12-31-2008

Exploration of multiple pathways for the development of immersive virtual reality environments

Kevin Laurence Garrison
Rowan University

Exploration of Multiple Pathways for the Development of Immersive Virtual Reality Environments

by

Kevin Laurence Garrison

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Electrical and Computer Engineering

Major: Engineering (Electrical Engineering)

Approved:

Members of the Committee:

_____
In Charge of Major Work

_____
For the Major Department

_____
For the College

Rowan University
Glassboro, New Jersey
2008

ABSTRACT

Kevin L Garrison
EXPLORATION OF MULTIPLE PATHWAYS FOR THE DEVELOPMENT OF IMMERSIVE VIRTUAL REALITY ENVIRONMENTS
2008
Dr. Shreekanth Mandayam
Master of Science in Engineering (Electrical Engineering)

The focus of this thesis is the study and recommendation of optimal techniques for developing immersive virtual environments for generic applications. The overarching objective is to ensure that virtual environments can be created and deployed, rapidly and accurately, using commercial off-the-shelf software. Specific subjective and objective criteria have been employed to determine trade-offs between multiple pathways for designing such environments and specific recommendations are made for the applicability of each. The efficacy of the techniques developed as part of this research work has been demonstrated by applying them to three widely differing areas – visualizing arbitrary 2D surface data, synthesis of particle aggregate models from computed tomography and simulation of NASA rocket engine test stands.

The objectives of this thesis were obtained by an examination of the current algorithms and software in use for the development of virtual environments. From these currently used methods, general methods were defined. The expansion of these general methods to include the inputs and situations of common applications, allowed for the development of methods for real-world examples. Results were obtained by evaluating

these methods against defined measurement criteria. These criteria measured the effectiveness of these methods for increasing the value of virtual reality, while reducing the cost.

In this thesis, two virtual environment platforms (vGeo® and Vizard®) were used to develop three applications. These applications were a surface plot, particle visualizations and test stand simulations. In most cases, the results found the open-ended Vizard® to be the better platform. vGeo®, a platform designed for data visualization, worked well for basic data visualization, but was not as effective as Vizard® for developing more complex visualization. This thesis found that in most cases, an open-ended development platform, with functionality for rapid development is ideal. These methods and evaluations can be applied to a more diverse set of application and datasets to build development platforms that are even more efficient.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

Virtual reality (VR) is a rapidly growing area in the field of scientific visualization. VR enables the use of 3D computer graphics to display datasets in an immersive, interactive and navigable environment [1]. Immersion, interaction and navigation allow the user to intuitively view the datasets, which can lead to better interpretation of the data. Furthermore, the overlaying of datasets in a 3D environment allows the user to do the work of a data fusion device, possibly exposing correlations in the data that would otherwise go unnoticed [2].

Like other scientific visualization methods, VR enables the extraction of information from raw data [3]. Data without a reference tends to be of little use. When the data is put into context, and useful meaning is derived, the data becomes information. This is done by processing, organizing, structuring and presenting data. Once information has been extracted from the data, it can be used to take further action.

It is difficult to determine beforehand if a particular VR environment will produce useful information [4]. The cost of developing a virtual environment may outweigh the advantages. Since it is not possible to predetermine exactly how much information the user will be able to visually extract from the virtual environment, reducing the cost of development could increase the overall usefulness of the environment. Therefore, it is necessary to streamline the virtual environment development process to ensure that the cost of a virtual environment does not outweigh its benefits.

1.1 Motivation

The goal of virtual reality is to allow a user to extract information from a dataset in an immersive, interactive and navigable 3D environment. Since the outcome of this extraction is based on what the user can visually extract from the data, predetermining the quality and quantity of the information extracted is difficult. If the cost of developing a virtual environment is great, while the information extracted is minimal or could have been extracted by simpler methods, then virtual reality is of little use. Therefore, it is important to simplify the development process, so that even if the information extracted is minimal, the overall gain of the environment is greater.

Another goal of virtual reality as a form of scientific visualization is to display data as realistically and accurately as possible. A virtual world that is perceptually realistic becomes more immersive for the user. Similarly, a virtual world that reacts realistically also becomes more interactive. Augmented user inputs can increase interaction within the world, as well as making the world more navigable. Ultimately, a realistic and accurate virtual environment has a greater possibility of increasing the chances of discovering valuable information within the raw datasets.

