute
    Dim mResult As MsgBoxResult
    Dim newElement As IXMLDOMElement
    Dim newText As IXMLDOMText

    ' Check for proper syntax.
    If (txtName.Text.Length = 0 Or txtID.Text.Length = 0 Or txtMat.Text.Length = 0 _
    Or txtMatLib.Text.Length = 0 Or txtXFile.Text.Length = 0) Then
        StatusBar1.Text = _
            "Error: One or more fields have been left blank. Entity not saved."
        Exit Sub
    End If
    If (txtDepth.Text.Length = 0 Or txtHeight.Text.Length = 0 _
    Or txtWidth.Text.Length = 0) Then
        StatusBar1.Text = "Error: Error involving mesh dimensions. _
            Please check values. Entity not saved."
        Exit Sub
    End If

    ' Save the data appropriately.
    If (bNewEntity) Then ' If bNewEntity, save as a new node.
        ' Check to see if the ID already exists.
        Dim bIDExists As Boolean
        For Each xNode In Nodes
            If (txtID.Text() = xNode.attributes.getNamedItem("ID").nodeValue()) Then
                mResult = MsgBox("Copy over old entity?", MsgBoxStyle.YesNo, _
                    "ID Already Exists")
                If (mResult = MsgBoxResult.Yes) Then
                    bIDExists = True
                    Exit For
                Else

```

```

        Exit Sub
    End If
End If
Next

' Delete old node.
If (bIDExists) Then
    DeleteNodeByID(txtID.Text)
End If

' Save data in a new node.
newNode = xDoc.createElement("entity")
' ID
newAttr = xDoc.createAttribute("ID")
newAttr.nodeValue = txtID.Text()
newNode.attributes.setNamedItem(newAttr)
' Name
newAttr = xDoc.createAttribute("name")
newAttr.nodeValue = txtName.Text()
newNode.attributes.setNamedItem(newAttr)
' Material Library
newElement = xDoc.createElement("mlib")
newText = xDoc.createTextNode(txtMatLib.Text)
newElement.appendChild(newText)
newNode.appendChild(newElement)
' Material ID
newElement = xDoc.createElement("mID")
newText = xDoc.createTextNode(txtMat.Text)
newElement.appendChild(newText)
newNode.appendChild(newElement)
' X-File
newElement = xDoc.createElement("xfile")
newText = xDoc.createTextNode(txtXFile.Text)
newElement.appendChild(newText)
newNode.appendChild(newElement)
' Immobile Flag
newElement = xDoc.createElement("immobile")
If (chkImmobile.Checked) Then
    newText = xDoc.createTextNode("1")
Else
    newText = xDoc.createTextNode("0")
End If
newElement.appendChild(newText)
newNode.appendChild(newElement)

' Size Variables
newElement = xDoc.createElement("size")
' Height
newAttr = xDoc.createAttribute("height")
newAttr.nodeValue = txtHeight.Text()
newElement.attributes.setNamedItem(newAttr)
' Width
newAttr = xDoc.createAttribute("width")
newAttr.nodeValue = txtWidth.Text()
newElement.attributes.setNamedItem(newAttr)
' Depth
newAttr = xDoc.createAttribute("depth")
newAttr.nodeValue = txtDepth.Text()
newElement.attributes.setNamedItem(newAttr)
' Aspect Ratio
newAttr = xDoc.createAttribute("keepratio")
If (chkAspect.Checked) Then
    newAttr.nodeValue = "1"
Else
    newAttr.nodeValue = "0"
End If
newElement.attributes.setNamedItem(newAttr)
' Add the size node.
newNode.appendChild(newElement)

' Add in the new information as a new node

```

```

xDoc.documentElement.appendChild(newNode)

' Update maxID.
newAttr = xDoc.createAttribute("maxID")
newAttr.nodeValue = _
    (xDoc.documentElement.attributes.getNamedItem("maxID").nodeValue() + 1)
xDoc.documentElement.attributes.setNamedItem(newAttr)

' Save xml to file.
xDoc.save(stFilePathAndName)

' Reset state variable and refresh the Entity List.
bNewEntity = False
RefreshEntityListBox()
SetEntityListBoxFocus(txtID.Text())

' Set the status bar.
StatusBar1.Text = "Status: New entity added successfully."
Else ' Otherwise, update old data.
For Each xNode In Nodes
    IF (txtID.Text = xNode.attributes.getNamedItem("ID").nodeValue()) Then
        ' Update 'Name'.
        newAttr = xDoc.createAttribute("name")
        newAttr.nodeValue = txtName.Text()
        xNode.attributes.setNamedItem(newAttr)
        ' Update the subnodes.
        For Each subNode In xNode.childNodes
            ' Update 'Material'.
            IF (subNode.nodeName = "mLib") Then
                subNode.firstChild.nodeValue = txtMatLib.Text
            End IF
            IF (subNode.nodeName = "mID") Then
                subNode.firstChild.nodeValue = txtMat.Text
            End IF
            ' Update 'X-File'.
            IF (subNode.nodeName = "xfile") Then
                subNode.firstChild.nodeValue = txtXFile.Text
            End IF
            ' Update 'Immobile'.
            IF (subNode.nodeName = "immobile") Then
                IF (chkImmobile.Checked) Then
                    subNode.firstChild.nodeValue = 1
                Else
                    subNode.firstChild.nodeValue = 0
                End IF
            End IF
            ' Update 'size' variables.
            IF (subNode.nodeName = "size") Then
                ' Height
                newAttr = xDoc.createAttribute("height")
                newAttr.nodeValue = txtHeight.Text()
                subNode.attributes.setNamedItem(newAttr)
                ' Width
                newAttr = xDoc.createAttribute("width")
                newAttr.nodeValue = txtWidth.Text()
                subNode.attributes.setNamedItem(newAttr)
                ' Depth
                newAttr = xDoc.createAttribute("depth")
                newAttr.nodeValue = txtDepth.Text()
                subNode.attributes.setNamedItem(newAttr)
                ' Original Height
                newAttr = xDoc.createAttribute("oheight")
                newAttr.nodeValue = oHeight.ToString()
                subNode.attributes.setNamedItem(newAttr)
                ' Original Width
                newAttr = xDoc.createAttribute("owidth")
                newAttr.nodeValue = oWidth.ToString()
                subNode.attributes.setNamedItem(newAttr)
                ' Original Depth
                newAttr = xDoc.createAttribute("odepth")
                newAttr.nodeValue = oDepth.ToString()
            End IF
        End For
    End IF
End For

```

```

        subNode.attributes.setNamedItem(newAttr)
        ' Aspect Ratio
        newAttr = xDoc.createAttribute("kepratio")
        If (chkAspect.Checked) Then
            newAttr.nodeValue = 1
        Else
            newAttr.nodeValue = 0
        End If
        subNode.attributes.setNamedItem(newAttr)
    End If
Next

' Save XML data.
xDoc.save(stFilePathAndName)
RefreshEntityListBox()
SetEntityListboxFocus(txtID.Text())

' Set the status bar.
StatusBar1.Text = "Status: Entity successfully updated."
Exit For
End If
' Set the status bar.
StatusBar1.Text = "Error: Update unsuccessful as the given _
ID matches no known existing node."
Next
End If
End Sub

Private Sub DeleteEntity(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDeleteEntity.Click
' Cancel delete if this is a new entity in progress.
' (should be impossible, but ...)
If (bNewEntity) Then
    StatusBar1.Text = "Status: Cannot delete an unsaved entity."
    Exit Sub
End If

' Confirm with user to delete the entity.
Dim dResult As MsgBoxResult
dResult = MsgBox("Delete currently selected Entity?", MsgBoxStyle.YesNo, _
"Entity Deletion")
If (dResult = MsgBoxResult.Yes) Then
    ' Delete the current node.
    DeleteNodeByID(txtID.Text())
    RefreshEntityListBox()
    xDoc.save(stFilePathAndName)
Else
    Exit Sub
End If

' Clear list of selections, clear form, change enables.
ClearAll()
RefreshEntityListBox()

MenuItem9.Enabled = False
MenuItem10.Enabled = False
MenuItem13.Enabled = False
MenuItem15.Enabled = False

btnSaveEntity.Enabled = False
btnDeleteEntity.Enabled = False

btnSetMaterial.Enabled = False
btnLoadMesh.Enabled = False

txtName.ReadOnly = True
txtHeight.ReadOnly = True
txtWidth.ReadOnly = True
txtDepth.ReadOnly = True
chkAspect.Enabled = False
chkImmobile.Enabled = False

```

```

' Set the status bar.
StatusBar1.Text = "Status: Entity successfully deleted."
End Sub

Public Sub OnMatSelect(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles lstMaterials.SelectedIndexChanged
    If (lstEntity.SelectedIndex > -1 Or bNewEntity) Then
        btnSetMaterial.Enabled = True
        MenuItem15.Enabled = True
    Else
        btnSetMaterial.Enabled = False
        MenuItem15.Enabled = False
    End If
End Sub

Public Sub OnEntitySelect(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles lstEntity.SelectedIndexChanged
    Dim xNode, subNode As IXMLDOMNode
    Dim idString As String
    Dim mResult As MsgBoxResult

    If ((bNewEntity = True)) Then
        If (lstEntity.SelectedIndices.Count() = 1) Then
            mResult = MsgBox("Discard new entity?", MsgBoxStyle.YesNo, _
                "Discard new entity?")
            If (mResult = MsgBoxResult.Yes) Then
                ' Continue.
            Else
                ClearEntityListSelections()
                Exit Sub
            End If
        Else
            Exit Sub
        End If
    End If

    bNewEntity = 0 ' Reset this global flag.

    If (lstEntity.SelectedIndices.Count() = 0) Then
        Exit Sub
    End If

    idString = ParseIDFromString(lstEntity.SelectedItem())

    For Each xNode In Nodes
        If (xNode.attributes.getNamedItem("ID").nodeValue() = idString) Then
            ' Set Name and ID fields.
            txtName.Text = xNode.attributes.getNamedItem("name").nodeValue()
            txtID.Text = xNode.attributes.getNamedItem("ID").nodeValue()

            For Each subNode In xNode.childNodes
                ' Set Material Library field.
                If (subNode.nodeName = "mLib") Then
                    txtMatLib.Text = subNode.text()
                End If
                ' Set Material ID field.
                If (subNode.nodeName = "mID") Then
                    txtMat.Text = subNode.text()
                End If
                ' Set .X Filename field.
                If (subNode.nodeName = "xfile") Then
                    txtXFile.Text = subNode.text()
                End If
                ' Set Immobile checkbox.
                If (subNode.nodeName = "immobile") Then
                    If (subNode.text()) Then
                        chkImmobile.Checked = True
                    Else
                        chkImmobile.Checked = False
                    End If
                End If
            End For
        End If
    End For

```

```

End If
' Set size values.
If (subNode.nodeName = "size") Then
    ' Used to prevent recalculations on initial setting of values.
    bFreezeAdjustAll = True
    LoadMeshValues(txtXFile.Text)

    txtHeight.Text = _
        subNode.attributes.getNamedItem("height").nodeValue()
    txtWidth.Text = _
        subNode.attributes.getNamedItem("width").nodeValue()
    txtDepth.Text = _
        subNode.attributes.getNamedItem("depth").nodeValue()
    bFreezeAdjustAll = False

    'SetOriginalDimensionsForMesh() ' Used for resizing.
    If (subNode.attributes.getNamedItem("keepratio").nodeValue())
    Then
        chkAspect.Checked = True
    Else
        chkAspect.Checked = False
    End If
End If
Next
Exit For ' Entity was found, exit the loop.
End If
Next

' Enable and disable commands.
txtName.ReadOnly = False
chkImmobile.Enabled = True
txtHeight.ReadOnly = False
txtWidth.ReadOnly = False
txtDepth.ReadOnly = False
chkAspect.Enabled = True

btnNewEntity.Enabled = True
MenuItem8.Enabled = True
btnSaveEntity.Enabled = True
MenuItem9.Enabled = True
btnDeleteEntity.Enabled = True
MenuItem10.Enabled = True

btnLoadMesh.Enabled = True
MenuItem13.Enabled = True

If (lstMaterials.SelectedIndex > -1) Then
    btnSetMaterial.Enabled = True
    MenuItem15.Enabled = True
Else
    btnSetMaterial.Enabled = False
    MenuItem15.Enabled = False
End If
End Sub

Private Sub LoadMesh(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnLoadMesh.Click
    Dim openFileDialog As New OpenFileDialog
    Dim stFilePath As String

    openFileDialog.InitialDirectory = "xmesh\"
    openFileDialog.Title = "Open .X Mesh"
    openFileDialog.Filter = "X-Mesh (*.x)|*.x"
    openFileDialog.FilterIndex = 1
    openFileDialog.RestoreDirectory = True

    If openFileDialog.ShowDialog() = DialogResult.OK Then
        ' Extract file strings from the dialog.
        stFilePath = openFileDialog.FileName
        Dim MyFile As FileInfo = New FileInfo(stFilePath)
    End If
End Sub

```

```

' Set the form data.
txtXFile.Text = MyFile.Name

' Calculate the mesh's bounding coordinates.
LoadMeshValues(MyFile.Name)

' Set the status bar.
StatusBar1.Text = "Status: X-Mesh loaded successfully."
Else
' Set the status bar.
StatusBar1.Text = "Status: X-Mesh not loaded."
End If
End Sub

Private Sub LoadMeshValues(ByVal stMesh As String)
' This is where the mesh is loaded.
Dim dxMaterials() As Direct3D.ExtendedMaterial

' Get the .x Mesh from file.
dxMesh = Direct3D.Mesh.FromFile("xmesh\" + stMesh, _
    MeshFlags.SystemMemory, dxDevice, dxMaterials)

' Set up a stream and lock it.
Dim dxStream As Microsoft.DirectX.GraphicsStream
Dim vb2 As Direct3D.VertexBuffer
vb2 = dxMesh.VertexBuffer
dxStream = vb2.Lock(0, 0, LockFlags.NoSystemLock)

' Calculate the mesh dimensions.
Dim v3min, v3max As Microsoft.DirectX.Vector3
Direct3D.Geometry.ComputeBoundingBox(dxStream, dxMesh.NumberVertices, _
    dxMesh.VertexFormat, v3min, v3max)

' Unlock the stream.
vb2.Unlock()
vb2.Dispose()

' Set the text box size properties to match the new mesh.

' Used to prevent recalculations on initial setting of values.
bFreezeAdjustAll = True
txtHeight.Text = CDb1(v3max.Y - v3min.Y).ToString("F")
txtWidth.Text = CDb1(v3max.X - v3min.X).ToString("F")
txtDepth.Text = CDb1(v3max.Z - v3min.Z).ToString("F")
bFreezeAdjustAll = False

chkAspect.Checked = True

' Stores the initial mesh values, for recalculation purposes.
SetOriginalDimensionsForMesh()
End Sub

Private Sub SetOriginalDimensionsForMesh()
' This sets global variables used when resizing meshes.
oDepth = txtDepth.Text
oHeight = txtHeight.Text
oWidth = txtWidth.Text
End Sub

Private Sub SetMaterial(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSetMaterial.Click
txtMat.Text = ParseIDFromString(lstMaterials.SelectedItem())
txtMatLib.Text = txtMatFile.Text
End Sub

Private Sub SetEntityListboxFocus(ByVal nID As String)
Dim mString As String = GetEntityNameFromID(nID) + " " + nID
lstEntity.SetSelected(lstEntity.Items.IndexOf(mString), True)
End Sub

```



```

Private Sub Height_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles txtHeight.TextChanged
    ' This check prevents calculations from occurring on initial load or
    ' while other fields are updating.
    If (bFreezeAdjust Or bFreezeAdjustAll) Then
        Exit Sub
    End If
    bFreezeAdjust = True

    Dim ratio As Double
    If (chkAspect.Checked) Then
        ' Check for 0 / blank values.
        If (txtHeight.Text.Length = 0) Then
            txtWidth.Text = CDb1(oWidth).ToString("F")
            txtDepth.Text = CDb1(oDepth).ToString("F")
        Else
            ' Adjust all boxes since the value is not null.
            ratio = CDb1(CDb1(txtHeight.Text) / oHeight)
            txtWidth.Text = CDb1(ratio * oWidth).ToString("F")
            txtDepth.Text = CDb1(ratio * oDepth).ToString("F")
        End If
    End If

    bFreezeAdjust = False
End Sub

```

```

Private Sub Width_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles txtWidth.TextChanged
    ' This check prevents calculations from occurring on initial load or
    ' while other fields are updating.
    If (bFreezeAdjust Or bFreezeAdjustAll) Then
        Exit Sub
    End If
    bFreezeAdjust = True

    Dim ratio As Double
    If (chkAspect.Checked) Then
        ' Check for 0 / blank values.
        If (txtWidth.Text.Length = 0) Then
            txtHeight.Text = CDb1(oHeight).ToString("F")
            txtDepth.Text = CDb1(oDepth).ToString("F")
        Else
            ' Adjust all boxes since the value is not null.
            ratio = CDb1(CDb1(txtWidth.Text / oWidth)).ToString("F")
            txtHeight.Text = CDb1(ratio * oHeight).ToString("F")
            txtDepth.Text = CDb1(ratio * oDepth).ToString("F")
        End If
    End If

    bFreezeAdjust = False
End Sub

```

```

Private Sub Depth_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles txtDepth.TextChanged
    ' This check prevents calculations from occurring on initial load
    ' or while other fields are updating.
    If (bFreezeAdjust Or bFreezeAdjustAll) Then
        Exit Sub
    End If
    bFreezeAdjust = True

    Dim ratio As Double
    If (chkAspect.Checked) Then
        ' Check for 0 / blank values.
        If (txtDepth.Text.Length = 0) Then
            txtWidth.Text = CDb1(oWidth).ToString("F")
            txtHeight.Text = CDb1(oHeight).ToString("F")
        Else
            ' Adjust all boxes since the value is not null.
            ratio = CDb1(CDb1(txtDepth.Text / oDepth)).ToString("F")
            txtWidth.Text = CDb1(ratio * oWidth).ToString("F")

```

```
        txtHeight.Text = CDb1(ratio * oHeight).ToString("F")
    End If
End If

    bFreezeAdjust = False
End Sub

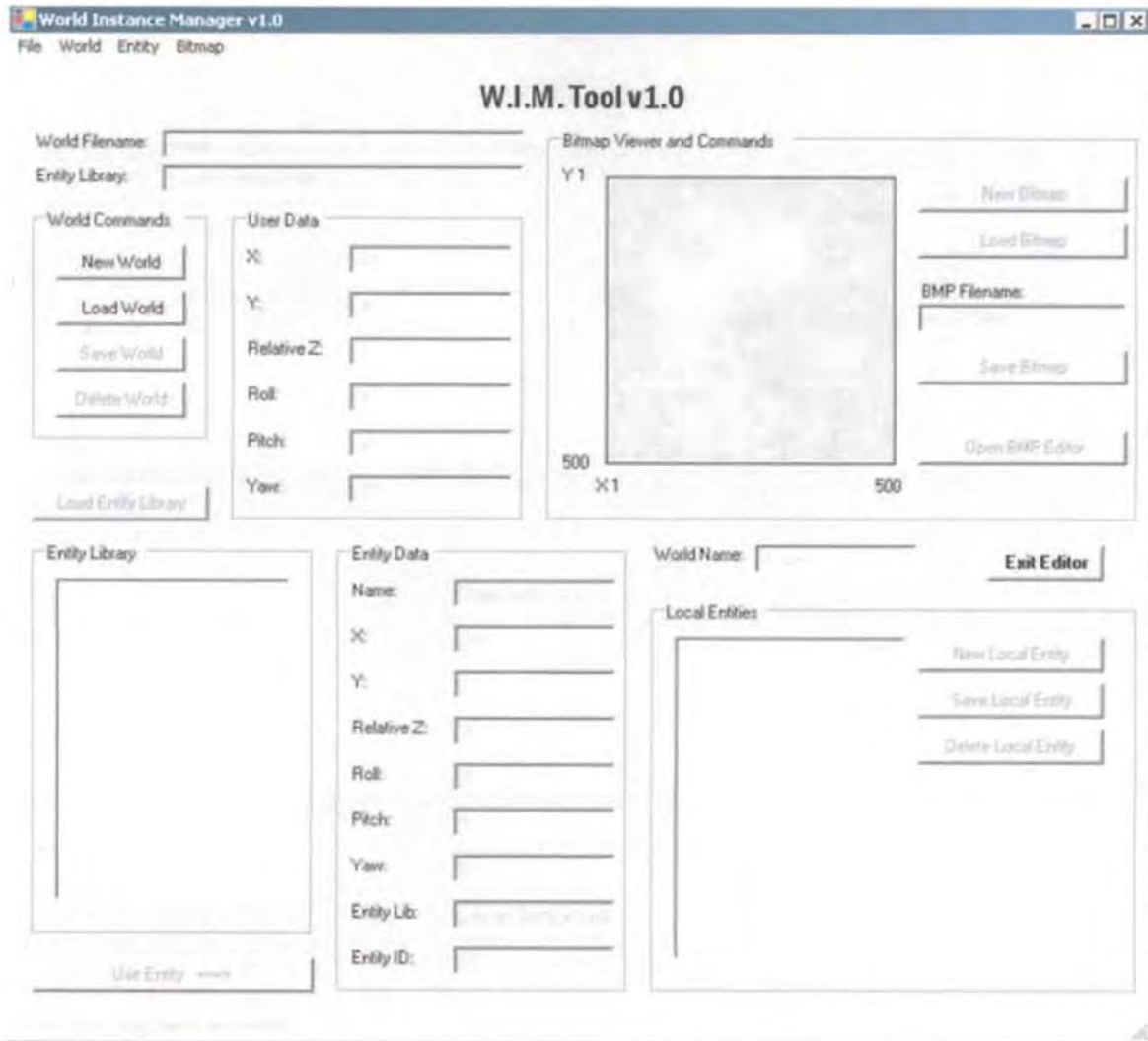
End Class
```


Section 6

W.I.M. Tool Code and Screenshots

W.I.M. Tool Screenshot

(1 of 3)



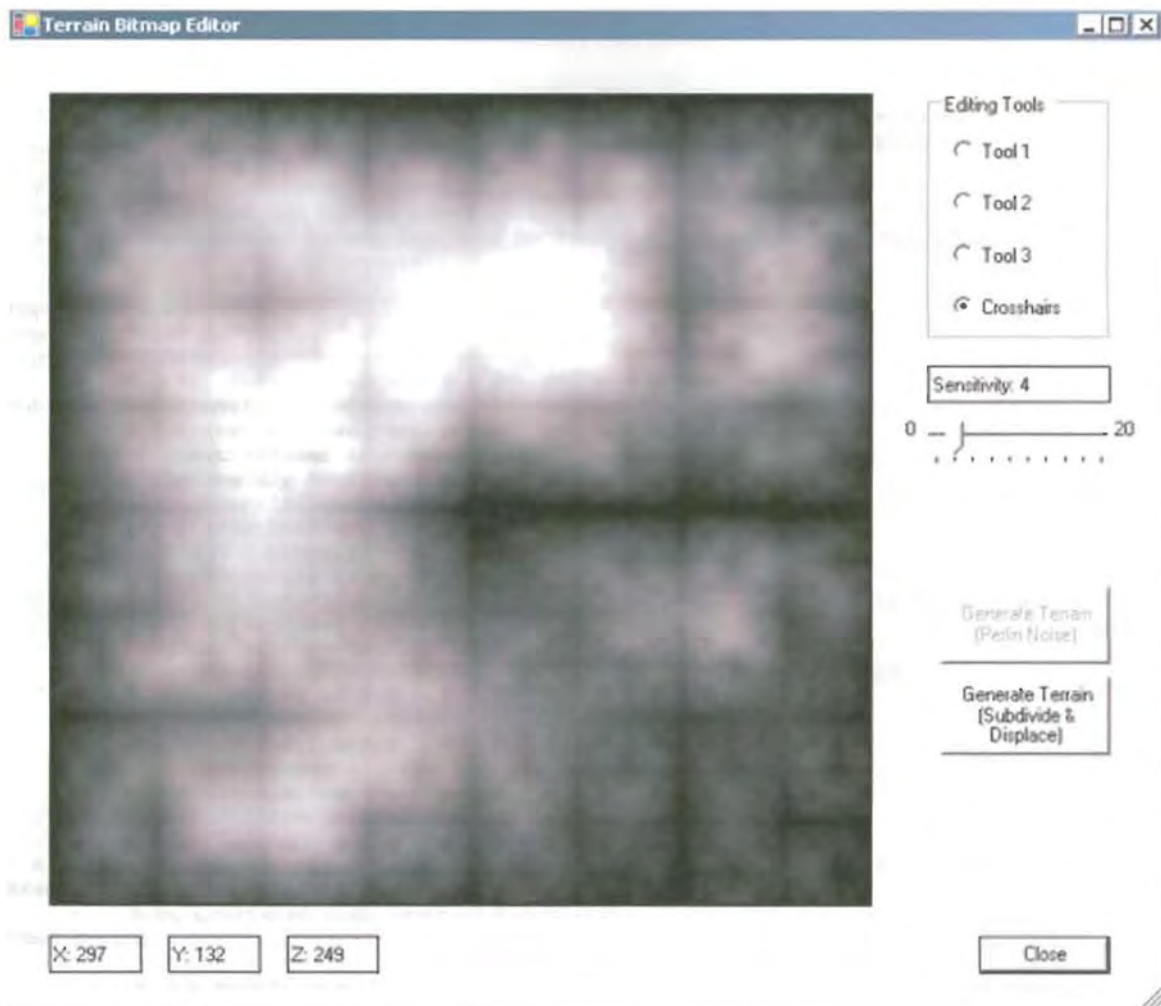
W.I.M. Tool Screenshot

(2 of 3)



W.I.M. Tool Screenshot

(3 of 3)




```

-----
' File: frmMain.vb
'
' Desc: This is the primary form for the WIM Tool.
'
' First created on:    February 11th, 2005
' Last modification:  February 28th, 2005
'
' Copyright (c) Jason M. Black (donblas@donblas.org)
-----
' Revision History:
'
' 02-11-05: File created. Load, New, Delete commands for World files added.
' 02-15-05: All functions added except the Bitmap ones (Entity manipulations
'           and entity library loading).
' 02-21-05: New/Load Bitmap functions added.
' 02-25-05: Bitmaps can now be saved to file!
' 02-28-05: Cleaned up bitmap instance handling. Can now save over old files.
-----

```

```

Imports System
Imports System.IO
Imports MSXML2

```

```

Public Class frmMain
    Inherits System.Windows.Forms.Form
    Dim stFilePathAndName As String
    Dim stFileNameOnly As String
    ' Entity Library filename strings.
    Dim stEntFilePathAndName, stEntFileNameOnly As String
    ' The World XML Library, and the Entity XML Library.
    Dim xDoc, xDocEntity As New XmlDocument
    ' The nodes in the World XML Library, and the nodes in the Entity XML Library.
    Dim Nodes, EntityNodes, LocalEntityNodes As IXMLDOMNodeList
    Dim UserNode, BitmapNode As IXMLDOMNode
    ' FALSE is normal, TRUE indicates a new entity process is engaged.
    Dim bNewLocalEntity As Boolean

    Dim F2 As New frmFilename
    Dim F3 As frmBitmap
    Public Shared objBitmap As Bitmap
    Public Shared GBitmapFilename As String

    ' All code in this "Region" was created by VS.NET as the graphical form was assembled.
    #Region " Windows Form Designer generated code "
    ' Auto-generated code removed for clarity.
    #End Region

    Private Sub ExitProgram(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnExit.Click, MenuItem2.Click
        Close()
    End Sub

    Private Sub LoadXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnLoadWorld.Click, MenuItem13.Click

        Dim openFileDialog1 As New OpenFileDialog
        Dim stFileName As String
        Dim xNode As IXMLDOMNode

        openFileDialog1.InitialDirectory = "xml\"
        openFileDialog1.Title = "Open World File"
        openFileDialog1.Filter = "World Instance File (*.wid)|*.wid"
        openFileDialog1.FilterIndex = 1
        openFileDialog1.RestoreDirectory = True

        If openFileDialog1.ShowDialog() = DialogResult.OK Then
            ' Extract file strings from the dialog.
            stFilePathAndName = openFileDialog1.FileName
            Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
            stFileNameOnly = MyFile.Name 'Global data.

```

```

txtWorldFile.Text = MyFile.Name ' Display to Screen.

' This loads the XML into xDoc.
xDoc.load(stFilePathAndName)

' This loads Node data into Nodes.
Nodes = xDoc.documentElement.childNodes
For Each xNode In Nodes
    If (xNode.nodeName = "bitmap") Then
        BitmapNode = xNode
    End If
    If (xNode.nodeName = "user") Then
        UserNode = xNode
    End If
    If (xNode.nodeName = "locals") Then
        LocalEntityNodes = xNode.childNodes
    End If
Next

' Clear out any old data.
ClearAll()
' Display all world data present.
DisplayWorldData()
' Enable 'Delete World', disable most other commands.
ChangeObjectsAfterLoad()

' Set the status bar.
StatusBar1.Text = "Status: World file loaded successfully."
End If
End Sub

Private Sub NewXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewWorld.Click, MenuItem12.Click

    Dim saveFileDialog1 As New SaveFileDialog
    Dim xNode As IXMLDOMNode

    saveFileDialog1.InitialDirectory = "xml\"
    saveFileDialog1.Filter = "World Instance File (*.wid)|*.wid"
    saveFileDialog1.FilterIndex = 1
    saveFileDialog1.RestoreDirectory = True

    If saveFileDialog1.ShowDialog() = DialogResult.OK Then
        ' Set up streams for writing.
        Dim filename As String = saveFileDialog1.FileName
        Dim myFileStream As New System.IO.FileStream(filename, _
            System.IO.FileMode.OpenOrCreate, System.IO.FileAccess.Write, _
            System.IO.FileShare.None)
        Dim XMLWriter As New System.IO.StreamWriter(myFileStream)

        ' Write XML data to file.
        XMLWriter.WriteLine("<?xml version=""1.0"" encoding=""UTF-8""?>")
        XMLWriter.WriteLine("<world name="""">")
        XMLWriter.WriteLine("  <bitmap filename=""""/>")
        XMLWriter.WriteLine("  <user x="""" y="""" z="""" roll="""" _
            pitch="""" yaw=""""/>")
        XMLWriter.WriteLine("  <locals>")
        XMLWriter.WriteLine("  </locals>")
        XMLWriter.WriteLine("</world>")

        ' Close the streams.
        XMLWriter.Close()
        myFileStream.Close()

        ' Load this new, empty file into the program.
        stFilePathAndName = saveFileDialog1.FileName
        Dim MyFile As FileInfo = New FileInfo(stFilePathAndName)
        stFileNameOnly = MyFile.Name 'Global data.
        txtWorldFile.Text = MyFile.Name ' Display to Screen.

        ' This loads the XML into xDoc.

```

```

xDoc.load(stFilePathAndName)

' This loads Node data into Nodes.
Nodes = xDoc.documentElement.childNodes
For Each xNode In Nodes
    If (xNode.nodeName = "bitmap") Then
        BitmapNode = xNode
    End If
    If (xNode.nodeName = "user") Then
        UserNode = xNode
    End If
    If (xNode.nodeName = "locals") Then
        LocalEntityNodes = xNode.childNodes
    End If
Next

' Clear all fields.
ClearAll()
' Display all world data present.
DisplayWorldData()
' Enable 'Delete World', disable most other commands.
ChangeObjectsAfterLoad()

' Set the status bar.
StatusBar1.Text = "Status: New file created and loaded successfully."
End If

End Sub

Private Sub DeleteXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnDeleteWorld.Click, MenuItem15.Click
    If (stFilePathAndName <> "") Then
        If (MsgBox("Delete " + stFileNameOnly + "?", MsgBoxStyle.OKCancel, _
            "Delete Confirmation") = MsgBoxResult.OK) Then
            ' Delete the file and clear the appropriate field.
            Kill(stFilePathAndName)
            txtWorldFile.Text = ""

            ' Set the status bar.
            StatusBar1.Text = "Status: " + stFileNameOnly _
                + " has been successfully deleted."

            ' Clear the editor and disable all commands that can't be used.
            ClearAll()
            DisableCommandsOnWorldDelete()
        End If
    End If
End Sub

Private Sub ClearAll()
    txtEntityLibFile.Text = ""

    txtUserX.Text = ""
    txtUserY.Text = ""
    txtUserRelZ.Text = ""
    txtUserRoll.Text = ""
    txtUserPitch.Text = ""
    txtUserYaw.Text = ""

    ClearLocalEntity()

    txtBitmapFilename.Text = ""
    txtWorldName.Text = ""

    lstEntity.Items.Clear()
    lstLocal.Items.Clear()

    If Not (boxThumb.Image Is Nothing) Then
        boxThumb.Image.Dispose()
        boxThumb.Image = Nothing
    End If

```

```

    txtWorldName.Text = ""
End Sub

Private Sub ClearLocalEntity()
    txtEntityName.Text = ""
    txtEntityX.Text = ""
    txtEntityY.Text = ""
    txtEntityRelZ.Text = ""
    txtEntityRoll.Text = ""
    txtEntityPitch.Text = ""
    txtEntityYaw.Text = ""
    txtEntityLibRef.Text = ""
    txtEntityIDRef.Text = ""
End Sub

Private Sub DisplayWorldData()
    Dim xNode As IXMLDOMNode
    Dim sLocal As String

    ' The name of the world.
    txtWorldName.Text = _
        xmlDoc.documentElement.attributes.getNamedItem("name").nodeValue()

    ' User data.
    txtUserX.Text = UserNode.attributes.getNamedItem("x").nodeValue()
    txtUserY.Text = UserNode.attributes.getNamedItem("y").nodeValue()
    txtUserRelZ.Text = UserNode.attributes.getNamedItem("z").nodeValue()
    txtUserRoll.Text = UserNode.attributes.getNamedItem("roll").nodeValue()
    txtUserPitch.Text = UserNode.attributes.getNamedItem("pitch").nodeValue()
    txtUserYaw.Text = UserNode.attributes.getNamedItem("yaw").nodeValue()

    ' Bitmap path.
    txtBitmapFilename.Text = _
        BitmapNode.attributes.getNamedItem("filename").nodeValue()
    GBitmapFilename = txtBitmapFilename.Text
    If Not (BitmapNode.attributes.getNamedItem("filename").nodeValue() = "") Then
        objBitmap = Image.FromFile("terrain\"
            + BitmapNode.attributes.getNamedItem("filename").nodeValue())
        boxThumb.Image = objBitmap
    End If

    ' Display the local entity list.
    For Each xNode In LocalEntityNodes
        sLocal = xNode.attributes.getNamedItem("name").nodeValue() + " ("
            + xNode.attributes.getNamedItem("x").nodeValue() + ", "
            + xNode.attributes.getNamedItem("y").nodeValue() + ")"
        lstLocal.Items.Add(sLocal)
    Next
End Sub

Private Sub ChangeObjectsAfterLoad()
    btnSaveWorld.Enabled = True
    btnDeleteWorld.Enabled = True
    MenuItem14.Enabled = True
    MenuItem15.Enabled = True

    btnLoadEntityLib.Enabled = True
    MenuItem9.Enabled = True

    btnNewLocalEntity.Enabled = True
    MenuItem16.Enabled = True

    btnNewBitmap.Enabled = True
    btnLoadBitmap.Enabled = True
    btnSaveBitmap.Enabled = True
    MenuItem6.Enabled = True
    MenuItem7.Enabled = True
    MenuItem18.Enabled = True
    If Not (txtBitmapFilename.Text = "") Then

```

```

        btnOpenBitmapEditor.Enabled = True
        MenuItem8.Enabled = True
    End If

    txtUserX.ReadOnly = False
    txtUserY.ReadOnly = False
    txtUserRelZ.ReadOnly = False
    txtUserRoll.ReadOnly = False
    txtUserPitch.ReadOnly = False
    txtUserYaw.ReadOnly = False

    txtWorldName.ReadOnly = False

End Sub

Private Sub DisableCommandsOnWorldDelete()
    txtWorldName.ReadOnly = True

    txtUserX.ReadOnly = True
    txtUserY.ReadOnly = True
    txtUserRelZ.ReadOnly = True
    txtUserRoll.ReadOnly = True
    txtUserPitch.ReadOnly = True
    txtUserYaw.ReadOnly = True

    txtEntityName.ReadOnly = True
    txtEntityX.ReadOnly = True
    txtEntityY.ReadOnly = True
    txtEntityRelZ.ReadOnly = True
    txtEntityRoll.ReadOnly = True
    txtEntityPitch.ReadOnly = True
    txtEntityYaw.ReadOnly = True
    txtEntityLibRef.ReadOnly = True
    txtEntityIDRef.ReadOnly = True

    txtBitmapFilename.ReadOnly = True
    txtWorldName.ReadOnly = True

    btnSaveWorld.Enabled = False
    btnDeleteWorld.Enabled = False
    MenuItem14.Enabled = False
    MenuItem15.Enabled = False

    btnLoadEntityLib.Enabled = False
    MenuItem9.Enabled = False

    btnUseEntity.Enabled = False
    btnNewLocalEntity.Enabled = False
    btnSaveLocalEntity.Enabled = False
    btnDeleteLocalEntity.Enabled = False
    MenuItem10.Enabled = False
    MenuItem11.Enabled = False
    MenuItem16.Enabled = False
    MenuItem17.Enabled = False

    btnNewBitmap.Enabled = False
    btnLoadBitmap.Enabled = False
    btnOpenBitmapEditor.Enabled = False
    btnSaveBitmap.Enabled = False
    MenuItem18.Enabled = False
    MenuItem6.Enabled = False
    MenuItem7.Enabled = False
    MenuItem8.Enabled = False
End Sub

Private Sub SaveXMLFile(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSaveWorld.Click, MenuItem14.Click
    ' Saves User and Bitmap information. Local Entity data stored by separate functions.
    Dim xNode, newUser, newBitmap As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute

```

```

' Check for proper syntax.
If (txtUserX.Text.Length = 0 Or txtUserY.Text.Length = 0
Or txtUserRelZ.Text.Length = 0 Or txtUserRoll.Text.Length = 0
Or txtUserPitch.Text.Length = 0 Or txtUserYaw.Text.Length = 0
Or txtBitmapFilename.Text.Length = 0 Or txtWorldName.Text.Length = 0) Then
    StatusBar1.Text = "Error: One or more fields have been left blank.
    World not saved."
    Exit Sub
End If

' Save data to BitmapNode.
newBitmap = xDoc.createElement("bitmap")

newAttr = xDoc.createAttribute("filename")
newAttr.nodeValue = txtBitmapFilename.Text()
newBitmap.attributes.setNamedItem(newAttr)

' Save data to UserNode.
newUser = xDoc.createElement("user")

newAttr = xDoc.createAttribute("x")
newAttr.nodeValue = txtUserX.Text()
newUser.attributes.setNamedItem(newAttr)

newAttr = xDoc.createAttribute("y")
newAttr.nodeValue = txtUserY.Text()
newUser.attributes.setNamedItem(newAttr)

newAttr = xDoc.createAttribute("z")
newAttr.nodeValue = txtUserRelZ.Text()
newUser.attributes.setNamedItem(newAttr)

newAttr = xDoc.createAttribute("roll")
newAttr.nodeValue = txtUserRoll.Text()
newUser.attributes.setNamedItem(newAttr)

newAttr = xDoc.createAttribute("pitch")
newAttr.nodeValue = txtUserPitch.Text()
newUser.attributes.setNamedItem(newAttr)

newAttr = xDoc.createAttribute("yaw")
newAttr.nodeValue = txtUserYaw.Text()
newUser.attributes.setNamedItem(newAttr)

' Save world filename.
newAttr = xDoc.createAttribute("name")
newAttr.nodeValue = txtWorldName.Text
xDoc.documentElement.attributes.setNamedItem(newAttr)

' Append information into the document.
xDoc.documentElement.removeChild(BitmapNode)
xDoc.documentElement.removeChild(UserNode)

BitmapNode = newBitmap
UserNode = newUser

xDoc.documentElement.appendChild(BitmapNode)
xDoc.documentElement.appendChild(UserNode)

' Save xml to file.
xDoc.save(stFilePathAndName)

' Set the status bar.
StatusBar1.Text = "Status: World file saved."
End Sub

' -----
' Listbox functions.
' -----

```

```

Private Sub RefreshEntityListBox()
    Dim xNode As IXMLDOMNode
    Dim newEntity As String
    lstEntity.Items.Clear()
    For Each xNode In EntityNodes
        newEntity = xNode.attributes.getNamedItem("name").nodeValue() + " " _
            + xNode.attributes.getNamedItem("ID").nodeValue()
        lstEntity.Items.Add(newEntity)
    Next xNode
End Sub

Private Sub RefreshLocalEntityListBox()
    Dim xNode As IXMLDOMNode
    Dim newEntity As String
    lstLocal.Items.Clear()
    For Each xNode In LocalEntityNodes
        newEntity = xNode.attributes.getNamedItem("name").nodeValue() + " (" _
            + xNode.attributes.getNamedItem("x").nodeValue() + ", " _
            + xNode.attributes.getNamedItem("y").nodeValue() + ")"
        lstLocal.Items.Add(newEntity)
    Next xNode
End Sub

Private Sub ClearEntityListSelections()
    For nIndex As Integer = 0 To (lstEntity.Items.Count() - 1)
        lstEntity.SetSelected(nIndex, False)
    Next nIndex
End Sub

Private Sub ClearLocalEntityListSelections()
    For nIndex As Integer = 0 To (lstLocal.Items.Count() - 1)
        lstLocal.SetSelected(nIndex, False)
    Next nIndex
End Sub

Private Sub SetLocalEntityListboxFocus(ByVal coor As Coor)
    Dim mString As String = GetEntityNameFromCoor(coor) + " (" + coor.x.ToString() _
        + ", " + coor.y.ToString() + ")"
    lstLocal.SetSelected(lstLocal.Items.IndexOf(mString), True)
End Sub

'-----
' Local entity helper functions.
'-----

Private Function GetNodeFromCoor(ByVal coor As Coor) As IXMLDOMNode
    Dim xNode As IXMLDOMNode
    For Each xNode In LocalEntityNodes
        If ((xNode.attributes.getNamedItem("x").nodeValue() = coor.x) _
            And (xNode.attributes.getNamedItem("y").nodeValue() = coor.y)) Then
            Return xNode
        End If
    Next
End Function

Private Sub DeleteNodeByCoor(ByVal coor As Coor)
    Dim xNode, subNode As IXMLDOMNode
    xNode = GetNodeFromCoor(coor)
    For Each subNode In Nodes
        If (subNode.nodeName = "locals") Then
            subNode.removeChild(xNode)
        End If
    Next
End Sub

Private Function GetEntityNameFromCoor(ByVal coor As Coor) As String
    Dim xNode As IXMLDOMNode
    For Each xNode In LocalEntityNodes
        If ((xNode.attributes.getNamedItem("x").nodeValue() = coor.x) _
            And (xNode.attributes.getNamedItem("y").nodeValue() = coor.y)) Then
            Return xNode.attributes.getNamedItem("name").nodeValue()
        End If
    Next
End Function

```

```

        End If
    Next
End Function

Private Function ParseNameFromString(ByVal aString As String) As String
    Dim nLength As Integer
    nLength = aString.IndexOf(" ")
    If (nLength < 0) Then
        ParseNameFromString = ""
        Exit Function
    End If
    ParseNameFromString = aString.Substring(0, nLength)
End Function

Private Function ParseIDFromString(ByVal aString As String) As String
    Dim nLength As Integer
    nLength = (aString.Length() - aString.IndexOf(" ")) - 1
    If (nLength < 0) Then
        ParseIDFromString = ""
        Exit Function
    End If
    ParseIDFromString = aString.Substring((aString.IndexOf(" ") + 1), nLength)
End Function

Private Function ParseCoorFromString(ByVal aString As String) As Coor
    Dim sTemp As String
    Dim cTemp As New Coor

    ' If there is no string, return 0,0 as invalid.
    If (aString.IndexOf(" ") = -1) Then
        cTemp.x = 0
        cTemp.y = 0
        ParseCoorFromString = cTemp
        Exit Function
    End If

    ' Parse the string for the coordinates.
    sTemp = aString.Substring((aString.IndexOf("(") + 1), _
        (aString.Length() - aString.IndexOf(")")) - 2)
    cTemp.x = sTemp.Substring(0, sTemp.IndexOf(","))
    cTemp.y = sTemp.Substring(sTemp.IndexOf(",") + 2, _
        (sTemp.Length() - sTemp.IndexOf(",")) - 2)
    ParseCoorFromString = cTemp
End Function

'-----
' Entity Functions
'-----

Private Sub LoadEntityLibrary(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLoadEntityLib.Click, MenuItem9.Click
    Dim openFileDialog1 As New OpenFileDialog
    Dim stFileName As String

    openFileDialog1.InitialDirectory = "xml\"
    openFileDialog1.Title = "Open Entity Library"
    openFileDialog1.Filter = "Entity Library (*.elb)|*.elb"
    openFileDialog1.FilterIndex = 1
    openFileDialog1.RestoreDirectory = True

    If openFileDialog1.ShowDialog() = DialogResult.OK Then
        ' Extract file strings from the dialog.
        stEntFilePathAndName = openFileDialog1.FileName
        Dim MyFile As FileInfo = New FileInfo(stEntFilePathAndName)
        stEntFileNameOnly = MyFile.Name ' Global data.
        txtEntityLibFile.Text = MyFile.Name ' Display to Screen.