Therefore, the goal of developing a virtual environment is to augment the realism and accuracy of the environment, while streamlining the development process. Increases in computing power allow these environments to become increasing complex [5]. However, this leads to more complex development requirements. This calls for well-defined processes for creating these environments. The goal now becomes to explore multiple pathways of both software and algorithms, for the design, development and deployment of immersive VR environments.

1.2 Objectives of this Thesis

The goal of this thesis is to perform a comparative study of different software and algorithms for use in the design, development and deployment of virtual environments. Limited work has been done in the area of defining efficient development paths for the creation of virtual environments. With increased computing power leading to virtual worlds that are more complex, the need for such a study becomes increasingly important. The objectives of the thesis are:

1. Examination of the current algorithms and software used to design, development and deployment of virtual environments.
2. Definition of a general method for developing virtual environments.
3. Expansion of these general methods to methods applicable to common inputs and situations.
4. Application of these methods to actual examples, including different types of datasets and applications.
5. Application of measurement criteria to evaluate the effectiveness of these methods in increasing the value of VR while reducing development cost.

The expected contribution of this thesis will be a measurable comparison of methods used to create these virtual environments. This will help in determining which algorithms and software are ideal for which datasets and applications. A variety of measurement criteria will be considered in determining the quality of methods used. These include ease of design, accuracy of data, process efficiency, integration with existing software and input/output specifications.

This comparative study will lead to information that will allow for efficient development, design and deployment of more advanced virtual environments. Keeping

the cost of development low while increasing the quality of the virtual environments will help to ensure the benefits of VR outweigh the costs. This will increase the overall value of VR as a form of scientific visualization.

1.3 Scope and Organization of this Thesis

This thesis evaluates algorithms and software for the development, design and deployment of virtual environments. Within the scope of this thesis, two actual applications are used as examples to evaluate the algorithms and software available for visualizing data.

1.3.1 Visualization of Synthesized 3D Particle Aggregates

When considering the properties of a mixture of sand particle aggregates, some are more easily measured than others are. Of the more easily measured properties, density, particle size and size distribution, hardness and specific gravity are the most common. However, an important property of particle aggregates, shape, is not easily described. It is especially not easy to describe numerically, making the development of discrete element models based on shape difficult.

Previous work has used Fourier-based 3D shape descriptors and the Algebraic Reconstruction Technique algorithm to construct synthesized 3D shapes from a series of 2D projections [6]. This method gives quantitative values to evaluate the shape of the particles in the mixture. However, this method does not retain all information concerning the shape, and question of how much retained information is necessary for a discrete element models arises. To evaluate this question, the synthesized particle is compared to a "gold standard," in this case, an X-ray microtomograph.

To compare the synthesized particle, mathematical method could be used to align and evaluate the difference between the particles. However, this may not give the user a sense of the quality of the synthesized particle compared to the "gold standard." Virtual Reality can be used to display the particles side by side, with the ability to move and manipulate the particles, until they are visually aligned side by side. This gives the user a true sense of the particles, and allows for the selection of better initial conditions, ensuring the alignment algorithm comes to the best possible result.

1.3.2 Simulation of NASA Rocket Test Stands

Tests are often performed on equipment used to represent hardware used in an actual system. This allows a designer to test the design before it is implemented on the actual system or to test just a subsystem of the actual system. If the test fails, then it only fails on the test equipment, and no damage occurs on the actual hardware. Test equipment can also be fitted with more sensors than the actual hardware, giving the designer more feedback. This gives the designer a safer and more efficient method of testing a design.

Within the scope of this thesis, the test equipment is a set two NASA rocket test stands at the Stennis Space Center. These test stands record a variety of information, including valve position, flow, pressure, temperature and electrical current during the testing of a rocket thruster. From these sensor readings, the test can be evaluated, and adjustments made to the thruster.

These sensor readings can be displayed in Virtual Reality to "replay" the hardware test [7]. This allows the user to watch a test multiple times and possibly see something that was missed during the actual test. Furthermore, logic can be added to the

virtual environment or imported from an existing model. This allows the user to run a virtual test within the virtual environment, removing the need for expensive and time-consuming physical tests.

This thesis is organized as follows. Chapter 1 describes the problems associated with developing, designing and deploying virtual environments as increasing computing power leads to more complex work and the efficiency issues inherent to such processes. Chapter 2 discusses Virtual Reality as a form of scientific visualization, and the general step required for integrating raw data into a new virtual environment. Chapter 3 describes the algorithms used when integrating particular types and formats of data into a virtual environment, as well as how those algorithms change depending on the virtual environment platform software used. Comparison criteria used for the analysis of these methods and software are also discussed. Chapter 4 contains the results of using these algorithms and software for real applications, mainly visualization of 3D synthesized particle aggregates and simulation of a NASA rocket test stand. The comparison criteria described in Chapter 4 are used to evaluate these applications. Chapter 5 contains a summary of accomplishments and recommendations for future work and is the conclusion of this thesis.

1.4 Expected Contributions

Increasing computing power has allowed for virtual environments that are more complex. These environments have the potential of allowing the user to extract greater amounts of information from the raw data. However, increasing complexity leads to greater development cost, potentially outweighing the benefits of virtual reality. The exploration in this thesis will lead to more efficient development, design and deployment of virtual environments. This will allow future virtual environments to be more realistically immersive and navigable, and interact more intuitively. Ultimately, this will increase the overall value of virtual reality as a form of scientific visualization.

Chapter 2 - Background

2.1 Previous Work

Much of the previous work in the area of virtual reality is in its application, and not its development. Table 2-1 shows a collection of previous work on the development and evaluation of virtual environments. Most work speaks of the effectiveness of expandable open-ended platforms. However, Ziegeler, et al. speak of the value of single purpose software with their MetVR application. Gabbard, et al. takes a user-first approach in virtual environment design.

Table 2-1: Previous work on the development and evaluation of virtual reality.

Research Group	Investigators	Research Focus
Rowan University	S. Papson, J. Bram, S. Mandayam	Developed a framework for the integration of Non-Destructive Evaluation (NDE) data into visual, interactive, and immersive displays, and analyses the risks involved [8][9].
Virginia Tech/ Georgia Institute of Technology	D. Bowman, L. Hodges,	Evaluated the development of immersive virtual reality, including areas of interaction and menu systems [10][11][12].
Iowa State University	C. Just, A. Bierbaum, A. Baker, C. Cruz-Neira	Developed an application, VR Juggler, for the development of virtual reality, emphasizing on simplifying application development, rapid development, a unified interface, extensibility and portability [13].
Mississippi State University / Jackson State University	S. Ziegeler, R.J.Moorhead, P. J. Croft, D. Lu	Compared the their own immersive meteorological visualization software, MetVR, against more generic visualization software, including Cave5D and vGeo® [14].
Virginia Polytechnic Institute	J .L.Gabbard. D. Hix J.E. Swan	Developed a structured method for designing and evaluating virtual environments with user-interaction at the forefront [15].

In his thesis [8], Papson develops a framework for the integration of large, complex, multi-dimensional datasets into virtual reality. He emphasizes on algorithms with time-evolutionary capabilities used for the enhancement of virtual environments. Of these algorithms, hidden Markov models gave the best results for the Non-Destructive Evaluation (NDE) data used. Bram utilized virtual reality in his thesis [9] to visualize the outcome of his "Divide-and-Conquer" strategy for the inversion of NDE signals in gas transmission pipelines. The visualization allowed for the side-by-side comparison of the graphical, measurement and functional data of the results,

Bowman has done much work in the area user interaction and menu systems in virtual environments. Guidelines for general virtual environment interaction provided for an empirical basis for choosing interaction techniques [10][12]. An empirical comparison on input methods and menu systems found that user comfort was just as important, if not more than the ease and speed of an input method [11]. The results of these studies showed measurable gains in usability in real applications. Gabbard, Hix and Swan find that visual quality of virtual reality is far outpacing the development of interaction techniques [15]. Virtual environment user interfaces are poorly designed and rarely tested on actual users. As a result, they have suggested a methodology for the user-centric development of virtual environment interfaces.

Some previous work has brought forth virtual environment development platforms. Iowa State's VR Juggler application [13] is designed around a need for a development platform that meets a number of specifications, including portability, an easy-to-learn interface and scalability. At joint group at Mississippi State University and Jackson State University developed a virtual environment development platform from a

different set of needs. Finding that the transition to visualizing meteorological data in 3D lost some of the ease of analysis available in 2D, MetVR was developed [14]. Contrary to the open design ideas of VR Juggler, the group that developed MetVR found a single purpose development platform to be more efficient and accurate.