        ' This loads the XML into xDoc.
        xDocEntity.load(stEntFilePathAndName)

        ' This loads Node data into Nodes.

```



```

EntityNodes = xDocEntity.documentElement.childNodes

' Loop through and display materials.
RefreshEntityListBox()

' Set the status bar.
StatusBar1.Text = "Status: Entity library loaded successfully."

' Disable the 'use entity' command.
btnUseEntity.Enabled = False
MenuItem10.Enabled = False
End If
End Sub

Private Sub OnEntitySelect(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstEntity.SelectedIndexChanged
' Enable 'Use Entity' command.
If (lstLocal.SelectedIndex > -1 Or bNewLocalEntity) Then
    btnUseEntity.Enabled = True
    MenuItem10.Enabled = True
Else
    btnUseEntity.Enabled = False
    MenuItem10.Enabled = False
End If
End Sub

Private Sub UseEntity(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnUseEntity.Click
' Load Data into the forms.
txtEntityName.Text = ParseNameFromString(lstEntity.SelectedItem())
txtEntityLibRef.Text = txtEntityLibFile.Text
txtEntityIDRef.Text = ParseIDFromString(lstEntity.SelectedItem())
End Sub

Private Sub OnLocalEntitySelect(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles lstLocal.SelectedIndexChanged
Dim xNode As IXMLDOMNode
Dim cString As Coord
Dim mResult As MsgBoxResult

If ((bNewLocalEntity = True)) Then
    If (lstLocal.SelectedIndices.Count() = 1) Then
        mResult = MsgBox("Discard new entity?", MsgBoxStyle.YesNo, _
            "Discard new entity?")
        If (mResult = MsgBoxResult.Yes) Then
            ' Continue.
        Else
            ClearLocalEntityListSelections()
            Exit Sub
        End If
    Else
        Exit Sub
    End If
End If

bNewLocalEntity = 0 ' Reset this global flag.

If (lstLocal.SelectedIndices.Count() = 0) Then
    Exit Sub
End If

cString = ParseCoordFromString(lstLocal.SelectedItem())

For Each xNode In LocalEntityNodes
    If ((xNode.attributes.getNamedItem("x").nodeValue() = cString.x) _
        And (xNode.attributes.getNamedItem("y").nodeValue() = cString.y)) Then
        ' Set attributes to text fields.
        txtEntityName.Text = xNode.attributes.getNamedItem("name").nodeValue()
        txtEntityX.Text = xNode.attributes.getNamedItem("x").nodeValue()
        txtEntityY.Text = xNode.attributes.getNamedItem("y").nodeValue()
        txtEntityRelZ.Text = xNode.attributes.getNamedItem("z").nodeValue()
    End If
Next

```

```

        txtEntityRoll.Text = xNode.attributes.getNamedItem("roll").nodeValue()
        txtEntityPitch.Text = xNode.attributes.getNamedItem("pitch").nodeValue()
        txtEntityYaw.Text = xNode.attributes.getNamedItem("yaw").nodeValue()
        txtEntityIDRef.Text = xNode.attributes.getNamedItem("eID").nodeValue()
        txtEntityLibRef.Text = xNode.attributes.getNamedItem("elib").nodeValue()
    Exit For ' Entity was found, exit the loop.
End If
Next

' Enable and disable commands.
txtEntityX.ReadOnly = False
txtEntityY.ReadOnly = False
txtEntityRelZ.ReadOnly = False
txtEntityRoll.ReadOnly = False
txtEntityPitch.ReadOnly = False
txtEntityYaw.ReadOnly = False

btnNewLocalEntity.Enabled = True
MenuItem16.Enabled = True
btnSaveLocalEntity.Enabled = True
MenuItem11.Enabled = True
btnDeleteLocalEntity.Enabled = True
MenuItem17.Enabled = True

If (lstEntity.SelectedIndex > -1) Then
    btnUseEntity.Enabled = True
    MenuItem10.Enabled = True
Else
    btnUseEntity.Enabled = False
    MenuItem10.Enabled = False
End If
End Sub

Private Sub NewLocalEntity(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewLocalEntity.Click, MenuItem16.Click
    ' Clear the form.
    ClearLocalEntity()

    ' Enable the fields.
    txtEntityX.ReadOnly = False
    txtEntityY.ReadOnly = False
    txtEntityRelZ.ReadOnly = False
    txtEntityRoll.ReadOnly = False
    txtEntityPitch.ReadOnly = False
    txtEntityYaw.ReadOnly = False

    ' Set the focus & change commands available.
    txtEntityX.Focus()

    btnNewLocalEntity.Enabled = False
    MenuItem16.Enabled = False

    btnSaveLocalEntity.Enabled = True
    MenuItem11.Enabled = True

    btnDeleteLocalEntity.Enabled = False
    MenuItem17.Enabled = False

    ' Enable this command if there is an entity selected in lstEntity.
    If (lstEntity.SelectedIndex > -1) Then
        btnUseEntity.Enabled = True
        MenuItem10.Enabled = True
    Else
        btnUseEntity.Enabled = False
        MenuItem10.Enabled = False
    End If

    ' Clear selections.
    ClearLocalEntityListSelections()

    ' Set the global creation flag to true.

```

```

    bNewLocalEntity = True
End Sub

Private Sub SaveLocalEntity(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSaveLocalEntity.Click, MenuItem11.Click
    Dim xNode, newNode As IXMLDOMNode
    Dim newAttr As IXMLDOMAttribute
    Dim mResult As MsgBoxResult
    Dim oldCoor As New Coor

    ' Check for proper syntax.
    If (txtEntityName.Text.Length = 0 Or txtEntityX.Text.Length = 0 _
Or txtEntityY.Text.Length = 0 Or txtEntityRelZ.Text.Length = 0 _
Or txtEntityRoll.Text.Length = 0 Or txtEntityPitch.Text.Length = 0 _
Or txtEntityYaw.Text.Length = 0 Or txtEntityIDRef.Text.Length = 0 _
Or txtEntityLibRef.Text.Length = 0) Then
        ' Set the status bar.
        StatusBar1.Text = "Error: One or more fields have been left blank. _
Local Entity not saved."
        Exit Sub
    End If

    ' Save the data appropriately.
    If (bNewLocalEntity) Then ' If bNewLocalEntity, save as a new node.
        ' Check to see if the location is already used.
        Dim bRemoveOccupant As Boolean
        For Each xNode In LocalEntityNodes
            If ((xNode.attributes.getNamedItem("x").nodeValue() = txtEntityX.Text) _
And (xNode.attributes.getNamedItem("y").nodeValue() = txtEntityY.Text)) _
Then
                mResult = MsgBox("Remove previous Local Entity at this location?", _
MsgBoxStyle.YesNo, "Coordinates Occupied")
                If (mResult = MsgBoxResult.Yes) Then
                    bRemoveOccupant = True
                    Exit For
                Else
                    Exit Sub
                End If
            End If
        Next

        oldCoor.x = txtEntityX.Text
        oldCoor.y = txtEntityY.Text

        ' Delete old node.
        If (bRemoveOccupant) Then
            DeleteNodeByCoor(oldCoor)
        End If

        ' Save data in a new node.
        newNode = xDoc.createElement("entity")
        ' Name
        newAttr = xDoc.createAttribute("name")
        newAttr.nodeValue = txtEntityName.Text()
        newNode.attributes.setNamedItem(newAttr)
        ' X, Y, Z
        newAttr = xDoc.createAttribute("x")
        newAttr.nodeValue = txtEntityX.Text()
        newNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("y")
        newAttr.nodeValue = txtEntityY.Text()
        newNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("z")
        newAttr.nodeValue = txtEntityRelZ.Text()
        newNode.attributes.setNamedItem(newAttr)
        ' Roll, Pitch, Yaw
        newAttr = xDoc.createAttribute("roll")
        newAttr.nodeValue = txtEntityRoll.Text()
        newNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("pitch")
        newAttr.nodeValue = txtEntityPitch.Text()

```

```

newNode.attributes.setNamedItem(newAttr)
newAttr = xDoc.createAttribute("yaw")
newAttr.nodeValue = txtEntityYaw.Text()
newNode.attributes.setNamedItem(newAttr)
' Material Reference ID
newAttr = xDoc.createAttribute("eID")
newAttr.nodeValue = txtEntityIDRef.Text()
newNode.attributes.setNamedItem(newAttr)
' Material Reference Library
newAttr = xDoc.createAttribute("elib")
newAttr.nodeValue = txtEntityLibRef.Text()
newNode.attributes.setNamedItem(newAttr)

' Add in the new information as a new node.
For Each xNode In Nodes
    If (xNode.nodeName = "locals") Then
        xNode.appendChild(newNode)
    Exit For
End If
Next

' Save XML to file.
xDoc.save(stFilePathAndName)

' Reset state variable and refresh the Local Entity List.
bNewLocalEntity = False
RefreshLocalEntityListBox()
SetLocalEntityListBoxFocus(oldCoor)

' Set the status bar.
StatusBar1.Text = "Status: New local entity added successfully."
Else ' Otherwise, update old data.
For Each xNode In LocalEntityNodes
    oldCoor = ParseCoorFromString(1stLocal.SelectedItem())
    If ((xNode.attributes.getNamedItem("x").nodeValue() = oldCoor.x) _
And (xNode.attributes.getNamedItem("y").nodeValue() = oldCoor.y)) Then
        ' Update fields ...
        ' Name
        newAttr = xDoc.createAttribute("name")
        newAttr.nodeValue = txtEntityName.Text()
        xNode.attributes.setNamedItem(newAttr)
        ' X, Y, Z
        newAttr = xDoc.createAttribute("x")
        newAttr.nodeValue = txtEntityX.Text()
        xNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("y")
        newAttr.nodeValue = txtEntityY.Text()
        xNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("z")
        newAttr.nodeValue = txtEntityRelZ.Text()
        xNode.attributes.setNamedItem(newAttr)
        ' Roll, Pitch, Yaw
        newAttr = xDoc.createAttribute("roll")
        newAttr.nodeValue = txtEntityRoll.Text()
        xNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("pitch")
        newAttr.nodeValue = txtEntityPitch.Text()
        xNode.attributes.setNamedItem(newAttr)
        newAttr = xDoc.createAttribute("yaw")
        newAttr.nodeValue = txtEntityYaw.Text()
        xNode.attributes.setNamedItem(newAttr)
        ' Material Reference ID
        newAttr = xDoc.createAttribute("eID")
        newAttr.nodeValue = txtEntityIDRef.Text()
        xNode.attributes.setNamedItem(newAttr)
        ' Material Reference Library
        newAttr = xDoc.createAttribute("elib")
        newAttr.nodeValue = txtEntityLibRef.Text()
        xNode.attributes.setNamedItem(newAttr)

        ' Save XML data

```

```

        xDoc.save(stFilePathAndName)
        RefreshLocalEntityListBox()
        oldCoor.x = txtEntityX.Text
        oldCoor.y = txtEntityY.Text
        SetLocalEntityListboxFocus(oldCoor)
        StatusBar1.Text = "Status: Local Entity successfully updated."
        Exit For
    End If
    ' Set the status bar:
    StatusBar1.Text = "Error: Update unsuccessful as the given _
        Coordinates match no known existing node."
Next
End If
End Sub

Private Sub DeleteLocalEntity(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDeleteLocalEntity.Click, MenuItem17.Click
    ' Cancel delete if this is a new local entity in progress (should be impossible ...)
    If (bNewLocalEntity) Then
        ' Set the status bar.
        StatusBar1.Text = "Status: Cannot delete an unsaved local entity."
        Exit Sub
    End If

    ' Confirm with user to delete the entity.
    Dim dResult As MsgBoxResult
    dResult = MsgBox("Delete currently selected Local Entity?", _
        MsgBoxStyle.YesNo, "Local Entity Deletion")
    If (dResult = MsgBoxResult.Yes) Then
        DeleteNodeByCoor(ParseCoorFromString(1stLocal.SelectedItem()))
        RefreshLocalEntityListBox()
        xDoc.save(stFilePathAndName)
        ' Set the status bar.
        StatusBar1.Text = "Status: Local entity successfully deleted."
    Else
        ' Set the status bar.
        StatusBar1.Text = "Status: Local entity deletion aborted."
        Exit Sub
    End If

    ' Clear list of selections, clear form, change enables.
    ClearLocalEntity()
    RefreshLocalEntityListBox()

    txtEntityX.ReadOnly = True
    txtEntityY.ReadOnly = True
    txtEntityRelZ.ReadOnly = True
    txtEntityRoll.ReadOnly = True
    txtEntityPitch.ReadOnly = True
    txtEntityYaw.ReadOnly = True

    btnNewLocalEntity.Enabled = True
    MenuItem16.Enabled = True
    btnSaveLocalEntity.Enabled = False
    MenuItem11.Enabled = False
    btnDeleteLocalEntity.Enabled = False
    MenuItem17.Enabled = False
End Sub

'-----
' Bitmap Functions
'-----

Private Sub NewBitmap(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnNewBitmap.Click
    Dim objGraphics As Graphics
    Dim sName As String

    ' Get the filename from the user.
    F2.ShowDialog(Me)

```

```

If (F2.GFilename.Text = "") Then
    StatusBar1.Text = "Status: Invalid name for new bitmap. Creation aborted."
    Exit Sub
End If
sName = F2.GFilename.Text + ".bmp"

' Create the graphics objects.
objBitmap = New Bitmap(500, 500, Imaging.PixelFormat.Format24bppRgb)
objGraphics = Graphics.FromImage(objBitmap)

' Draw on the bitmap via the graphics object.
objGraphics.FillRectangle(Brushes.Black, 1, 1, 500, 500)
' Set the image property of the picturebox to the bitmap.
boxThumb.Image = objBitmap

' Save the image to file.
txtBitmapFilename.Text = sName
GBitmapFilename = sName ' Global access.
objBitmap.Save("terrain\" + sName, System.Drawing.Imaging.ImageFormat.Bmp)

' Clean up resources.
objGraphics.Dispose()

btnOpenBitmapEditor.Enabled = True
MenuItem8.Enabled = True

' Set the status bar.
StatusBar1.Text = "Status: New bitmap created."
End Sub

Private Sub LoadBitmap(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnLoadBitmap.Click
    ' Check if the image is 500 x 500. then load.
    Dim openFileDialog1 As New OpenFileDialog

    openFileDialog1.InitialDirectory = "terrain\"
    openFileDialog1.Title = "Open Terrain Bitmap"
    openFileDialog1.Filter = "Bitmap Image (*.bmp)|*.bmp"
    openFileDialog1.FilterIndex = 1
    openFileDialog1.RestoreDirectory = True

    If openFileDialog1.ShowDialog() = DialogResult.OK Then
        ' Extract file strings from the dialog.
        Dim MyFile As FileInfo = New FileInfo(openFileDialog1.FileName)

        ' Check for valid image size.
        objBitmap = objBitmap.FromFile("terrain\" + MyFile.Name)
        If ((objBitmap.Size.Height() <> 500) Or (objBitmap.Size.Width() <> 500)) Then
            ' Set the status bar.
            StatusBar1.Text = "Status: Invalid bitmap for terrain"
            Exit Sub
        End If

        ' Display filename and image.
        txtBitmapFilename.Text = MyFile.Name
        GBitmapFilename = MyFile.Name ' Global access.
        boxThumb.Image = objBitmap

        btnOpenBitmapEditor.Enabled = True
        MenuItem8.Enabled = True

        ' Set the status bar.
        StatusBar1.Text = "Status: Terrain bitmap"
    End If
End Sub

Private Sub OpenBitmapEditor(ByVal sender As
System.EventArgs) Handles btnOpenBitmapEditor.C
    ' Check to see if a bitmap is loaded.
    If (txtBitmapFilename.Text = "") Then
        ' Set the status bar.

```

```

        StatusBar1.Text = "Status: Cannot open bitmap editor with no bitmap loaded."
    Exit Sub
End If

' Open bitmap editing form
F3 = New frmBitmap
F3.ShowDialog(Me)
End Sub

Private Sub SaveBitmap(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnSaveBitmap.Click, MenuItem18.Click
' Save the bitmap to file.

' Check for no bitmap.
If (txtBitmapFilename.Text = "") Then
    Exit Sub
End If

' Save to file.
Dim dlg As SaveFileDialog = New SaveFileDialog
dlg.Title = "Save BMP file"
dlg.InitialDirectory = "terrain\"
dlg.Filter = "bmp files (*.bmp)|*.bmp|All files (*.*)|*.*"
If dlg.ShowDialog = DialogResult.OK Then
    Dim MyFile As FileInfo = New FileInfo(dlg.FileName)
    If (MyFile.Name = txtBitmapFilename.Text) Then
        ' Create the graphics objects.
        Dim tmpBitmap As Bitmap = New Bitmap(500, 500, _
            Imaging.PixelFormat.Format24bppRgb)
        Dim tmpGraphics As Graphics = Graphics.FromImage(tmpBitmap)

        ' Copy the bitmap to the temporary space via the graphics object.
        tmpGraphics.DrawImage(objBitmap, 0, 0)

        ' Erase the old bitmap / picture box references.
        tmpGraphics.Dispose()
        boxThumb.Image.Dispose()
        objBitmap.Dispose()

        ' Copy the bitmap back over to objBitmap.
        objBitmap = New Bitmap(500, 500, Imaging.PixelFormat.Format24bppRgb)
        tmpGraphics = Graphics.FromImage(objBitmap)
        tmpGraphics.DrawImage(tmpBitmap, 0, 0)
        tmpGraphics.Dispose()
        tmpBitmap.Dispose()

        ' Save the original bitmap to file.
        objBitmap.Save(dlg.FileName, System.Drawing.Imaging.ImageFormat.Bmp)

        ' Set the image property of the picturebox to the bitmap.
        boxThumb.Image = objBitmap
    Else
        ' Save the original bitmap to file.
        objBitmap.Save(dlg.FileName, System.Drawing.Imaging.ImageFormat.Bmp)
        ' Set the image property of the picturebox to the bitmap.
        boxThumb.Image = objBitmap
    End If
    GBitmapFilename = MyFile.Name
    txtBitmapFilename.Text = MyFile.Name
End If

' Set the status bar.
StatusBar1.Text = "Status: Terrain bitmap saved successfully."
End Sub
End Class

Public Class Coord
    Public x As Integer
    Public y As Integer
End Class

```

```
'-----  
' File: filename.vb  
'  
' Desc: This is a filename entry form for the WIM Tool.  
'  
' First created on:    February 21st, 2005  
' Last modification:  February 21st, 2005  
'  
' Copyright (c) Jason M. Black (donblas@donblas.org)  
'-----  
' Revision History:  
'  
' 02-21-05: Form created.  
'-----  
  
Public Class frmFilename  
    Inherits System.Windows.Forms.Form  
  
    Public Shared GFilename As New TextBox  
  
' All code in this "Region" was created by VS.NET as the graphical form was assembled.  
#Region " Windows Form Designer generated code "  
    ' Auto-generated code removed for clarity.  
#End Region  
  
    Private Sub Accept(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles btnOkay.Click  
        GFilename.Text = txtFilename.Text  
        Me.Close()  
    End Sub  
  
    Private Sub Cancel(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles btnCancel.Click  
        GFilename.Text = ""  
        Me.Close()  
    End Sub  
  
End Class
```



```

'-----
' File: bitmap.vb
'
' Desc: This is the bitmap editing form for the WIM Tool.
'
' First created on:    February 21st, 2005
' Last modification:  March 12th, 2005
'
' Copyright (c) Jason M. Black (donblas@donblas.org)
'-----
' Revision History:
'
' 02-21-05: Form created.
' 02-22-05: Work on manipulating bitmaps continues!
' 02-25-05: Saving the bitmap is now done on the main form.
' 02-27-05: The mouse cursor tools now load and have proper hot spots.
'           There is now a Z label that reports the elevation of a pixel.
' 02-28-05: This form now works on multiple uses since it destructs on exit.
'           Added the CursorFactory from MSDN in order to allow colored mouse
'           cursors.
' 03-12-05: Added in the circle tool manipulations as well as a control for
'           their sensitivity. Perlin noise generation functions in, but not
'           working properly. Added in a 'Subdivide and Deviate' terrain
'           function, tends to look very fractaled.
'-----

```

```

Public Class frmBitmap
    Inherits System.Windows.Forms.Form
    Dim OffsetX, OffsetY As Integer
    Dim X, Y As Integer

    ' All code in this "Region" was created by VS.NET as the graphical form was assembled.
    #Region " Windows Form Designer generated code "
        ' Auto-generated code removed for clarity.
    #End Region

    Private Sub OnFormLoad(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles MyBase.Load, MyBase.Activated
        ' Copy bitmap to the picturebox from the primary form.
        boxTerrain.Image = frmMain.objBitmap

        Me.Cursor = System.Windows.Forms.Cursors.Default()
        RadioButton1.Checked = False
        RadioButton2.Checked = False
        RadioButton3.Checked = False
        RadioButton4.Checked = True
    End Sub

    Private Sub CloseBitmapEditor(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btnClose.Click
        boxTerrain.Dispose() ' This is good as long as it doesn't damage the Bitmap.

        Me.Dispose()
    End Sub

    Private Sub MouseMovesOverBitmap(ByVal sender As System.Object, ByVal e As
        System.Windows.Forms.MouseEventHandler) Handles boxTerrain.MouseMove
        ' Calculate the position on the terrain the mouse is hovering over.
        OffsetX = frmBitmap.ActiveForm.Left() + boxTerrain.Left() + 4
        OffsetY = frmBitmap.ActiveForm.Top() + boxTerrain.Top() + 23
        X = boxTerrain.MousePosition.X() - OffsetX
        Y = boxTerrain.MousePosition.Y() - OffsetY
        lblX.Text = "X: " + X.ToString()
        lblY.Text = "Y: " + Y.ToString()

        ' Calculate the elevation of the pixel being hovered over.
        ' Elevations are white = 255 units above base, black = 0 units above base.
        Dim s As System.Drawing.Color = frmMain.objBitmap.GetPixel(X - 1, Y - 1)
        lblZ.Text = "Z: " + s.R.ToString()

        If (e.Button = MouseButtons.Left Or e.Button = MouseButtons.Right) Then

```

```

        MouseClickOnBitmap(sender, e)
    End If
End Sub

Private Sub MouseEntersBitmap(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles boxTerrain.MouseEnter
    If (RadioButton1.Checked) Then
        Me.Cursor = CursorFactory.Create("cursors/1.cur") ' Size 7.
    ElseIf (RadioButton2.Checked) Then
        Me.Cursor = CursorFactory.Create("cursors/2.cur") ' Size 11.
    ElseIf (RadioButton3.Checked) Then
        Me.Cursor = CursorFactory.Create("cursors/3.cur") ' Size 15.
    Else
        ' Default mouse cursor.
        Me.Cursor = System.Windows.Forms.Cursors.Cross()
    End If
End Sub

Private Sub MouseExitsBitmap(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles boxTerrain.MouseLeave
    Me.Cursor = System.Windows.Forms.Cursors.Default()

    lblX.Text = "X:"
    lblY.Text = "Y:"
    lblZ.Text = "Z:"
End Sub

Private Sub MouseClickOnBitmap(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles boxTerrain.MouseDown
    ' Calculate what pixel the mouse was clicked on. X and Y range from 1 to 500.
    OffsetX = frmBitmap.ActiveForm.Left() + boxTerrain.Left() + 4
    OffsetY = frmBitmap.ActiveForm.Top() + boxTerrain.Top() + 23
    X = boxTerrain.MousePosition.X() - OffsetX
    Y = boxTerrain.MousePosition.Y() - OffsetY

    ' Transform the pixel that was clicked on.
    Dim bms As New BitmapManipStruct(frmMain.objBitmap)

    ' Last three arguments in these functions: Boolean for mouse button (false = left,
    ' true = right). First integer is the change in pixel value when the tool is used.
    ' Second integer is the range of the circle tool as a radius.
    If (RadioButton1.Checked And e.Button = MouseButton.Left) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, False, tbSensitivity.Value * 2, 2)
    ElseIf (RadioButton1.Checked And e.Button = MouseButton.Right) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, True, tbSensitivity.Value * 2, 2)
    ElseIf (RadioButton2.Checked And e.Button = MouseButton.Left) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, False, tbSensitivity.Value * 2, 4)
    ElseIf (RadioButton2.Checked And e.Button = MouseButton.Right) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, True, tbSensitivity.Value * 2, 4)
    ElseIf (RadioButton3.Checked And e.Button = MouseButton.Left) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, False, tbSensitivity.Value * 2, 6)
    ElseIf (RadioButton3.Checked And e.Button = MouseButton.Right) Then
        TFCircleTool(bms, frmMain.objBitmap.Width, frmMain.objBitmap.Height, X - 1, _
            Y - 1, True, tbSensitivity.Value * 2, 6)
    Else
        Exit Sub
    End If

    bms.Unlock()
    boxTerrain.Image = frmMain.objBitmap

    ' Do not save to file, this is done back on the main form.
End Sub

Private Sub ChangeSensitivity(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles tbSensitivity.Scroll

```

```

    lblSens.Text = "Sensitivity: " + (tbSensitivity.Value * 2).ToString()
End Sub

Private Sub PerlinNoise(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnPerlin.Click
    Dim dTemp As Double

    Dim bms As New BitmapManipStruct(frmMain.objBitmap)
    bms.Lock()

    For Y = 1 To 10
        For X = 1 To 10
            dTemp = PerlinNoise2D(CDbl(X), CDbl(Y))
            ' This allows us to scale our results to -1.0 to 1.0.
            While (System.Math.Abs(dTemp) > 1.0 Or Not (System.Math.Abs(dTemp) <> 0))
                dTemp = PerlinNoise2D(CDbl(X), CDbl(Y))
            End While
            ' Write noise pixel to file.
            TFWriteNoisePixel(bms, X, Y, dTemp)
        Next X
    Next Y

    bms.Unlock()
    boxTerrain.Image = frmMain.objBitmap
End Sub

Private Sub SubdivideDisplace(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSubDis.Click
    ' Our workspace will be 512 x 512 and we'll just crop the extra 12.
    Dim nSize As Short = 512
    ' Our plane will begin at an elevation of 128, where the range is 0 to 255.
    Dim nStartElevation As Short = 0

    Dim bms As New BitmapManipStruct(frmMain.objBitmap)
    bms.Lock()

    ' Begin the Subdivide and Displace algorithm.
    SDHelper(nSize, nStartElevation, nStartElevation, nStartElevation, _
        nStartElevation, nStartElevation, nStartElevation, nStartElevation, _
        nStartElevation, bms, 1, 512, 1, 512)

    bms.Unlock()
    boxTerrain.Image = frmMain.objBitmap
End Sub

Private Function SDHelper(ByVal nSize As Short, ByVal nLeft As Short, ByVal nRight As
Short, ByVal nTop As Short, ByVal nBottom As Short, ByVal nTL As Short, ByVal nTR As
Short, ByVal nBL As Short, ByVal nBR As Short, ByVal bms As BitmapManipStruct, ByVal xmin
As Short, ByVal xmax As Short, ByVal ymin As Short, ByVal ymax As Short)
    ' Randomize number generator.
    Dim RNG As New Random

    ' Calculate the center.
    Dim nCenter As Integer = (nLeft + nTop + nRight + nBottom) / 4

    ' Smallest box has been reached, so write 4 values to file.
    If (nSize = 2) Then
        ' Top-Left value.
        If (xmin <= 500 And ymin <= 500) Then ' This line ignores the extra cells.
            If Not (RNG.Next(0, 9)) Then ' Slightly adjust the pixel 10% of the time.
                If (RNG.Next(0, 1)) Then
                    TFWritePixel(bms, xmin, ymin, nCenter + 1)
                Else
                    TFWritePixel(bms, xmin, ymin, nCenter - 1)
                End If
            Else
                ' Write 'center' height 90% of the time.
                TFWritePixel(bms, xmin, ymin, nCenter)
            End If
        End If
    End If
End Function

```

```

' Bottom-Left value.
If (xMin <= 500 And yMax <= 500) Then
  If Not (RNG.Next(0, 9)) Then ' Slightly adjust the pixel 10% of the time.
    If (RNG.Next(0, 1)) Then
      TFWritePixel(bms, xMin, yMax, nCenter + 1)
    Else
      TFWritePixel(bms, xMin, yMax, nCenter - 1)
    End If
  Else ' Write 'center' height 90% of the time.
    TFWritePixel(bms, xMin, yMax, nCenter)
  End If
End If

' Top-Right value.
If (xMax <= 500 And yMin <= 500) Then
  If Not (RNG.Next(0, 9)) Then ' Slightly adjust the pixel 10% of the time.
    If (RNG.Next(0, 1)) Then
      TFWritePixel(bms, xMax, yMin, nCenter + 1)
    Else
      TFWritePixel(bms, xMax, yMin, nCenter - 1)
    End If
  Else ' Write 'center' height 90% of the time.
    TFWritePixel(bms, xMax, yMin, nCenter)
  End If
End If

' Bottom-Right value.
If (xMax <= 500 And yMax <= 500) Then
  If Not (RNG.Next(0, 9)) Then ' Slightly adjust the pixel 10% of the time.
    If (RNG.Next(0, 1)) Then
      TFWritePixel(bms, xMax, yMax, nCenter + 1)
    Else
      TFWritePixel(bms, xMax, yMax, nCenter - 1)
    End If
  Else ' Write 'center' height 90% of the time.
    TFWritePixel(bms, xMax, yMax, nCenter)
  End If
End If

' 4 pixels written, so exit this branch.
Exit Function
End If

' Calculate four new midpoints.
Dim nL, nR, nT, nB As Short

' Left midpoint.
If (RNG.Next(0, 1)) Then
  nL = ((nCenter + nLeft) / 2) - RNG.Next(0, nSize / 2)
Else
  nL = ((nCenter + nLeft) / 2) + RNG.Next(0, nSize / 2)
End If

' Right midpoint.
If (RNG.Next(0, 1)) Then
  nR = ((nCenter + nRight) / 2) - RNG.Next(0, nSize / 2)
Else
  nR = ((nCenter + nRight) / 2) + RNG.Next(0, nSize / 2)
End If

' Top midpoint.
If (RNG.Next(0, 1)) Then
  nT = ((nCenter + nTop) / 2) - RNG.Next(0, nSize / 2)
Else
  nT = ((nCenter + nTop) / 2) + RNG.Next(0, nSize / 2)
End If

' Bottom midpoint.
If (RNG.Next(0, 1)) Then
  nB = ((nCenter + nBottom) / 2) - RNG.Next(0, nSize / 2)
Else

```

```

        nB = ((nCenter + nBottom) / 2) + RNG.Next(0, nSize / 2)
    End If

    Dim xMid, yMid As Short
    xMid = (xMax + xMin - 1) / 2
    yMid = (yMax + yMin - 1) / 2

    ' Top Left.
    SDHelper(nSize / 2, ((nLeft + nTL) / 2), nT, ((nTop + nTL) / 2), nL, nTL, _
        nTop, nLeft, nCenter, bms, xMin, xMid, yMin, yMid)
    ' Top Right.
    SDHelper(nSize / 2, nT, ((nRight + nTR) / 2), ((nTop + nTR) / 2), nR, nTop, _
        nTR, nCenter, nRight, bms, xMid + 1, xMax, yMin, yMid)
    ' Bottom Left.
    SDHelper(nSize / 2, ((nLeft + nBL) / 2), nB, nL, ((nBottom + nBL) / 2), _
        nLeft, nCenter, nBL, nBottom, bms, xMin, xMid, yMid + 1, yMax)
    ' Bottom Right.
    SDHelper(nSize / 2, nB, ((nRight + nBR) / 2), nR, ((nBottom + nBR) / 2), _
        nCenter, nRight, nBottom, nBR, bms, xMid + 1, xMax, yMid + 1, yMax)
End Function
End Class

' The following class adapted from MSDN
' http://msdn.microsoft.com/library/default.asp _
' ?url=/archive/en-us/dnaraskdr/html/askgull1182003.asp

Public Class CursorFactory
    Private Declare Unicode Function LoadCursorFromFile Lib "user32.dll" Alias _
        "LoadCursorFromFileW" (ByVal filename As String) As IntPtr

    Public Shared Function Create(ByVal filename As String) As Cursor
        Dim hCursor As IntPtr
        Dim result As Cursor = Nothing

        Try
            hCursor = LoadCursorFromFile(filename)
            If Not IntPtr.Zero.Equals(hCursor) Then
                result = New Cursor(hCursor)
            Else
                ' Could not create a cursor.
                Throw New ApplicationException("Could not create cursor from file " _
                    & filename)
            End If
        Catch ex As Exception
            ' If I had an error log the command would go here.
        End Try

        Return result
    End Function
End Class

```

```

-----
' File: bitmap_manip.vb
'
' Desc: This class/function library is used to store bitmap manipulation
'       functions. I did this to clear up space in bitmap.vb.
'
' First created on:    February 22nd, 2005
' Last modification:  March 13th, 2005
'
' Copyright (c) Jason M. Black (donblas@donblas.org)
-----
' Revision History:
'
' 02-22-05: File created. BitmapManipStruct created.
' 03-12-05: Circle tool manipulations added.
-----

```

Module bitmap_manip

```

Public Class BitmapManipStruct ' Use only with 24-bit RGB bitmaps!
    Public BitmapBytes() As Byte
    Public nStride As Integer
    Dim TheBitmap As Bitmap
    Dim BitmapData As System.Drawing.Imaging.BitmapData
    Dim nTotalSize As Integer

    Public Sub New(ByVal bmpIn As Bitmap)
        TheBitmap = bmpIn
    End Sub

    Public Sub Lock()
        ' Lock the bitmap for writing.
        Dim rect As New Rectangle(0, 0, TheBitmap.Width, TheBitmap.Height)
        ' There are 256 shades of grey!
        BitmapData = TheBitmap.LockBits(rect, Imaging.ImageLockMode.ReadWrite, _
            Imaging.PixelFormat.Format24bppRgb)

        ' Allocate room for the data.
        nTotalSize = BitmapData.Stride * BitmapData.Height
        ReDim BitmapBytes(nTotalSize)
        nStride = BitmapData.Stride

        ' Copy the data into the BitmapBytes array.
        System.Runtime.InteropServices.Marshal.Copy(BitmapData.Scan0, BitmapBytes, _
            0, nTotalSize)
    End Sub

    Public Sub Unlock()
        ' Copy the data back into the original Bitmap.
        System.Runtime.InteropServices.Marshal.Copy(BitmapBytes, 0, _
            BitmapData.Scan0, nTotalSize)

        ' Unlock the bitmap from writing.
        TheBitmap.UnlockBits(BitmapData)

        ' Release allocated data.
        BitmapBytes = Nothing
        BitmapData = Nothing
    End Sub
End Class

' Inverts the color of an entire bitmap.
Public Sub TFInvertBitmap(ByVal bms As BitmapManipStruct, ByVal nWidth As Integer,
    ByVal nHeight As Integer)
    Dim nRow, nColumn, nPixel As Integer
    Dim pixel As Integer

    bms.Lock()

    pixel = 0
    For nRow = 0 To nHeight - 1

```

```

        For nColumn = 0 To nWidth - 1
            For nPixel = 0 To 2
                bms.BitmapBytes(pixel) = 255 - bms.BitmapBytes(pixel)
                pixel += 1
            Next nPixel
        Next nColumn
    Next nRow
End Sub

' Makes a pixel white.
Public Sub TFWhitePixel(ByVal bms As BitmapManipStruct, ByVal nWidth As Integer,
ByVal nHeight As Integer, ByVal X As Integer, ByVal Y As Integer)
    Dim n As Integer
    Dim pixel As Integer

    bms.Lock()

    pixel = (X * 3) + (Y * bms.nStride)
    For n = 0 To 2
        bms.BitmapBytes(pixel) = 255
        pixel += 1
    Next n
End Sub

' Used for writing perlin noise. Or could be used to write any specific pixel.
' Lock() not included.
Public Sub TFWriteNoisePixel(ByVal bms As BitmapManipStruct, ByVal X As Integer,
ByVal Y As Integer, ByVal dNoise As Double)
    Dim n As Integer
    Dim pixel, value As Integer

    value = System.Math.Floor((dNoise + 1.0) * 128)

    pixel = ((X - 1) * 3) + ((Y - 1) * bms.nStride)
    For n = 0 To 2
        bms.BitmapBytes(pixel) = value
        pixel += 1
    Next n
End Sub

' Used in the Subdivide and and Displace terrain generation method.
Public Sub TFWritePixel(ByVal bms As BitmapManipStruct, ByVal X As Integer, ByVal Y
As Integer, ByVal nValue As Integer)
    Dim n As Integer
    Dim pixel As Integer

    If (nValue > 255) Then
        nValue = 255
    End If

    If (nValue < 0) Then
        nValue = 0
    End If

    pixel = ((X - 1) * 3) + ((Y - 1) * bms.nStride)
    For n = 0 To 2
        bms.BitmapBytes(pixel) = nValue
        pixel += 1
    Next n
End Sub