2.2 Virtual Reality

2.2.1 Data Visualization

Data visualization can be used to extract information from even the simplest of datasets [16]. One of the simplest visualizations is an X-Y scatter plot. The visualization starts with two corresponding sets arrays of data, such as the one-dimensional position of an object and the time at with the object was at that position. Each point is plotted on a grid, as seen in Figure 2-1. From this plot, a trend in the data can be seen that may not have been noticed in just the list of numbers shown in Table 2-2.

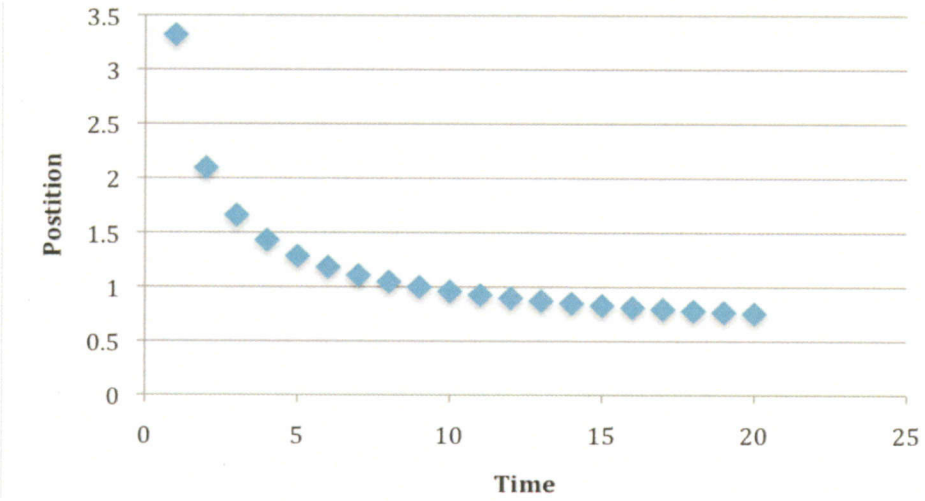
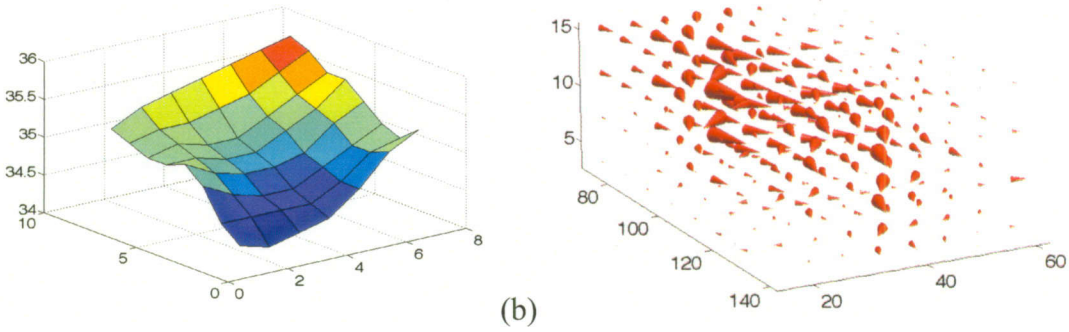


Figure 2-1: Example of data visualized as an X-Y scatter plot.

Table 2-2: Example of data for the X-Y scatter plot.

Time	1	2	3	4	5	6	7	8	9	10
Position	3.3	2.1	1.7	1.4	1.3	1.2	1.1	1	1	1
Time	11	12	13	14	15	16	17	18	19	20
Position	0.9	0.9	0.9	0.9	0.8	0.8	0.8	0.8	0.8	0.8

As datasets become more complex, more advanced visualizations must be used. Advanced scientific visualization (ASV) is used to handle such datasets [5]. While the X-Y scatter may be sufficient for a 2D data, ASV can be used to display visualization of increasing complexity and dimensionality. An X-Y scatter plot gives way to a surface plot or a vector field, shown in Figure 2-2. These visualizations allow the user to better understand the data.



(a) (b)

Figure 2-2: Example of advanced scientific visualization. (a) Surface plot. (b) Vector field.

2.2.2 Virtual Reality as Advanced Scientific Visualization

Virtual reality is the use of hardware, software, and interaction devices to create the psychophysical experience of being surrounded by a computer-generated environment [5]. This experience gives the user the sense that the environment is real. The key to virtual reality is divided into three parts: Immersion, Interaction and Navigability [1] [8], seen in Figure 2-3. These components make the VR environment an intuitive environment for visualizing data, making it ideal for use in ASV.

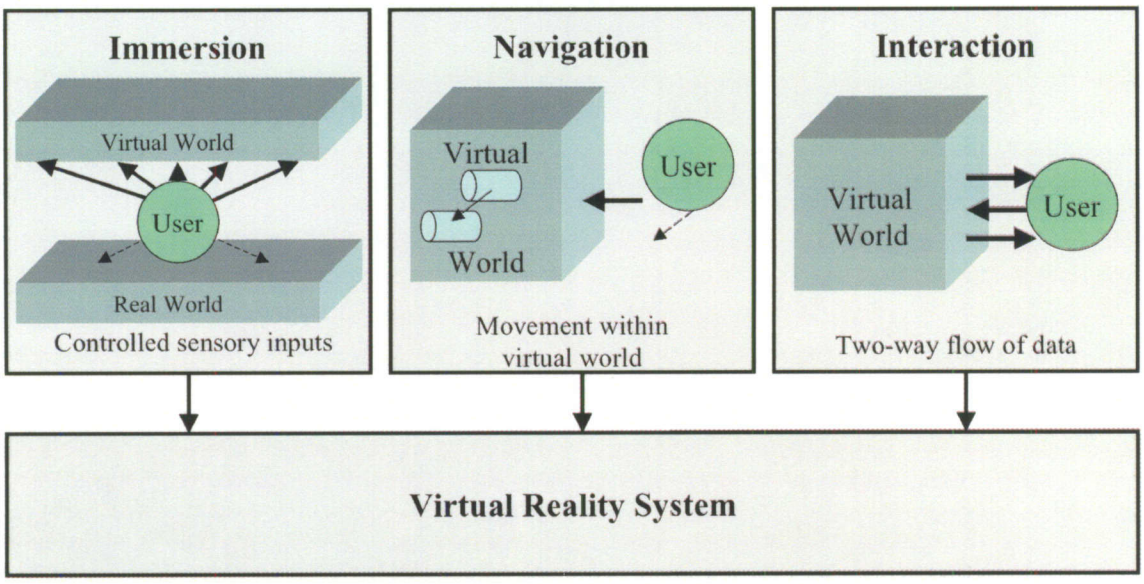


Figure 2-3: Components of virtual reality [8].

Immersion is the sense of presence that makes the user feel that they are actually in the environment [5]. Immersion gives the user of the virtual environment a better sense of awareness, context and spatial judgment. Immersion is accomplished by blocking out real world sensory inputs, while enhancing the virtual sensory inputs [8]. This blocking can be reached with a combination of software and hardware, including stereoscopic displays, head and motion tracking, spatial sound and haptics [17].

Interaction is the tracking of users actions to trigger events in the virtual environment [18]. It can also be described as the filter between user input, data and display [19]. Interaction can be accomplished though a combination of input devices and software triggers. Commonly, interaction is performed with a motion device, such as a computer mouse or 3D tracking source, and a button to trigger an action. These input functions are simple motions, meant to trigger much more complex responses in the environment [20]. Increasing the quality and resolution of the interaction increases the overall immersive effects of the environment [4].

Navigation is the most common form of interaction in virtual environments [21] and can be divided into two components [22]. The cognitive component of navigation, *wayfinding*, gives spatial awareness, allowing users to use virtual reality to view datasets that could not be viewed in standard 2D visualizations. With that awareness, the user can take advantage of the motor component of navigation, called *travel*. This is what allows the user to move around the environment as if it were real life. Navigation is performed though the means of a user-input device, such as motion sensing device, computer mouse or floor sensors.

2.2.3 Hardware and Software

Hardware is the core of virtual reality. The hardware, with software, creates a virtual environment with interactive elements [23]. Generally, this hardware includes displays, sensors, and user-tracking and navigation devices. Most important to the virtual reality system is the display. Displays can be non-immersive, semi-immersive, like the one shown in Figure 2-4, or fully immersive, like a multiwall, user-surrounding CAVE or head-mount displays (HMD) [8]. Larger displays and displays that surround the user create a better sense of immersion.