' Controls the interaction between the mouse cursor and the bitmap.
Public Sub TFCircleTool(ByVal bms As BitmapManipStruct, ByVal nWidth As Integer,
ByVal nHeight As Integer, ByVal X As Integer, ByVal Y As Integer, ByVal bButton As
Boolean, ByVal nStrength As Integer, ByVal nRange As Integer)
    ' bButton: false if left, true is right.
    Dim nRow, nColumn, nPixel As Integer
    Dim pixel As Integer

    bms.Lock()

```

```

For nRow = Y - nRange To Y + nRange
  For nColumn = X - nRange To X + nRange
    ' Prevents altering memory outside the bitmap.
    If (nRow > 0 And nColumn > 0) Then

      pixel = (nColumn * 3) + (nRow * bms.nStride)
      For nPixel = 0 To 2
        If ((nColumn > X - nRange And nColumn < X + nRange) _
          Or (nRow > Y - nRange And nRow < Y + nRange)) Then
          If (bButton) Then ' Right click.
            ' Prevent bitmap byte error.
            If (bms.BitmapBytes(pixel) < 256 - nStrength) Then
              bms.BitmapBytes(pixel) = bms.BitmapBytes(pixel) _
                + nStrength
            Else
              bms.BitmapBytes(pixel) = 255
            End If
          Else ' Left click.
            ' Prevent bitmap byte error.
            If (bms.BitmapBytes(pixel) > -1 + nStrength) Then
              bms.BitmapBytes(pixel) = bms.BitmapBytes(pixel) _
                - nStrength
            Else
              bms.BitmapBytes(pixel) = 0
            End If
          End If
        Else
          If (bButton) Then ' Right click.
            ' Prevent bitmap byte error.
            If (bms.BitmapBytes(pixel) < 256 - (nStrength / 2)) Then
              bms.BitmapBytes(pixel) = bms.BitmapBytes(pixel) _
                + (nStrength / 2) ' Corners.
            Else
              bms.BitmapBytes(pixel) = 255
            End If
          Else ' Left click.
            ' Prevent bitmap byte error.
            If (bms.BitmapBytes(pixel) > -1 + (nStrength / 2)) Then
              bms.BitmapBytes(pixel) = bms.BitmapBytes(pixel) _
                - (nStrength / 2) ' Corners.
            Else
              bms.BitmapBytes(pixel) = 0
            End If
          End If
        End If
        pixel += 1
      Next nPixel
    End If
  Next nColumn
Next nRow
End Sub

End Module

```


Section 7

Interactive Terrain Simulator Code and Screenshots







Thank you for using:

I.T.S.

The Interactive Terrain Simulator

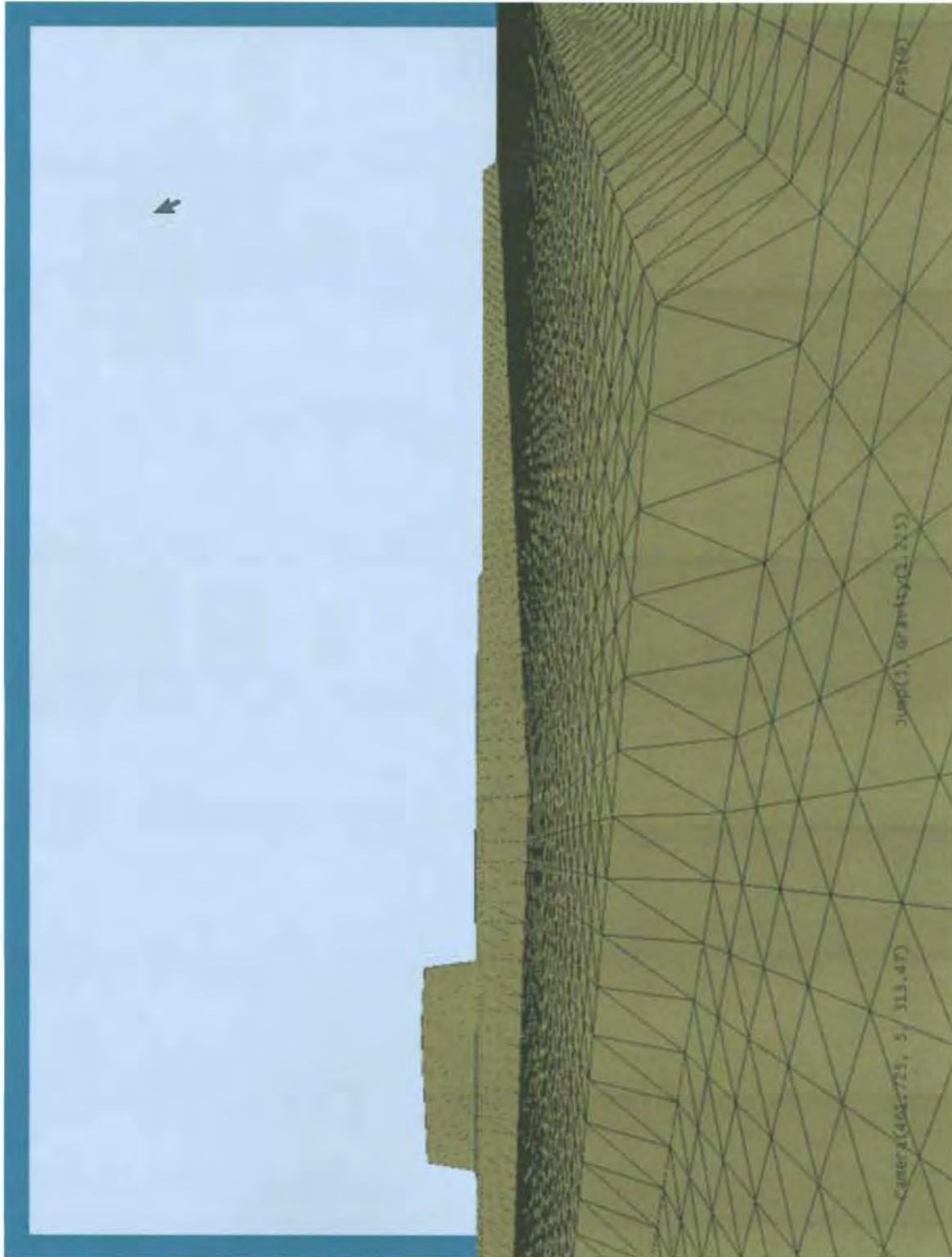
Credits

- Dr. Eileen Fésaro, for her advising for the year and a half from this project's conception to its conclusion.
- Jeremy Lotkaer, for many late evenings of discussion on the project's structure and countless tutorials concerning XML.
- Nicole Gugliucci, for encouragement, motivation, and being a muse.
- Char and Karen Black, for their support of the project's author during the journey.
- The Honors Committee, for taking the time to help with and review this project.
- Microsoft, for Visual Studio, DirectX, and the ever confusing yet useful MSDN knowledge base.
- Google, for their helpful search engine.

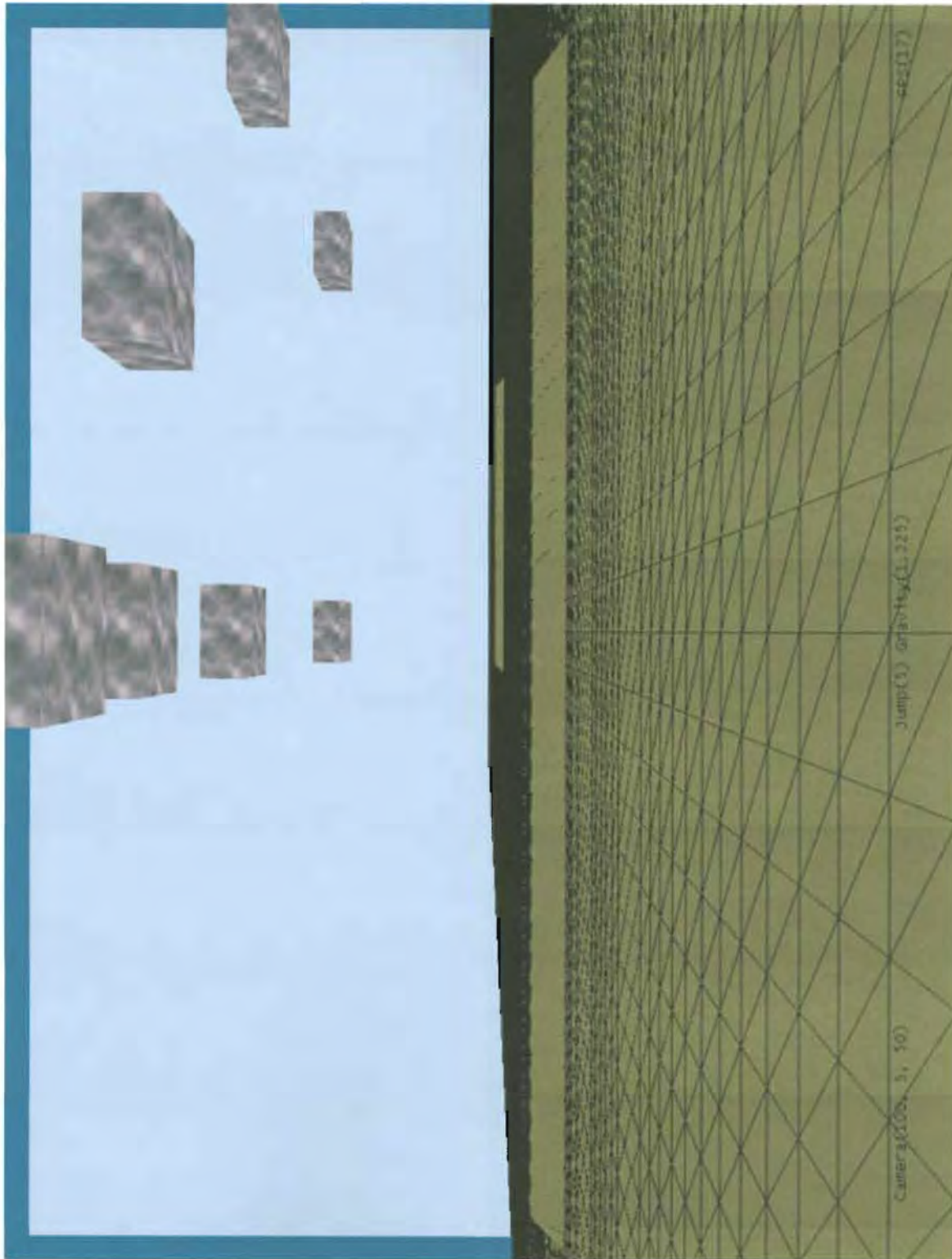
I.T.S. Screenshot

(5 of 14)

Below: This is a generic scene in a simulation.



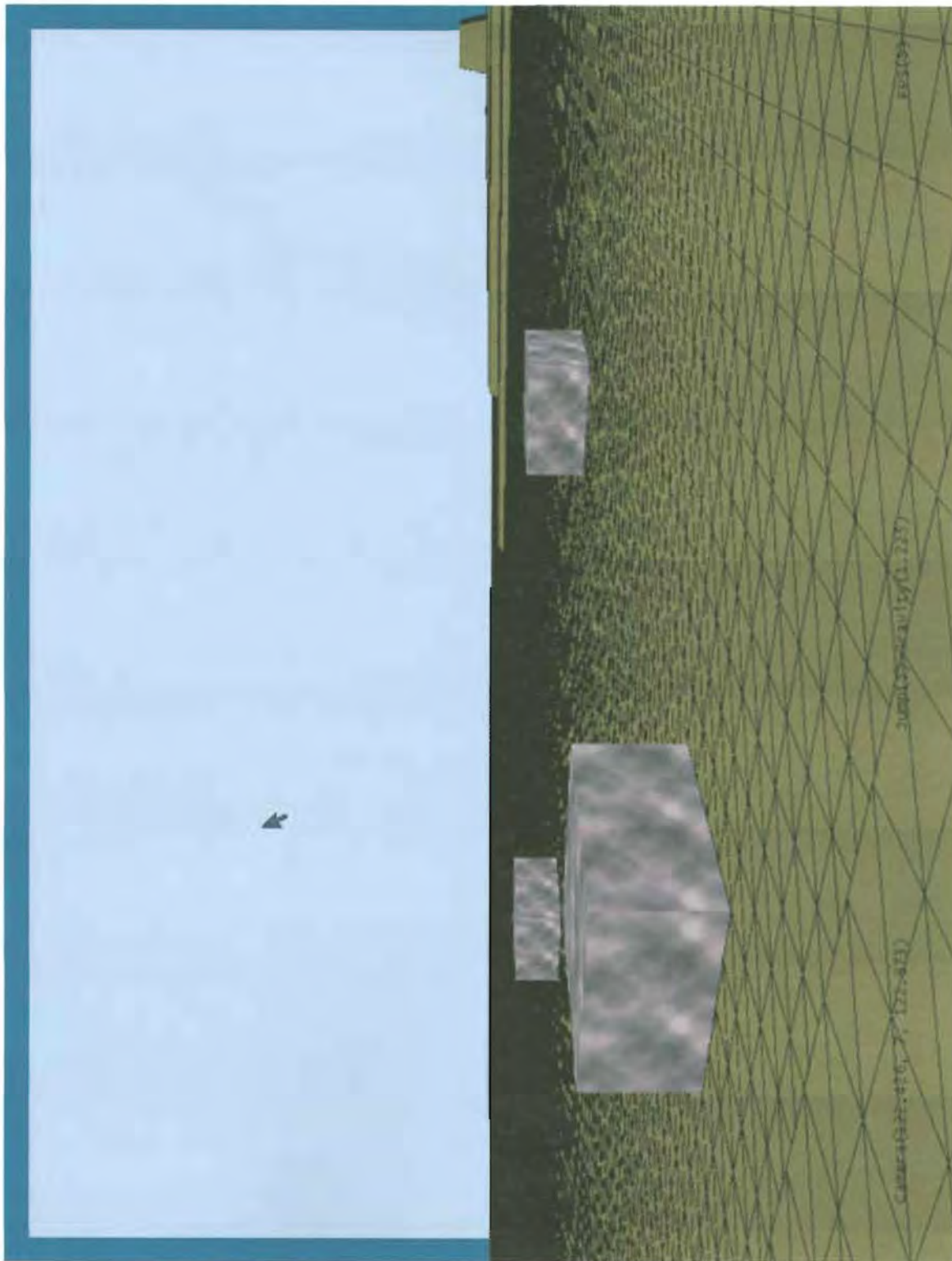
Below: These boxes are entities being affected by gravity.



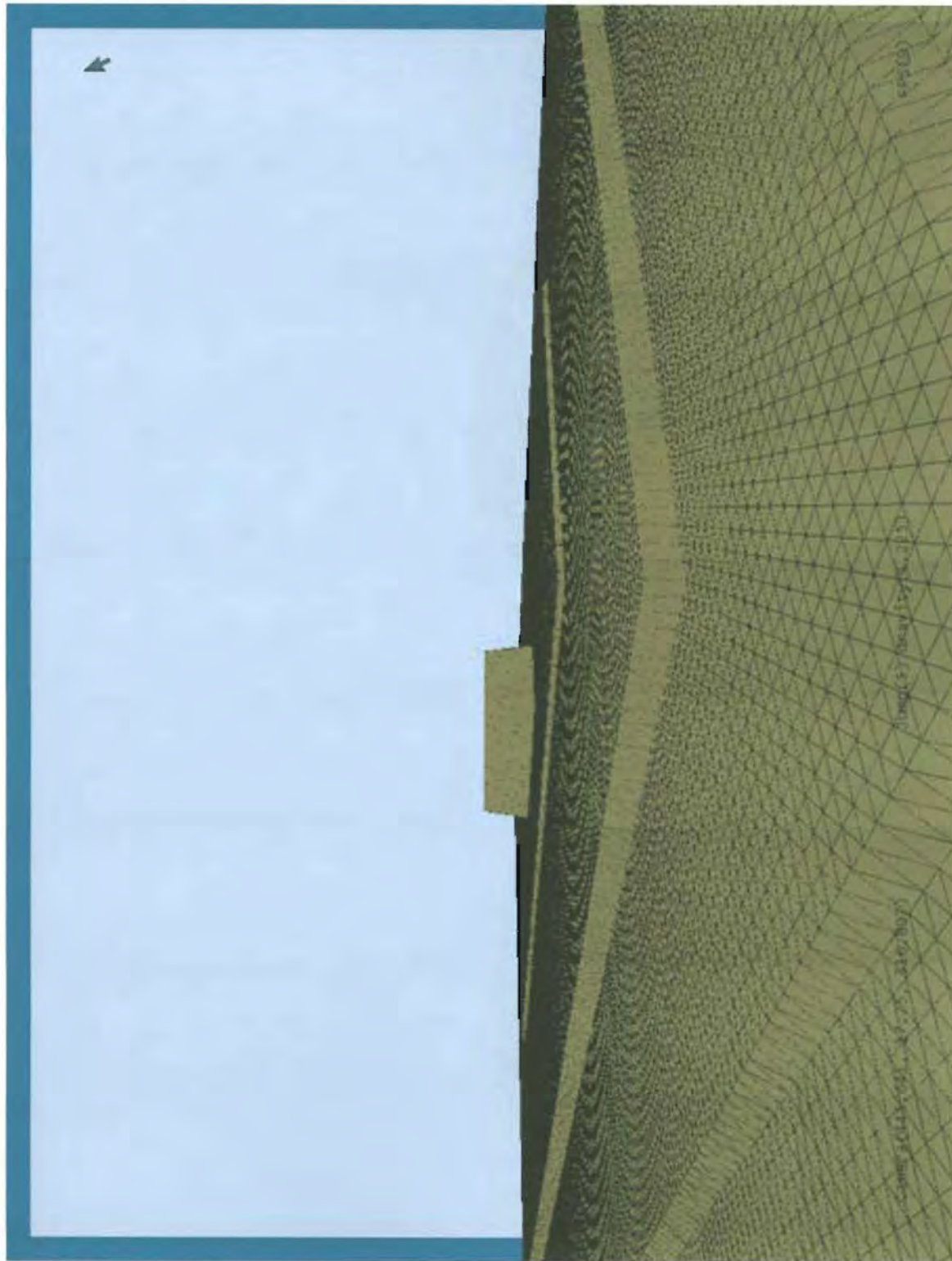
I.T.S. Screenshot

(7 of 14)

Below: The box entities have reached the ground.



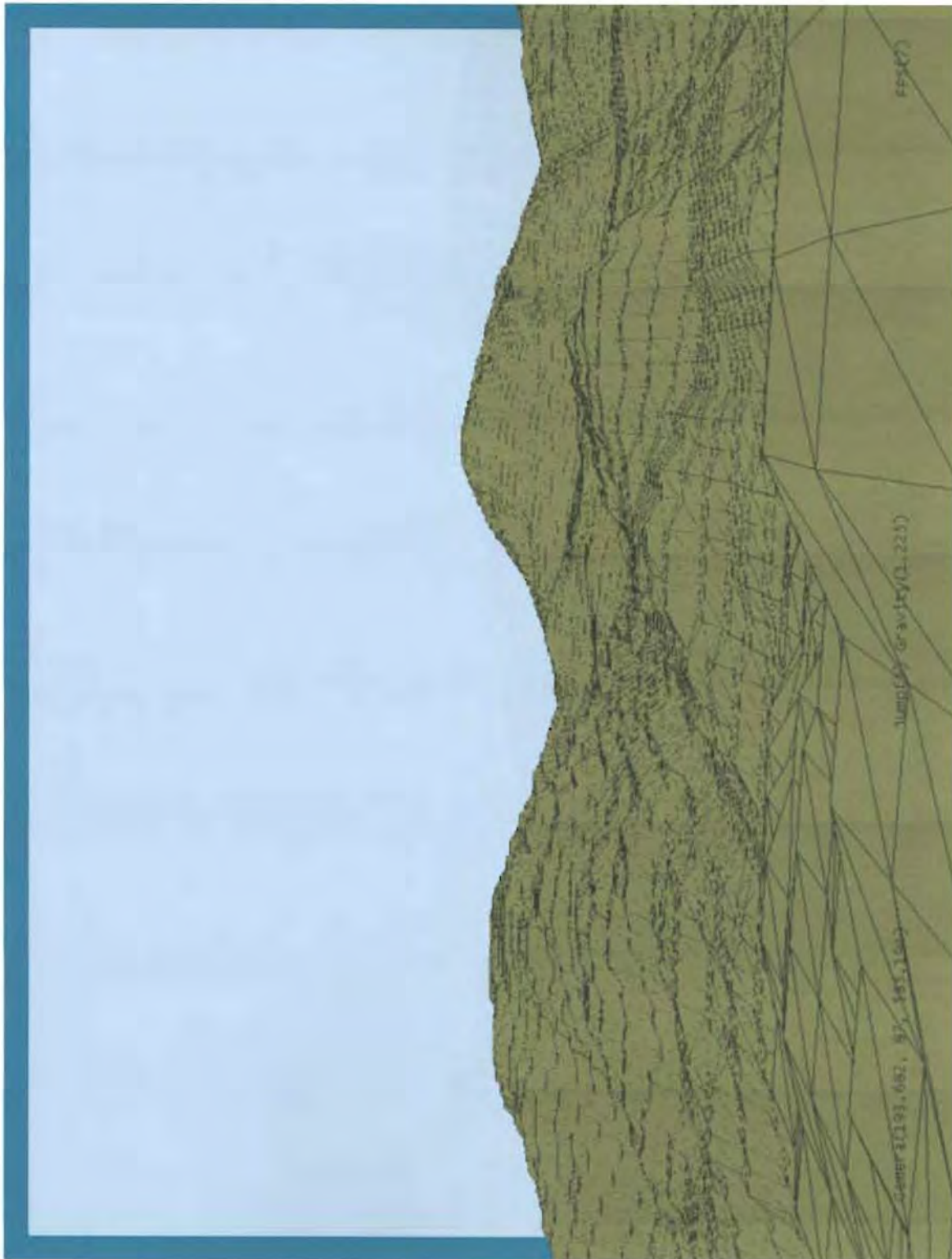
Below: More of the terrain can be seen when the camera jumps.



I.T.S. Screenshot

(9 of 14)

Below: This terrain was created by the W.I.M. Tool.



I.T.S. Screenshot

(10 of 14)

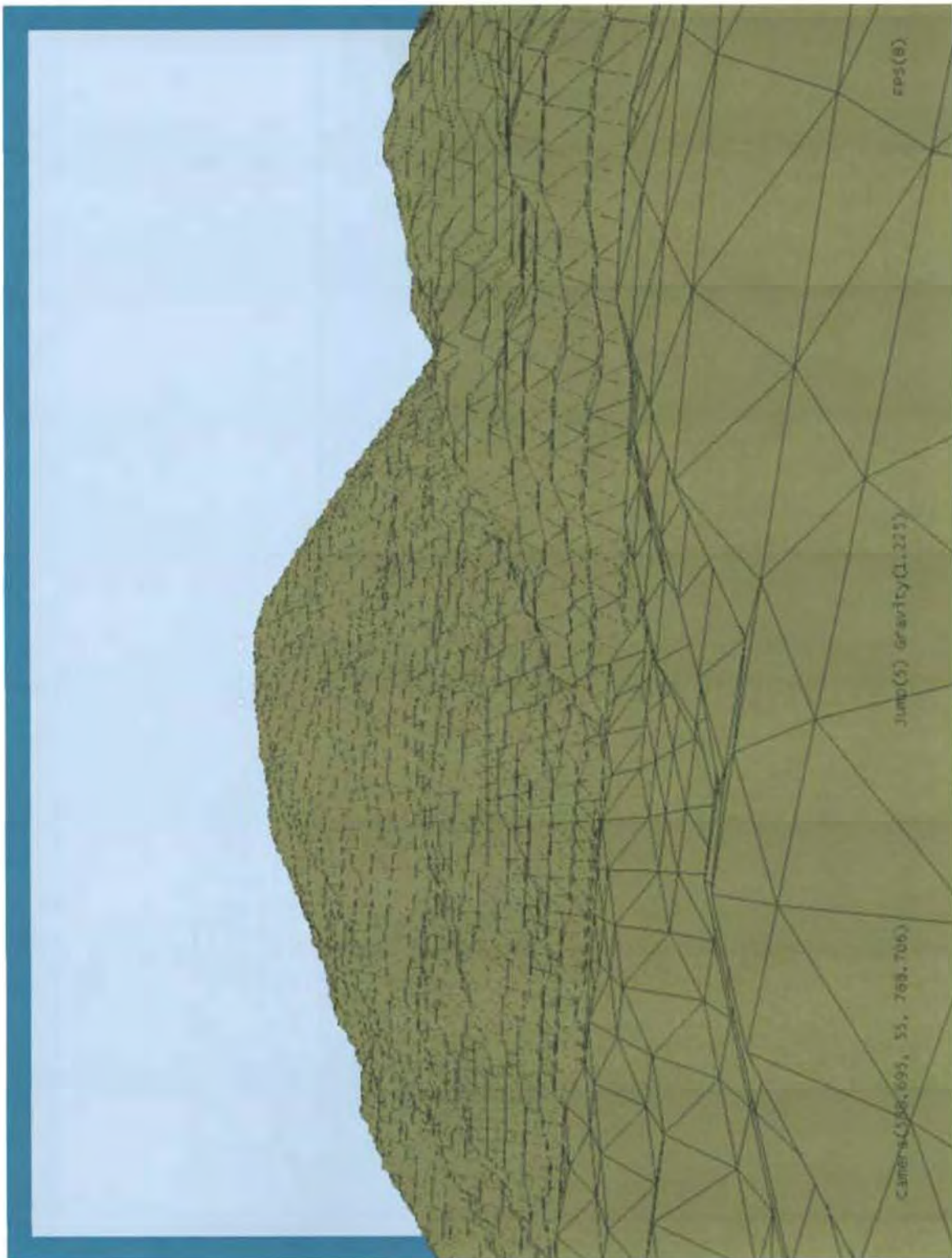
Below: More terrain created by the W.I.M. Tool.



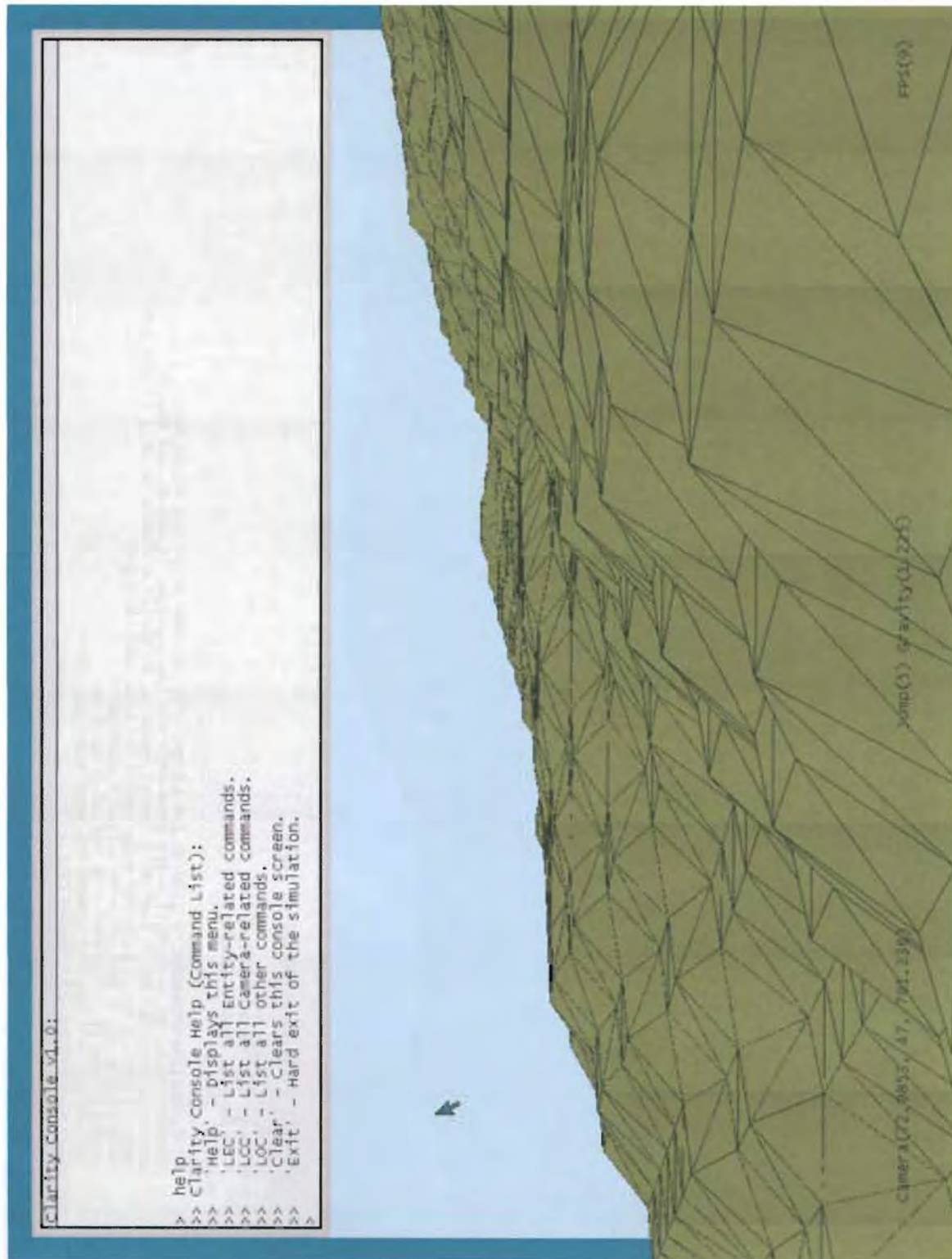
I.T.S. Screenshot

(11 of 14)

Below: The camera is pointed at nearby hills in this W.I.M. Tool terrain.



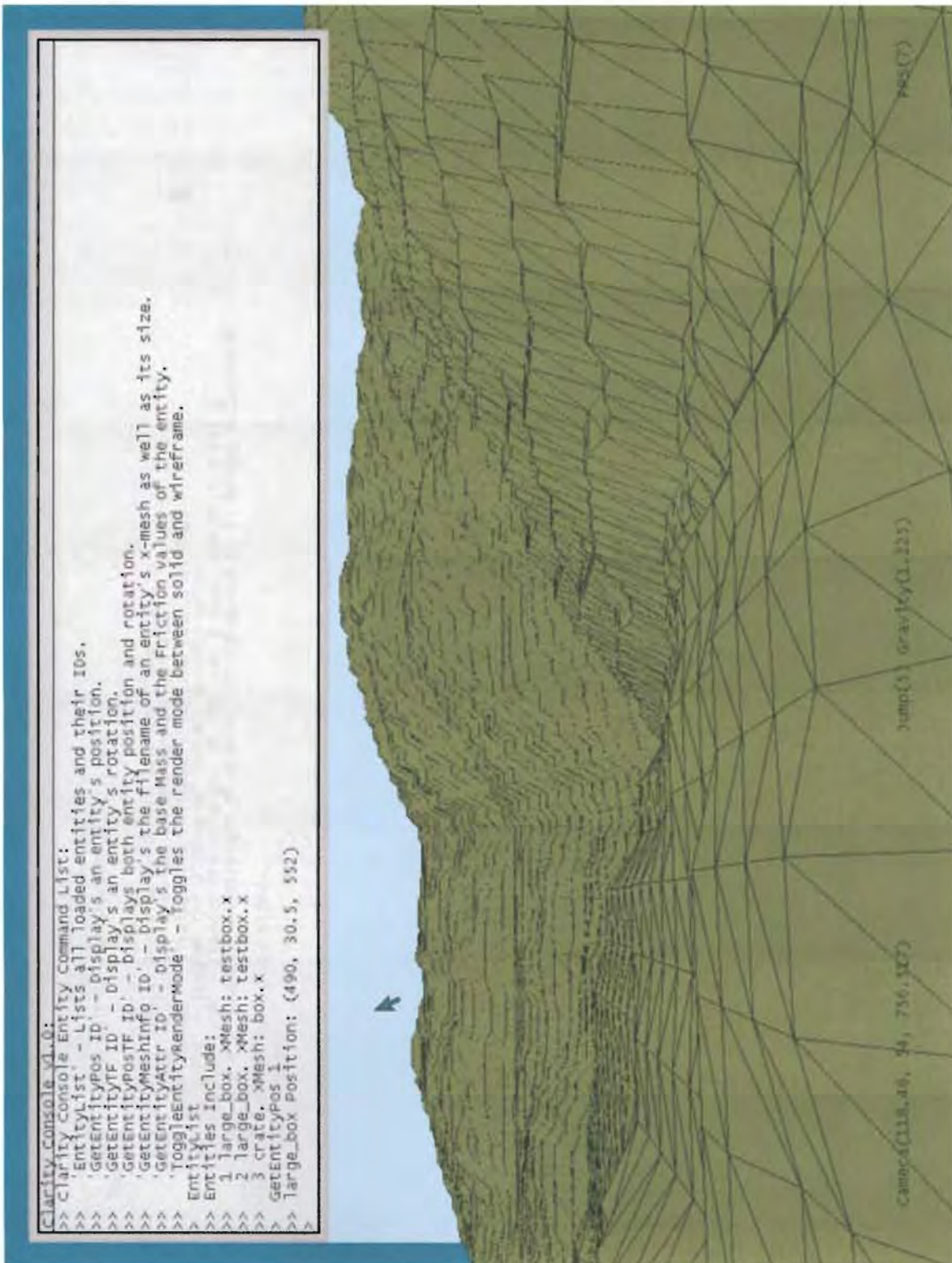
Below: The command console displays the help menu.



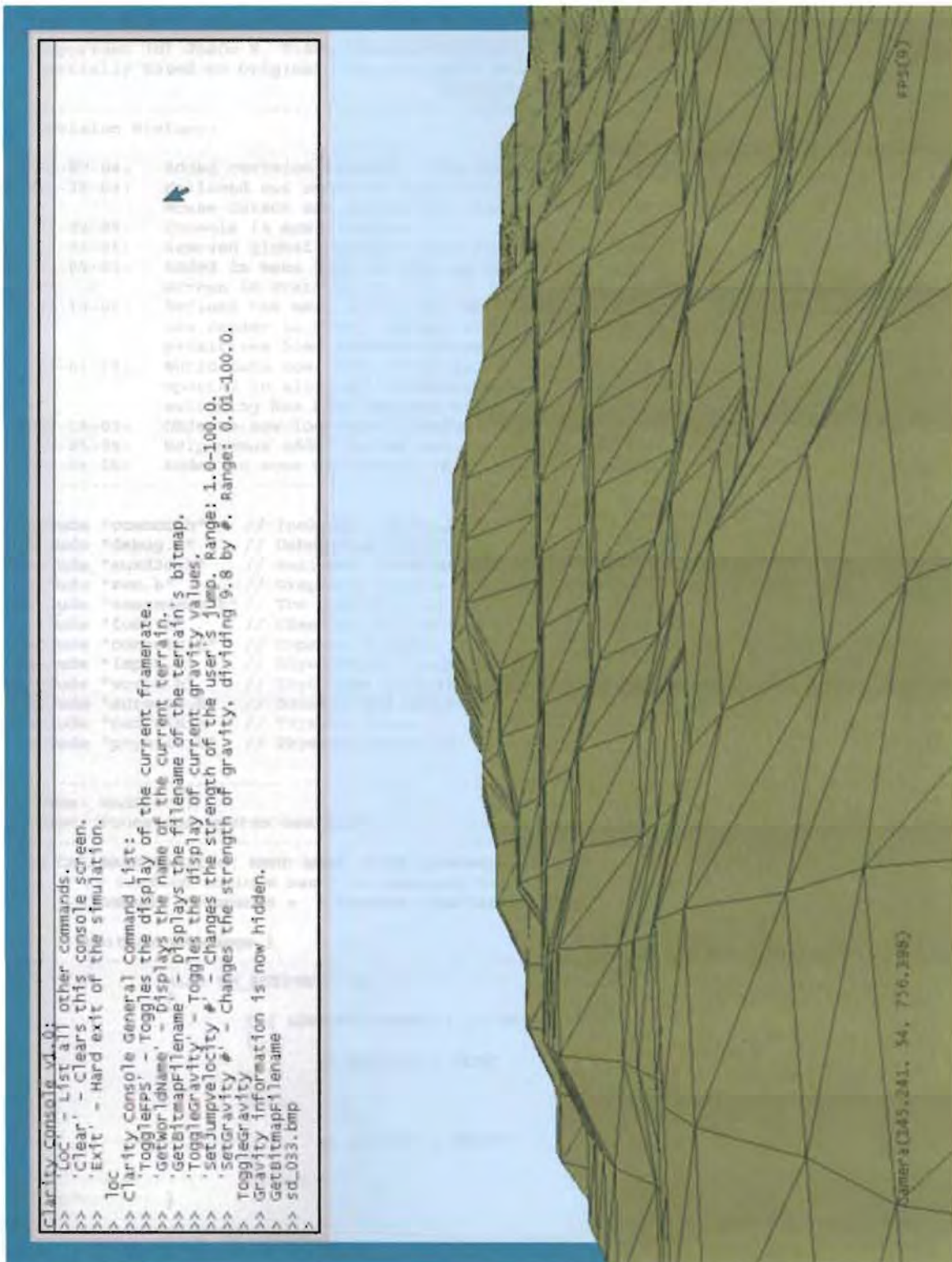
I.T.S. Screenshot

(13 of 14)

Below: The command console contains various entity-related commands.



Below: Visual information and hidden information are both managed here.



```

-----
// File:      origins.cpp
//
// Desc:      This is the main file for the Interactive Terrain Simulator.
//
// First created on:  December 27th, 2004
// Last modification: April 6th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//                                     Direct3D Game Programming"
-----
// Revision History:
//
// 12-27-04:  Added revision history. File created. Converted code to DX9.
// 12-28-04:  Hollowed out code and commented everything.
//           Mouse cursor and background images are working.
// 01-02-05:  Console is now visible.
// 01-04-05:  Removed global objects since they are all now Singletons.
// 01-05-05:  Added in menu text as well as the listing of .wid files on the
//           screen in state #2.
// 01-13-05:  Refined the menu calls. Set up the camera position. Primitives
//           now render in color. Meshes also render, and both meshes and
//           primitives load textures properly.
// 03-01-05:  World data now loads properly, and the console has been
//           updated to allow an interface with this data. Also, the state
//           switching has been updated to reflect these changes.
// 03-18-05:  Objects now load and transform properly.
// 03-25-05:  Help menus added to the now cleaned-up console.
// 04-06-05:  Added in some management of terrain/world data.
-----

#include "common.h"    // Includes, globals.
#include "debug.h"    // Debugging functions.
#include "auxd3d.h"   // Auxiliary functions, Separated for cleanliness.
#include "zen.h"      // Graphics classes and functions.
#include "zencamera.h" // The camera class.
#include "font.h"     // CZenFont for 3D fonts.
#include "console.h"  // Console classes.
#include "input.h"    // DirectInput classes.
#include "world.h"    // Structure to load, hold, and access Sim data.
#include "screens.h"  // Screens and menus.
#include "terrain.h"  // Terrain class.
#include "physics.h"  // Physics functions. (Gravity)

-----
// Name: WndProc()
// Desc: Processes system messages.
-----
long CALLBACK WndProc( HWND hWnd, UINT uMessage, WPARAM wParam, LPARAM lParam )
{
    // Perform actions based on messages from the Windows OS.
    CConsole * Console = CConsole::Instance();

    switch( uMessage )
    {
        case WM_ACTIVATE:
        {
            if( LOWORD( wParam ) != WA_INACTIVE )
            {
                g_bActive = TRUE;
            }
            else
            {
                g_bActive = FALSE;
            }
            return 0;
        }

        case WM_CREATE:    // CreateWindow() was called.
        {
            // GDI area for GDI version of SimInit().

```

```

        return 0;
    }

    case WM_PAINT: // Message to redraw the window.
    {
        ValidateRect( hWnd, NULL );
        return 0;
    }

    case WM_KEYDOWN:
    {
        Console->OnKeyDown( wParam );

        switch( wParam )
        {
            case VK_LEFT:
            {
                break;
            }
            case VK_RIGHT:
            {
                break;
            }
            case VK_UP:
            {
                break;
            }
            case VK_DOWN:
            {
                break;
            }
        }
        return 0;
    }

    case WM_CHAR:
    {
        Console->OnChar( (char)wParam );
        return 0;
    }

    case WM_DESTROY: // The window is closing.
    {
        PostQuitMessage( 0 ); // Exit program.
        return 0;
    }

    case WM_SETCURSOR:
    {
        CZenMouse * g_Mouse = CZenMouse::Instance();
        return g_Mouse->HandleSetCursor();
    }

    default:
    {
        // Let Windows handle this message.
        return DefWindowProc( hWnd, uMessage, wParam, lParam );
    }
}

}

//-----
// Name: WinMain()
// Desc: Entry point to the program. Initializes everything, and goes into a
//       message-processing loop. Idle time is used to render the scene.
//-----
int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, PSTR pstrCmdLine, int
iCmdShow )
{
    HWND hWnd; // The handle to our main window.
    MSG msg; // The message windows is sending us.
    WNDCLASSEX wc; // The window class used to create our window.

```

```

// The name of our class and also the title to our window.
static char strAppName[] = "ITS - Interactive Terrain Simulator";

// Fill in the window class with the attributes for our main window.