Depth information can be provided with stereoscopic displays. Two eyes give humans a sense of depth perception. Providing this information is very important to the users sense of immersion on the virtual world. Stereoscopic vision can be simulated by generating two visualizations from separate angles, corresponding to the angles at which the eyes would in the real world. Getting the separate images to each is a more challenging task. HMDs already have two displays, one for each eye. For large displays,

a variety of glass can be used. Projected two overlapping images, with different polarization can be viewed stereoscopically with glasses containing corresponding polarized lenses. A time-sharing method can be used in which the display alternates between frames of each angle. Without glasses, the screen appears to flicker, but with LCD glasses, like the ones seen in Figure 2-5, each angle can be seen separate. The LCDs in each lens alternate, blocking the image from the eye that should not be viewing the angle. An emitter in the display keeps the visualization and the glasses synchronized.

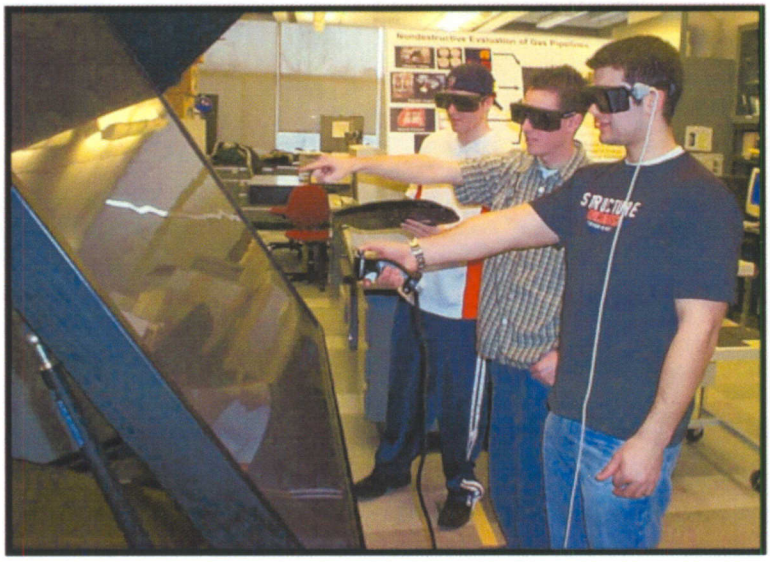


Figure 2-4: A semi-immersive virtual reality system [8].

Figure 2-5: LCD shuttering glasses used for stereoscopic viewing.

User input is also very important to a virtual environment, allowing the user to navigate and interact with the world. At the forefront of user input is motion tracking. To give the user a better sense of immersion, motion sensing mimics the users real world motion in the virtual world. Motion tracking can be performed by a variety of systems, including optical and radio signal. Fully immersive motion tracking gives a full range of motion known as six-degrees-of freedom, which includes both translation and rotation in all three dimensions. Additional input devices can be used to enhance the experience. These include traditional computer inputs, like a keyboard or mouse, motion tracking inputs, or gaming inputs, such as a joystick or floor pad.

At the center of the virtual reality hardware is a computer workstation. This workstation works as the central hub of the systems. The computer accepts user input, calculates changes based on those inputs and updates the visualization accordingly. Rendering out the environment is the most intensive tasks performed by the machine, traditionally requiring expensive hardware. However, recently, computing power has become less expensive, as has virtual reality accessory hardware [23]. This has led to more complex virtual reality systems and environment becoming more accessible. Within the scope of this thesis, two software packages are run on the computer workstation: vGeo® (www.mechdyne.com) and Vizard® (www.worldviz.com/vizard.)

2.2.3.1 vGeo®

Virtual Global Explorer and Observatory (vGeo®) was developed by VRCO. vGeo® is a data visualization platform meant for time varying 3D environments and large, multivariate datasets [24]. It can accept data in its raw numerical form or as imported models. Raw data can be scalar or vector, and be one, two or three-dimensional. Models are imported utilizing the Virtual Reality Modeling Language (VRML) standard. Data and models may also include time steps and animations. Visualizations, shown in Figure 2-2, include surface plots, vector fields, isosurfaces, numerical and text displays. Although vGeo® excels at visualizing raw data, it is not set up for custom interaction and functionality. Interaction is limited to navigation and turning dataset on and off. vGeo® is best suited for preset animations and static models and data.