// The size of this structure in bytes.
wc.cbSize = sizeof( WNDCLASSEX );
// The style of the window.
wc.style = CS_HREDRAW | CS_VREDRAW | CS_OWNDC;
// Useless information. Just set to zero.
wc.cbClsExtra = 0;
// Useless information. Just set to zero.
wc.cbWndExtra = 0;
// The name of our event handler.
wc.lpfnWndProc = WndProc;
// A handle to the applications instance.
wc.hInstance = hInstance;
// The handle to the brush to use for the window background.
wc.hbrBackground = (HBRUSH)GetStockObject( DKGRAY_BRUSH );
// A handle to the icon to use for the window.
wc.hIcon = LoadIcon( NULL, IDI_APPLICATION );
// A handle to a smaller version of the apps icon.
wc.hIconSm = LoadIcon( NULL, IDI_APPLICATION );
// A handle to the cursor to use while the mouse is over our window.
wc.hCursor = LoadCursor( NULL, IDC_CROSS );
// A handle to the resource to use as our menu.
wc.lpszMenuName = NULL;
// The human readable name for this class.
wc.lpszClassName = strAppName;

// Register the class with windows.
RegisterClassEx( &wc );

// Create the window based on the previous class.
hWnd = CreateWindowEx( WS_EX_TOPMOST, // Advanced style settings.
    strAppName, // The name of the class.
    strAppName, // The window caption.
    WS_POPUP | WS_SYSMENU | WS_VISIBLE, // The window style.
    CW_USEDEFAULT, // The initial x position.
    CW_USEDEFAULT, // The initial y position.
    512, 512, // The initial width / height.
    NULL, // Handle to parent window.
    NULL, // Handle to the menu.
    hInstance, // Handle to the apps instance.
    NULL ); // Advanced context.

g_hWndMain = hWnd;
g_hInstMain = hInstance;

// Display the window we just created.
ShowWindow( hWnd, iCmdShow );
// Draw the window contents for the first time.
UpdateWindow( hWnd );

if( FAILED( SimInit() ) )
{
    // Simulation initialization - exit simulation if there is an error.
    Debug( "Simulation initialization failed. Exiting." );
    SimCleanup();
    return E_FAIL;
}

while( TRUE )
{
    // This is the windows message loop.
    if( PeekMessage( &msg, NULL, 0, 0, PM_REMOVE ) )
    {
        // If there is a message to process ...
        if( msg.message == WM_QUIT )
        {
            // This means we should exit the loop.

            break;
        }
        // Change the format of certain messages.
    }
}

```

```

        TranslateMessage( &msg );
        // Pass the message to WndProc() for processing.
        DispatchMessage( &msg );
    }
    else
    {
        SimLoop();
    }
}

SimCleanup();

// Return control to Windows with the exit code.
return msg.wParam;
}

//-----
// Name: SimInit()
// Desc: Initialize everything needed in the simulation. Should only be called
//       once per simulation.
//-----
int SimInit()
{
    HRESULT r = 0;
    CZenMouse * g_Mouse = CZenMouse::Instance();
    CConsole * Console = CConsole::Instance();

    // Create the IDirect3D9 pointer.
    g_pD3D = Direct3DCreate9( D3D_SDK_VERSION );
    if( !Trycatch((void*)g_pD3D, "g_pD3D in SimInit().") )
    {
        Debug("D3D object creation failed.");
        return E_FAIL;
    }

    // Create the D3D device pointer.
    r = InitDirect3DDevice( g_hWndMain, 1024, 768, FALSE, D3DFMT_A8R8G8B8, _
        g_pD3D, &g_pDevice );
    if( FAILED( r ) )
    {
        Debug( "Direct3D Device initialization failed." );
        return E_FAIL;
    }

    // Set up viewing information.
    CreateViewport();
    SetProjectionMatrix();

    // Clear the back buffer.
    g_pDevice->Clear( 0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, _
        D3DCOLOR_XRGB( 0, 0, 100 ), 1.0f, 0 );

    // Get a pointer to the back buffer.
    r = g_pDevice->GetBackBuffer( 0, 0, D3DBACKBUFFER_TYPE_MONO, &g_pBackSurface );
    if( FAILED( r ) )
    {
        Debug( "Couldn't get the backbuffer." );
        return E_FAIL;
    }

    // Load the alphabet to be used for text.
    LoadAlphabet( "img\\Alphabet10.bmp", 8, 13 );

    // Initialize timing for frame rate counters, etc.
    srand( GetTickCount() );
    InitTiming();

    // Initialize the console.
    Console->Initialize( g_pDevice, g_pBackSurface );
    Console->SetParserCallback( ConsoleParser );
}

```

```

// Set the vertex format for rendering.
g_pDevice->SetFVF( ZENVERTEX_TYPE ); // Used to be SetVertexShader()

// Initialize the DirectInput devices.
r = InitializeInput();
if( FAILED( r ) )
{
    Debug( "Unable to initialize input in SimInit()." );
    return E_FAIL;
}
g_Mouse->ShowCursor( TRUE );
// I shouldn't need this line since WM_SETCURSOR should. But Windows
// never calls it from there.
g_Mouse->HandleSetCursor();

// Set the default texture for undefined X-Meshes.
r = D3DXCreateTextureFromFile( g_pDevice, "img\\DefaultTexture.bmp", _
    &g_pDefaultTexture );
if( FAILED( r ) )
{
    Debug( "Unable to load default texture in SimInit()." );
    return E_FAIL;
}

// Place the camera into its initial position.
CZenCamera * g_Camera = CZenCamera::Instance();
g_Camera->SetPosition( 15, 5, 15 );

// Set the ambient light level, and other lighting and rendering information.
g_pDevice->SetRenderState( D3DRS_AMBIENT, D3DCOLOR_XRGB( 200, 200, 200 ) );
SetAmbientLight( D3DCOLOR_XRGB( 0, 0, 100 ) );
g_pDevice->SetRenderState( D3DRS_LIGHTING, FALSE ); // Allows color to show.
g_pDevice->SetRenderState( D3DRS_CULLMODE, D3DCULL_CCW );

// Initialize font objects for the screens.
HFONT hFont;
CZenFont Font[ MAX_FONTS ]; // Create up to 26 fonts since MAX_FONTS = 26.
FontBank * FontBank = Fontbank::Instance();

// FW_NORMAL -> FW_BOLD for bold text.
// Next two BOOL values are Italics and Underline.

hFont = CreateFont( 36, 0, 0, 0, FW_NORMAL, 0, 0, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[0].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 36pt Black TNR
FontBank->AddFont(1, Font[0]);

hFont = CreateFont( 24, 0, 0, 0, FW_NORMAL, 0, 0, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[1].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 24pt Black TNR
FontBank->AddFont(2, Font[1]);

hFont = CreateFont( 12, 0, 0, 0, FW_NORMAL, 0, 0, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[2].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 12pt Black TNR
FontBank->AddFont(3, Font[2]);

hFont = CreateFont( 24, 0, 0, 0, FW_NORMAL, 0, 1, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[3].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 24pt Black TNR, Underline
FontBank->AddFont(4, Font[3]);

hFont = CreateFont( 72, 0, 0, 0, FW_NORMAL, 0, 0, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[4].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 72pt Black TNR
FontBank->AddFont(5, Font[4]);

```

```

hFont = CreateFont( 16, 0, 0, 0, FW_NORMAL, 0, 0, 0, ANSI_CHARSET,
    OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
    DEFAULT_PITCH | FF_ROMAN, "Times New Roman" );
Font[5].Initialize(hFont, D3DCOLOR_XRGB(0,0,0)); // 16pt Black TNR
FontBank->AddFont(6, Font[5]);

// Call a helper function that performs state-dependant initialization.
InitScene();

return S_OK;
}

//-----
// Name: InitScene()
// Desc: Helper function to SimInit(). Used to initialize state-specific data.
//-----
HRESULT InitScene()
{
    // Font pointer retrieval.
    Fontbank * FontBank = Fontbank::Instance();
    CZenFont * FontTNR36 = FontBank->GetFont(1);
    CZenFont * FontTNR24 = FontBank->GetFont(2);
    CZenFont * FontTNR12 = FontBank->GetFont(3);
    CZenFont * FontTNR24U = FontBank->GetFont(4);
    CZenFont * FontTNR72 = FontBank->GetFont(5);
    CZenFont * FontTNR16 = FontBank->GetFont(6);

    // Get pointers to the necessary singletons.
    CZenCamera * g_Camera = CZenCamera::Instance();
    WorldSingleton * World = WorldSingleton::Instance();
    TerrainSingleton * Terrain = TerrainSingleton::Instance();

    // For world loading in State #1.
    int nCounter = 0;
    vector<string> vecWIDFiles;
    vecWIDFiles.clear(); // Is this necessary? Precaution.
    vecWIDFiles.resize(255);
    char * tmpString;
    bool bNoFiles = false;

    // Screen creation.
    Screen * theScreen = Screen::Instance();
    int tWidth, tHeight;

    switch(g_nStateFlag)
    {
    case 0: // Title Screen.
        theScreen->Clear();

        FontTNR36->GetBoundingBox("I.T.S.", tWidth, tHeight);
        theScreen->SetText(1, FontTNR36, "I.T.S.", _
            ((g_DeviceWidth / 2) - (tWidth / 2)), 30);

        FontTNR36->GetBoundingBox("The Interactive Terrain Simulator", _
            tWidth, tHeight);
        theScreen->SetText(2, FontTNR36, "The Interactive Terrain Simulator", _
            ((g_DeviceWidth / 2) - (tWidth / 2)), 80);

        FontTNR24->GetBoundingBox("(Project Origins)", tWidth, tHeight);
        theScreen->SetText(6, FontTNR24, "(Project Origins)", _
            ((g_DeviceWidth / 2) - (tWidth / 2)), 120);

        theScreen->SetText(3, FontTNR24, "Main Menu:", 350, 300);

        theScreen->SetText(4, FontTNR24, "Load a Simulation", 400, 340);
        theScreen->SetFunc(4, LoadWorldScreen);

        theScreen->SetText(5, FontTNR24, "Exit the Simulator", 400, 380);
        theScreen->SetFunc(5, ExitSimulator);
    }
}

```

```

LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );
break;
case 1: // Load World Screens.
theScreen->Clear();

FontTNR36->GetBoundingBox("Simulation World Loader", tWidth, tHeight);
theScreen->SetText(1, FontTNR36, "Simulation World Loader", _
((g_DeviceWidth / 2) - (tWidth / 2)), 80);

FontTNR24U->GetBoundingBox("Simulation Variables", tWidth, tHeight);
theScreen->SetText(2, FontTNR24U, "Simulation Variables", _
((g_DeviceWidth / 4) - (tWidth / 2)), 200);

FontTNR24U->GetBoundingBox("World Data Files", tWidth, tHeight);
theScreen->SetText(3, FontTNR24U, "World Data Files", _
((g_DeviceWidth / 4) * 3) - (tWidth / 2)), 200);

// List the World Files available.
vecWIDFiles = GetWIDFileNames();
while(nCounter < 20) // Print 20 filenames, maximum.
{
    if(vecWIDFiles.size() == 0)
    {
        Debug("Empty vector!");
        bNoFiles = true;
        break;
    }
    if(vecWIDFiles[nCounter] == "")
    {
        Debug("Null pointer in vector!");
        break;
    }
    // Send the string to the Screen object to be printed later.
    tmpString = new char[80];
    strcpy(tmpString, vecWIDFiles[nCounter].c_str());
    theScreen->SetText((4 + nCounter), FontTNR16, tmpString, 700, _
(220 + (nCounter * 15)));
    theScreen->SetWorldFunc((4 + nCounter), CallLoadWorld);
    // Saves the filename.
    theScreen->SetWorldFile((4 + nCounter), vecWIDFiles[nCounter]);
    // Increment the counter.
    nCounter++;
}

// If there are no .wid files to load ...
if(bNoFiles)
{
    tmpString = new char[80];
    strcpy(tmpString, "The .wid directory is empty.");
    theScreen->SetText((4 + nCounter), FontTNR16, tmpString, 700, _
(220 + (nCounter * 15)));
}

LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );

g_bTerrainLoaded = false; // Invalidate any previously loaded terrain.
break;
case 2: // Waiting Screen.
// This state is not currently in use in the simulator. More advanced timing and
// management code must be in place before this state can be used properly.
theScreen->Clear();

FontTNR72->GetBoundingBox("Please Wait:", tWidth, tHeight);
theScreen->SetText(1, FontTNR72, "Please Wait:", _
((g_DeviceWidth / 2) - (tWidth / 2)), ((g_DeviceHeight / 2) - 150));

FontTNR72->GetBoundingBox("Simulation Loading ...", tWidth, tHeight);
theScreen->SetText(2, FontTNR72, "Simulation Loading ...", _
((g_DeviceWidth / 2) - (tWidth / 2)), ((g_DeviceHeight / 2) - 50));

LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );

```



```

        break;
    case 3: // Simulation.
        theScreen->Clear();
        LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );

        if(!g_bTerrainLoaded)
        {
            // Load data into the TerrainSingleton's Vertex Buffers.

            // Create the actual terrain.
            if(Terrain->CreateVertexBuffer() == 0)
            {
                // Exit if there was an error.
                PostQuitMessage( 0 );
            }
            // Create a slightly elevated wireframe.
            if(Terrain->CreateElevatedVertexBuffer() == 0)
            {
                // Exit if there was an error.
                PostQuitMessage( 0 );
            }

            // Get the starting position for the camera.
            int cx, cy, cz;
            cx = World->TheUser.x;
            cy = World->TheUser.y;
            cz = World->TheUser.z;

            g_Camera->SetPosition( cx*2, cz + Terrain->GetHeight( (float)cx,
                (float)cy), cy*2 );

            g_bTerrainLoaded = true;
        }

        break;
    case 4: // Pause Screen.
        theScreen->Clear();

        FontTNR36->GetBoundingBox("Simulation Paused", tWidth, tHeight);
        theScreen->SetText(1, FontTNR36, "Simulation Paused",
            ((g_DeviceWidth / 2) - (tWidth / 2)), 80);

        theScreen->SetText(2, FontTNR24, "Pause Menu:", 350, 300);

        theScreen->SetText(3, FontTNR24, "Resume Simulation", 400, 340);
        theScreen->SetFunc(3, ResumeSim);

        theScreen->SetText(4, FontTNR24, "Exit to World Loading Screen",
            400, 380);
        theScreen->SetFunc(4, ExitToWorldScreen);

        theScreen->SetText(5, FontTNR24, "Exit the Simulator", 400, 420);
        theScreen->SetFunc(5, ExitSimulator);

        LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );
        break;
    case 5: // Exit Screen.
        theScreen->Clear();

        // Title.
        FontTNR36->GetBoundingBox("Thank you for using:", tWidth, tHeight);
        theScreen->SetText(1, FontTNR36, "Thank you for using:",
            ((g_DeviceWidth / 2) - (tWidth / 2)), 80);

        FontTNR36->GetBoundingBox("I.T.S.", tWidth, tHeight);
        theScreen->SetText(2, FontTNR36, "I.T.S.",
            ((g_DeviceWidth / 2) - (tWidth / 2)), 140);

        FontTNR36->GetBoundingBox("The Interactive Terrain Simulator",
            tWidth, tHeight);
        theScreen->SetText(3, FontTNR36, "The Interactive Terrain Simulator",
            ((g_DeviceWidth / 2) - (tWidth / 2)), 200);

```

```

// Credits.
FontTNR24U->GetBoundingBox("Credits", tWidth, tHeight);
theScreen->SetText(4, FontTNR24U, "Credits", _
((g_DeviceWidth / 2) - (tWidth / 2)), 400);

LoadBitmapToSurface( "img\\bg_state.bmp", &g_pBackground, g_pDevice );
break;
default:
theScreen->Clear();
LoadBitmapToSurface( "img\\background.bmp", &g_pBackground, g_pDevice );
break;
}

return S_OK;
}

//-----
// Name: DestroyScene()
// Desc: Helper function to SimCleanup(), Used for state-specific
//       deconstruction.
//-----
HRESULT DestroyScene()
{
    // All variables other than the background are handled by their own
    // classes. The init/destruct pair will be better organized in the future.
    if( Trycatch((void*)g_pBackground, "g_pBackground in DestroyScene()") )
    {
        g_pBackground->Release();
    }
    return S_OK;
}

//-----
// Name: SimLoop()
// Desc: This is the simulation loop that is constantly being called. Once
//       activated, the loop handles timing, input, and rendering to the
//       display.
//-----
int SimLoop()
{
    if( !g_bActive )
    {
        // Only run the loop if Windows has activated it.
        return S_OK;
    }

    FrameCount(); // Keep track of the framerate. Used for display.
                 // Synchronization could be added for stable framerates.
    HandleInput(); // Read in the user's input and act accordingly.
    SimRender(); // Render the world (or screens) to the display.
    return S_OK;
}

//-----
// Name: HandleInput()
// Desc: This is where the logic for the DirectInput devices goes. Options are
//       given depending on the state of the simulation.
//-----
HRESULT HandleInput()
{
    // Get the necessary singleton pointers.
    CZenMouse * g_Mouse = CZenMouse::Instance();
    CZenKeyboard * g_Keyboard = CZenKeyboard::Instance();

    // Blocks keyboard input from the console when it is not activated.
    if(!g_bConsoleOn)
    {
        if( g_Keyboard->IsKeyDown( DIK_ESCAPE ) )
        {
            // Exit the program when the 'Esc' key is pressed.
            PostQuitMessage( 0 );
        }
    }
}

```

```

if(!g_bPauseLock)    // Preserves exclusivity.
{
    if( (g_nStateFlag == 3) && g_Keyboard->IsKeyDown( DIK_P ) )
    {
        // Pauses the program if in Simulation mode.
        PauseSim();
    }
    else if( (g_nStateFlag == 4) && g_Keyboard->IsKeyDown( DIK_P ) )
    {
        // Resumes the program if in Pause mode.
        ResumeSim();
    }
}

// Retrieve the current state of the mouse.
g_Mouse->Poll();
g_Mouse->UpdateCursorPos();

// Mouse logic goes here.

// Retrieve data from the necessary objects.
Screen * theScreen = Screen::Instance();           // The menu screen.
list<Text> * lstText = theScreen->GetTextList();     // A list of all text items.
int mx, my;                                         // For cursor position.
g_Mouse->GetCursorPosition(mx, my);                // Fill in cursor position.

// This section deals with menu text and attached functionality.
if(g_nStateFlag != 3) // There is no menu text in the simulation ...
{
    // The following code highlights text with attached commands.
    int x, y;
    for(list<Text>::iterator i = lstText->begin(); i != lstText->end(); i++)
    {
        // Skip this text if there's no functionality.
        if(i->GetFuncPtr() == 0)
        {
            continue;
        }
        i->GetFontPtr()->GetBoundingBox(i->GetTextPtr(), x, y);
        if((mx > i->GetX()) && (my > i->GetY()) &&
            (mx < (i->GetX() + x)) && (my < (i->GetY() + y)))
        {
            i->GetFontPtr()->SetColor( D3DCOLOR_XRGB(255, 0, 0) );
            if(g_Mouse->IsButtonDown( 0 ))
            {
                VoidFuncPtr FuncPtr = i->GetFuncPtr();
                FuncPtr();
            }
            break;
        }
        i->GetFontPtr()->RestoreColor();
    }
    // Handles the WorldFuncPtr.
    for(list<Text>::iterator i = lstText->begin(); i != lstText->end(); i++)
    {
        // Skip this text if there's no functionality.
        if(i->GetWorldFuncPtr() == 0)
        {
            continue;
        }
        i->GetFontPtr()->GetBoundingBox(i->GetTextPtr(), x, y);
        if((mx > i->GetX()) && (my > i->GetY()) && (mx < (i->GetX() + x)) &&
            (my < (i->GetY() + y)))
        {
            i->GetFontPtr()->SetColor( D3DCOLOR_XRGB(0, 128, 0) );
            if(g_Mouse->IsButtonDown( 0 ))
            {
                WorldFuncPtr FuncPtr = i->GetWorldFuncPtr();
                FuncPtr( i->GetWorldFile() );
            }
            break;
        }
    }
}

```

```

        i->GetFontPtr()->RestoreColor();
    }
}

// Check for keyboard input related to the camera.
if(!g_bConsoleOn) && (g_nStateFlag == 3)
{
    // Camera is only available in Simulation Mode with no console.
    CZenCamera * g_Camera = CZenCamera::Instance();
    TerrainSingleton * Terrain = TerrainSingleton::Instance();

    float x, y, z, vx, vy, vz;

    g_Camera->GetPosition(x, y, z);
    g_Camera->GetVelocity(vx, vy, vz);
    if((vx == 0) && (vy == 0) && (vz == 0))
    {
        // Is the user jumping?
        if( g_Keyboard->IsKeyDown( DIK_SPACE ) )
        {
            CameraJump(); // This is the initial jump.
            g_bCameraHitGround = false;
            g_Camera->SetYaw(0.0f);
        }
        else
        {
            // Do everything else.

            g_Camera->GetRight(x, y, z);
            if( g_Keyboard->IsKeyDown( DIK_Q ) _
            || g_Keyboard->IsKeyDown( DIK_LEFT ) )
            {
                // Strafe left.
                g_Camera->Move(-x*g_fCameraSpeed, 0, _
                -z*g_fCameraSpeed);
            }
            if( g_Keyboard->IsKeyDown( DIK_E ) _
            || g_Keyboard->IsKeyDown( DIK_RIGHT ) )
            {
                // Strafe right.
                g_Camera->Move(x*g_fCameraSpeed, 0, _
                z*g_fCameraSpeed);
            }

            // Update the camera position based on the terrain.
            g_Camera->GetPosition(x, y, z);
            float fHeight = Terrain->GetHeight(x, z);
            if(fHeight != -1.0)
            {
                g_Camera->SetPosition(x, fHeight + 5.0f, z);
            }
            else // Out of Bounds Error,
            {
                g_Camera->SetPosition(x, y, z);

                // Allow the camera to go up/down if camera
                // is unlocked and out of bounds.
                if( g_Keyboard->IsKeyDown( DIK_R ) )
                {
                    // Hover up.
                    g_Camera->Move(0, g_fCameraSpeed, 0);
                }
                if( g_Keyboard->IsKeyDown( DIK_F ) )
                {
                    // Hover down.
                    g_Camera->Move(0, -g_fCameraSpeed, 0);
                }
            }

            g_Camera->GetLookPoint(x, y, z);
            if( g_Keyboard->IsKeyDown( DIK_W ) _
            || g_Keyboard->IsKeyDown( DIK_UP ) )
            {
                // Move forward.
                g_Camera->Move(x*g_fCameraSpeed, 0, _
                z*g_fCameraSpeed);
            }
        }
    }
}

```

```

        if( g_Keyboard->IsKeyDown( DIK_S ) || g_Keyboard->IsKeyDown( DIK_DOWN ) )
        {
            // Move backward.
            g_Camera->Move(-x*g_fCameraSpeed, 0, -z*g_fCameraSpeed);
        }

        g_Camera->SetYaw(0.0f);
        if( g_Keyboard->IsKeyDown( DIK_A ) )
        {
            // Turn left.
            g_Camera->SetYaw(-g_fCameraYaw);
        }
        if( g_Keyboard->IsKeyDown( DIK_D ) )
        {
            // Turn right.
            g_Camera->SetYaw(g_fCameraYaw);
        }
    } // End of normal block.
}
else
{
    if(y + vy < Terrain->GetHeight(x, z) + 5.0)
    {
        g_bCameraHitGround = true;
    }

    // Not a jump, just handle gravity.
    CameraGravity(g_bCameraHitGround);
}

// Update the camera.
g_Camera->Update();
}

return S_OK;
}

//-----
// Name: SimRender()
// Desc: Render to the screen based on the current state.
//-----
int SimRender()
{
    HRESULT r = 0;

    // Clear the buffers before rendering.
    g_pDevice->Clear( 0, 0, D3DCLEAR_TARGET | D3DCLEAR_ZBUFFER, _
        D3DCOLOR_XRGB( 0, 0, 10 ), 1.0f, 0 );

    // Confirm that the device is valid before continuing.
    if( !Trycatch((void*)g_pDevice, "g_pDevice in SimRender()") )
    {
        Debug( "Cannot render because there is no device." );
        return E_FAIL;
    }

    // Don't render if the device isn't able to render anything.
    r = ValidateDevice();
    if( FAILED( r ) )
    {
        return E_FAIL;
    }

    // Copy the background to the 2D rendering surface.
    D3DXLoadSurfaceFromSurface( g_pBackSurface, 0, 0, g_pBackground, 0, 0, _
        D3DX_FILTER_NONE, 0 );

    // Sets the vertex format before rendering.
    g_pDevice->SetFVF( ZENVERTEX_TYPE ); // Used to be SetVertexShader().

    // This resets the world matrix before doing a transform.
    D3DXMATRIX WorldMatrix;

```

```

D3DXMatrixIdentity( &WorldMatrix );
g_pDevice->SetTransform( D3DTS_WORLD, &WorldMatrix );

// Previous location of 2D code. It was moved so it would render over the 3D.

// Start the 3D rendering process.
g_pDevice->BeginScene();

// Output any necessary menu items here.
Screen * theScreen = Screen::Instance(); // The menu screen.
list<Text> * lstText = theScreen->GetTextList(); // A list of all text items.
for(list<Text>::iterator j = lstText->begin(); j != lstText->end(); j++)
{
    j->Render();
}

// Render the terrain.
TerrainSingleton * Terrain = TerrainSingleton::Instance();
if((g_nStateFlag == 3) && (!Terrain->bIsEmpty))
{
    // Only render if the state is right and the data exists.
    g_pDevice->SetRenderState( D3DRS_FILLMODE, D3DFILL_WIREFRAME );
    Terrain->Render(true);
    g_pDevice->SetRenderState( D3DRS_FILLMODE, D3DFILL_SOLID );
    Terrain->Render(false);
}

// Render the world's entities.
WorldSingleton * World = WorldSingleton::Instance();
if((g_nStateFlag == 3) && (!Terrain->bIsEmpty))
{
    // Render all of World's objects.
    D3DXMATRIX TransMatrix, TempMatrix, RotMatrix, ScaleMatrix;

    // Determine how to render the entities.
    if(g_bEntityWireframe)
    {
        g_pDevice->SetRenderState( D3DRS_FILLMODE, D3DFILL_WIREFRAME );
    }
    else
    {
        g_pDevice->SetRenderState( D3DRS_FILLMODE, D3DFILL_SOLID );
    }

    // Transform and render each entity.
    for(list<LocalEntity *>::iterator i = World->lstLocalEntities.begin(); _
    i != World->lstLocalEntities.end(); i++)
    {
        // Start with a clean transformation matrix.
        D3DXMatrixIdentity( &WorldMatrix );
        // Scale the entity.
        D3DXMatrixScaling(&ScaleMatrix, _
        (float)(*i)->width / (float)(*i)->owidth, _
        (float)(*i)->height / (float)(*i)->oheight, _
        (float)(*i)->depth / (float)(*i)->odepth);
        D3DXMatrixMultiply(&WorldMatrix, &WorldMatrix, &ScaleMatrix);
        // Rotate the entity.
        D3DXMatrixRotationX(&RotMatrix, _
        (float)(*i)->roll * (float)(6.28 / 360)); // Roll
        D3DXMatrixRotationY(&TempMatrix, _
        (float)(*i)->yaw * (float)(6.28 / 360)); // Yaw
        D3DXMatrixMultiply(&RotMatrix, &RotMatrix, &TempMatrix);
        D3DXMatrixRotationZ(&TempMatrix, _
        (float)(*i)->pitch * (float)(6.28 / 360.0)); // Pitch
        D3DXMatrixMultiply(&RotMatrix, &RotMatrix, &TempMatrix);
        D3DXMatrixMultiply(&WorldMatrix, &WorldMatrix, &RotMatrix);
        // Translate the entity.
        EntityGravity(*i);
        D3DXMatrixTranslation(&TransMatrix, (float)((*i)->x) * 2, _
        ((*i)->z + Terrain->GetHeight((float)(*i)->x*2, _
        (float)(*i)->y*2)), (float)((*i)->y) * 2);
        // Do translation after rotation!
        D3DXMatrixMultiply(&WorldMatrix, &WorldMatrix, &TransMatrix);
    }
}

```

```

        // Set the matrix and render the entity.
        g_pDevice->SetTransform(D3DTS_WORLD, &WorldMatrix);
        (*i)->xmesh.Render();
    }
    g_pDevice->SetRenderState( D3DRS_FILLMODE, D3DFILL_SOLID );
}

// End the 3D rendering process.
g_pDevice->EndScene();

// Lock the 3D surface for rendering.
D3DLOCKED_RECT Locked;
g_pBackSurface->LockRect( &Locked, 0, 0 );

// Add any 2D rendering to the screen here that should cover up
// the 3D rendering. HUD, etc.
if(g_bShowFPS && (g_nStateFlag == 3))
{
    // Displays the Frames Per Second in the lower right.
    stringstream ss;
    ss << "FPS(" << g_FrameRate << ")";
    PrintString( (g_DeviceWidth - 75), (g_DeviceHeight - 50),
        (char *)ss.str().c_str(), TRUE, D3DCOLOR_ARGB( 255, 255, 0, 255 ),
        (DWORD*)Locked.pBits, Locked.Pitch );
}

if(g_bShowCameraLoc && (g_nStateFlag == 3))
{
    // Displays the camera's current location in the lower left.
    CZenCamera * g_Camera = CZenCamera::Instance();
    float x, y, z;
    g_Camera->GetPosition(x, y, z);

    stringstream ss;
    ss << "Camera(" << x << ", " << y << ", " << z << ")";

    PrintString( 50, (g_DeviceHeight - 50), (char *)ss.str().c_str(), TRUE,
        D3DCOLOR_ARGB( 255, 255, 0, 255 ), (DWORD*)Locked.pBits, Locked.Pitch );
}

if(g_bShowGravity && (g_nStateFlag == 3))
{
    // Displays gravity information in the bottom center.
    stringstream ss;
    ss << "Jump(" << g_dJumpVelocity << ") Gravity(" << g_dGravity << ")";

    PrintString( (g_DeviceHeight / 2) + 50, (g_DeviceHeight - 50),
        (char *)ss.str().c_str(), TRUE, D3DCOLOR_ARGB( 255, 255, 0, 255 ),
        (DWORD*)Locked.pBits, Locked.Pitch );
}

// Unlock the 2D surface since rendering is complete.
g_pBackSurface->UnlockRect();

// The console rendering call comes next to last since it should appear
// above the simulation.
CConsole * Console = CConsole::Instance();
Console->Render();

// Present the back buffer to the primary surface.
r = g_pDevice->Present( NULL, NULL, NULL, NULL );

// This block of code causes the simulator to pause on the Credits screen
// before closing the program, allowing the user to view the credits.
DWORD dwCurrentTime;
if(g_nStateFlag == 5)
{
    static DWORD dwStartTime = timeGetTime();
    dwCurrentTime = timeGetTime();
    if((dwCurrentTime - dwStartTime) >= 7000) // Milliseconds:
    {
        PostQuitMessage( 0 );
    }
}

return S_OK;

```

```

}

//-----
// Name: SimCleanup()
// Desc: This function is the counterpart to SimInit() and destroys all ptrs
//       created in SimInit().
//-----
int SimCleanup()
{
    if( Trycatch((void*)g_pCursorSurf, "g_pCursorSurf in SimCleanup()") )
    {
        g_pCursorSurf->Release();
    }

    DestroyScene();           // Helper function. Paired with InitScene().

    ShutdownInput();         // Break down DirectInput.

    CConsole * Console = CConsole::Instance();
    Console->Shutdown();      // Break down the console.

    UnloadAlphabet();        // Break down the alphabet we created for text.

    // Release the default X-Mesh texture pointer.
    if( Trycatch((void*)g_pDefaultTexture, "g_pDefaultTexture in SimCleanup()") )
    {
        g_pDefaultTexture->Release();
    }

    // Release the 2D surface pointer.
    if( Trycatch((void*)g_pBackSurface, "g_pBackSurface in SimCleanup()") )
    {
        g_pBackSurface->Release();
    }

    // Release the IDirect3DDevice9 pointer.
    if( Trycatch((void*)g_pDevice, "g_pDevice in SimCleanup()") )
    {
        g_pDevice->Release();
    }

    // Release the IDirect3D9 pointer.
    if( Trycatch((void*)g_pD3D, "g_pD3D in SimCleanup()") )
    {
        g_pD3D->Release();
    }

    return S_OK;
}

//-----
// Name: ConsoleParser()
// Desc: This is where commands can be added to the console.
//-----
int ConsoleParser( CCommand* pCommand )
{
    CConsole * Console = CConsole::Instance();
    char* pstrCmd = pCommand->pstrCommand;
    char* pstrParams[MAX_PARAMS];
    memcpy( &pstrParams, &(pCommand->pstrParams), sizeof( pstrParams ) );
    int NumParams = pCommand->NumParams;

    string sTemp;
    stringstream ss;
    list<LocalEntity *>::iterator i;

    // Exit the Simulator.
    if( MATCH( pstrCmd, "exit" ) || MATCH( pstrCmd, "quit" ) )
    {
        // This exit is a hard exit.
        PostQuitMessage(0);
        return 0;
    }
}

```



```

}

// Clear the console screen.
else if( MATCH( pstrCmd, "cls" ) || MATCH( pstrCmd, "clear" ) )
{
    Console->Clear();
    return 0;
}

// Toggle the visibility of a framerate count in the lower-right.
else if( MATCH( pstrCmd, "togglefps" ) )
{
    g_bShowFPS = !g_bShowFPS;
    return 0;
}

// List all of the command groups.
else if( MATCH( pstrCmd, "help" ) )
{
    Console->OutputString( "Clarity Console Help (Command List): ", false );
    Console->OutputString( " 'Help' - Displays this menu.", false );
    Console->OutputString( " 'LEC' - List all Entity-related commands.", _
        false );
    Console->OutputString( " 'LCC' - List all Camera-related commands.", _
        false );
    Console->OutputString( " 'LOC' - List all other commands.", false );
    Console->OutputString( " 'Clear' - Clears this console screen.", false );
    Console->OutputString( " 'Exit' - Hard exit of the simulation.", false );
    return 0;
}

// LEC: List all entity commands.
else if( MATCH( pstrCmd, "lec" ) )
{
    Console->OutputString( "Clarity Console Entity Command List: ", false );
    Console->OutputString( " 'EntityList' - Lists all loaded entities _
        and their IDs.", false );
    Console->OutputString( " 'GetEntityPos ID' - Display's an entity's _
        position.", false );
    Console->OutputString( " 'GetEntityTF ID' - Display's an entity's _
        rotation.", false );
    Console->OutputString( " 'GetEntityPosTF ID' - Displays both entity _
        position and rotation.", false );
    Console->OutputString( " 'GetEntityMeshInfo ID' - Display's the _
        filename of an entity's x-mesh as well as its size.", false );
    Console->OutputString( " 'GetEntityAttr ID' - Display's the base Mass _
        and the Friction values of the entity.", false );
    Console->OutputString( " 'ToggleEntityRenderMode' - Toggles the _
        render mode between solid and wireframe.", false );
    return 0;
}

// LCC: List all camera commands.
else if( MATCH( pstrCmd, "lcc" ) )
{
    Console->OutputString( "Clarity Console Camera Command List: ", false );
    Console->OutputString( " 'ToggleCamPos' - Toggles the display of the _
        camera's position.", false );
    Console->OutputString( " 'GetUserPos' - Return's the coordinates of _
        the user (camera).", false );
    Console->OutputString( " 'SetCameraSpeed X' - As the name implies. _
        Range: 1.0-100.0.", false );
    Console->OutputString( " 'SetCameraYaw X' - Sets how fast the _
        camera turns. Range: 0.01-0.50.", false );
    Console->OutputString( " 'ToggleCameraLock' - (Un)restricts the camera _
        to the terrain area.", false );
    return 0;
}

// LOC: List all other commands.
else if( MATCH( pstrCmd, "loc" ) )

```

```

    Console->OutputString( "Clarity Console General Command List: ", false );
    Console->OutputString( " 'ToggleFPS' - Toggles the display of the _
        current framerate.", false );
    Console->OutputString( " 'GetWorldName' - Displays the name of the _
        current terrain.", false );
    Console->OutputString( " 'GetBitmapFilename' - Displays the filename of _
        the terrain's bitmap.", false );
    Console->OutputString( " 'ToggleGravity' - Toggles the display of _
        current gravity values.", false );
    Console->OutputString( " 'SetJumpVelocity #' - Changes the strength of _
        the user's jump. Range: 1.0-100.0.", false );
    Console->OutputString( " 'SetGravity #' - Changes the strength of _
        gravity, dividing 9.8 by #. Range: 0.01-100.0.", false );
    return 0;
}

// Get the pointers to the necessary Singletons.
WorldSingleton * World = WorldSingleton::Instance();
TerrainSingleton * Terrain = TerrainSingleton::Instance();

/** Only allow these commands if the singletons have been filled with data. */

// Get the World Name.
if( MATCH( pstrCmd, "getworldname" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        Console->OutputString( (char *)World->sWorldName.c_str(), false );
        return S_OK;
    }
}

// Get Bitmap Filename.
else if( MATCH( pstrCmd, "getbitmapfilename" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        Console->OutputString( (char *)World->sBitmapFilename.c_str(), _
            false );
        return S_OK;
    }
}

// Get User Information.
else if( MATCH( pstrCmd, "getuserpos" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        CZenCamera * g_Camera = CZenCamera::Instance();
        float x, y, z;
        g_Camera->GetPosition(x, y, z);
    }
}

```

```

        ss << "User Position: (" << x << ", " << y << ", " << z << ")";

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Get Entity Information.
else if( MATCH( pstrCmd, "entitylist" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        // Add in a counter later to handle the case where there are too
        // many entities to display at once.
        Console->OutputString( "Entities Include: ", false );

        for(i = World->lstLocalEntities.begin(); _
            i != World->lstLocalEntities.end(); i++)
        {
            ss << " " << (*i)->ID << " " << (*i)->name _
                << ". XMesh: " << (*i)->xfile;
            Console->OutputString( (char *)ss.str().c_str(), false );
            ss.str("");
        }

        return S_OK;
    }
}

// Get Entity position and transformation.
else if( MATCH( pstrCmd, "getentitypostf" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
                for this command.", true );
            return S_OK;
        }

        int ID;
        stringstream ssID;
        ssID << pstrParams[0];
        ssID >> ID;

        for(i = World->lstLocalEntities.begin(); _
            i != World->lstLocalEntities.end(); i++)
        {
            if(ID == (*i)->ID)
            {
                ss << (*i)->name << ": Pos(" << (*i)->x * 2 _
                    << ", " << (*i)->z _
                    + Terrain->GetHeight((float)(*i)->x * 2, _
                    (float)(*i)->y * 2) << ", " << (*i)->y * 2;
                ss << "), Roll(" << (*i)->roll << "), Pitch(" _
                    << (*i)->pitch;
                ss << "), Yaw(" << (*i)->yaw << ")";
                break;
            }
        }
    }
}

```

```

    }
}

Console->OutputString( (char *)ss.str().c_str(), false );
return S_OK;
}

// Get Entity position.
else if( MATCH( pstrCmd, "getentitypos" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
for this command.", true );
            return S_OK;
        }

        int ID;
        stringstream ssID;
        ssID << pstrParams[0];
        ssID >> ID;

        for(i = World->lstLocalEntities.begin(); _
i != World->lstLocalEntities.end(); i++)
        {
            if(ID == (*i)->ID)
            {
                ss << (*i)->name << " Position: (" _
                << (*i)->x * 2.0 << ", " _
                << (*i)->z + Terrain->GetHeight((float)(*i)->x _
                * 2, (float)(*i)->y * 2) << ", " _
                << (*i)->y * 2.0 << ")";
                break;
            }
        }

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Get Entity transformation.
else if( MATCH( pstrCmd, "getentitytf" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
for this command.", true );
            return S_OK;
        }

        int ID;
        stringstream ssID;
        ssID << pstrParams[0];

```

```

        ssID >> ID;

        for(i = World->lstLocalEntities.begin(); _
        i != World->lstLocalEntities.end(); i++)
        {
            if(ID == (*i)->ID)
            {
                ss << (*i)->name << " Transformation: Roll(" _
                << (*i)->roll << " ), Pitch(" << (*i)->pitch;
                ss << " ), Yaw(" << (*i)->yaw << " )";
                break;
            }
        }

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Get Entity X-Mesh information.
else if( MATCH( pstrCmd, "getentitymeshinfo" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
        WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
            for this command.", true );
            return S_OK;
        }

        int ID;
        stringstream ssID;
        ssID << pstrParams[0];
        ssID >> ID;

        for(i = World->lstLocalEntities.begin(); _
        i != World->lstLocalEntities.end(); i++)
        {
            if(ID == (*i)->ID)
            {
                ss << (*i)->name << ": XMesh(" << (*i)->xfile _
                << " ), Width(" << (*i)->width;
                ss << " ), Height(" << (*i)->height _
                << " ), Depth(" << (*i)->depth<< " )";
                break;
            }
        }

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Get Entity attributes of mass and friction.
else if( MATCH( pstrCmd, "getentityattr" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
        WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {

```

```

    if( NumParams != 1 )
    {
        Console->OutputString( "Incorrect number of parameters _
            for this command.", true );
        return S_OK;
    }

    int ID;
    stringstream ssID;
    ssID << pstrParams[0];
    ssID >> ID;

    for(i = World->lstLocalEntities.begin(); _
        i != World->lstLocalEntities.end(); i++)
    {
        if(ID == (*i)->ID)
        {
            ss << (*i)->name << ": Mass(" << (*i)->mass _
                << ")", Friction(" << (*i)->friction << " );
            break;
        }
    }

    Console->OutputString( (char *)ss.str().c_str(), false );
    return S_OK;
}

// Set the speed for the camera.
else if( MATCH( pstrCmd, "setcameraspeed" ) || MATCH( pstrCmd, "scs" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
                for this command.", true );
            return S_OK;
        }

        float fSpeed;
        stringstream sss;
        sss << pstrParams[0];
        sss >> fSpeed;

        g_fCameraSpeed = fSpeed;
        ss << "Camera Speed set to: " << fSpeed;

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Set the turn speed for the camera.
else if( MATCH( pstrCmd, "setcamerayaw" ) || MATCH( pstrCmd, "scy" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )

```

```

    {
        Console->OutputString( "Incorrect number of parameters _
            for this command.", true );
        return S_OK;
    }

    float fSpeed;
    stringstream sss;
    sss << pstrParams[0];
    sss >> fSpeed;

    g_fCameraYaw = fSpeed;
    ss << "Camera Yaw set to: " << fSpeed;

    Console->OutputString( (char *)ss.str().c_str(), false );
    return S_OK;
}

// Lock or unlock the camera's restriction to the terrain's surface.
else if( MATCH( pstrCmd, "togglecameralock" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        g_bCameraLocked = !g_bCameraLocked;

        if(g_bCameraLocked)
        {
            ss << "The camera is now locked above the terrain.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        else
        {
            ss << "The camera is now unlocked.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        return S_OK;
    }
}

// Toggle the display of the camera information on the screen.
else if( MATCH( pstrCmd, "togglecampos" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        g_bShowCameraLoc = !g_bShowCameraLoc;
        if(g_bShowCameraLoc)
        {
            ss << "The camera location is now visible.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        else
        {
            ss << "The camera location is now hidden.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        return S_OK;
    }
}
}

```

```

// Toggle how entities are rendered: solid or wireframe.
else if( MATCH( pstrCmd, "toggleentityrendermode" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
        WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        g_bEntityWireframe = !g_bEntityWireframe;
        if(g_bEntityWireframe)
        {
            ss << "Entity render state changed to: Wireframe.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        else
        {
            ss << "Entity render state changed to: Normal.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        return S_OK;
    }
}

// Toggle the display of gravity information to the screen.
else if( MATCH( pstrCmd, "togglegravity" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
        WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        g_bShowGravity = !g_bShowGravity;
        if(g_bShowGravity)
        {
            ss << "Gravity information is now visible.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        else
        {
            ss << "Gravity information is now hidden.";
            Console->OutputString( (char *)ss.str().c_str(), false );
        }
        return S_OK;
    }
}

// Change the power of the camera jump.
else if( MATCH( pstrCmd, "setjumpvelocity" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
        WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
            for this command.", true );
            return S_OK;
        }
    }
}

```



```

        double dStrength;
        stringstream sss;
        sss << pstrParams[0];
        sss >> dStrength;

        g_dJumpVelocity = dStrength;
        ss << "Initial user jump velocity set to: " << dStrength;

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

// Change the strength of the environment's gravity.
else if( MATCH( pstrCmd, "setgravity" ) )
{
    if(World->bIsEmpty)
    {
        Console->OutputString( "Command unavailable while the _
            WorldSingleton is empty.", true );
        return S_OK;
    }
    else
    {
        if( NumParams != 1 )
        {
            Console->OutputString( "Incorrect number of parameters _
                for this command.", true );
            return S_OK;
        }

        double dFactor;
        stringstream sss;
        sss << pstrParams[0];
        sss >> dFactor;

        g_dGravityFactor = dFactor;
        g_dGravity = 9.8 / g_dGravityFactor;
        ss << "Gravity value of 9.8 now divided by: " << dFactor;

        Console->OutputString( (char *)ss.str().c_str(), false );
        return S_OK;
    }
}

return -1;
}

HRESULT InitializeInput()
{
    if( Trycatch((void*)g_pDI, "g_pDI in InitializeInput()") )
    {
        g_pDI->Release();
    }

    HRESULT r = 0;

    // Create the IDirectInput8 object.
    r = DirectInput8Create( g_hInstMain, DIRECTINPUT_VERSION, _
        IID_IDirectInput8, (void**)&g_pDI, NULL );
    if( FAILED( r ) )
    {
        Debug( "Failed to create DirectInput," );
        return E_FAIL;
    }

    // Initialize the keyboard.
    CZenKeyboard * g_Keyboard = CZenKeyboard::Instance();
    r = g_Keyboard->Initialize();
    if( FAILED( r ) )
    {

```

```

        Debug( "Keyboard initialization failed." );
        return E_FAIL;
    }

    // Initialize the mouse.
    CZenMouse * g_Mouse = CZenMouse::Instance();
    r = g_Mouse->Initialize();
    if( FAILED( r ) )
    {
        Debug( "Mouse initialization failed." );
        return E_FAIL;
    }

    return S_OK;
}

HRESULT ShutdownInput()
{
    // Get rid of the IDirectInput8 object.
    if( Trycatch((void*)g_pDI, "g_pDI in ShutdownInput()") )
    {
        g_pDI->Release();
        g_pDI = 0;
    }
    return E_FAIL;
}

vector<string> GetWIDFileNames()
{
    WIN32_FIND_DATA FindFileData;
    HANDLE hFind = INVALID_HANDLE_VALUE;
    char szDirectory[80] = "xml\\*.wid";
    DWORD dwError;
    vector<string> szFileNames;

    hFind = FindFirstFile(szDirectory, &FindFileData);

    if (hFind == INVALID_HANDLE_VALUE)
    {
        Debug("Invalid Handle in GetWIDFileNames()!");
        return szFileNames; // Return an empty vector
    }
    else
    {
        Debug(FindFileData.cFileName);
        szFileNames.push_back(FindFileData.cFileName);
        while (FindNextFile(hFind, &FindFileData) != 0)
        {
            Debug(FindFileData.cFileName);
            szFileNames.push_back(FindFileData.cFileName);
        }

        dwError = GetLastError();
        FindClose(hFind);
        if (dwError != ERROR_NO_MORE_FILES)
        {
            Debug("Unknown error in GetWIDFileNames().");
            return szFileNames;
        }
    }
    return szFileNames; // Is this even needed?
}

```

```

//-----
// File:      Common.h
//
// Desc:      This file contains the common include files for ProjectOrigins.cpp
//            that were not created by the author.
//
// First created on:  December 27th, 2004
// Last modification:  January 5th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 12-27-04:  Added revision history. File created.
// 12-29-04:  All global variables commented.
// 01-02-05:  Added a function pointer definition for the console.
// 01-05-05:  Added a world function pointer definition for loading world
//            files. Also added some other global variables.
//-----

#define WIN32_LEAN_AND_MEAN
// #define QUIET_MODE

// Includes.
#include <Windows.h>
#include <commctrl.h>
#include <stdio.h>
#include <math.h>
#include <mmsystem.h> // timeGetTime()

#include <stdlib.h>
#include <malloc.h>
#include <memory.h>
#include <tchar.h>

// STL Includes.
#include <list>
#include <vector>
#include <string>
#include <sstream>

#import "msxml4.dll"

// DX9 Includes.
#include <D3DX9.h>
#include "DXUtil.h"
#include "D3DEnumeration.h"
#include "D3DSettings.h"
#include "D3DApp.h"
#include "D3DFile.h"
#include "D3DFont.h"
#include "D3DUtil.h"
#include "dinput.h"

using namespace std;

// Constants.
const int MAX_CHARS_PER_LINE = 256; // Max length of a line of text in the console.
const int MAX_PARAMS = 25; // For the console. Params of console commands.
const int MAX_FONTS = 20; // Maximum number of fonts in the Fontbank class.
const int PAUSE_WAIT = 250; // Milliseconds between issuing Pause or
// Resume commands in screens.h.

#define MATCH(a, b) (!strcmp( a, b )) // Used for matching console commands.

// Fixed-Function Vertex structure.
#define ZENVERTEX_TYPE (D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_DIFFUSE | D3DFVF_SPECULAR |
| D3DFVF_TEX1 )

/** Typedefs */
class CCommand;
class CZenFrame;

```

```

void PrintString( int, int, char*, BOOL, D3DCOLOR, DWORD*, int);
typedef int (*CONSOLE_PARSER_CALLBACK)( CCommand* pCommand ); // Console parser.
typedef int (*FRAME_MOVEMENT_CALLBACK)( CZenFrame* pFrame, void* Parameter );
typedef void(*VoidFuncPtr)(); // A void and null function pointer.
typedef void(*WorldFuncPtr)( string sWorldFile ); // A call to a world-loading function.

/**/ Global Flags /**/
BOOL g_bActive; // Activation flag from windows to start my loop.
int g_DeviceHeight = 0; // Dimension of the D3D Device.
int g_DeviceWidth = 0; // Dimension of the D3D Device.
int g_nStateFlag = 0; // Simulation state. Global and independant since
// the primary functions directly deal with states.

bool g_bShowFPS = true; // Display the FPS.
bool g_bShowCameraLoc = true; // Display the Camera Location.
bool g_bShowGravity = true; // Display gravity information.
static UINT g_LightCounter = 0; // Used for the CZenLight class.
bool g_bConsoleOn = false; // Used to halt Simulator input while the
// console is active.

bool g_bPauseLock = false; // Used to prevent input while switching states
// between Sim and Pause.

float g_fCameraSpeed = 5.0; // The speed at which the camera moves when the
// user inputs actions.

float g_fCameraYaw = 0.15; // The speed at which the camera turns.
bool g_bCameraLocked = true; // The camera cannot leave the bounded region
// above the terrain.

bool g_bTerrainLoaded = false; // This variable is 'true' when there is valid,
// loaded terrain.

/**/ Global Variables /**/
LPDIRECT3D9 g_pD3D(0); // This is a pointer to the D3D object.
LPDIRECT3DDEVICE9 g_pDevice(0); // This is a pointer to the D3D Device.
D3DPRESENT_PARAMETERS g_SavedPresParams; // D3D parameters. Stored in
// D3DDeviceInit().

HWND g_hWndMain(0); // This is a handle to the main window.
// Initialized in WinMain().

HINSTANCE g_hInstMain(0); // Global copy of the handle to this program
// instance passed to WinMain().

LPDIRECTINPUT8 g_pDI = 0; // DirectInput object.
bool g_bEntityWireframe = false; // When true, renders entities as wireframes.
DWORD g_dwTerrainColor = 0x0000FF00; // Color of the terrain.
DWORD g_dwTerrainWireColor = 0x00000000; // Color of the terrain's wireframe.

LPDIRECT3DSURFACE9 g_pBackground = 0; // The background image surface.
LPDIRECT3DSURFACE9 g_pBackSurface = 0; // Related to g_pBackground. Probably used
// for 2D rendering.

LPDIRECT3DTEXTURE9 g_pDefaultTexture = 0; // Default X-Mesh texture.
LPDIRECT3DSURFACE9 g_pCursorSurf = 0; // Image for the mouse pointer.

// Function prototypes.
HRESULT InitScene();
HRESULT DestroyScene();
int SimInit();
int SimLoop();
int SimCleanup();
int SimRender();
HRESULT HandleInput();
int ConsoleParser( CCommand* pCommand );
HRESULT InitializeInput();
HRESULT ShutdownInput();
vector<string> GetWIDFileNames();

```

```

-----
// File:      Debug.h
//
// Desc:      This is a file for testing purposes (drivers, debug calls, etc.)
//
// First created on:  November 19th 2004
// Last modification:  December 29th, 2004
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 11-19-04:   Created this file. Added Debug().
// 12-28-04:   Added Trycatch() and successfully tested it.
// 12-29-04:   Added DebugPtr() and successfully tested it.
//-----

void Debug(char * szDebug)
{
    OutputDebugString( "Error: " );
    OutputDebugString( szDebug );
    OutputDebugString( "\n" );
}

void Debug(const char * szDebug)
{
    OutputDebugString( "Error: " );
    OutputDebugString( szDebug );
    OutputDebugString( "\n" );
}

void DebugPtr(void * ptr)
{
    // Outputs the address of a ptr to Debug().
    stringstream str;
    str << (void*)ptr;
    string tempStr;
    str >> tempStr;
    char * cString = (char *)tempStr.c_str();
    Debug(cString);
}

bool Trycatch(void * ptr, string pName)
{
    // Returns '1' if the ptr is valid, throws an exception and returns '0' otherwise.
    try
    {
        if(!ptr)
        {
            pName = pName + " - invalid pointer.";
            throw( pName.c_str() );
        }
        return 1;
    }
    catch( const char * str )
    { Debug(str); return 0; }
}

bool QTrycatch(void * ptr, string pName) // Quiet version.
{
    // Returns '1' if the ptr is valid, throws an exception and returns '0' otherwise.
    try
    {
        if(!ptr)
        {
            pName = pName + " - invalid pointer.";
            throw( pName.c_str() );
        }
        return 1;
    }
    catch( const char * str ) // Debug(str) would go here, but this is quiet.
    { return 0; }
}

```

```

//-----
// File:      auxd3d.h
//
// Desc:      This file contains all of the auxiliary functions that pertain to
//            the Direct3D functions in the main .cpp file.
//
// First created on:  December 28th, 2004
// Last modification: December 29th, 2004
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//            Direct3D Game Programming"
//-----
// Revision History:
//
// 12-28-04:  Added revision history. File created. Converted code to DX9.
// 12-29-04:  Cleaned up and recommented all of the code in this file.
//-----

/*****
* Section:      Direct3D Initialization Helper Function
*****/

int InitDirect3DDevice( HWND hWndTarget, int Width, int Height, BOOL bWindowed, D3DFORMAT
FullScreenFormat, LPDIRECT3D9 pd3D, LPDIRECT3DDEVICE9* ppDevice )
{
    // Structure to hold information about the rendering method.
    D3DPRESENT_PARAMETERS d3dpp;
    // Structure to hold information about the current display mode.
    D3DDISPLAYMODE d3ddm;
    HRESULT r = 0;

    if( *ppDevice )        // If the device already exists, release it.
    {
        (*ppDevice)->Release();
    }

    // Initialize the structure to 0.
    ZeroMemory( &d3dpp, sizeof( D3DPRESENT_PARAMETERS ) );

    // Get the settings for the current display mode.
    r = pd3D->GetAdapterDisplayMode( D3DADAPTER_DEFAULT, &d3ddm );
    if( FAILED( r ) )
    {
        Debug( "Could not get display adapter information." );
        return E_FAIL;
    }

    // The width of the back buffer in pixels.
    d3dpp.BackBufferWidth = Width;
    // The height of the buffer in pixels.
    d3dpp.BackBufferHeight = Height;
    // The format of the back buffer.
    d3dpp.BackBufferFormat = bWindowed ? d3ddm.Format : FullScreenFormat;
    // The number of back buffers.
    d3dpp.BackBufferCount = 1;

    // The type of multisampling.
    d3dpp.MultiSampleType = D3DMULTISAMPLE_NONE;
    // The swap effect.
    d3dpp.SwapEffect = D3DSWAPEFFECT_COPY;

    // The handle to the window that we want to render to.
    d3dpp.hDeviceWindow = hWndTarget;
    // Windowed or fullscreen?
    d3dpp.Windowed = bWindowed;

    // Let Direct3D manage the depth buffer.
    d3dpp.EnableAutoDepthStencil = TRUE;
    // Set the depth buffer format to 16 bits.
    d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
}

```

```

// Use the default refresh rate available.
d3dpp.FullScreen_RefreshRateInHz = D3DPRESENT_RATE_DEFAULT;

// Present the information as fast as possible.
d3dpp.PresentationInterval = bWindowed ? 0 : D3DPRESENT_INTERVAL_ONE;
// Allow the back buffer to be accessed for 2D rendering.
d3dpp.Flags = D3DPRESENTFLAG_LOCKABLE_BACKBUFFER;

// Acquire a pointer to IDirect3DDevice9.
r = pD3D->CreateDevice( D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, hWndTarget,
    D3DCREATE_SOFTWARE_VERTEXPROCESSING, &d3dpp, ppDevice );
if( FAILED( r ) )
{
    Debug( "Could not create the render device." );
    return E_FAIL;
}

// Save global copies of the device dimensions.
g_DeviceHeight = Height;
g_DeviceWidth = Width;

// Save a copy of the pres-params for use in device validation later.
g_SavedPresParams = d3dpp;

return S_OK;
}

/*****
* Section:      Direct3D Device Validation
*****/

HRESULT InitScene();

// Use this function to reinit any surfaces that were lost when the device was lost.
HRESULT RestoreGraphics()
{
    InitScene(); // I don't think anything else is needed here.
    return S_OK;
}

// Call every frame to check if the device is valid.
// If it is not then it is reaquired if possible.
HRESULT ValidateDevice()
{
    HRESULT r = 0;

    // Test the current state of the device.
    r = g_pDevice->TestCooperativeLevel();
    if( FAILED( r ) )
    {
        // If the device is lost then return failure.
        if( r == D3DERR_DEVICELOST )
        {
            return E_FAIL;
        }

        // If the device is ready to be reset then attempt to do so.
        if( r == D3DERR_DEVICENOTRESET )
        {
            // Release the back surface so it can be recreated.
            g_pBackSurface->Release();

            // Reset the device.
            r = g_pDevice->Reset( &g_SavedPresParams );
            if( FAILED( r ) )
            {
                // If the device was not reset then exit the simulation.
                Debug( "Could not reset device." );
                PostQuitMessage( E_FAIL );
                return E_FAIL;
            }
        }
    }
}

```

```

    }

    // Require a pointer to the new back buffer.
    r = g_pDevice->GetBackBuffer( 0, 0, D3DBACKBUFFER_TYPE_MONO, _
        &g_pBackSurface );
    if( FAILED( r ) )
    {
        Debug( "Unable to require the back buffer." );
        PostQuitMessage( 0 );
        return E_FAIL;
    }

    g_pDevice->Clear( 0, NULL, D3DCLEAR_TARGET, D3DCOLOR_XRGB( 0, 0, _
        0 ), 0.0f, 0 );
    RestoreGraphics();
}
return S_OK;
}

/*****
+ Section:      Matrix Manipulation
*****/

HRESULT CreateViewport()
{
    // Creates a viewport.
    HRESULT r = 0;

    if( !g_pDevice )
    {
        return E_FAIL;
    }

    D3DVIEWPORT9 Viewport;

    Viewport.X = 0;
    Viewport.Y = 0;
    Viewport.Width = g_DeviceWidth;
    Viewport.Height = g_DeviceHeight;
    Viewport.MinZ = 0.0f;
    Viewport.MaxZ = 1.0f;

    r = g_pDevice->SetViewport( &Viewport );

    return r;
}

void SetProjectionMatrix()
{
    // Sets up the projection matrix.
    D3DXMATRIX ProjectionMatrix;
    ZeroMemory( &ProjectionMatrix, sizeof( D3DXMATRIX ) );

    float ScreenAspect = (float)g_DeviceWidth / (float)g_DeviceHeight;
    float FOV = D3DX_PI / 4;

    D3DXMatrixPerspectiveFovLH( &ProjectionMatrix, FOV, ScreenAspect, 1.0f, 1000.0f );

    g_pDevice->SetTransform( D3DTS_PROJECTION, &ProjectionMatrix );
}

```



```

//-----
// File:      zen.h
//
// Desc:      This file contains the bulk of the Zen classes and functions.
//
// First created on:  December 28th, 2004
// Last modification:  January 13th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//                                     Direct3D Game Programming"
//-----
// Revision History:
//
// 12-28-04:  Added revision history. File created. Converted code to DX9.
// 12-29-04:  Began to separate content out of this file into other .h files.
// 01-02-05:  Cleaned up and recommented the code up to CZenMaterial.
// 01-03-05:  Switched D3DPOOL_DEFAULT to D3DPOOL_SYSTEMMEM in
//            LoadBitmapToSurface. This fixed the slow console problem.
//            Finished cleaning and recommented the code in this file.
// 01-03-05:  Added SetColor(), RestoreColor() and GetBoundingBox() to
//            the CZenFont class. This allows for text display on menus.
// 01-04-05:  Converted the CZenCamera class to a Singleton.
// 01-13-05:  Removed the Point, Line, and related classes since they're not
//            needed. Also made adjustments to the camera and cube classes.
//-----

/*****
* Section:    2D Direct3D Graphics Functions
*****/

int LoadBitmapToSurface( char* PathName, LPDIRECT3DSURFACE9* ppSurface, LPDIRECT3DDEVICE9
pDevice )
{
    // Loads a bitmap to a surface.
    HRESULT r;
    HBITMAP hBitmap;
    BITMAP Bitmap;

    // Load the bitmap using the GDI to get information.
    hBitmap = (HBITMAP)LoadImage( NULL, PathName, IMAGE_BITMAP, 0, 0, _
        LR_LOADFROMFILE | LR_CREATEDIBSECTION );

    if( hBitmap == NULL ) // The file probably does not exist.
    {
        Debug( "Unable to load bitmap." );
        return E_FAIL;
    }

    // Get information about the object.
    GetObject( hBitmap, sizeof( BITMAP ), &Bitmap );
    // Unload the bitmap from memory.
    DeleteObject( hBitmap );

    // Create a surface using information from LoadImage().
    r = pDevice->CreateOffscreenPlainSurface( Bitmap.bmWidth, Bitmap.bmHeight, _
        D3DFMT_A8R8G8B8, D3DPOOL_SYSTEMMEM, ppSurface, NULL );
    if( FAILED( r ) )
    {
        Debug( "Unable to create surface for bitmap load." );
        return E_FAIL;
    }

    // Load the image directly to the new surface.
    r = D3DXLoadSurfaceFromFile( *ppSurface, NULL, NULL, PathName, NULL, _
        D3DX_FILTER_NONE, 0, NULL ); // 0xFF999999.
    if( FAILED( r ) )
    {
        Debug( "Unable to load file to surface." );
        return E_FAIL;
    }
}

```

```

    return S_OK;
}

/*****
* Section:      Timing and Framerate Functions
*****/

INT64 g_Frequency = 0;          // The number of high performance ticks per second.
int g_FrameCount = 0;          // The number of elapsed frames this counting period.
int g_FrameRate = 0;           // The number of elapsed frames this second.
float g_FrameDeviance = 0;     // Percentage the frame rate has changed from 25fps.

HRESULT InitTiming()
{
    // Get the number of counts per second.
    QueryPerformanceFrequency( (LARGE_INTEGER*)&g_Frequency );

    // If the frequency is 0 then this system does not have high performance timers.
    if( g_Frequency == 0 )
    {
        Debug( "The system does not support high resolution timing." );
        return E_FAIL;
    }
    return S_OK;
}

void Pause( int Milliseconds )
{
    // Pause the simulation for a certain time.
    INT64 SecondsDelay = (INT64)Milliseconds * 1000;
    INT64 StartTime;
    INT64 CurrentTime;

    QueryPerformanceCounter( (LARGE_INTEGER*)&StartTime );

    while(1)
    {
        QueryPerformanceCounter( (LARGE_INTEGER*)&CurrentTime );
        if( (CurrentTime - StartTime) > (INT64)SecondsDelay )
        {
            break;
        }
    }
}

float GetNumTicksPerMs()
{
    // Returns the number of ticks in a millisecond.
    return ((float)g_Frequency / 1000.0f);
}

void FrameCount()
{
    INT64 NewCount = 0;          // The current count.
    static INT64 LastCount = 0; // The last count.
    INT64 Difference = 0;       // The difference since the last count.

    // Get the current frame count.
    QueryPerformanceCounter( (LARGE_INTEGER*)&NewCount );

    // If the count is 0 then this system does not have high performance timers.
    if( NewCount == 0 )
    {
        Debug( "The system does not support high resolution timing" );
    }

    // Increase the frame count.
    g_FrameCount++;

    // Compute the difference since the last count.
    Difference = NewCount - LastCount;
}

```

```

// If more than a second has passed.
if( Difference >= g_Frequency )
{
    g_FrameRate = g_FrameCount; // Record the number of elapsed frames.
    g_FrameCount = 0; // Reset the counter.
    LastCount = NewCount; // Update the last count.
}

g_FrameDeviance = (float)g_FrameRate / 25.0f;
}

/*****
* Section: 3D Direct3D Functions
*****/

/** This class holds vertex information. */
class CZenVertex
{
public:
    CZenVertex();
    CZenVertex( float x, float y, float z, float nx, float ny, float nz,
        D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv);
    ~CZenVertex();

public:
    void Set( float x, float y, float z, float nx, float ny, float nz,
        D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv);

public:
    D3DVECTOR m_Position;
    D3DVECTOR m_Normal;
    D3DCOLOR m_DiffuseColor;
    D3DCOLOR m_SpecularColor;
    float m_tu, m_tv;
};

CZenVertex::CZenVertex()
{
    ZeroMemory( &m_Position, sizeof( D3DVECTOR ) );
    ZeroMemory( &m_Normal, sizeof( D3DVECTOR ) );
    ZeroMemory( &m_SpecularColor, sizeof( D3DCOLOR ) );

    m_DiffuseColor = D3DCOLOR_ARGB( 255, 255, 255, 255 );

    m_tu = m_tv = 0.0f;
}

CZenVertex::CZenVertex( float x, float y, float z, float nx, float ny, float nz,
    D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv)
{
    m_Position.x = x;
    m_Position.y = y;
    m_Position.z = z;

    m_Normal.x = nx;
    m_Normal.y = ny;
    m_Normal.z = nz;

    m_DiffuseColor = DiffuseColor;
    m_SpecularColor = SpecularColor;
    m_tu = tu;
    m_tv = tv;
}

CZenVertex::~CZenVertex()
{
    // Nothing to destruct.
}

void CZenVertex::Set( float x, float y, float z, float nx, float ny, float nz,
    D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv)
{
    m_Position.x = x;

```

```

    m_Position.y = y;
    m_Position.z = z;

    m_Normal.x = nx;
    m_Normal.y = ny;
    m_Normal.z = nz;

    m_DiffuseColor = DiffuseColor;
    m_SpecularColor = SpecularColor;
    m_tu = tu;
    m_tv = tv;
}

/** This class is a base class for all other objects. */
class CZenObject
{
public:
    CZenObject();
    ~CZenObject();
    CZenObject( CZenObject& OtherObject );

public:
    virtual HRESULT Render();
    void SetNext( CZenObject* pNext ){ m_pNext = pNext; }
    void* GetNext(){ return m_pNext; }
    void* GetParentFrame(){ return m_pParentFrame; }
    void SetParentFrame( void* pFrame ){ m_pParentFrame = pFrame; }
    virtual int GetSize(){ return sizeof( *this ); }

public:
    char* m_strName;
    void* m_pParentFrame;
protected:
    CZenObject* m_pNext;
};

CZenObject::CZenObject( CZenObject& OtherObject )
{
    m_strName = OtherObject.m_strName;
    m_pParentFrame = OtherObject.m_pParentFrame;
    m_pNext = OtherObject.m_pNext;
}

CZenObject::CZenObject()
{
    m_strName = 0;
    m_pNext = 0;
    m_pParentFrame = NULL;
}

CZenObject::~CZenObject()
{
    if( m_strName )
    {
        delete m_strName;
    }
}

HRESULT CZenObject::Render()
{
    return S_OK;
}

/** A face (triangle) objects. Contains geometry and rendering functionality. */
class CZenFace : public CZenObject
{
public:
    CZenFace();
    ~CZenFace();
    CZenFace( CZenFace& OtherFace );

public:
    void SetProps( int Vertex, float x, float y, float z, float nx, float ny,
        float nz, D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv );
};

```

```

    HRESULT SetTexture( LPDIRECT3DTEXTURE9 pTexture );
    HRESULT SetTexture( char* strPathName );
    HRESULT Render();
    int GetSize(){ return sizeof( *this ); }
protected:
    CZenVertex m_Vertices[3];
    LPDIRECT3DTEXTURE9 m_pTexture;
    BOOL m_bTextureSet;
};

CZenFace::CZenFace( CZenFace& OtherFace )
{
    m_bTextureSet = OtherFace.m_bTextureSet;
    m_pTexture = OtherFace.m_pTexture;
    m_pTexture->AddRef();
    CopyMemory( &m_Vertices, &OtherFace.m_Vertices, sizeof( m_Vertices ) );
}

CZenFace::CZenFace()
{
    m_pTexture = 0;
    m_bTextureSet = FALSE;
}

CZenFace::~CZenFace()
{
    if( Trycatch((void*)m_pTexture, "m_pTexture in ~CZenFace()") )
    {
        m_pTexture->Release();
    }
}

HRESULT CZenFace::SetTexture( LPDIRECT3DTEXTURE9 pTexture )
{
    // Sets the texture using a texture from memory.
    // Make sure a valid texture was specified.
    if( !Trycatch((void*)pTexture, "pTexture in CZenFace::SetTexture()") )
    {
        m_bTextureSet = FALSE;
        return E_FAIL;
    }

    // If a texture is already assigned then release it.
    if( Trycatch((void*)m_pTexture, "m_pTexture in CZenFace::SetTexture()") )
    {
        m_pTexture->Release();
    }

    // Set the new texture.
    m_pTexture = pTexture;
    m_pTexture->AddRef();
    m_bTextureSet = TRUE;

    return S_OK;
}

HRESULT CZenFace::SetTexture( char* strPathName )
{
    // Set the texture using a texture from the disk.
    HRESULT r = 0;

    // Release the current texture if one is set.
    if( Trycatch((void*)m_pTexture, "m_pTexture in CZenFace::SetTexture()") )
    {
        m_pTexture->Release();
    }

    // Load the texture from disk.
    r = D3DXCreateTextureFromFile( g_pDevice, strPathName, &m_pTexture );
    if( SUCCEEDED( r ) )
    {
        m_bTextureSet = TRUE;
        return S_OK;
    }
}

```

```

    }

    return E_FAIL;
}

// Sets the properties for the face vertices.
void CZenFace::SetProps( int Vertex, float x, float y, float z, float nx, float ny,
float nz, D3DCOLOR DiffuseColor, D3DCOLOR SpecularColor, float tu, float tv )
{
    m_Vertices[Vertex].m_Position.x = x;
    m_Vertices[Vertex].m_Position.y = y;
    m_Vertices[Vertex].m_Position.z = z;

    m_Vertices[Vertex].m_Normal.x = nx;
    m_Vertices[Vertex].m_Normal.y = ny;
    m_Vertices[Vertex].m_Normal.z = nz;

    m_Vertices[Vertex].m_DiffuseColor = DiffuseColor;
    m_Vertices[Vertex].m_SpecularColor = SpecularColor;
    m_Vertices[Vertex].m_tu = tu;
    m_Vertices[Vertex].m_tv = tv;
}

HRESULT CZenFace::Render()
{
    HRESULT r = 0;
    LPDIRECT3DVERTEXBUFFER9 pVB = 0;

    // Create the vertex buffer.
    r = g_pDevice->CreateVertexBuffer( sizeof( CZenVertex ) * 3, D3DUSAGE_WRITEONLY, _
ZENVERTEX_TYPE, D3DPOOL_DEFAULT, &pVB, NULL );
    if( FAILED( r ) )
    {
        return E_FAIL;
    }

    BYTE* pData = 0; // Pointer to the vertex buffer.

    // Lock the vertex buffer.
    r = pVB->Lock( 0, 0, (void**)&pData, 0 );
    if( FAILED( r ) )
    {
        pVB->Release();
        return E_FAIL;
    }

    // Copy the face vertices into the buffer.
    CopyMemory( pData, (void*)&m_Vertices, sizeof( CZenVertex ) * 3 );

    // Unlock the vertex buffer.
    pVB->Unlock();

    // Setup the texture for the face.
    if( QTrycatch((void*)m_pTexture, "m_pTexture in CZenFace::Render()") )
    {
        g_pDevice->SetTexture( 0, m_pTexture );
    }

    // Connect the vertex buffer to a rendering stream.
    g_pDevice->SetStreamSource( 0, pVB, 0, sizeof( CZenVertex ) );

    // Render the face.
    g_pDevice->DrawPrimitive( D3DPT_TRIANGLELIST, 0, 1 );

    // Reset the texture.
    if( QTrycatch((void*)m_pTexture, "m_pTexture in CZenFace::Render()") )
    {
        g_pDevice->SetTexture( 0, NULL );
    }
}

```

```

        // Release the vertex buffer.
        pVB->Release();

        return S_OK;
    }

    /** This is the material (lighting surface) of an object. */
    class CZenMaterial
    {
    public:
        CZenMaterial();
        ~CZenMaterial();

    public:
        void SetDiffuse( float r, float g, float b );
        void SetAmbient( float r, float g, float b );
        void SetSpecular( float r, float g, float b, float Power );
        void SetEmissive( float r, float g, float b );
        HRESULT Update();

    public:
        D3DMATERIAL9 m_Material;
    };

    CZenMaterial::CZenMaterial()
    {
        ZeroMemory( &m_Material, sizeof( D3DMATERIAL9 ) );

        m_Material.Diffuse.r = 1.0f;
        m_Material.Diffuse.g = 1.0f;
        m_Material.Diffuse.b = 1.0f;

        m_Material.Ambient.r = 0.5f;
        m_Material.Ambient.g = 0.5f;
        m_Material.Ambient.b = 0.5f;
    }

    CZenMaterial::~CZenMaterial()
    {
        // There is nothing to destruct.
    }

    void CZenMaterial::SetDiffuse( float r, float g, float b )
    {
        m_Material.Diffuse.r = r;
        m_Material.Diffuse.g = g;
        m_Material.Diffuse.b = b;
    }

    void CZenMaterial::SetAmbient( float r, float g, float b )
    {
        m_Material.Ambient.r = r;
        m_Material.Ambient.g = g;
        m_Material.Ambient.b = b;
    }

    void CZenMaterial::SetEmissive( float r, float g, float b )
    {
        m_Material.Emissive.r = r;
        m_Material.Emissive.g = g;
        m_Material.Emissive.b = b;
    }

    void CZenMaterial::SetSpecular( float r, float g, float b, float Power )
    {
        m_Material.Specular.r = r;
        m_Material.Specular.g = g;
        m_Material.Specular.b = b;
        m_Material.Power = Power;
    }
}

```

```

HRESULT CZenMaterial::Update()
{
    // Set this material as active.
    return g_pDevice->SetMaterial( &m_Material );
}

/* This object contains a 3D X-Mesh. */
class CZenMesh : public CZenObject
{
public:
    CZenMesh();
    ~CZenMesh();
    CZenMesh( CZenMesh& OtherMesh );

public:
    int m_NumMats;

protected:
    LPD3DXMESH m_pMesh; // The mesh.
    LPDIRECT3DTEXTURE9 * m_pTextures; // The mesh's textures.
    CZenMaterial* m_pMaterials; // The mesh's materials.

public:
    HRESULT LoadXFile( char* pstrPathName ); //Loads a mesh.
    HRESULT Render(); // Renders the mesh.
    void SetMaterial( CZenMaterial* pMaterial );
    int GetSize(){ return sizeof( *this ); }
    LPD3DXMESH GetMesh() { return m_pMesh; }
};

CZenMesh::CZenMesh( CZenMesh& OtherMesh )
{
    m_NumMats = OtherMesh.m_NumMats;
    m_pMesh = OtherMesh.m_pMesh;
    m_pMesh->AddRef();

    m_pTextures = new LPDIRECT3DTEXTURE9[ m_NumMats ];
    CopyMemory( m_pTextures, OtherMesh.m_pTextures, sizeof( m_pTextures ) );

    for( int i = 0 ; i < m_NumMats ; i++ )
    {
        m_pTextures[i]->AddRef();
    }

    m_pMaterials = new CZenMaterial[ m_NumMats ];
    CopyMemory( m_pMaterials, OtherMesh.m_pMaterials, sizeof( m_pMaterials ) );
}

CZenMesh::CZenMesh()
{
    m_pMesh = 0;
    m_NumMats = 0;
    m_pTextures = 0;
    m_pMaterials = 0;
}

CZenMesh::~CZenMesh()
{
    if( Trycatch((void*)m_pMesh, "m_pMesh in ~CZenMesh()") )
    {
        m_pMesh->Release();
    }

    if( Trycatch((void*)m_pTextures, "m_pTextures in ~CZenMesh()") )
    {
        delete[] m_pTextures;
    }

    if( Trycatch((void*)m_pMaterials, "m_pMaterials in ~CZenMesh()") )
    {
        delete[] m_pMaterials;
    }
}

```



```

void CZenMesh::SetMaterial( CZenMaterial* pMaterial )
{
    // Sets all the materials for the object to the specified material.
    for( int i = 0 ; i < m_NumMats ; i++ )
    {
        CopyMemory( &m_pMaterials[i].m_Material, &pMaterial->m_Material, _
            sizeof( D3DMATERIAL9 ) );
    }
}

HRESULT CZenMesh::LoadXFile( char* pstrPathName )
{
    HRESULT r = 0;
    LPD3DXBUFFER pMaterialBuffer = 0;

    // Load the x file from disk. 6th parameter is new in DX9.
    r = D3DXLoadMeshFromX( pstrPathName, D3DXMESH_SYSTEMMEM, g_pDevice, 0, _
        &pMaterialBuffer, 0, (DWORD*)&m_NumMats, &m_pMesh );
    if( FAILED( r ) )
    {
        Debug( "Failed to load .X File with filename:" );
        Debug( pstrPathName );
        return E_FAIL;
    }

    // Create a new texture array.
    m_pTextures = new LPDIRECT3DTEXTURE9[ m_NumMats ];
    // Create a new material array.
    m_pMaterials = new CZenMaterial[ m_NumMats ];

    // Get a pointer to the start of the material buffer.
    D3DXMATERIAL* pMaterials = (D3DXMATERIAL*)pMaterialBuffer->GetBufferPointer();

    // Loop for each material in the buffer.
    for( int i = 0 ; i < m_NumMats ; i++ )
    {
        // Extract the material from the buffer.
        m_pMaterials[i].m_Material = pMaterials[i].MatD3D;
        // Brighten the material.
        m_pMaterials[i].m_Material.Ambient = m_pMaterials[i].m_Material.Diffuse;

        // If a texture is not set for this material.
        if( !pMaterials[i].pTextureFilename )
        {
            // Set the texture to the default texture.
            m_pTextures[i] = g_pDefaultTexture;

            // Iterate to the next loop because there is no texture.
            continue;
        }

        // Create a new texture from the filename supplied.
        r = D3DXCreateTextureFromFile( g_pDevice, _
            pMaterials[i].pTextureFilename, &m_pTextures[i] );
        if( FAILED( r ) )
        {
            Debug( "Unable to load texture for mesh with filename:" );
            Debug( pMaterials[i].pTextureFilename );
            // If the texture load failed then set it to the default texture.
            m_pTextures[i] = g_pDefaultTexture;
        }
    }

    pMaterialBuffer->Release(); // Release the material buffer.

    return S_OK;
}

HRESULT CZenMesh::Render()
{
    HRESULT r = E_FAIL;

```

```

for( int i = 0 ; i < m_NumMats ; i++ )      // Loop for each material.
{
    // Set this material as active.
    m_pMaterials[i].Update();
    // Set this texture as active.
    g_pDevice->SetTexture( 0, m_pTextures[i] );
    // Render this subset of the mesh.
    r = m_pMesh->DrawSubset(i);

    // Reset the vertex shader.
    g_pDevice->SetFVF( ZENVERTEX_TYPE ); // Used to be SetVertexShader().
}

// Return the result of the render operation.
return r;
}

/* A Frame object provides a context of position for sets of objects and other frames. */
class CZenFrame
{
public:
    CZenFrame();
    ~CZenFrame();

public:
    void* m_pParameter;

protected:
    D3DXMATRIX m_mLocal;           // The local matrix for this frame.
    D3DXVECTOR3 m_vPosition;      // The position of this frame.
    D3DXVECTOR3 m_vVelocity;      // The velocity of this frame.
    float m_Yaw, m_Pitch, m_Roll; // The orientation of this frame.

    CZenObject* m_pObjectList;    // The list of objects in this frame.
    CZenFrame* m_pNext;           // Pointer to the next frame in the list.

    CZenFrame* m_pChildFrameList;
    CZenFrame* m_pParentFrame;

    FRAME_MOVEMENT_CALLBACK m_pfnCallback;
    BOOL m_bCallback;

public:
    HRESULT SetCallback( FRAME_MOVEMENT_CALLBACK pfnCallback );

    void SetVelocity( float x, float y, float z );
    void GetVelocity( float& x, float& y, float& z );

    void SetPosition( float x, float y, float z );
    void GetPosition( float& x, float& y, float& z );

    // Returns the local matrix for this frame.
    void GetLocal( D3DXMATRIX& pMatrix );

    void SetYaw( float Yaw ){ m_Yaw = Yaw; }
    void GetYaw( float& Yaw ){ Yaw = m_Yaw; }

    void SetPitch( float Pitch ){ m_Pitch = Pitch; }
    void GetPitch( float& Pitch ){ Pitch = m_Pitch; }

    void SetRoll( float Roll ){ m_Roll = Roll; }
    void GetRoll( float& Roll ){ Roll = m_Roll; }

    // Update the position of the objects.
    void Update();

    // Add an object to the frame.
    HRESULT AddObject( CZenObject* pNewObject );

    // Render the objects.
    HRESULT Render();

    // Set/Get the next pointer for use in the list.
    void SetNext( CZenFrame* pNext ){ m_pNext = pNext; }

```

```

    CZenFrame* GetNext(){ return m_pNext; }

    HRESULT AddFrame( CZenFrame* pNewFrame );
protected:
    void SetParent( CZenFrame* pParent ){ m_pParentFrame = pParent; }
    CZenFrame* GetParent(){ return m_pParentFrame; }
};

CZenFrame::CZenFrame()
{
    // Set the orientation to 0.
    m_Yaw = 0.0f;
    m_Pitch = 0.0f;
    m_Roll = 0.0f;

    // Set the position and velocity to 0
    m_vPosition = D3DXVECTOR3( 0.0f, 0.0f, 0.0f );
    m_vVelocity = D3DXVECTOR3( 0.0f, 0.0f, 0.0f );

    // Init the local matrix to an identity.
    D3DXMatrixIdentity( &m_mLocal );

    // Zero out the object list.
    m_pObjectList = 0;

    m_pNext = 0;
    m_pChildFrameList = 0;
    m_pParentFrame = 0;

    m_pfnCallback = 0;
    m_bCallback = FALSE;
}

CZenFrame::~CZenFrame()
{
    // There is nothing to deconstruct.
}

HRESULT CZenFrame::SetCallback( FRAME_MOVEMENT_CALLBACK pfnCallback )
{
    if( !Trycatch((void*)pfnCallback, "pfnCallback in CZenFrame::SetCallback()") )
    {
        m_bCallback = FALSE;
        m_pfnCallback = NULL;
        return E_FAIL;
    }
    m_pfnCallback = pfnCallback;
    m_bCallback = TRUE;
    return S_OK;
}

HRESULT CZenFrame::AddFrame( CZenFrame* pNewFrame )
{
    // Make sure the new frame is valid
    if( !Trycatch((void*)pNewFrame, "pNewFrame in CZenFrame::AddFrame()") )
    {
        Debug( "Failed in attempt to add an invalid child frame." );
        return E_FAIL;
    }

    pNewFrame->SetParent( this );

    if( !Trycatch((void*)m_pChildFrameList, "m_pChildFrameList in CZenFrame::AddFrame()") )
    {
        m_pChildFrameList = pNewFrame;
    }
    else
    {
        CZenFrame* pTempFrame = m_pChildFrameList;

```

```

        while( pTempFrame->GetNext() )
        {
            pTempFrame = pTempFrame->GetNext();
        }
        pTempFrame->SetNext( pNewFrame );
    }

    return S_OK;
}

HRESULT CZenFrame::AddObject( CZenObject* pNewObject )
{
    // Return if the new object is invalid.
    if( !Trycatch((void*)pNewObject, "pNewObject in CZenFrame::AddObject()") )
    {
        return E_FAIL;
    }

    // Tell the object it has a new parent frame.
    pNewObject->SetParentFrame( this );

    // If the object list does not exist yet ...
    if( !Trycatch((void*)m_pObjectList, "m_pObjectList in CZenFrame::AddObject()") )
    {
        // ... set this object to the start of the list.
        m_pObjectList = pNewObject;
    }
    else
    {
        // ... the list has already been created.
        // Add this object to the end of the list.

        // Get a pointer to the start of the list.
        CZenObject* pObject = m_pObjectList;

        // Find the last object in the list.
        while( pObject->GetNext() )
        {
            pObject = (CZenObject*)pObject->GetNext();
        }

        // Add this to the last item in the list.
        pObject->SetNext( pNewObject );
    }

    return S_OK;
}

HRESULT CZenFrame::Render()
{
    // Update the frame and set the new world transform matrix.
    Update();

    CZenFrame* pFrame = m_pChildFrameList;

    while( pFrame )
    {
        pFrame->Render();
        pFrame = pFrame->GetNext();
    }

    // Return if this frame has no visuals to render.
    if( !Trycatch((void*)m_pObjectList, "m_pObjectList in CZenFrame::Render()") )
    {
        return S_OK;
    }

    // Get a pointer to the start of the list.
    CZenObject* pObject = m_pObjectList;

```

```

// Reset the transform in case those pesky children modified it.
g_pDevice->SetTransform( D3DTS_WORLD, &m_mLocal );

// Loop for each object in the list.
while( Trycatch((void*)pObject, "pObject in CZenFrame::AddRender()") )
{
    // Render the object.
    pObject->Render();
    // Increment to the next object in the list.
    pObject = (CZenObject*)pObject->GetNext();
}

return S_OK;
}

void CZenFrame::Update()
{
    if( m_bCallback )
    {
        m_pfnCallback( this, m_pParameter );
    }

    // Create some temporary matrices for the rotation and
    // translation transformations.
    D3DXMATRIX mRotX, mRotY, mRotZ, mTrans, mRotTemp;

    // Update the position by the velocity.
    m_vPosition.x += m_vVelocity.x;
    m_vPosition.y += m_vVelocity.y;
    m_vPosition.z += m_vVelocity.z;

    // Set the translation matrix.
    D3DXMatrixTranslation( &mTrans, m_vPosition.x, m_vPosition.y, m_vPosition.z );

    // Set the rotation around the x axis.
    D3DXMatrixRotationX( &mRotX, m_Pitch );
    // Set the rotation around the y axis.
    D3DXMatrixRotationY( &mRotY, m_Yaw );
    // Set the rotation around the z axis.
    D3DXMatrixRotationZ( &mRotZ, m_Roll );

    // Concatenate the y axis and x axis rotation matrices.
    D3DXMatrixMultiply( &mRotTemp, &mRotX, &mRotY );
    // Concatenate the xy axes and z axis rotation matrices.
    D3DXMatrixMultiply( &mRotTemp, &mRotZ, &mRotTemp );
    // Concatenate the xyz axes and translation matrices.
    D3DXMatrixMultiply( &mTrans, &mRotTemp, &mTrans );

    // Update the copy of the local matrix.
    m_mLocal = mTrans;

    if( GetParent() )
    {
        D3DXMATRIX mParent;
        GetParent()->GetLocal( mParent );
        D3DXMatrixMultiply( &m_mLocal, &m_mLocal, &mParent );
    }

    // Set the world matrix.
    g_pDevice->SetTransform( D3DTS_WORLD, &m_mLocal );
}

void CZenFrame::GetLocal( D3DXMATRIX& Matrix )
{
    // Returns the local transform matrix.
    Update();
    Matrix = m_mLocal;
}

void CZenFrame::GetVelocity( float& x, float& y, float& z )
{
    // Returns the velocity of the frame.
    x = m_vVelocity.x;

```

```

        y = m_vVelocity.y;
        z = m_vVelocity.z;
    }

void CZenFrame::SetVelocity( float x, float y, float z )
{
    // Sets the velocity of the frame.
    m_vVelocity.x = x;
    m_vVelocity.y = y;
    m_vVelocity.z = z;
}

void CZenFrame::GetPosition( float& x, float& y, float& z )
{
    // Returns the position of the frame.
    x = m_vPosition.x;
    y = m_vPosition.y;
    z = m_vPosition.z;
}

void CZenFrame::SetPosition( float x, float y, float z )
{
    // Sets the position of the frame.
    m_vPosition.x = x;
    m_vPosition.y = y;
    m_vPosition.z = z;
}

/* This class creates a light object. */
class CZenLight : public CZenObject
{
public:
    CZenLight();
    ~CZenLight();
    CZenLight( CZenLight& OtherLight );

public:
    D3DLIGHT9 m_Light;

protected:
    int m_ID; // The lights index.
    BOOL m_bIsOn; // Is the light on?

public:
    // Sets the light's color properties.
    void SetDiffuse( float r, float g, float b );
    void SetSpecular( float r, float g, float b );
    void SetAmbient( float r, float g, float b );

    // Turns the light on or off.
    void Enable( BOOL bEnable );
    // Returns the status of the light.
    BOOL IsOn(){ return m_bIsOn; }
    // Updates the status of the light.
    HRESULT Render();
    // Returns the size of the light object (in bytes).
    int GetSize(){ return sizeof( *this ); }
};

CZenLight::CZenLight( CZenLight& OtherLight )
{
    m_bIsOn = OtherLight.m_bIsOn;
    m_ID = OtherLight.m_ID;
    m_Light = OtherLight.m_Light;
    CopyMemory( &m_Light, &OtherLight.m_Light, sizeof( D3DLIGHT9 ) );
}

CZenLight::CZenLight()
{
    // Zero out the D3DLIGHT9 structure.
    ZeroMemory( &m_Light, sizeof( D3DLIGHT9 ) );

    // Set the initial type to point.
    m_Light.Type = D3DLIGHT_POINT;
}

```

```

    // Set the initial color white.
    m_Light.Diffuse.r = 1.0f;
    m_Light.Diffuse.g = 1.0f;
    m_Light.Diffuse.b = 1.0f;

    // Set the attenuation.
    m_Light.Attenuation0 = 1.0f;

    // Set the initial range to 100 units.
    m_Light.Range = 100.0f;

    // Set the index based on a static counter.
    m_ID = g_LightCounter++;

    // Set the light status tracker to off.
    m_bIsOn = FALSE;
}

CZenLight::~CZenLight()
{
    // Nothing to destruct.
}

void CZenLight::SetAmbient( float r, float g, float b )
{
    // Sets the ambient color of the light.
    m_Light.Ambient.r = r;
    m_Light.Ambient.g = g;
    m_Light.Ambient.b = b;
}

void CZenLight::SetDiffuse( float r, float g, float b )
{
    // Sets the diffuse color of the light.
    m_Light.Diffuse.r = r;
    m_Light.Diffuse.g = g;
    m_Light.Diffuse.b = b;
}

void CZenLight::SetSpecular( float r, float g, float b )
{
    // Sets the specular color of the light.
    m_Light.Specular.r = r;
    m_Light.Specular.g = g;
    m_Light.Specular.b = b;
}

void CZenLight::Enable( BOOL bEnable )
{
    // Turns the light on or off.
    m_bIsOn = bEnable; // Update the tracking variable.
    g_pDevice->LightEnable( m_ID, bEnable ); // Change the status of the light.
}

HRESULT CZenLight::Render()
{
    // Puts the light at the same location as the parent frame.
    // The local matrix of the parent frame.
    D3DXMATRIX ParentMatrix;
    // The position of this light = Center of frame.
    D3DXVECTOR3 Position = D3DXVECTOR3( 0, 0, 0 );

    if( Trycatch((void*)m_pParentFrame, "m_pParentFrame in CZenLight::Render()") )
    {
        // Get the position from the parent frame.
        ((CZenFrame*)m_pParentFrame)->GetLocal( ParentMatrix );
        // Transform the lights position by the matrix.
        D3DXVec3TransformCoord( &Position, &Position, &ParentMatrix );
        // Update the position.
        m_Light.Position = Position;
    }
    else
    {
        // Set the light to be at the origin.
        m_Light.Position.x = 0;
        m_Light.Position.y = 0;
    }
}

```

```
        m_Light.Position.z = 0;
        Debug( "Light being rendered without a parent frame." );
    }

    // Update the light with Direct3D.
    g_pDevice->SetLight( m_ID, &m_Light );

    return S_OK;
}

void SetAmbientLight( D3DCOLOR AmbientColor )
{
    g_pDevice->SetRenderState( D3DRS_AMBIENT, AmbientColor );
}
```



```

-----
// File:         zencamera.h
//
// Desc:         This file contains the Zen camera class.
//
// First created on:  March 2nd, 2005
// Last modification:  March 3rd, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//                                     Direct3D Game Programming"
//
-----
// Revision History:
//
// 03-02-05:     This file created. CZenCamera removed from zen.h. This class
//               may receive slight modification later, but as far as I can
//               tell, there is no need to alter Peter Walsh's design in this
//               case.
// 03-03-05:     Adjusted the Move() function to constrain the camera.
//
-----

/** CZenCamera encapsulates the View matrix into a 'camera' class. */

class CZenCamera : public CZenObject
{
public:
    static CZenCamera* Instance();
protected:
    CZenCamera();
    ~CZenCamera();
    CZenCamera( CZenCamera& OtherCamera );
public:
    // Sets/Gets the up vector.
    void SetUp( float x, float y, float z );
    void GetUp( float& x, float& y, float& z );

    // Sets/Gets the right vector.
    void SetRight( float x, float y, float z );
    void GetRight( float& x, float& y, float& z );

    // Sets/Gets the velocity.
    void SetVelocity( float x, float y, float z );
    void GetVelocity( float& x, float& y, float& z );

    // Sets/Gets Position.
    void SetPosition( float x, float y, float z );
    void GetPosition( float& x, float& y, float& z );

    // Sets/Gets the look point.
    void SetLookPoint( float x, float y, float z );
    void GetLookPoint( float& x, float& y, float& z );

    // Updates the position of the camera.
    void Update();

    // Moves the camera.
    void Move( float x, float y, float z );

    // Sets the roll, pitch, or yaw.
    void SetRoll( float Roll );
    void SetYaw( float Yaw );
    void SetPitch( float Pitch );

    // Returns the roll, pitch, or yaw.
    void GetRoll( float& Roll );
    void GetYaw( float& Yaw );
    void GetPitch( float& Pitch );

    // Resets the camera back to the origin.
    void Reset();
}

```

```

HRESULT Render();

// Returns the size (in bytes) of this object.
int GetSize(){ return sizeof( *this ); }
protected:
// The roll, pitch, and yaw for the camera's orientation.
float m_Roll;
float m_Pitch;
float m_Yaw;

// The position.
D3DXVECTOR3 m_Position;
// The look-at vector.
D3DXVECTOR3 m_LookAt;
// The up vector.
D3DXVECTOR3 m_Up;
// The right vector.
D3DXVECTOR3 m_Right;

// The camera's velocity.
D3DXVECTOR3 m_Velocity;
private:
static CZenCamera* _instance;
};

CZenCamera* CZenCamera::_instance = 0;

CZenCamera* CZenCamera::Instance()
{
    if(_instance == 0)
    {
        _instance = new CZenCamera;
    }
    return _instance;
}

CZenCamera::CZenCamera( CZenCamera& OtherCamera )
{
    m_LookAt = OtherCamera.m_LookAt;
    m_Pitch = OtherCamera.m_Pitch;
    m_Position = OtherCamera.m_Position;
    m_Right = OtherCamera.m_Right;
    m_Roll = OtherCamera.m_Roll;
    m_Up = OtherCamera.m_Up;
    m_Velocity = OtherCamera.m_Velocity;
    m_Yaw = OtherCamera.m_Yaw;
}

CZenCamera::CZenCamera()
{
    // Set the position.
    // The camera is one unit behind the origin.
    m_Position = D3DXVECTOR3( 0.0f, 0.0f, -1.0f );
    // Set the velocity to 0.
    m_Velocity = D3DXVECTOR3( 0.0f, 0.0f, 0.0f );
    // Set the lookat vector to straight ahead.
    // Look straight ahead.
    m_LookAt = D3DXVECTOR3( 0.0f, 0.0f, 1.0f );
    // Set the right vector to right.
    m_Right = D3DXVECTOR3( 1.0f, 0.0f, 0.0f );
    // Set the up vector to up.
    m_Up = D3DXVECTOR3( 0.0f, 1.0f, 0.0f );
    // Set the roll, pitch, and yaw to 0.
    m_Roll = m_Pitch = m_Yaw = 0.0f;
}

CZenCamera::~CZenCamera()
{
    // Nothing to destroy.
}

```

```

void CZenCamera::Reset()
{
    // Move the camera back to the origin.
    m_Position      = D3DXVECTOR3( 0.0f, 0.0f, -1.0f ); // -
    m_LookAt        = D3DXVECTOR3( 0.0f, 0.0f, 1.0f ); // z +
    m_Right         = D3DXVECTOR3( 1.0f, 0.0f, 0.0f ); // x
    m_Up            = D3DXVECTOR3( 0.0f, 1.0f, 0.0f ); // y
    m_Roll = m_Pitch = m_Yaw = 0.0f;
}

void CZenCamera::Update()
{
    // Updates the Direct3D view matrix.
    // Update the x position.
    m_Position.x += m_Velocity.x * m_Right.x;
    m_Position.y += m_Velocity.x * m_Right.y;
    m_Position.z += m_Velocity.x * m_Right.z;

    // Update the y position.
    m_Position.x += m_Velocity.y * m_Up.x;
    m_Position.y += m_Velocity.y * m_Up.y;
    m_Position.z += m_Velocity.y * m_Up.z;

    // Update the z position.
    m_Position.x += m_Velocity.z * m_LookAt.x;
    m_Position.y += m_Velocity.z * m_LookAt.y;
    m_Position.z += m_Velocity.z * m_LookAt.z;

    D3DXMATRIX mPitch, mRoll, mYaw;

    // Normalize and Regenerate the Look, Right, and Up Vectors.
    D3DXVec3Normalize( &m_LookAt, &m_LookAt );
    D3DXVec3Cross( &m_Right, &m_Up, &m_LookAt );
    D3DXVec3Normalize( &m_Right, &m_Right );
    D3DXVec3Cross( &m_Up, &m_LookAt, &m_Right );
    D3DXVec3Normalize( &m_Up, &m_Up );

    // Set up the y-axis rotation.
    D3DXMatrixRotationAxis( &mYaw, &m_Up, m_Yaw );
    D3DXVec3TransformCoord( &m_LookAt, &m_LookAt, &mYaw );
    D3DXVec3TransformCoord( &m_Right, &m_Right, &mYaw );

    // Set up the x-axis rotation.
    D3DXMatrixRotationAxis( &mPitch, &m_Right, m_Pitch );
    D3DXVec3TransformCoord( &m_LookAt, &m_LookAt, &mPitch );
    D3DXVec3TransformCoord( &m_Up, &m_Up, &mPitch );

    // Set up the z-axis rotation.
    D3DXMatrixRotationAxis( &mRoll, &m_LookAt, m_Roll );
    D3DXVec3TransformCoord( &m_Right, &m_Right, &mRoll );
    D3DXVec3TransformCoord( &m_Up, &m_Up, &mRoll );

    D3DXMATRIX mView;

    // Init the view matrix to an identity.
    D3DXMatrixIdentity( &mView );

    // Fill in the view matrix.
    mView(0,0) = m_Right.x;
    mView(0,1) = m_Up.x;
    mView(0,2) = m_LookAt.x;

    mView(1,0) = m_Right.y;
    mView(1,1) = m_Up.y;
    mView(1,2) = m_LookAt.y;

    mView(2,0) = m_Right.z;
    mView(2,1) = m_Up.z;
    mView(2,2) = m_LookAt.z;

    mView(3,0) = - D3DXVec3Dot( &m_Position, &m_Right );
    mView(3,1) = - D3DXVec3Dot( &m_Position, &m_Up );
    mView(3,2) = - D3DXVec3Dot( &m_Position, &m_LookAt );
}

```

```

        // Set the view matrix.
        g_pDevice->SetTransform( D3DTS_VIEW, &mView );
    }

HRESULT CZenCamera::Render()
{
    // We do not render the camera so no code goes here!
    return S_OK;
}

void CZenCamera::Move( float x, float y, float z )
{
    // Moves the camera relative to its current position.
    m_Position.x += x;
    m_Position.y += y;
    m_Position.z += z;

    if(g_bCameraLocked)
    {
        if(m_Position.x > 983) { m_Position.x = 983; }
        if(m_Position.x < 15) { m_Position.x = 15; }
        if(m_Position.z > 983) { m_Position.z = 983; }
        if(m_Position.z < 15) { m_Position.z = 15; }
        if(m_Position.y > 275) { m_Position.y = 275; }
        if(m_Position.y < 5) { m_Position.y = 5; }
    }
}

void CZenCamera::SetPosition( float x, float y, float z )
{
    // Sets the position of the camera.
    m_Position.x = x;
    m_Position.y = y;
    m_Position.z = z;
}

void CZenCamera::GetPosition( float& x, float& y, float& z )
{
    // Gets the position for the camera.
    x = m_Position.x;
    y = m_Position.y;
    z = m_Position.z;
}

void CZenCamera::SetRoll( float Roll )
{
    // Sets the roll of the camera.
    m_Roll = Roll;
}

void CZenCamera::GetRoll( float& Roll )
{
    // Gets the roll for the camera.
    Roll = m_Roll;
}

void CZenCamera::GetYaw( float& Yaw )
{
    // Gets the yaw for the camera.
    Yaw = m_Yaw;
}

void CZenCamera::SetYaw( float Yaw )
{
    // Sets the yaw for the camera.
    m_Yaw = Yaw;
}

void CZenCamera::GetPitch( float& Pitch )
{
    // Gets the pitch for the camera.
    Pitch = m_Pitch;
}

void CZenCamera::SetPitch( float Pitch )
{
    // Sets the pitch for the camera.
    m_Pitch = Pitch;
}

```

```

void CZenCamera::SetLookPoint( float x, float y, float z )
{
    // Sets the point for the camera to look at.
    m_LookAt.x = x;
    m_LookAt.y = y;
    m_LookAt.z = z;
}

void CZenCamera::GetLookPoint( float& x, float& y, float& z )
{
    // Gets the look vector.
    x = m_LookAt.x;
    y = m_LookAt.y;
    z = m_LookAt.z;
}

void CZenCamera::SetUp( float x, float y, float z )
{
    // Sets the up direction.
    m_Up.x = x;
    m_Up.y = y;
    m_Up.z = z;
}

void CZenCamera::GetUp( float& x, float& y, float& z )
{
    // Gets the up vector.
    x = m_Up.x;
    y = m_Up.y;
    z = m_Up.z;
}

void CZenCamera::SetVelocity( float x, float y, float z )
{
    // Sets the velocity.
    m_Velocity.x = x;
    m_Velocity.y = y;
    m_Velocity.z = z;
}

void CZenCamera::GetVelocity( float& x, float& y, float& z )
{
    // Gets the velocity.
    x = m_Velocity.x;
    y = m_Velocity.y;
    z = m_Velocity.z;
}

void CZenCamera::SetRight( float x, float y, float z )
{
    // Sets the right vector.
    m_Right.x = x;
    m_Right.y = y;
    m_Right.z = z;
}

void CZenCamera::GetRight( float& x, float& y, float& z )
{
    // Gets the right vector.
    x = m_Right.x;
    y = m_Right.y;
    z = m_Right.z;
}

```

```

//-----
// File:      font.h
//
// Desc:      This file contains the CZenFont class for rendering 3D text.
//
// First created on:  January 4th, 2005
// Last modification: January 4th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//                                     Direct3D Game Programming"
//-----
// Revision History:
//
// 01-04-05:  This file was created to store CZenFont.
//-----

/* This class abstracts away font creation and use. */
class CZenFont
{
public:
    CZenFont();
    ~CZenFont();

public:
    D3DCOLOR m_FontColor;
    D3DCOLOR m_OrigColor;
    int m_Align;

protected:
    LPD3DXFONT m_pFont;
    BOOL m_bInitialized;

public:
    HRESULT Initialize( HFONT hFont, D3DCOLOR FontColor );
    HRESULT SetColor( D3DCOLOR FontColor );
    void RestoreColor();
    HRESULT OutputText( char* pString, int x, int y );
    void GetBoundingBox( char* pString, int & x, int & y );
    CZenFont* GetPtrToSelf();
};

CZenFont::CZenFont()
{
    m_pFont = 0;
    // Set the default font color to black.
    m_FontColor = D3DCOLOR_XRGB( 0, 0, 0 );
    m_OrigColor = D3DCOLOR_XRGB( 0, 0, 0 );
    // The object has not been initialized yet.
    m_bInitialized = FALSE;
    // Align font to the left by default.
    m_Align = DT_LEFT;
}

CZenFont::~CZenFont()
{
    #ifdef QUIET_MODE
    // Release the font if it has been created.
    if( QTrycatch((void*)m_pFont, "m_pFont in ~CZenFont()") )
    {
        // This causes a program crash if enabled.
        // Is it because of deleting data with a pointer still on it?

        //m_pFont->Release();
    }
    #else
    // Release the font if it has been created.
    if( QTrycatch((void*)m_pFont, "m_pFont in ~CZenFont()") )
    {
        // This causes a program crash if enabled.
        // Is it because of deleting data with a pointer still on it?

        //m_pFont->Release();
    }
}

```

```

        #endif
    }

    // Add a trycatch to the string.
    void CZenFont::GetBoundingBox( char* pString, int & x, int & y )
    {
        // Return if the class has not been initialized yet.
        if( !m_bInitialized )
        {
            return;
        }

        // Fill in the rect structure with the dest coords for the font.
        RECT FontRect = { 0, 0, 0, 0 };

        m_pFont->Begin();        // Used for pseudo-rendering.

        m_pFont->DrawText( pString, -1, &FontRect, DT_CALCRECT, 0 );
        x = FontRect.right - FontRect.left;
        y = FontRect.bottom - FontRect.top;

        m_pFont->End();          // Used for pseudo-rendering.
    }

    CZenFont* CZenFont::GetPtrToSelf()
    {
        return this;
    }

    // Add a trycatch to the string.
    HRESULT CZenFont::OutputText( char* pString, int x, int y )
    {
        // Return if the class has not been initialized yet.
        if( !m_bInitialized )
        {
            return E_FAIL;
        }

        HRESULT r;

        // Fill in the rect structure with the dest coords for the font.
        RECT FontRect = { x, y, 0, 0 };

        // Tell the font we are about to begin rendering.
        m_pFont->Begin();

        // Calculate the rectangle for the string.
        m_pFont->DrawText( pString, -1, &FontRect, DT_CALCRECT, 0 );

        // Render the string to the screen.
        r = m_pFont->DrawText( pString, -1, &FontRect, m_Align, m_FontColor );

        // We are done rendering.
        m_pFont->End();

        return r;
    }

    HRESULT CZenFont::SetColor( D3DCOLOR FontColor )
    {
        if( !Trycatch((void*)m_pFont, "m_pFont in CZenFont::Initialize()") )
        {
            Debug("Cannot set CZenFont color before initializing.");
            return E_FAIL;
        }

        // Save the color.
        m_FontColor = FontColor;

        return S_OK;
    }

```

```

void CZenFont::RestoreColor( )
{
    m_FontColor = m_OrigColor;
}

HRESULT CZenFont::Initialize( HFONT hFont, D3DCOLOR FontColor )
{
    HRESULT r;

    #ifdef QUIET_MODE
    // Release the font if it has already been created.
    if( QTrycatch((void*)m_pFont, "m_pFont in CZenFont::Initialize()") )
    {
        m_pFont->Release();
        m_pFont = 0;
    }
    #else
    // Release the font if it has already been created.
    if( Trycatch((void*)m_pFont, "m_pFont in CZenFont::Initialize()") )
    {
        m_pFont->Release();
        m_pFont = 0;
    }
    #endif

    // Release the font if it has already been created.
    if( Trycatch((void*)m_pFont, "m_pFont in CZenFont::Initialize()") )
    {
        m_pFont->Release();
        m_pFont = 0;
    }

    // Create the font.
    r = D3DXCreateFont( g_pDevice, hFont, &m_pFont );
    if( FAILED( r ) )
    {
        Debug( "Could not create font" );
        return E_FAIL;
    }

    // Save the color.
    m_FontColor = FontColor;
    m_OrigColor = FontColor;

    // Set initialization flag to true.
    m_bInitialized = TRUE;

    return S_OK;
}

```



```

//-----
// File:      console.h
//
// Desc:      This file contains the Zen font engine.
//
// First created on:  December 28th, 2004
// Last modification:  March 1st, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Zen of
//                                     Direct3D Game Programming"
//-----
// Revision History:
//
// 12-28-04:  Added revision history. File created. Converted code to DX9.
// 01-02-05:  Console is now visible and working. Cleaned up and recommented
//            all of the code in this file.
// 01-03-05:  Created a new, cleaner alphabet to be loaded into the console.
//            Fixed the console speed issue with syntax in zen.h. Created a
//            gray console background until transparency works, if ever.
// 01-04-05:  Converted the CConsole class to a Singleton.
// 03-01-05:  Tweaked some of the console code to format nicer.
//-----
// To Do:    - Allow background image to be transparent.
//           - Make console display more messages, blinking cursor, etc.
//-----

/*****
* Section:  Font Engine
*****/

int g_AlphabetWidth = 0;           // The width of the Alphabet bitmap.
int g_AlphabetHeight = 0;        // The height of the Alphabet bitmap.
int g_AlphabetLetterWidth = 0;   // The width of a letter.
int g_AlphabetLetterHeight = 0;  // The height of a letter.
int g_AlphabetLettersPerRow = 0; // The number of letters per row.

// The surface holding the alphabet bitmap.
LPDIRECT3DSURFACE9 g_pAlphabetSurface = 0;

// Has the alphabet bitmap been loaded yet?
BOOL g_bAlphabetLoaded = FALSE;

HRESULT LoadAlphabet( char* strPathName, int LetterWidth, int LetterHeight )
{
    // Make sure a valid path was specified.
    if( !strPathName )
    {
        return E_FAIL;
    }

    // Make sure the size of the letters is greater than 0.
    if( !LetterWidth || !LetterHeight )
    {
        return E_FAIL;
    }

    HRESULT r = 0;

    // Load the bitmap into memory.
    r = LoadBitmapToSurface( strPathName, &g_pAlphabetSurface, g_pDevice );
    if( FAILED( r ) )
    {
        Debug( "Unable to load alphabet bitmap" );
        return E_FAIL;
    }

    // Holds information about the alphabet surface.
    D3DSURFACE_DESC d3dsd;

```

```

// Get information about the alphabet surface.
g_pAlphabetSurface->GetDesc( &d3dsd );

// Update globals with the letter dimensions.
g_AlphabetWidth = d3dsd.Width;
g_AlphabetHeight = d3dsd.Height;
g_AlphabetLetterWidth = LetterWidth;
g_AlphabetLetterHeight = LetterHeight;

// Compute the number of letters in a row.
g_AlphabetLettersPerRow = g_AlphabetWidth / g_AlphabetLetterWidth;

// Set the loaded flag to TRUE.
g_bAlphabetLoaded = TRUE;

return S_OK;
}

HRESULT UnloadAlphabet()
{
    if( Trycatch((void*)g_pAlphabetSurface, "g_pAlphabetSurface in UnloadAlphabet()")
    {
        // If the alphabet exists ...
        g_pAlphabetSurface->Release(); // Release the surface.
        g_pAlphabetSurface = 0; // NULL the pointer.
        g_bAlphabetLoaded = FALSE; // Set the loaded flag to FALSE.
    }
    return S_OK;
}

// Print a character to a surface using the loaded alphabet.
void PrintChar( int x, int y, char Character, BOOL bTransparent, D3DCOLOR ColorKey,
DWORD* pDestData, int DestPitch )
{
    HRESULT r = 0;

    div_t Result; // Holds the result of divisions.

    // The offset into the alphabet image.
    int OffsetX = 0, OffsetY = 0;

    POINT LetterDestPoint = { 0, 0 }; // The destination point for the letter.
    RECT LetterRect = { 0, 0, 0, 0 }; // The source rectangle for the letter.

    // If the alphabet has not been loaded yet then exit.
    if( !g_bAlphabetLoaded )
    {
        return;
    }

    // The characters are specified in ASCII code, which begins at 32 so
    // we want to decrement this value by 32 to make it zero based.
    Character -= 32;

    // Divide the character code by the number of letters per row. The quotient will
    // get the vertical offset and the remainder will get the horizontal offset.
    Result = div( Character, g_AlphabetLettersPerRow );

    // Get the horizontal offset by multiplying the remainder by the width of the letter.
    OffsetX = Result.rem * g_AlphabetLetterWidth;
    // Get the vertical offset by multiplying the quotient by the height of the letter.
    OffsetY = Result.quot * g_AlphabetLetterHeight;

    // Fill in the source rectangle with the computed offsets.
    SetRect( &LetterRect, OffsetX, OffsetY,
OffsetX + g_AlphabetLetterWidth, OffsetY + g_AlphabetLetterHeight );

    // Fill in the destination point.
    LetterDestPoint.x = x;
    LetterDestPoint.y = y;
}

```

```

D3DLOCKED_RECT LockedAlphabet; // Holds info about the alphabet surface.

// Lock the source surface.
r = g_pAlphabetSurface->LockRect( &LockedAlphabet, 0, D3DLOCK_READONLY );
if( FAILED( r ) )
{
    Debug( "Couldnt lock alphabet surface for PrintChar()" );
    return;
}

// Get a DWORD pointer to each surface.
DWORD* pAlphaData = (DWORD*)LockedAlphabet.pBits;

// Convert the BYTE pitch pointer to a DWORD ptr.
LockedAlphabet.Pitch /=4;
DestPitch /= 4;

// Compute the offset into the alphabet.
int AlphaOffset = OffsetY * LockedAlphabet.Pitch + OffsetX;
// Compute the offset into the destination surface.
int DestOffset = y * DestPitch + x;

// Loop for each row in the letter.
for( int cy = 0 ; cy < g_AlphabetLetterHeight ; cy++ )
{
    // Loop for each column in the letter.
    for( int cx = 0 ; cx < g_AlphabetLetterWidth ; cx++ )
    {
        if( bTransparent )
        {
            if( pAlphaData[ AlphaOffset ] != ColorKey )
            {
                // If this alphabet pixel is not transparent then
                // copy the pixel.
                pDestData[ DestOffset ] = pAlphaData[ AlphaOffset ];
            }
        }
        else
        {
            pDestData[ DestOffset ] = pAlphaData[ AlphaOffset ];
        }
        AlphaOffset++;
        DestOffset++;
    }

    // Move the offsets to the start of the next row.
    DestOffset += DestPitch - g_AlphabetLetterWidth;
    AlphaOffset += LockedAlphabet.Pitch - g_AlphabetLetterWidth;
}

// Unlock the surface.
g_pAlphabetSurface->UnlockRect();
}

void PrintString( int x, int y, char* String, BOOL bTransparent, D3DCOLOR ColorKey,
DWORD* pDestData, int DestPitch )
{
    // Loop for each character in the string.
    for( UINT i = 0 ; i < strlen( String ) ; i++ )
    {
        // Print the current character.
        int cx = x + (g_AlphabetLetterWidth * i);

        if( cx + g_AlphabetLetterWidth > g_DeviceWidth )
        {
            continue;
        }
        PrintChar(cx, y, String[i], bTransparent, ColorKey, pDestData, DestPitch);
    }
}

```

```

/*****
* Section: Console
*****/

/** This is a row of text. */
class CEntry
{
public:
    CEntry();
    ~CEntry();

protected:
    char* m_pstrText;    // The text buffer for this entry.
    CEntry* m_pNext;    // Pointer to next entry( row ).
    int m_nVerticalPos; // The y position to render.

public:
    // Draws the text using the GDI to the console surface.
    int RenderText( int NumHorzChars, DWORD* pData, int DestPitch );

    CEntry* GetNext(); // Returns the next entry(row).
    void SetNext( CEntry* pNext ); // Sets the next entry(row).
    int OnChar( char Key ); // Adds a character to the text buffer.
    int GetText( char* pstrText, int Length ); // Returns the text in the buffer.
    int SetText( char* pstrText ); // Clears and then sets the text in the buffer.

    int GetTextLength(){ return strlen( m_pstrText ); }
    void SetVerticalPos( int Pos ){ m_nVerticalPos = Pos; }
    int GetVerticalPos(){ return m_nVerticalPos; }
};

CEntry::CEntry()
{
    m_pstrText = new char[MAX_CHARSPERLINE];
    if( m_pstrText )
    {
        ZeroMemory( m_pstrText, sizeof( char[MAX_CHARSPERLINE] ) );
    }
    else
    {
        m_pstrText = 0;
    }

    m_pNext = 0;
    m_nVerticalPos = 0;
}

CEntry::~CEntry()
{
    if( m_pstrText )
    {
        delete m_pstrText;
        m_pstrText = 0;
    }
}

CEntry* CEntry::GetNext()
{
    // Returns the next entry (row).
    return m_pNext;
}

void CEntry::SetNext( CEntry* pNext )
{
    // Sets the next entry(row).
    m_pNext = pNext;
}

// This function copies the text in the buffer to the supplied string variable.
int CEntry::GetText( char* pstrText, int Length )
{
    // Make sure the length doesnt overrun our buffer.
    if( Length > MAX_CHARSPERLINE )
    {
        Length = MAX_CHARSPERLINE;
    }
}

```

```

    }

    // Copy the string.
    CopyMemory( pstrText, m_pstrText, Length );

    return S_OK;
}

int CEntry::SetText( char* pstrText )
{
    // This function sets the text in the buffer.

    // Clear out any text that is already there.
    ZeroMemory( m_pstrText, sizeof( char[MAX_CHARSPERLINE] ) );

    // Make sure the supplied text doesn't overrun our buffer.
    int Length = strlen( pstrText );
    if( Length > MAX_CHARSPERLINE )
    {
        Length = MAX_CHARSPERLINE;
    }

    // Copy the supplied text into our buffer.
    CopyMemory( m_pstrText, pstrText, strlen( pstrText ) );

    return S_OK;
}

// This function draws the text to the console surface using the GDI.
int CEntry::RenderText( int NumHorzChars, DWORD* pData, int DestPitch )
{
    // Only draw the number of characters that fit on the screen.
    int Length = strlen( m_pstrText );
    if( Length > NumHorzChars )
    {
        Length = NumHorzChars;
    }

    PrintString( 10, m_nVerticalPos - 5, m_pstrText, TRUE,
                D3DCOLOR_ARGB( 255, 255, 0, 255 ), pData, DestPitch );

    return S_OK;
}

// This function is called whenever a character key is pressed that needs
// to be added to the buffer.
int CEntry::OnChar( char Key )
{
    if( Key == '\b' ) // Check if the backspace key was pressed.
    {
        char pstrTemp[MAX_CHARSPERLINE]; // Make a temporary string holder.
        int Length = strlen( m_pstrText ); // Get the length of the buffer.

        if( Length == 0 ) { return S_OK; } // If the buffer is empty then return.
        Length--; // Reduce the length of the string by one.

        // Copy the string(-1) to the temp string.
        CopyMemory( pstrTemp, m_pstrText, Length );
        // Zero out the buffer.
        ZeroMemory( m_pstrText, sizeof( char[MAX_CHARSPERLINE] ) );
        // Copy the text back into the buffer.
        CopyMemory( m_pstrText, pstrTemp, Length );
    }
    else // A normal key was pressed.
    {
        // Make sure the buffer has not overflowed.
        if( strlen( m_pstrText ) > MAX_CHARSPERLINE )
        {
            return E_FAIL;
        }
        // Append the text buffer with the character.
        strcat( m_pstrText, &Key, 1 );
    }
}

```

```

    }

    return S_OK;
}

/** This holds pre-parsed commands. */
class CCommand
{
public:
    char* pstrCommand;           // Name of the command.
    int NumParams;              // Number of parameters.
    char* pstrParams[MAX_PARAMS]; // Parameters.
    CCommand()
    {
        pstrCommand = 0;
        NumParams = 0;
        ZeroMemory( &pstrParams, sizeof( pstrParams ) );
    }
    ~CCommand()
    {
        if( pstrCommand ) // Destroy the command string if it exists.
        {
            delete pstrCommand;
        }

        // Destroy any parameter strings if they exist.
        for( int i = 0 ; i < MAX_PARAMS ; i++ )
        {
            if( pstrParams[i] )
            {
                delete pstrParams[i];
            }
        }
    }
};

/** Console class. */
class CConsole
{
public:
    static CConsole* Instance();
protected:
    CConsole();
    ~CConsole();
public:
    void Shutdown();
    HRESULT Initialize( LPDIRECT3DDEVICE9 pDevice, LPDIRECT3DSURFACE9 pTargetSurface );
    void Render();
    BOOL GetVisibility(){ return m_bVisible; }
    void SetVisibility( BOOL bVisible ) { m_bVisible = bVisible; }
    void OutputString( char* pString, bool bType );
    void Clear();
    int OnChar( char Key );
    int OnKeyDown( WPARAM wParam );
    void SetParserCallback( CONSOLE_PARSER_CALLBACK pfnCallback );
    void PreParse( char* pstrText, CCommand* pCommand );

    // Command helper functions.
    int ParseStringForNumber( char* pString );
protected:
    void RotateEntries();
protected:
    BOOL m_bInitialized; // Has the console been initialized?
    int m_Width; // The width of the console surface.
    int m_Height; // The height of the console surface.

    // Pointer to the console surface.
    LPDIRECT3DSURFACE9 m_pConsoleSurface;
    // Pointer to the background bitmap surface.
    LPDIRECT3DSURFACE9 m_pConsoleBackgroundSurf;
};

```

```

// Pointer to the target render surface( eg back buffer ).
LPDIRECT3DSURFACE9 m_pTargetSurface;
LPDIRECT3DDEVICE9 m_pDevice;           // Pointer to the device.
BOOL m_bVisible;                       // Is the console visible?

CEntry* m_pActiveEntry;                // The active entry(accepts key input).
CEntry* m_pEntryList;                  // The list of old entries.

// Pointer to an external console parser.
CONSOLE_PARSER_CALLBACK m_pfnCallback;
// Is there an external parser?
BOOL m_bParserCallback;

private:
    static CConsole* _instance;
};

CConsole* CConsole::_instance = 0;

CConsole* CConsole::Instance()
{
    if(_instance == 0)
    {
        _instance = new CConsole;
    }
    return _instance;
}

CConsole::CConsole()
{
    m_pConsoleSurface = 0;
    m_pConsoleBackgroundSurf = 0;
    m_pTargetSurface = 0;

    m_Width = 0;
    m_Height = 0;

    m_bInitialized = FALSE;
    m_bVisible = FALSE;

    m_pActiveEntry = 0;
    m_pEntryList = 0;

    m_pfnCallback = 0;
    m_bParserCallback = FALSE;
}

CConsole::~CConsole()
{
    if( m_pConsoleSurface )           // Release the console surface.
    {
        m_pConsoleSurface->Release();
        m_pConsoleSurface = 0;
    }

    if( m_pConsoleBackgroundSurf )   // Release the background bitmap surface.
    {
        m_pConsoleBackgroundSurf->Release();
        m_pConsoleBackgroundSurf = 0;
    }

    if( m_pTargetSurface )           // Release the target surface.
    {
        m_pTargetSurface->Release();
        m_pTargetSurface = 0;
    }

    // Destroy all of the entries.
    CEntry* pEntry = m_pEntryList;
    CEntry* pTemp = 0;

    if( !pEntry ) { return; }

```

```

while( pEntry->GetNext() )
{
    pTemp = pEntry->GetNext();    // Copy the next entry into a temp pointer.
    delete pEntry;                // Delete the current pointer.
    pEntry = pTemp;               // Set the current pointer to the next pointer.
}

}

void CConsole::Shutdown()
{
    this->-CConsole();            // This should work, since 'this' points to the console.
    m_pEntryList = 0;
    m_bInitialized = FALSE;
}

void CConsole::SetParserCallback( CONSOLE_PARSER_CALLBACK pfnCallback )
{
    // Sets the external command parser.
    m_pfnCallback = pfnCallback; // Set the function pointer.
    m_bParserCallback = TRUE;    // Flag of whether Ptr has been set.
}

void CConsole::Clear()
{
    // Clears the contents of all the entries.
    CEntry* pEntry = m_pEntryList;    // Create a pointer to the first entry.

    while( pEntry->GetNext() )
    {
        pEntry->SetText( "" );        // Set the entries text to nothing.
        pEntry = pEntry->GetNext();    // Get the next entry.
    }
}

int CConsole::OnChar( char Key )
{
    // Handles character input.
    static char LastKey = 0;          // This holds the last key pressed.

    if( !m_bVisible ) { return 0; }    // Ignore the keypress if invisible.
    if( Key == '\r' ) { return 0; }    // Ignore the enter key.
    if( Key == '\t' ) { Key = ' '; }    // Change the tab key to a space key.

    // Only allow one space.
    if( (Key == ' ') && (LastKey == ' ') )
    {
        return 0;
    }

    // Make sure the first character in a line is not a space.
    if( (Key == ' ') && (LastKey == 0) )
    {
        return 0;
    }

    // Only send the message to the active entry if the Last Key is not being reset.
    if( Key != 0 )
    {
        m_pActiveEntry->OnChar( Key );
    }

    LastKey = Key; // Update the last key pressed variable.

    return 0;
}

int CConsole::OnKeyDown( WPARAM wParam )
{
    // Handles non character keyboard input.
    int Result = 0;                // Holds result of parse operation.

    switch( wParam )                // Figure out which key was pressed.
    {
        case VK_F11:                // The F11 key was pressed.

```



```

    {
        SetVisibility( !GetVisibility() );
        g_bConsoleOn = !g_bConsoleOn;
        break;
    }
    case VK_RETURN: // The enter key was pressed.
    {
        // Ignore if the console is not visible.
        if( !m_bVisible ) { return 0; }
        OnChar( 0 ); // Reset the last keypressed.

        if( m_bParserCallback ) // Check if a parser has been set.
        {
            // Create a temporary string.
            char* String = new char[MAX_CHARS_PERLINE];
            // Get the string from the active entry.
            m_pActiveEntry->GetText( String, MAX_CHARS_PERLINE );

            // If nothing was typed just ignore it.
            if( !MATCH( String, " " ) && !MATCH( String, " " ) )
            {
                // Create a new command class.
                CCommand Command;
                // Convert the string into a command.
                PreParse( String, &Command );
                // Send the command to the parser.
                Result = m_pfnCallback( &Command );
                if( FAILED( Result ) )
                {
                    OutputString( "Unknown Command.", true );
                }
            }
            delete String; // Destroy the temporary string.
        }
        RotateEntries(); // Move all the entries up.
        break;
    }
    case VK_LEFT: // The left arrow key was pressed.
    {
        OnChar( '\b' ); // Treat this like a backspace press.
        break;
    }
}
return 0;
}

// Converts the command string into a command and a list of parameters.
void CConsole::PreParse( char* pstrText, CCommand* pCommand )
{
    string sTemp = pstrText;
    string sTemp2(sTemp, 3, sTemp.length());
    pstrText = (char *)sTemp2.c_str();

    // The parameter separators are the comma and space characters.
    char Separators[] = " , ";
    char* Token; // String to hold the current parameter.
    int TokenCount = 0; // The number of parameters.

    strlwr( pstrText ); // Convert the string to lowercase.
    Token = strtok( pstrText, Separators ); // Get the command string.

    if( Token ) // The line was not blank.
    {
        // Set the command string to the token.
        // Create a string in the command to hold the command string.
        pCommand->pstrCommand = new char[ strlen( Token ) + 1 ];
        // Copy the string into the command string.
        strcpy( pCommand->pstrCommand, Token );
    }
    else // The line was blank.
    {
        // Create a single character to hold a blank character.

```

```

    // This will notify the keypress event that the line was empty.
    pCommand->pstrCommand = new char;
    // Coy a blank character into a new.
    strcpy( pCommand->pstrCommand, " " );
    // Set the number of parameters to zero.
    pCommand->NumParams = 0;
    // Return: there is nothing left to do.
    return;
}

// Get the next token in the string.
Token = strtok( NULL, Separators );

// Loop for the rest of the tokens.
while( Token != NULL )
{
    // Allocate memory in the command for this parameter.
    pCommand->pstrParams[TokenCount] = new char[ strlen( Token ) + 1 ];
    // Coy the parameter into the allocated memory.
    strcpy( pCommand->pstrParams[TokenCount], Token );
    // Get the next token in the string.
    Token = strtok( NULL, Separators );
    // Increase the parameter count.
    TokenCount++;
    // Make sure there are not too many parameters.
    if( TokenCount > MAX_PARAMS )
    {
        break;
    }
}

// Set the number of parameters in command.
pCommand->NumParams = TokenCount;
}

void CConsole::OutputString( char* pString, bool bType )
{
    // Output a string to the console.
    // false/0 is normal, true/1 is an error.
    RotateEntries(); // Move the entries up.
    string sNew = pString;
    if(bType)
    {
        // This is a special console message.
        sNew = "> " + sNew;
    }
    else
    {
        // This is normal output.
        sNew = ">> " + sNew;
    }
    m_pActiveEntry->SetText( (char *)sNew.c_str() ); // Set the new active entry.
}

// Moves the entries up one row. The top entry is deleted and a
// new one is created for the bottom.
void CConsole::RotateEntries()
{
    CEntry* pEntry = m_pEntryList; // Get a pointer to the first entry.
    CEntry* pTempEntry = 0; // A temporary entry pointer.
    CEntry* pNewEntry = 0; // A pointer to the new entry.

    while( pEntry->GetNext() ) // Loop for each entry.
    {
        // Set the temp entry to the current entry.
        pTempEntry = pEntry;
        // Increase the vertical position of the current entry.
        pEntry->SetVerticalPos(pEntry->GetVerticalPos() - g_AlphabetLetterHeight);
        // Set the current entry to the next entry.
        pEntry = pEntry->GetNext();
    }

    // The pEntry variable now points to the last
    // entry in the list and pTempEntry to the 2nd last.
}

```

```

delete pEntry; // Delete the last entry in the list.
pEntry = pTempEntry; // Get a pointer to the new last entry.
pEntry->SetNext( NULL ); // Set the new last entries pNext pointer to zero.

pNewEntry = new CEntry; // Create a new entry for the top of the list.

// Set new entries pNext variable to last first entry.
pNewEntry->SetNext( m_pEntryList );
// Set the start of the Entry list to the new entry.
m_pEntryList = pNewEntry;

// Set the active entry to the new entry.
m_pActiveEntry = pNewEntry;
// Set the vertical position of the new entry to the bottom of the console.
m_pActiveEntry->SetVerticalPos( (m_Height-5) - g_AlphabetLetterHeight );

// Set the console prefix for the new line.
m_pActiveEntry->SetText("> ");
}

HRESULT CConsole::Initialize( LPDIRECT3DDEVICE9 pDevice, LPDIRECT3DSURFACE9
pTargetSurface )
{
    if( !pDevice ) // Make sure a valid device was specified.
    {
        return E_FAIL;
    }

    if( !pTargetSurface ) // Make sure a valid target surface was specified.
    {
        return E_FAIL;
    }

    HRESULT r = 0;

    // Keep a local pointer of the device.
    m_pDevice = pDevice;

    // Keep a local copy of the target surface.
    m_pTargetSurface = pTargetSurface;
    m_pTargetSurface->AddRef();

    D3DSURFACE_DESC d3dsd; // Holds information about the target surface.
    m_pTargetSurface->GetDesc( &d3dsd ); // Get information about the target surface.

    // Set the dimensions of the console.
    m_Width = d3dsd.Width - 40;
    m_Height = 240;

    // Create a surface for the console.
    r = m_pDevice->CreateOffscreenPlainSurface( m_Width, m_Height, _
        D3DFMT_A8R8G8B8, D3DPool_SYSTEMMEM, &m_pConsoleSurface, NULL );
    if( FAILED( r ) )
    {
        Debug( "Unable to create image surface for console." );
        Shutdown();
        return E_FAIL;
    }

    // Load the background bitmap for the console.
    r = LoadBitmapToSurface( "img\\console_bg.bmp", &m_pConsoleBackgroundSurf, _
        m_pDevice );
    if( FAILED( r ) )
    {
        Debug( "Unable to load console background image." );
        Shutdown();
        return E_FAIL;
    }

    m_pEntryList = new CEntry; // Start the list with a new entry.

```

```

CEntry* pEntry = m_pEntryList; // Get a temp pointer to the new entry.
m_pActiveEntry = m_pEntryList; // Set the active entry to the start of the list.

// Set the console prefix for the new line.
m_pActiveEntry->SetText("> ");

// Compute the number of visible rows of text.
int VisibleRows = (m_Height / g_AlphabetLetterHeight) - 1;

// Loop for each possible visible row of text.
for( int i = 1 ; i < VisibleRows ; i++ )
{
    // Create a new entry.
    pEntry->SetNext( new CEntry );
    // Set its vertical position to above the previous row.
    pEntry->SetVerticalPos( (m_Height-5) - (i * g_AlphabetLetterHeight) );
    // Get a pointer to the new entry.
    pEntry = pEntry->GetNext();
}

// Set the initialized flag to 'true'.
m_bInitialized = TRUE;

return S_OK;
}

void CConsole::Render()
{
    // Make sure the console has been initialized.
    if( !m_bInitialized ) { return; }

    // If the console is not visible then return.
    if( !m_bVisible ) { return; }

    // Set the source rectangle.
    RECT SourceRect = { 0, 0, m_Width, m_Height };
    // Set the destination point.
    POINT DestPoint = { 20, 20 };

    // Copy the background surface to the console surface.
    D3DXLoadSurfaceFromSurface( m_pConsoleSurface, NULL, NULL,
        m_pConsoleBackgroundSurf, NULL, NULL, D3DX_FILTER_POINT, 0 );

    // Get a pointer to the start of the entry list.
    CEntry* pEntry = m_pEntryList;
    D3DLOCKED_RECT LockedRect;
    m_pConsoleSurface->LockRect( &LockedRect, 0, 0 );

    // Print the title.
    char * szTitleString = "Clarity Console v1.0:";
    PrintString( 12, 10, szTitleString, TRUE, D3DCOLOR_ARGB( 255, 255, 0, 255 ), _
        (DWORD*)LockedRect.pBits, LockedRect.Pitch );
    // Loop for each entry in the list.
    while( pEntry->GetNext() )
    {
        pEntry->RenderText( MAX_CHARS_PERLINE, (DWORD*)LockedRect.pBits, _
            LockedRect.Pitch );
        pEntry = pEntry->GetNext(); // MOVE to the next entry to render.
    }

    m_pConsoleSurface->UnlockRect();

    // Copy the console to the target surface.
    m_pDevice->UpdateSurface( m_pConsoleSurface, &SourceRect, _
        m_pTargetSurface, &DestPoint );
}

```

```

//-----
// File:      input.h
//
// Desc:      This file contains both DirectInput classes.
//
// First created on:  December 28th, 2004
// Last modification: January 4th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
// Partially Based on Original Code By: Peter Walsh, author of "The Ten of
//                                     Direct3D Game Programming"
//-----
// Revision History:
//
// 12-28-04:  Added revision history. File created. Converted code to DX9.
// 12-29-04:  Cleaned up and recommented all of the code in this file.
// 01-04-05:  Converted both input classes to Singletons.
//-----

class CZenMouse
{
public:
    static CZenMouse* Instance();
protected:
    CZenMouse();
    ~CZenMouse();
protected:
    LPDIRECTINPUTDEVICE8 m_pMouseDev;    // The mouse device.
    BOOL m_bInitialized;                // Has the mouse been initialized?
    BOOL m_bShowCursor;                // Is the cursor visible?
    DIMOUSESTATE m_MouseData;          // Used to store mouse data from Windows.
    POINT m_Position;                  // The cursor position.
public:
    HRESULT Initialize();
    HRESULT Poll();                    // Is the mouse in focus?
    POINT GetMousePos();
    BOOL IsButtonDown( int Button );
    BOOL HandleSetCursor();            // Handles the WM_SETCURSORS message. Or should.
    void ShowCursor( BOOL bShow ){ m_bShowCursor = bShow; }
    void SetCursorPosition( int x, int y );
    void GetCursorPosition( int& x, int& y );
    void MoveCursor( int x, int y );
    void UpdateCursorPos();            // Updated from movement data from Windows.
private:
    static CZenMouse* _instance;
};

CZenMouse* CZenMouse::_instance = 0;

CZenMouse* CZenMouse::Instance()
{
    if(_instance == 0)
    {
        _instance = new CZenMouse;
    }
    return _instance;
}

CZenMouse::CZenMouse()
{
    m_pMouseDev = 0;
    m_bInitialized = FALSE;
}

CZenMouse::~CZenMouse()
{
    if( Trycatch((void*)m_pMouseDev, "m_pMouseDev in ~CZenMouse()") )
    {
        m_pMouseDev->Unacquire();
        m_pMouseDev->Release();
    }
}

```

```

}

void CZenMouse::SetCursorPosition( int x, int y )
{
    // The mouse cursor needs to stay inside of the screen.
    if( x < 0 ) { x = 0; }
    if( y < 0 ) { y = 0; }
    if( x > g_DeviceWidth-1 ) { x = g_DeviceWidth-1; }
    if( y > g_DeviceHeight-1 ) { y = g_DeviceHeight-1; }

    // Update the position tracker.
    m_Position.x = x;
    m_Position.y = y;

    // Update the position in the D3D device.
    g_pDevice->SetCursorPosition( x, y, 0 );
}

void CZenMouse::GetCursorPosition( int& x, int& y )
{
    x = m_Position.x;
    y = m_Position.y;
}

void CZenMouse::MoveCursor( int x, int y )
{
    m_Position.x += x;
    m_Position.y += y;

    // Update the position in the D3D device.
    g_pDevice->SetCursorPosition( x, y, 0 );
}

void CZenMouse::UpdateCursorPos()
{
    // Get the relative movement out of the DIMOUSESTATE structure.
    m_Position.x += m_MouseData.lX;
    m_Position.y += m_MouseData.lY;

    // Make sure the point is within screen bounds.
    if( m_Position.x < 0 ) { m_Position.x = 0; }
    if( m_Position.y < 0 ) { m_Position.y = 0; }
    if( m_Position.x > g_DeviceWidth-1 ) { m_Position.x = g_DeviceWidth-1; }
    if( m_Position.y > g_DeviceHeight-1 ) { m_Position.y = g_DeviceHeight-1; }

    // Update the position in the D3D device.
    g_pDevice->SetCursorPosition( m_Position.x, m_Position.y, 0 );
}

BOOL CZenMouse::HandleSetCursor()
{
    // This function handles the WM_SETCURSOR message. Supposedly,
    // there are some problems calling it from WM_SETCURSOR.
    if( !m_bInitialized )
    {
        // Exit the function if the mouse isn't initialized.
        return FALSE;
    }

    if( m_bShowCursor )
    {
        // If the cursor is set to be visible...
        SetCursor( NULL ); // Turn off the Windows cursor.
        g_pDevice->ShowCursor( TRUE ); // Turn on the custom cursor.
        return TRUE; // This tells Windows not to control the cursor.
    }

    return FALSE; // This tells Windows to control the cursor.
}

POINT CZenMouse::GetMousePos()
{
    POINT MousePos;
}

```

```

    // Get mouse position data from the buffer.
    MousePos.x = m_MouseData.lX;
    MousePos.y = m_MouseData.lY;

    return MousePos;
}

// 0 is the primary mouse button, 1 is secondary, 2 is middle.
BOOL CZenMouse::IsButtonDown( int Button )
{
    // Return the button status from the buffer.
    if( m_MouseData.rgbButtons[Button] & 0x80 )
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

HRESULT CZenMouse::Poll()
{
    HRESULT r = 0;

    if( !m_bInitialized )
    {
        // Exit the function if the mouse isn't initialized.
        return E_FAIL;
    }

    // Get the state of the mouse.
    r = m_pMouseDev->GetDeviceState( sizeof( DIMOUSESTATE ), &m_MouseData );
    if( FAILED( r ) )
    {
        if( r == DIERR_INPUTLOST ) // If the mouse has moved focus ...
        {
            while( r == DIERR_INPUTLOST )
            {
                // Reacquire the mouse.
                r = m_pMouseDev->Acquire();
            }

            // Try to test the state again.
            if( SUCCEEDED( r ) )
            {
                m_pMouseDev->GetDeviceState( sizeof( DIMOUSESTATE ), _
                    &m_MouseData );
            }
            else
            {
                return FALSE;
            }
        }
        else
        {
            return E_FAIL;
        }
    }

    return S_OK;
}

HRESULT CZenMouse::Initialize()
{
    HRESULT r = 0;

    // Return if the DirectInput object does not exist.
    if( !Trycatch((void*)g_pDI, "g_pDI in CZenMouse::Initialize()") )
    {
        return E_FAIL;
    }
}

```

```

// Release the mouse device if it has already been created.
if( Trycatch((void*)m_pMouseDev, "m_pMouseDev in CZenMouse::Initialize()") )
{
    m_pMouseDev->Release();
}

// Create the mouse device.
r = g_pDI->CreateDevice( GUID_SysMouse, &m_pMouseDev, NULL );
if( FAILED( r ) )
{
    Debug( "Unable to create mouse device." );
    return E_FAIL;
}

// Set the data format for the mouse.
r = m_pMouseDev->SetDataFormat( &c_dfDIMouse );
if( FAILED( r ) )
{
    Debug( "Unable to set the mouse data format." );
    return E_FAIL;
}

// Set the cooperative level for the mouse.
r = m_pMouseDev->SetCooperativeLevel( g_hwndMain, _
DISCL_EXCLUSIVE | DISCL_FOREGROUND );
if( FAILED( r ) )
{
    Debug( "Unable to set the cooperative level for the mouse." );
    return E_FAIL;
}

// Acquire the physical mouse into the device.
r = m_pMouseDev->Acquire();
if( FAILED( r ) )
{
    Debug( "Unable to acquire mouse." );
    return E_FAIL;
}

// Create a new surface for the mouse pointer image.
g_pDevice->CreateOffscreenPlainSurface( 16, 32, D3DFMT_A8R8G8B8, _
D3DPOOL_SCRATCH, &g_pCursorSurf, NULL );
// Load the image file from disk.
D3DXLoadSurfaceFromFile( g_pCursorSurf, 0, 0, "img\\blue_mouse.bmp", _
0, D3DX_FILTER_NONE, 0, 0 );
// Set the hotspot for the cursor.
g_pDevice->SetCursorProperties( 0, 0, g_pCursorSurf );

// Set the initialization flag to true.
m_bInitialized = TRUE;

return S_OK;
}

class CZenKeyboard
{
public:
    static CZenKeyboard* Instance();
protected:
    CZenKeyboard();
    ~CZenKeyboard();
protected:
    LPDIRECTINPUTDEVICES8 m_pKeyDev;
    char m_KeyBuffer[256];
    BOOL m_bInitialized;
public:
    HRESULT Initialize();
    BOOL IsKeyDown( int Key );
private:
    static CZenKeyboard* _instance;
};

```



```

CZenKeyboard* CZenKeyboard::_instance = 0;

CZenKeyboard* CZenKeyboard::Instance()
{
    if(_instance == 0)
    {
        _instance = new CZenKeyboard;
    }
    return _instance;
}

CZenKeyboard::CZenKeyboard()
{
    ZeroMemory( &m_KeyBuffer, sizeof( m_KeyBuffer ) );
    m_pKeyDev = 0;
    m_bInitialized = FALSE;
}

CZenKeyboard::~CZenKeyboard()
{
    if( Trycatch((void*)m_pKeyDev, "m_pKeyDev in ~CZenKeyboard()") )
    {
        m_pKeyDev->Unacquire();
        m_pKeyDev->Release();
    }
}

HRESULT CZenKeyboard::Initialize()
{
    HRESULT r;

    // Return if the DirectInput object does not exist.
    if( !Trycatch((void*)g_pDI, "g_pDI in CZenKeyboard::Initialize()") )
    {
        return E_FAIL;
    }

    // Release the device if it has already been created.
    if( Trycatch((void*)m_pKeyDev, "m_pKeyDev in CZenKeyboard::Initialize()") )
    {
        m_pKeyDev->Unacquire();
        m_pKeyDev->Release();
    }

    // Create the device for the keyboard.
    r = g_pDI->CreateDevice( GUID_SysKeyboard, &m_pKeyDev, NULL );
    if( FAILED( r ) )
    {
        Debug( "Failed to create key device." );
        return E_FAIL;
    }

    // Set the data format for the device.
    r = m_pKeyDev->SetDataFormat( &c_dfDIKeyboard );
    if( FAILED( r ) )
    {
        Debug( "Unable to set the keyboard data format." );
        return E_FAIL;
    }

    // Set the cooperative level.
    r = m_pKeyDev->SetCooperativeLevel( g_hWndMain, _
        DISCL_FOREGROUND | DISCL_NONEXCLUSIVE );
    if( FAILED( r ) )
    {
        Debug( "Unable to set keyboard cooperative level." );
        return E_FAIL;
    }
}

```

```

// Acquire the device.
r = m_pKeyDev->Acquire();
if( FAILED( r ) )
{
    Debug( "Unable to acquire the keyboard." );
    return E_FAIL;
}

// Set the initialization flag to true.
m_bInitialized = TRUE;

return S_OK;
}

BOOL CZenKeyboard::IsKeyDown( int Key )
{
    HRESULT r = 0;

    // Make sure the keyboard has been initialized.
    if( !m_bInitialized )
    {
        return FALSE;
    }

    // Get the state of the keyboard into the key buffer.
    r = m_pKeyDev->GetDeviceState( sizeof( m_KeyBuffer ), &m_KeyBuffer );
    if( FAILED( r ) )
    {
        if( r == DIERR_INPUTLOST ) // If the device is not acquired...
        {
            while( r == DIERR_INPUTLOST ) // ... then reacquire the device.
            {
                r = m_pKeyDev->Acquire();
            }

            if( SUCCEEDED( r ) )
            {
                m_pKeyDev->GetDeviceState( sizeof( m_KeyBuffer ), _
                    &m_KeyBuffer );
            }
            else
            {
                return FALSE;
            }
        }
        else // ... otherwise it was some other error.
        {
            return FALSE;
        }
    }

    // Check if the key was set.
    if( m_KeyBuffer[Key] & 0x80 )
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}
}

```

```

-----
// File:      world.h
//
// Desc:      This file contains the classes and functions for the World. This
//            includes: WorldSingleton, User, and LocalEntity.
//
// First created on:  October 26th, 2004
// Last modification: February 28th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 11-16-04:  Added revision history. WorldSingleton class is defined.
// 02-27-05:  WorldSingleton class finished. User and LocalEntity added.
//            String conversion and data loading functions implemented.
// 02-28-05:  Added in loading of referenced XML files, and .x meshes.
//            Can now load bitmap data into memory.
// 03-01-05:  Synchronized this file's code with the main simulation.
//-----

/*****
/***** World Singleton *****/
/*****

struct User
{
    int x, y, z;
    double roll, pitch, yaw;
};

struct LocalEntity
{
    string name;
    int x, y;
    float z;
    double roll, pitch, yaw;
    int eid, mid;
    string elib, mlib;
    string xfile;
    bool immobile;
    double height, width, depth;
    double oheight, owidth, odepth;
    double mass, friction;
    float velocity;
    CZenMesh xmesh;
    int ID;
    bool bOnGround;
};

class WorldSingleton
{
public:
    // Returns a pointer to the WorldSingleton.
    static WorldSingleton* Instance();

    // COM wrapper.
    bool WIDFuncCOMWrapper(string filename);

    // Load data to memory.
    bool LoadWIDFile(string filename);
    bool LoadEntityData(LocalEntity * LocalEntity);
    bool LoadMaterialData(LocalEntity * LocalEntity);
    BYTE* LoadBitmap();

    // Data members.
    string sWorldName;
    string sBitmapFilename;
    User TheUser;
    list<LocalEntity *> lstLocalEntities;
    BYTE* HeightMap;
};

```

```

    long ByteRowWidth;    // A row offset in HeightMap.

    bool bIsEmpty;
protected:
    WorldSingleton();
    ~WorldSingleton();
private:
    int nMaxID;           // Used for entity IDs.

    static WorldSingleton* _instance;

    // String conversion functions.
    string ConvertBSTRtToString(_bstr_t bstrString);
    BSTR TB(const char * temp);
    int StringToInt(string temp);
    double StringToDouble(string temp);
};

WorldSingleton* WorldSingleton::_instance = 0;

WorldSingleton* WorldSingleton::Instance()
{
    if(_instance == 0)
    {
        _instance = new WorldSingleton;
    }
    return _instance;
}

WorldSingleton::WorldSingleton()
{
    bIsEmpty = true;
    nMaxID = 1;
}

WorldSingleton::~WorldSingleton()
{
    // Destructor!
}

/** Type Definitions */
typedef WorldSingleton CWorldSingleton;

string WorldSingleton::ConvertBSTRtToString(_bstr_t bstrString)
{
    // Convert a BSTR to a string.
    return (LPCTSTR)bstrString;
}

BSTR WorldSingleton::TB(const char * temp)
{
    // Convert a string to a BSTR.
    _bstr_t bs1 = temp;
    return bs1.copy();
}

int WorldSingleton::StringToInt(string temp)
{
    int n;
    stringstream ssBuffer;
    ssBuffer << temp;
    ssBuffer >> n;
    return n;
}

double WorldSingleton::StringToDouble(string temp)
{
    double n;
    stringstream ssBuffer;
    ssBuffer << temp;
    ssBuffer >> n;
    return n;
}

```

```

bool WorldSingleton::WIDFuncCOMWrapper(string filename)
{
    // The following 'Co' functions are for purposes of handling COM.
    CoInitialize(NULL);
    { // Extra braces for scope only.
        bool bTest = LoadWIDFile(filename);
        if(!bTest)
        {
            Debug( "Loading this .wid file failed." );
            return false;
        }
    }
    CoUninitialize();

    return true;
}

bool WorldSingleton::LoadWIDFile(string filename)
{
    // Variable declaration.
    MSXML2::IXMLDOMNodePtr xNode, xLocalNode, xTemp;
    MSXML2::IXMLDOMNodeListPtr NodeList, EntityList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    LocalEntity * tempLocalEntity;
    _bstr_t bstrTemp;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    filename = "xml\\" + filename;
    bool bLoadXML = xmlDoc->load(filename.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML WID file failed to load." );
        return false;
    }

    // Load 'world|name attribute.
    bstrTemp = _
        xmlDoc->documentElement->attributes->getNamedItem(TB("name"))->nodeValue;
    sWorldName = ConvertBSTRToTString(bstrTemp);

    // Loop through world's data nodes.
    NodeList = xmlDoc->documentElement->childNodes;
    long lNodeCount;
    NodeList->get_length(&lNodeCount);
    for (int i = 0; i < lNodeCount; i++)
    {
        // Get next child node.
        NodeList->get_item(i, &xNode);
        sData = ConvertBSTRToTString(xNode->GetnodeName());

        if(sData == "locals")
        {
            EntityList = xNode->childNodes;
            long lEntityCount;
            EntityList->get_length(&lEntityCount);
            for (int j = 0; j < lEntityCount; j++)
            {
                // Get next child node.
                EntityList->get_item(j, &xLocalNode);

                // Point the temp pointer to a new struct.
                tempLocalEntity = new LocalEntity;
            }
        }
    }
}

```

```

// Load all of the values from file into the new structure.
tempLocalEntity->name = ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "name"))->nodeValue );
tempLocalEntity->x = StringToInt( ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "x"))->nodeValue ) );
tempLocalEntity->y = StringToInt( ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "y"))->nodeValue ) );
tempLocalEntity->z = StringToInt( ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "z"))->nodeValue ) );
tempLocalEntity->roll = StringToDouble(
    ConvertBSTRToTString(
        (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
            "roll"))->nodeValue ) );
tempLocalEntity->pitch = StringToDouble(
    ConvertBSTRToTString(
        (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
            "pitch"))->nodeValue ) );
tempLocalEntity->yaw = StringToDouble(
    ConvertBSTRToTString(
        (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
            "yaw"))->nodeValue ) );
tempLocalEntity->eid = StringToInt( ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "eID"))->nodeValue ) );
tempLocalEntity->elib = ConvertBSTRToTString(
    (_bstr_t)xLocalNode->attributes->getNamedItem(TB(
        "elib"))->nodeValue );
tempLocalEntity->velocity = 0.0;
tempLocalEntity->bOnGround = false;

// Load entity information into memory.
if( LoadEntityData(tempLocalEntity) == false )
{
    return false;
}

// Save the new structure to the list.
tempLocalEntity->ID = nMaxID;
nMaxID++;
lstLocalEntities.push_back(tempLocalEntity);

// Clear the temp pointer.
tempLocalEntity = 0;
}
Debug( "Found locals node!" );
}
else if(sData == "bitmap")
{
    // Load bitmap data from file.
    bstrTemp = xNode->attributes->getNamedItem(TB(
        "filename"))->nodeValue;
    sBitmapFilename = ConvertBSTRToTString(bstrTemp);
}
else if(sData == "user")
{
    // Load user data from file into the User structure.
    TheUser.x = StringToInt( ConvertBSTRToTString(
        (_bstr_t)xNode->attributes->getNamedItem(TB("x"))->nodeValue ) );
    TheUser.y = StringToInt( ConvertBSTRToTString(
        (_bstr_t)xNode->attributes->getNamedItem(TB("y"))->nodeValue ) );
    TheUser.z = StringToInt( ConvertBSTRToTString(
        (_bstr_t)xNode->attributes->getNamedItem(TB("z"))->nodeValue ) );
    TheUser.roll = StringToDouble( ConvertBSTRToTString(
        (_bstr_t)xNode->attributes->getNamedItem(TB(
            "roll"))->nodeValue ) );
    TheUser.pitch = StringToDouble( ConvertBSTRToTString(
        (_bstr_t)xNode->attributes->getNamedItem(TB(

```

```

        "pitch"))->nodeValue ) );
    TheUser.yaw = StringToDouble( ConvertBSTRToTString( _
        (_bstr_t)xNode->attributes->getNamedItem(TB( _
        "yaw"))->nodeValue ) );
    }
    else
    {
        Debug("An invalid node has been found while loading the WID file.");
        return false;
    }
}

Debug( "The world file has been loaded successfully." );

// Load the bitmap to memory.
HeightMap = LoadBitmap();

Debug( "The height map has been loaded successfully." );

bIsEmpty = false;

return true;
}

bool WorldSingleton::LoadEntityData(LocalEntity * LocalEntity)
{
    // Will in mid, mlib, xfile, immobile, height, width, depth.
    // Variable declaration.
    MSXML2::IXMLDOMNodePtr xNode, xSubNode;
    MSXML2::IXMLDOMNodeListPtr EntityList, SubList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    int nID;
    bool bFound;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    string sTemp = "xml\\" + LocalEntity->elib;
    bool bLoadXML = xmlDoc->load(sTemp.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML ELB file failed to load." );
        return false;
    }

    // Loop through the <entity> objects.
    EntityList = xmlDoc->documentElement->childNodes;
    long lEntityCount;
    EntityList->get_length(&lEntityCount);
    for (int i = 0; i < lEntityCount; i++)
    {
        // Get next child node.
        EntityList->get_item(i, &xNode);
        nID = StringToInt( ConvertBSTRToTString( _
            (_bstr_t)xNode->attributes->getNamedItem(TB("ID"))->nodeValue ) );
        if(nID == LocalEntity->eid)
        {
            // Load entity data from file into the structure.
            SubList = xNode->childNodes;
            long lSubCount;
            SubList->get_length(&lSubCount);
            for(int j = 0; j < lSubCount; j++)
            {
                // Get next child node
                SubList->get_item(j, &xSubNode);
                sData = ConvertBSTRToTString(xSubNode->GetnodeName());

                if(sData == "mlib")
                {

```

```

LocalEntity->mlib = ConvertBSTRToTString( _
    (_bstr_t)xSubNode->text );
}
else if(sData == "mID")
{
    LocalEntity->mid = StringToInt( _
        ConvertBSTRToTString( (_bstr_t)xSubNode->text ) );
}
else if(sData == "xfile")
{
    LocalEntity->xfile = ConvertBSTRToTString( _
        (_bstr_t)xSubNode->text );
    string temp = "xmesh\\" + LocalEntity->xfile;
    // Load a x mesh.
    LocalEntity->xmesh.LoadXFile((char *)temp.c_str());
}
else if(sData == "immobile")
{
    int n = StringToInt( ConvertBSTRToTString( _
        (_bstr_t)xSubNode->text ) );
    if(n == 1)
    {
        LocalEntity->immobile = true;
    }
    else
    {
        LocalEntity->immobile = false;
    }
}
else if(sData == "size")
{
    LocalEntity->height = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "height"))->nodeValue ) );
    LocalEntity->width = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "width"))->nodeValue ) );
    LocalEntity->depth = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "depth"))->nodeValue ) );
    LocalEntity->oheight = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "oheight"))->nodeValue ) );
    LocalEntity->owidth = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "owidth"))->nodeValue ) );
    LocalEntity->odepth = StringToDouble( _
        ConvertBSTRToTString( _
            (_bstr_t)xSubNode->attributes->getNamedItem(TB( _
                "odepth"))->nodeValue ) );
}
else
{
    Debug( "An invalid node has been found while _
        loading an ELB file." );
    return false;
}
}

// Load material information into memory.
if( LoadMaterialData(LocalEntity) == false )
{
    return false;
}

// We found the entity ...

```



```

        bFound = true;
        break;
    }
}

if(bFound == false) { return false; }

return true;
}

bool WorldSingleton::LoadMaterialData(LocalEntity * LocalEntity)
{
    // Fill in mass and friction.
    // Variable declaration.
    MSXML2::IXMLDOMNodePtr xNode;
    MSXML2::IXMLDOMNodeListPtr MaterialList;
    MSXML2::IXMLDOMDocumentPtr xmlDoc;
    string sData;
    int nID;
    bool bFound;

    // Create the XML document and load it from file.
    xmlDoc.CreateInstance("MSXML2.DOMDocument.4.0");
    xmlDoc->async = false;
    string sTemp = "xml\\" + LocalEntity->mllib;
    bool bLoadXML = xmlDoc->load(sTemp.c_str());

    // Make sure the document loaded.
    if(!bLoadXML)
    {
        Debug( "XML MLB file failed to load." );
        return false;
    }

    // Loop through the <entity> objects.
    MaterialList = xmlDoc->documentElement->childNodes;
    long lMatCount;
    MaterialList->get_length(&lMatCount);
    for (int i = 0; i < lMatCount; i++)
    {
        // Get next child node.
        MaterialList->get_item(i, &xNode);
        nID = StringToInt( ConvertBSTRToTString(
            (_bstr_t)xNode->attributes->getNamedItem(TB("ID"))->nodeValue ) );

        if(nID == LocalEntity->mid)
        {
            LocalEntity->mass = StringToDouble( ConvertBSTRToTString(
                (_bstr_t)xNode->attributes->getNamedItem(TB(
                    "mass"))->nodeValue ) );
            LocalEntity->friction = StringToDouble( ConvertBSTRToTString(
                (_bstr_t)xNode->attributes->getNamedItem(TB(
                    "friction"))->nodeValue ) );
            bFound = true;
            break;
        }
    }

    if(bFound == false) { return false; }

    return true;
}

BYTE* WorldSingleton::LoadBitmap()
// A big thank you to 'Eric Carr', whose code this function is based on.
// http://www.gamedev.net/community/forums/profile.asp?mode=display&id=6748
// http://www.wannawiki.com/wiki/index.php/Loading_a_24bit_bitmap_in_Direct_Draw
{
    BITMAPINFOHEADER infoheader;
    BYTE *bitmapData;
    BYTE *bitmapDone;
    FILE *bitmapFile;

```

```

BYTE          red, blue, green;
Int           padding;
String        sTemp = "terrain\\" + sBitmapFilename;
char *        filename = (char *)sTemp.c_str();

// Open the bitmap in order to read from it.
bitmapFile = fopen(filename, "rb");
fseek(bitmapFile, sizeof(BITMAPFILEHEADER), SEEK_SET);
fread(&infoheader, sizeof(BITMAPINFOHEADER), 1, bitmapFile);

// Save infoheader.biWidth to a structure in the WorldSingleton.
ByteRowWidth = infoheader.biWidth;

// Get the padding at the end of the bitmap.
padding = 4 - ((infoheader.biWidth * 3) % 4);
if(padding == 4)
{
    padding = 0;
}

// Create space for the bitmap's original and transformed information.
bitmapData = new BYTE[infoheader.biWidth * infoheader.biHeight];
bitmapDone = new BYTE[infoheader.biWidth * infoheader.biHeight];

for( int y = 0; y < infoheader.biHeight; ++y)
{
    for( int x = 0; x < infoheader.biWidth; ++x)
    {
        fread(&blue, sizeof(BYTE), 1, bitmapFile);
        fread(&green, sizeof(BYTE), 1, bitmapFile);
        fread(&red, sizeof(BYTE), 1, bitmapFile);

        // Write each pixel of data into the bitmapData structure.
        bitmapData[y*infoheader.biWidth + x] = red; // white = 255, black = 0.
    }

    // Skip past the padding in the file.
    fseek(bitmapFile, padding, SEEK_CUR);
}

// Transform bitmapData's information into the proper layout.
int heightIndex = 0;
for( y = infoheader.biHeight - 1; y >= 0; --y)
{
    for( int x = 0; x < infoheader.biWidth; ++x)
    {
        bitmapDone[heightIndex*infoheader.biWidth + x] = _
            bitmapData[y*infoheader.biWidth + x];
    }
    ++heightIndex;
}

// Clean up before returning the bitmap.
delete bitmapData;
fclose(bitmapFile);

return bitmapDone;
}

```

```

//-----
// File:      screens.h
//
// Desc:      All of the classes involving screens are defined here. These
//            include: Fontbank, Screen, and Text.
//
// First created on:  October 25th, 2004
// Last modification:  March 1st, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 11-16-04:  Added revision history. Text, Screen and ScreenFactory classes
//            are defined.
// 01-03-05:  Completely rewrote this file from scratch. Implemented the
//            entirety of the Text and Screen classes. Tested text output to
//            the screen.
// 01-04-05:  Added the Fontbank Singleton class to store CZenFont objects.
// 01-05-05:  Updated the Screen and Text classes to be able to handle the
//            storing and retrieval of a WorldFuncPtr (a pointer to a function
//            that can load world data) and a string to store the filename of
//            the .wid file where the world data is stored. Also added more
//            VoidFuncPtr functions for the menus.
// 01-13-05:  Refined the void menu functions.
// 03-01-05:  Screen handling while loading world data works properly.
//-----

/*****
* Section:      Text Class - represents a row of text in a menu.
*****/

class Text
{
public:
    Text();
    Text(int, CZenFont, char *, int, int);
    ~Text();
    Text(const Text& t);           // Copy constructor.
    void operator = (const Text &t); // Overloaded assignment operator.

public:
    void SetAttributes(int, CZenFont, char *, int, int);
    int GetID();
    void SetFuncPtr(VoidFuncPtr);
    VoidFuncPtr GetFuncPtr();
    void SetWorldFuncPtr(WorldFuncPtr);
    WorldFuncPtr GetWorldFuncPtr();
    void SetWorldFile(string);
    string GetWorldFile();
    CZenFont * GetFontPtr();
    char * GetTextPtr();
    int GetX();
    int GetY();
    void Render();

protected:
    int m_nID;
    CZenFont m_Font;
    char * m_pTextString;
    int m_x;
    int m_y;
    // This points to a function that the text is attached to.
    VoidFuncPtr m_pfnFuncPtr;
    // This points to a function that handles world files.
    WorldFuncPtr m_pfnWorldFuncPtr;
    string sWorldFilename;
};

Text::Text ()
{
    m_pTextString = 0;
    m_pfnFuncPtr = 0;
}

```

```

    m_pfnWorldFuncPtr = 0;
    sWorldFilename = "";
}

Text::Text(int ID, CZenFont Font, char * pTextString, int x, int y)
{
    m_nID = ID;
    m_Font = Font;
    m_pTextString = pTextString;
    m_x = x;
    m_y = y;

    m_pfnFuncPtr = 0;
    m_pfnWorldFuncPtr = 0;
    sWorldFilename = "";
}

Text::~Text()
{
    #ifndef QUIET_MODE
    if( QTrycatch((void*)m_pTextString, "m_pTextString in ~Text()") )
    {
        m_pTextString = 0;
    }
    if( QTrycatch((void*)m_pfnFuncPtr, "m_pfnFuncPtr in ~Text()") )
    {
        m_pfnFuncPtr = 0;
    }
    if( QTrycatch((void*)m_pfnWorldFuncPtr, "m_pfnWorldFuncPtr in ~Text()") )
    {
        m_pfnWorldFuncPtr = 0;
    }
    #else
    if( Trycatch((void*)m_pTextString, "m_pTextString in ~Text()") )
    {
        m_pTextString = 0;
    }
    if( Trycatch((void*)m_pfnFuncPtr, "m_pfnFuncPtr in ~Text()") )
    {
        m_pfnFuncPtr = 0;
    }
    if( Trycatch((void*)m_pfnWorldFuncPtr, "m_pfnWorldFuncPtr in ~Text()") )
    {
        m_pfnWorldFuncPtr = 0;
    }
    #endif
}

void Text::SetAttributes(int ID, CZenFont Font, char * pTextString, int x, int y)
{
    m_nID = ID;
    m_Font = Font;
    m_pTextString = pTextString;
    m_x = x;
    m_y = y;
}

Text::Text(const Text& t)
{
    this->m_nID = t.m_nID;
    this->m_Font = t.m_Font;
    this->m_pTextString = t.m_pTextString;
    this->m_x = t.m_x;
    this->m_y = t.m_y;
    this->m_pfnFuncPtr = t.m_pfnFuncPtr;
    this->m_pfnWorldFuncPtr = t.m_pfnWorldFuncPtr;
    this->sWorldFilename = t.sWorldFilename;
}

void Text::operator = (const Text &t)
{

```

```

    this->m_nID = t.m_nID;
    this->m_Font = t.m_Font;
    this->m_pTextString = t.m_pTextString;
    this->m_x = t.m_x;
    this->m_y = t.m_y;
    this->m_pfnFuncPtr = t.m_pfnFuncPtr;
    this->m_pfnWorldFuncPtr = t.m_pfnWorldFuncPtr;
    this->sWorldFilename = t.sWorldFilename;
}

int Text::GetID()
{
    return m_nID;
}

void Text::SetFuncPtr(VoidFuncPtr FuncPtr)
{
    m_pfnFuncPtr = FuncPtr;
}

VoidFuncPtr Text::GetFuncPtr()
{
    return m_pfnFuncPtr;
}

void Text::SetWorldFuncPtr(WorldFuncPtr FuncPtr)
{
    m_pfnWorldFuncPtr = FuncPtr;
}

WorldFuncPtr Text::GetWorldFuncPtr()
{
    return m_pfnWorldFuncPtr;
}

void Text::SetWorldFile(string sString)
{
    sWorldFilename = sString;
}

string Text::GetWorldFile()
{
    return sWorldFilename;
}

CZenFont * Text::GetFontPtr()
{
    return &m_Font;
}

char * Text::GetTextPtr()
{
    return m_pTextString;
}

int Text::GetX()
{
    return m_x;
}

int Text::GetY()
{
    return m_y;
}

void Text::Render()
{
    m_Font.OutputText(m_pTextString, m_x, m_y);
}

```

```

/*****
* Section:   Screen class - represents a screen / menu.
*****/

class Screen
{
public:
    static Screen* Instance();
    void Clear();
    HRESULT SetText(int, CZenFont *, char *, int, int);
    HRESULT SetFunc(int, VoidFuncPtr);
    HRESULT SetWorldFunc(int, WorldFuncPtr);
    HRESULT SetWorldFile(int, string);
    list<Text> * GetTextList();

protected:
    Screen();
    ~Screen();

protected:
    list<Text> m_lstScreenText; // This is a list of all text displayed.

private:
    static Screen* _instance;
};

Screen* Screen::_instance = 0;

Screen* Screen::Instance()
{
    if(_instance == 0)
    {
        _instance = new Screen;
    }
    return _instance;
}

Screen::Screen()
{
    // Nothing to construct.
}

Screen::~Screen()
{
    m_lstScreenText.clear();
}

void Screen::Clear()
{
    m_lstScreenText.clear();
}

HRESULT Screen::SetText(int ID, CZenFont * Font, char * pTextString, int x, int y)
{
    Text tmpText(ID, *Font, pTextString, x, y);
    for(list<Text>::iterator i = m_lstScreenText.begin(); _
i != m_lstScreenText.end(); i++)
    {
        if(i->GetID() == ID) // This ID already exists.
        {
            Debug("Duplicate ID found while creating Text for _
Screen construction.");
            return E_FAIL;
        }
    }
    m_lstScreenText.push_back(tmpText);
    return S_OK;
}

HRESULT Screen::SetFunc(int ID, VoidFuncPtr FuncPtr)
{
    for(list<Text>::iterator i = m_lstScreenText.begin(); _
i != m_lstScreenText.end(); i++)
    {

```

```

        if(i->GetID() == ID) // A match has been found.
        {
            i->SetFuncPtr(FuncPtr);
            return S_OK;
        }
    }
    Debug("Tried to set a VoidFuncPtr with an invalid ID.");
    return E_FAIL;
}

HRESULT Screen::SetWorldFunc(int ID, WorldFuncPtr FuncPtr)
{
    for(list<Text>::iterator i = m_lstScreenText.begin(); _
    i != m_lstScreenText.end(); i++)
    {
        if(i->GetID() == ID) // A match has been found.
        {
            i->SetWorldFuncPtr(FuncPtr);
            return S_OK;
        }
    }
    Debug("Tried to set a WorldFuncPtr with an invalid ID.");
    return E_FAIL;
}

HRESULT Screen::SetWorldFile(int ID, string sString)
{
    for(list<Text>::iterator i = m_lstScreenText.begin(); _
    i != m_lstScreenText.end(); i++)
    {
        if(i->GetID() == ID) // A match has been found.
        {
            i->SetWorldFile(sString);
            return S_OK;
        }
    }
    Debug("Tried to set a WorldFile with an invalid ID.");
    return E_FAIL;
}

list<Text> * Screen::GetTextList()
{
    return &m_lstScreenText;
}

/*****
* Section: Fontbank - Stores CZenFont objects by ID.
*****/

class Fontbank
{
public:
    static Fontbank* Instance();
    void AddFont(int, CZenFont);
    CZenFont* GetFont(int);
protected:
    Fontbank();
    ~Fontbank();
protected:
    vector<CZenFont> m_Fonts;
private:
    static Fontbank* _instance;
};

Fontbank* Fontbank::_instance = 0;

Fontbank* Fontbank::Instance()
{
    if(_instance == 0)
    {
        _instance = new Fontbank;
    }
}

```

```

    }
    return _instance;
}

Fontbank::Fontbank()
{
    m_Fonts.resize(MAX_FONTS);
}

Fontbank::~Fontbank()
{
    m_Fonts.clear();
}

void Fontbank::AddFont(int ID, CZenFont Font)
{
    m_Fonts[ID] = Font;
}

CZenFont* Fontbank::GetFont(int ID)
{
    return m_Fonts[ID].GetPtrToSelf();
}

/*****
* Section:    void Menu functions.
*****/

void ExitSimulator() // Called from States 0, 4. (Main, Pause)
{
    static bool called = 0; // Do not need to reset this. Only called once, max.
    if(!called)
    {
        g_nStateFlag = 5;
        DestroyScene();
        InitScene();
        Debug("ExitSimulator() called.");
        called = 1;
    }
}

void LoadWorldScreen() // Called from State 0. (Main)
{
    static bool called = 0; // Do not need to reset. Only called once, max.
    if(!called)
    {
        g_nStateFlag = 1;
        DestroyScene();
        InitScene();
        Debug("LoadWorldData() called.");
        called = 1;
    }
}

void ResumeSim() // Called from State 4. (Pause)
{
    g_bPauseLock = true;

    DWORD dwLastPauseTime = timeGetTime();
    DWORD dwCurrentTime = timeGetTime();
    while(1)
    {
        if((dwCurrentTime - dwLastPauseTime) >= PAUSE_WAIT)
        {
            break;
        }
        dwCurrentTime = timeGetTime();
    }

    g_nStateFlag = 3;
    DestroyScene();
}

```



```

    InitScene();
    Debug("ResumeSim() called.");

    g_bPauseLock = false;
}

void CallLoadWorld( string sWorldFile )    // Called from State 1. (Load World)
{
    static bool called = 0;
    if(sWorldFile == "ResetCalled") { called = 0; return; }    // Reset.
    if(!called)
    {
        Debug("CallLoadWorld() called for filename:");
        Debug(sWorldFile.c_str());

        CZenCamera * g_Camera = CZenCamera::Instance();
        g_Camera->Reset();
        // This is where the call to World.LoadWorldFromFile(sWorldFile) would go.
        WorldSingleton * World = WorldSingleton::Instance();
        // Removes previously loaded entities from the WorldSingleton.
        World->lstLocalEntities.clear();
        if(World->WIDFuncCOMWrapper(sWorldFile))
        {
            // Everything loaded correctly.
            g_nStateFlag = 3;
            DestroyScene();
            InitScene();
        }
        else
        {
            // Something failed to load.
            // Clear out WorldSingleton and retry? Not necessary.

            // Exit for now.
            PostQuitMessage( 0 );
        }
        called = 1;
    }
}

void ExitToWorldScreen()    // Called from State 4. (Pause)
{
    g_nStateFlag = 1;
    DestroyScene();
    InitScene();
    Debug("ExitToWorldScreen() called.");
    CallLoadWorld("ResetCalled"); // Allows access to the menu item again.

    // Destruct simulation and then proceed to switch to state #1?
}

void PauseSim()    // Called from HandleInput() in origins.cpp.
{
    g_bPauseLock = true;

    DWORD dwLastPauseTime = timeGetTime();
    DWORD dwCurrentTime = timeGetTime();
    while(1)
    {
        if((dwCurrentTime - dwLastPauseTime) >= PAUSE_WAIT)
        {
            break;
        }
        dwCurrentTime = timeGetTime();
    }

    g_nStateFlag = 4;
    DestroyScene();
    InitScene();
    Debug("PauseSim() called.");

    g_bPauseLock = false;
}

```

```

//-----
// File:      terrain.h
//
// Desc:      The classes needed to manipulate and render the terrain.
//
// First created on:  March 1st, 2005
// Last modification: March 12th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 03-01-05:  This file was created. A rough outline of how to organize and
//            render the terrain has been put together.
// 03-02-05:  The creation of the 250K Vertices has been tested successfully.
// 03-02-05:  The very complicated GetHeight() function now returns the
//            height of the terrain at any location.
// 03-12-05:  Added in a second Vertex Buffer to contain an elevated version
//            of the terrain in order to render a wiremap on top of the
//            terrain as a temporary alternative to shadows and shading.
//-----

```

```
#define DEBUG 0
```

```
const double SHORT_SIZE = 2.0;
const double RAISE_WIREFRAME = 0.03;
```

```
class TerrainSingleton
{
public:
    // Returns a pointer to the TerrainSingleton.
    static TerrainSingleton* Instance();

    // Functions.
    bool CreateVertexBuffer();
    bool CreateElevatedVertexBuffer();
    bool Render(bool);
    float GetHeight(float x, float y);

    // Vertex buffer.
    CZenVertex zvTerrain[500][500];
    CZenVertex zvTerrainWire[500][500];

    bool bIsEmpty;
protected:
    TerrainSingleton();
    ~TerrainSingleton();
private:
    static TerrainSingleton* _instance;
    LPDIRECT3DVERTEXBUFFER9 pVB[499]; // Buffers to hold vertices.
    LPDIRECT3DVERTEXBUFFER9 pVBW[499]; // Buffers to hold vertices.
};

```

```
TerrainSingleton* TerrainSingleton::_instance = 0;
```

```
TerrainSingleton* TerrainSingleton::Instance()
{
    if(_instance == 0)
    {
        _instance = new TerrainSingleton;
    }
    return _instance;
}

```

```
TerrainSingleton::TerrainSingleton()
{
    bIsEmpty = true;
}

```

```
TerrainSingleton::~TerrainSingleton()
{
}

```

```

    // Destructor!
}

/** Type Definitions */
typedef TerrainSingleton CTerrainSingleton;

bool TerrainSingleton::CreateVertexBuffer()
{
    WorldSingleton * World = WorldSingleton::Instance();

    if(World->bIsEmpty)
    {
        Debug("The TerrainSingleton tried to created a VB without _
            the WorldSingleton being initialized.");
        return 0;
    }

    int nSum;
    float fMagnitude;

    // Add vertex positions to my vertex list. Also compute vertex normals.
    for(int i = 0; i < 500; i++)
    {
        for(int j = 0; j < 500; j++)
        {
            // *x, *z, *y (height)
            zvTerrain[i][j].m_Position.x = j*2;
            zvTerrain[i][j].m_Position.z = 998 - i*2; // Switch y and z.
            zvTerrain[i][j].m_Position.y =
                (int)World->HeightMap[i * World->ByteRowWidth + j] / SHORT_SIZE;
            // Vertex normals!
            nSum = (zvTerrain[i][j].m_Position.x * zvTerrain[i][j].m_Position.x) +
                (zvTerrain[i][j].m_Position.y * zvTerrain[i][j].m_Position.y) +
                (zvTerrain[i][j].m_Position.z * zvTerrain[i][j].m_Position.z);
            fMagnitude = sqrt((float)nSum);

            zvTerrain[i][j].m_Normal.x = zvTerrain[i][j].m_Position.x / fMagnitude;
            zvTerrain[i][j].m_Normal.y = zvTerrain[i][j].m_Position.y / fMagnitude;
            zvTerrain[i][j].m_Normal.z = zvTerrain[i][j].m_Position.z / fMagnitude;

            // Color.
            zvTerrain[i][j].m_DiffuseColor = g_dwTerrainColor;
            zvTerrain[i][j].m_SpecularColor = g_dwTerrainColor;
        }
    }

    // Arrange vertices in a specific order.
    DWORD m_dwSizeofVertices;
    CZenVertex zvStrip[1000]; // Should I add a safety to clear out data each loop?

    for(int k = 0; k < 499; k++)
    {
        // The strip is filled with the appropriate vertices.
        for(int m = 0; m < 500; m++)
        {
            zvStrip[(m * 2) + 1] = zvTerrain[k][m];
            zvStrip[m * 2] = zvTerrain[k+1][m];
        }

        // Calculate the size of the vertex strip.
        m_dwSizeofVertices = sizeof(zvStrip);

        // // Create the vertex buffer.
        if( FAILED( g_pDevice->CreateVertexBuffer( m_dwSizeofVertices, _
            D3DUSAGE_WRITEONLY, ZENVERTEX_TYPE, D3DPOOL_MANAGED, &vB[k], NULL ) ) )
        {
            Debug("The creation of a VB in the TerrainSingleton failed.");
            return 0;
        }

        // Lock the buffer, copy the data in, unlock.
    }
}

```

```

CZenVertex* pVertices = new CZenVertex;;

// The third parameter changed from BYTE** to VOID** in DX9.
if( FAILED( pVB[k]->Lock( 0, m_dwSizeofVertices, _
(VOID**) &pVertices, 0 ) ) )
{
    Debug("The filling of a VB in the TerrainSingleton failed.");
    return 0;
}
memcpy( pVertices, zvStrip, m_dwSizeofVertices);

pVB[k]->Unlock();
}

bIsEmpty = false;
return 1;
}

bool TerrainSingleton::CreateElevatedVertexBuffer()
{
    WorldSingleton * World = WorldSingleton::Instance();

    if(World->bIsEmpty)
    {
        Debug("The TerrainSingleton tried to created a VB-W without _
the WorldSingleton being initialized.");
        return 0;
    }

    int nSum;
    float fMagnitude;

    // Add vertex positions to my vertex list. Also compute vertex normals.
    for(int i = 0; i < 500; i++)
    {
        for(int j = 0; j < 500; j++)
        {
            // *x, *z, *y (height)
            zvTerrainWire[i][j].m_Position.x = j*2;
            zvTerrainWire[i][j].m_Position.z = 998 - i*2; // Switch y and z.
            zvTerrainWire[i][j].m_Position.y = ((int)World->HeightMap[i _
* World->ByteRowWidth + j] + RAISE_WIREFRAME) / SHORT_SIZE;

            // Vertex normals:
            nSum = (zvTerrainWire[i][j].m_Position.x * zvTerrainWire[i][j].m_Position.x) +
(zvTerrainWire[i][j].m_Position.y * zvTerrainWire[i][j].m_Position.y) +
(zvTerrainWire[i][j].m_Position.z * zvTerrainWire[i][j].m_Position.z);
            fMagnitude = sqrt((float)nSum);

            zvTerrainWire[i][j].m_Normal.x = zvTerrainWire[i][j].m_Position.x / fMagnitude;
            zvTerrainWire[i][j].m_Normal.y = zvTerrainWire[i][j].m_Position.y / fMagnitude;
            zvTerrainWire[i][j].m_Normal.z = zvTerrainWire[i][j].m_Position.z / fMagnitude;

            // Color.
            zvTerrainWire[i][j].m_DiffuseColor = g_dwTerrainWireColor;
            zvTerrainWire[i][j].m_SpecularColor = g_dwTerrainWireColor;
        }
    }

    // Arrange vertices in a specific order.
    DWORD m_dwSizeofVertices;
    CZenVertex zvStrip[1000]; // Should I add a safety to clear out data each loop?

    for(int k = 0; k < 499; k++)
    {
        // The strip is filled with the appropriate vertices.
        for(int m = 0; m < 500; m++)
        {
            zvStrip[(m * 2) + 1] = zvTerrainWire[k][m];
            zvStrip[m * 2] = zvTerrainWire[k+1][m];
        }
    }
}

```

```

// Calculate the size of the vertex strip.
m_dwSizeofVertices = sizeof(zvStrip);

// // Create the vertex buffer.
if( FAILED( g_pDevice->CreateVertexBuffer( m_dwSizeofVertices, _
D3DUSAGE_WRITEONLY, ZENVERTEX_TYPE, D3DPOOL_MANAGED, &pVBW[k], NULL ) ) )
{
    Debug("The creation of a VB in the TerrainSingleton failed.");
    return 0;
}

// Lock the buffer, copy the data in, unlock.
CZenVertex* pVertices = new CZenVertex;;

// The third parameter changed from BYTE** to VOID** in DX9.
if( FAILED( pVBW[k]->Lock( 0, m_dwSizeofVertices, _
(VOID**)&pVertices, 0 ) ) )
{
    Debug("The filling of a VB in the TerrainSingleton failed.");
    return 0;
}
memcpy( pVertices, zvStrip, m_dwSizeofVertices);

pVBW[k]->Unlock();
}

bIsEmpty = false;
return 1;
}

bool TerrainSingleton::Render(bool bWire)
{
    if(bWire)
    {
        for(int k = 0; k < 499; k++)
        {
            // Passing an FVF to IDirect3DDevice9::SetFVF specifies
            // a legacy FVF with stream 0.
            g_pDevice->SetFVF( ZENVERTEX_TYPE );
            g_pDevice->SetStreamSource( 0, pVBW[k], 0, sizeof(CZenVertex) );
            g_pDevice->DrawPrimitive( D3DPT_TRIANGLESTRIP, 0, 998);
        }
    }
    else
    {
        for(int k = 0; k < 499; k++)
        {
            // Passing an FVF to IDirect3DDevice9::SetFVF specifies
            // a legacy FVF with stream 0.
            g_pDevice->SetFVF( ZENVERTEX_TYPE );
            g_pDevice->SetStreamSource( 0, pVBW[k], 0, sizeof(CZenVertex) );
            g_pDevice->DrawPrimitive( D3DPT_TRIANGLESTRIP, 0, 998);
        }
    }

    return 1;
}

// Depending on where the camera is, there are 30 possible mathematical equations
// necessary in order to calculate the proper height of the terrain at any given point.
float TerrainSingleton::GetHeight(float x, float z) // Not affected by SHORT_SIZE.
{
    if(x < 0 || x > 998 || z < 0 || z > 998) // Out of Bounds.
    {
        return -1.0;
    }

    float fResult;
    float fx, fz;
    if(x != 0) { fx = x / 2.0f; }
    else{ fx = 0; }

```

```

if(z != 0) { fz = z / 2.0f; }
else{ fz = 0; }

int xMin, xMax, zMin, zMax;
xMin = floor(fx);
xMax = ceil(fx);
zMin = floor(fz);
zMax = ceil(fz);

float fxRem = fx - xMin;
float fzRem = fz - zMin;

if((xMin == xMax) && (zMin == zMax))
{
    // We are on a vertex.
    fResult = zvTerrain[499-zMin][xMin].m_Position.y;
    #if DEBUG
    Debug("Case 1");
    #endif
}
else if(xMin == xMax)
{
    // We are on a vertical line between vertices.
    int y1 = zvTerrain[499-zMin][xMin].m_Position.y; // x is arbitrary.
    int y2 = zvTerrain[499-zMax][xMin].m_Position.y; // x is arbitrary.
    if(y1 == y2) // The ground here is flat.
    {
        fResult = y1;
        #if DEBUG
        Debug("Case 2");
        #endif
    }
    else if(y1 < y2) // Looking north, slope is up.
    {
        fResult = y1 + fzRem * (y2 - y1);
        #if DEBUG
        Debug("Case 3");
        #endif
    }
    else // Looking north, slope is down.
    {
        fResult = y1 - fzRem * (y1 - y2);
        #if DEBUG
        Debug("Case 4");
        #endif
    }
}
else if(zMin == zMax)
{
    // We are on a horizontal line between vertices.
    int y1 = zvTerrain[499-zMin][xMin].m_Position.y; // z is arbitrary.
    int y2 = zvTerrain[499-zMin][xMax].m_Position.y; // z is arbitrary.
    if(y1 == y2) // The ground here is flat.
    {
        fResult = y1;
        #if DEBUG
        Debug("Case 5");
        #endif
    }
    else if(y1 < y2) // Looking east, slope is up.
    {
        fResult = y1 + fxRem * (y2 - y1);
        #if DEBUG
        Debug("Case 6");
        #endif
    }
    else // Looking east, slope is down.
    {
        fResult = y1 - fxRem * (y1 - y2);
        #if DEBUG
        Debug("Case 7");
        #endif
    }
}
}

```

```

else if(((fxRem + fzRem) > 0.99f) && ((fxRem + fzRem) < 1.01f))
{
    // We are on a diagonal line between the TL and BR vertices in a square.
    int y1 = zvTerrain[499-zMax][xMin].m_Position.y;    // TL vertex.
    int y2 = zvTerrain[499-zMin][xMax].m_Position.y;    // BR vertex.
    if(y1 == y2)    // The ground here is flat.
    {
        fResult = y1;
        #if DEBUG
        Debug("Case 8");
        #endif
    }
    else if(y1 < y2)    // Looking south-east, slope is up.
    {
        fResult = y1 + fxRem * (y2 - y1);    // fx is a good estimate.
        #if DEBUG
        Debug("Case 9");
        #endif
    }
    else    // Looking south-east, slope is down.
    {
        fResult = y1 - fxRem * (y1 - y2);    // fx is a good estimate.
        #if DEBUG
        Debug("Case 10");
        #endif
    }
}
else if((fxRem + fzRem) > 1.0f)    // 13 cases of triangle slopes.
{
    // The camera is in the top-right triangle of the current cell.
    int y1 = zvTerrain[499-zMax][xMin].m_Position.y;    // TL vertex.
    int y2 = zvTerrain[499-zMax][xMax].m_Position.y;    // TR vertex.
    int y3 = zvTerrain[499-zMin][xMax].m_Position.y;    // BR vertex.
    if((y1 == y2) && (y2 == y3))    // Case 1.
    {
        fResult = y1;
        #if DEBUG
        Debug("Case 11");
        #endif
    }
    else if(y1 == y2)
    {
        if(y3 < y2)    // Case 5.
        {
            // Looking north, slope is up.
            fResult = y3 + fzRem * (y2 - y3);
            #if DEBUG
            Debug("Case 12");
            #endif
        }
        else    // Case 2.
        {
            // Looking north, slope is down.
            fResult = y3 - fzRem * (y3 - y2);
            #if DEBUG
            Debug("Case 13");
            #endif
        }
    }
    else if(y2 == y3)
    {
        if(y1 < y2)    // Case 7.
        {
            // Looking east, slope is up.
            fResult = y1 + fxRem * (y2 - y1);
            #if DEBUG
            Debug("Case 14");
            #endif
        }
        else    // Case 4.
        {
            // Looking east, slope is down.
            fResult = y1 - fxRem * (y1 - y2);
            #if DEBUG
            Debug("Case 15");
            #endif
        }
    }
}

```

```

}
else if(y1 == y3)
{
    if(y1 < y2) // Case 3.
    {
        // Looking north-east, slope is up.
        fResult = y1 + ((fxRem + fzRem) - 1.0) * (y2 - y1);
        #if DEBUG
        Debug("Case 16");
        #endif
    }
    else // Case 6.
    {
        // Looking north-east, slope is down.
        fResult = y1 - ((fxRem + fzRem) - 1.0) * (y1 - y2);
        #if DEBUG
        Debug("Case 17");
        #endif
    }
}
else if((y1 < y2) && (y2 < y3)) // Case 8.
{
    fResult = y1 + (fxRem * (y2 - y1)) + ((1.0 - fzRem) * (y3 - y2));
    #if DEBUG
    Debug("Case 18");
    #endif
}
else if((y3 < y2) && (y2 < y1)) // Case 13.
{
    fResult = y3 + (fzRem * (y2 - y3)) + ((1.0 - fxRem) * (y1 - y2));
    #if DEBUG
    Debug("Case 19");
    #endif
}
else if((y2 < y1) && (y1 < y3)) // Case 10.
{
    fResult = y1 - (fxRem * (y1 - y2)) + ((1.0 - fzRem) * (y3 - y2));
    #if DEBUG
    Debug("Case 20");
    #endif
}
else if((y2 < y3) && (y3 < y1)) // Case 11.
{
    fResult = y3 - (fzRem * (y3 - y2)) + ((1.0 - fxRem) * (y1 - y2));
    #if DEBUG
    Debug("Case 21");
    #endif
}
else if((y3 < y1) && (y1 < y2)) // Case 12.
{
    fResult = y1 + (fxRem * (y1 - y3)) - ((1.0 - fzRem) * (y2 - y3));
    #if DEBUG
    Debug("Case 22");
    #endif
}
else if((y1 < y3) && (y3 < y2)) // Case 9.
{
    fResult = y3 + (fzRem * (y3 - y1)) - ((1.0 - fxRem) * (y2 - y1));
    #if DEBUG
    Debug("Case 23");
    #endif
}
else
{
    Debug("An error has occurred while calculating the _
        terrain height.");
    fResult = -1.0;
}
}
else // (fxRem + fzRem) < 1.0f
{
    // The camera is in the bottom-left triangle of the current cell.
    int y1 = zvTerrain[499-zMin][xMin].m_Position.y; // BL vertex.
    int y2 = zvTerrain[499-zMax][xMin].m_Position.y; // TL vertex.
}

```



```

int y3 = zvTerrain[499-zMin][xMax].m_Position.y; // BR vertex.
if((y1 == y2) && (y2 == y3)) // Case 1.
{
    fResult = y1;
    #if DEBUG
    Debug("Case 24");
    #endif
}
else if(y1 == y2)
{
    if(y3 < y2) // Case 5.
    {
        // Looking east, slope is down.
        fResult = y3 + (1.0 - fxRem) * (y2 - y3);
        #if DEBUG
        Debug("Case 25");
        #endif
    }
    else // Case 2.
    {
        // Looking east, slope is up.
        fResult = y3 - (1.0 - fxRem) * (y3 - y2);
        #if DEBUG
        Debug("Case 26");
        #endif
    }
}
else if(y2 == y3)
{
    if(y1 < y2) // Case 7.
    {
        // Looking south-west, slope is down.
        fResult = y1 + (fxRem + fzRem) * (y2 - y1);
        #if DEBUG
        Debug("Case 27");
        #endif
    }
    else // Case 4.
    {
        // Looking south-west, slope is up.
        fResult = y1 - (fxRem + fzRem) * (y1 - y2);
        #if DEBUG
        Debug("Case 28");
        #endif
    }
}
else if(y1 == y3)
{
    if(y1 < y2) // Case 3.
    {
        // Looking north, slope is up.
        fResult = y1 + fzRem * (y2 - y1);
        #if DEBUG
        Debug("Case 29");
        #endif
    }
    else // Case 6.
    {
        // Looking north, slope is down.
        fResult = y1 - fzRem * (y1 - y2);
        #if DEBUG
        Debug("Case 30");
        #endif
    }
}
else if((y1 < y2) && (y2 < y3)) // Case 8.
{
    fResult = y2 + (fxRem * (y3 - y1)) - ((1.0 - fzRem) * (y2 - y1));
    #if DEBUG
    Debug("Case 31");
    #endif
}
else if((y3 < y2) && (y2 < y1)) // Case 13.
{
    fResult = y2 - (fxRem * (y1 - y3)) + ((1.0 - fzRem) * (y1 - y2));
    #if DEBUG
    Debug("Case 32");
    #endif
}

```

```

        #endif
    }
    else if((y2 < y1) && (y1 < y3))           // Case 10.
    {
        fResult = y1 + (fxRem * (y3 - y1)) - (fzRem * (y1 - y2));
        #if DEBUG
        Debug("Case 33");
        #endif
    }
    else if((y2 < y3) && (y3 < y1))           // Case 11.
    {
        fResult = y3 + ((1.0 - fxRem) * (y1 - y3)) - (fzRem * (y1 - y2));
        #if DEBUG
        Debug("Case 34");
        #endif
    }
    else if((y3 < y1) && (y1 < y2))           // Case 12.
    {
        fResult = y1 - (fxRem * (y1 - y3)) + (fzRem * (y2 - y1));
        #if DEBUG
        Debug("Case 35");
        #endif
    }
    else if((y1 < y3) && (y3 < y2))           // Case 9.
    {
        fResult = y3 - ((1.0 - fxRem) * (y3 - y1)) + (fzRem * (y2 - y1));
        #if DEBUG
        Debug("Case 36");
        #endif
    }
    else
    {
        Debug("An error has occurred while calculating the _
            terrain height.");
        fResult = -1.0;
    }
}
return fResult;
}

```

```

//-----
// File:      physics.h
//
// Desc:      This file contains physics functions.
//
// First created on:  March 25th, 2005
// Last modification:  April 7th, 2005
//
// Copyright (c) Jason M. Black (donblas@donblas.org)
//-----
// Revision History:
//
// 03-25-05:  File created. Added in gravity for the user/camera.
// 04-07-05:  Added in gravity for entities.
//-----

bool g_bCameraHitGround = true;
double g_dJumpVelocity = 5.0;
double g_dGravityFactor = 8.0;
double g_dGravity = 9.8 / g_dGravityFactor;

CZenCamera * Camera = CZenCamera::Instance();
TerrainSingleton * Terrain = TerrainSingleton::Instance();

void CameraJump()
{
    Camera->SetVelocity(0, (float)g_dJumpVelocity, 0);
}

void CameraGravity(bool bHitGround)
{
    float x, y, z;

    if(!bHitGround)
    {
        Camera->GetVelocity(x, y, z);
        float newVelocity = y - (float)g_dGravity;
        Camera->SetVelocity(x, newVelocity, z);
    }
    else
    {
        Camera->GetVelocity(x, y, z);
        Camera->SetVelocity(x, 0, z);

        Camera->GetPosition(x, y, z);
        Camera->SetPosition(x, Terrain->GetHeight(x, z) + 5.0, z);
    }
}

// This searched for the highest vertex in a given area where x,y is a central
// point of a square with the width and depth specified. This is used
// in order to detect when a flat, square surface collides with the terrain.
float FindHighestTerrainVertex(float x, float y, float width, float depth)
{
    int i, j;
    float fGreatestHeight = Terrain->GetHeight(x, y);

    float left, right, front, back;
    left = x - (0.5 * width);
    right = x + (0.5 * width);
    front = y - (0.5 * depth);
    back = y + (0.5 * depth);

    // Check covered vertices.
    for(i = (int)ceil(left); i < (int)floor(right); i++)
    {
        for(j = (int)ceil(front); j < (int)floor(back); j++)
        {
            if( Terrain->GetHeight((float)i, (float)j) > fGreatestHeight )
            {
                fGreatestHeight = Terrain->GetHeight((float)i, (float)j);
            }
        }
    }
}

```

```

    }
}

// Check edges.
for(i = (int)ceil(left); i < (int)floor(right); i++)
{
    if( Terrain->GetHeight((float)i, back) > fGreatestHeight )
    {
        fGreatestHeight = Terrain->GetHeight((float)i, back);
    }
    if( Terrain->GetHeight((float)i, front) > fGreatestHeight )
    {
        fGreatestHeight = Terrain->GetHeight((float)i, front);
    }
}

for(j = (int)ceil(front); j < (int)floor(back); j++)
{
    if( Terrain->GetHeight(left, (float)j) > fGreatestHeight )
    {
        fGreatestHeight = Terrain->GetHeight(left, (float)j);
    }
    if( Terrain->GetHeight(right, (float)j) > fGreatestHeight )
    {
        fGreatestHeight = Terrain->GetHeight(right, (float)j);
    }
}

// Check corners.
if( Terrain->GetHeight(left, front) > fGreatestHeight )
{
    fGreatestHeight = Terrain->GetHeight(left, front);
}
if( Terrain->GetHeight(left, back) > fGreatestHeight )
{
    fGreatestHeight = Terrain->GetHeight(left, back);
}
if( Terrain->GetHeight(right, front) > fGreatestHeight )
{
    fGreatestHeight = Terrain->GetHeight(right, front);
}
if( Terrain->GetHeight(right, back) > fGreatestHeight )
{
    fGreatestHeight = Terrain->GetHeight(right, back);
}

return fGreatestHeight;
}

// This functions causes gravity to effect all loaded entities.
void EntityGravity(LocalEntity * obj)
{
    if(obj->bOnGround)
    {
        // This object is done falling.
        return;
    }

    if(obj->z > 0)
    {
        // Object is at least above its central vertex.
        float fTerrain, fCentral;
        fTerrain = FindHighestTerrainVertex(obj->x*2, obj->y*2, obj->width, _
            obj->depth);
        fCentral = Terrain->GetHeight(obj->x*2, obj->y*2);

        if( ( (fTerrain - 0.01) < (fCentral + obj->z) ) &&
            ( (fTerrain + 0.01) > (fCentral + obj->z) ) )
        {
            // The entity is touching terrain.
            obj->bOnGround = true;
            obj->velocity = 0;
            return;
        }
    }
}

```

```
    }
    else
    {
        // The entity is still falling.
        obj->velocity -= (float)g_dGravity;
        if((obj->z + obj->velocity) > fTerrain)
        {
            // Keep falling!
            Debug("Falling!");
            obj->z += obj->velocity;
        }
        else
        {
            // Finish falling.
            Debug("Finished Falling!");
            obj->z = (fTerrain - fCentral);
            obj->bOnGround = true;
            obj->velocity = 0;
        }
    }
}
else
{
    // Object is definitely resting on the ground.
    obj->bOnGround = true;
    return;
}
}
```

Honors Project Index

Page numbers in **bold** reference a definition; page numbers in *italics* reference code; and *RFP* indicates that a document or feature was Removed From Project and does not appear in the final project.

- acceptance tests, 112-122
- agent, **22**
- bibliography, *See references*
- calendar, 35-56
- character, **22**, 26, 45, 112, 121
- code, *137-324*
 - E.D.G.E. Tool, *159-178*
 - I.T. Simulator, *209-324*
 - Material Editor, *137-158*
 - W.I.M. Tool, *179-208*
- core sentence, **25**
- data dictionary, 63-65
- definitions, **22**
- DeleD, 27
- development tools, 29
 - E.D.G.E. Tool, 9, 11-13, 20, 29, 33, 36, 41-42, 49, 64, 71-74, 112, 115-116, *159-178*
 - Material Editor, 20, 29, 33, 36, 41-42, 48, 66-70, 112-115, *137-158*
 - W.I.M. Tool, 8-9, 11-13, 20, 29, 33, 36, 41, 43, 50-51, 75-82, 112, 117-119, *179-208*
- documentation, 3-136
 - DDD (Detailed Design Document), 37-136
 - Design & Integration Document, *See DDD*
 - Proposal, 3-14
 - Requirements & Specification Document, *See SPMP*
 - SPMP (Software Project Management Plan), 15-36
 - User Manuals, *RFP*
- entity, 9-12, 22, 42-45
- Entity Instance, 8-12, *See entity*
- Entity Library, 9-10, 12, 42-43, 49, 64, 115-117, 133
- Entity Manager,
 - See development tools, E.D.G.E. Tool*
- Entity System, 8, **10**, 12, *Also See files*
- Entity Template, 8-11, *RFP*
- environment, 5, 7-8, 11-12, 19, 22, 25-26, 43-46
- Environment Manager,
 - See development tools, W.I.M. Tool*
- Environment Test Data Set, *See acceptance tests*
- files, **63-65**
 - .mlb (Material Library), 42, 48, 63, 112-116, 134
 - .elb (Entity Library), 9-10, 12, 42-43, 49, 64, 115-117, 133
 - .wid (World Instance Data), 9-10, 12, 50, 59, 65, 117-122, 132
- first person view, **22**
- goals, **20**, 25-26
- hardware, 27
- material, 19-20, **22**, 25, 42, 63
 - attributes, 5-6, 10-11, 19, 42, 48, 63, 66-67, 112-115
 - properties, *See material, attributes*
- Material Library, 42, 48, 63, 112-116, 134
- milestones, **23-24**, 27-34
 - Content Complete, 32
 - Debugging, 32-33
 - Detailed Design Documentation, 28-29
 - First Production, 29-30
 - Fourth Production, 31-32
 - Honors Project Defense, 33-34
 - Literature Review, 28
 - Second Production, 30-31
 - Specification Documentation, 27
 - Third Production, 31
 - Tool Creation, 29
 - User Manuals, *RFP*, 33
- mobile, **9**
- NPC (Non-Player Character), **22**
- object, *See entity*
- PC (Player Character), **22**
- physics, 7, 20, **25-26**, 31, 62, 111, 322-324
 - movement, 8, 20, 31, 120
 - pushing, 20, 26, *RFP*
 - jumping, 20, 26, 62, 111, 121
- references, 21
- rule sets, 8, 11, 13, *RFP*
- screens, 12, 30, 41, 44-46, 120
- screenshots, 139-140, 161-162, 181-183, 211-224
- simulator, **1-324**
- software, 27
- test driver, 123-131
- tests, *See acceptance tests*
- timeline, *See calendar*
- Unit Demo, *See simulator*
- User Interaction Module, *See simulator*
- user manuals, 5, 13, 19-20, 23, 33, 36
- Visual Basic, *See development tools*
- world, *See environment*
- World Instance Data, 9-10, 12, 50, 59, 65, 117-122, 132
- XML, 8-10, 13, 30, 42-44, 48-50, 63-67, 71-72, 75-76, 115-116
 - Files, *See files*
 - MSXML, 27, 123, 125, -128, 141, 163 184, 250, 302-304