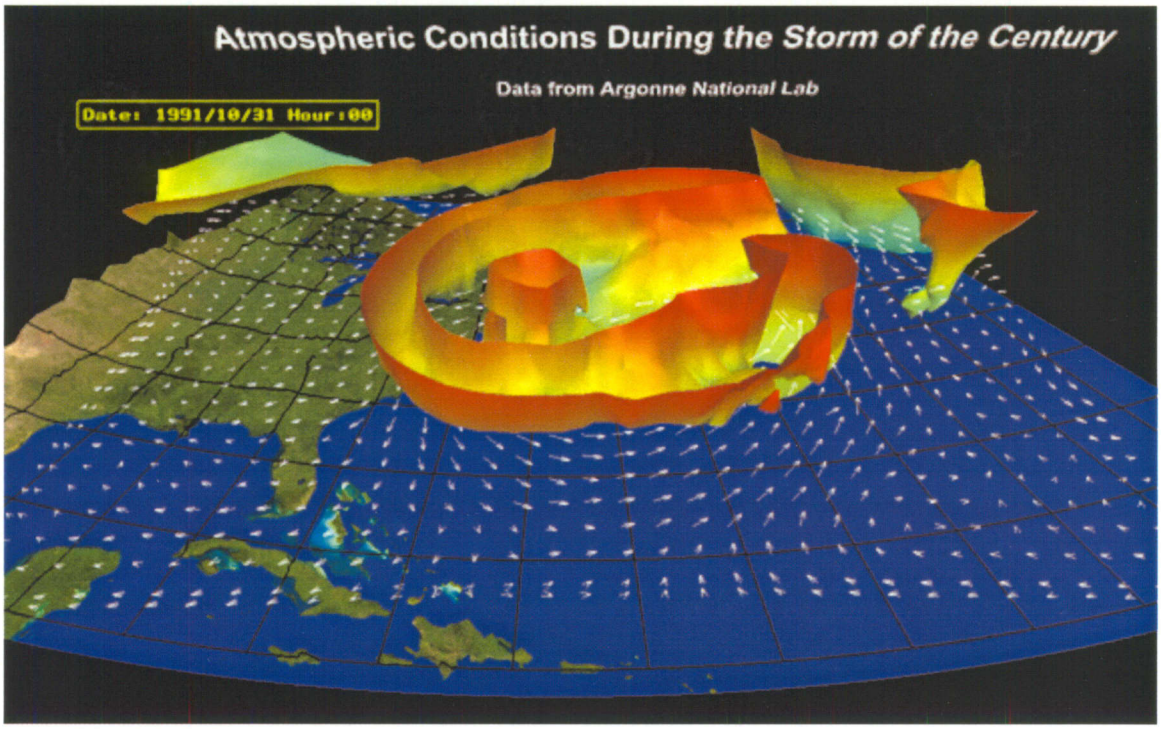


Figure 2-6: An example vGeo® environment entitled "Storm World"

2.2.3.2 Vizard®

Vizard® is a general visualization platform designed by WorldViz. Using a combination of low-level OpenGL functions and the Python scripting language, Vizard® becomes a very open-ended visualization platform. Python is a high-level scripting language with the ability to access OpenGL, a library of code designed to communicate directly with the computer's graphic card. This combination makes Vizard® an easy to use tool for programming complex visualization. Vizard® has the advantage of being able to load and scale a model with a few lines of code, while still having the ability to program complex interactions between user input, processed data and 3D visuals. A downfall of open-ended nature of Vizard® is its lack of built-in visualization functions. Unlike vGeo®, Vizard® requires raw data visualization functionality to be scripted. However, this allows the Vizard® to be programmed to display much more complex and interactive visualizations.

2.2.4 3D Modeling Basics

3D modeling and computer graphics are an important part of virtual reality. Computer graphics create surfaces in a virtual space with the appearance of real objects. These surfaces consist of many mathematically described polygons [25]. In computer graphics terms, a polygon is a plane specified by a set of three coordinate positions in 3D space. These coordinates, called vertices, are shared with adjacent polygons, which together form a surface. While each polygon in itself is a flat plane, a combination of many small polygons can form a rounded surface.

Just as in real life, models in computer graphics need to be illuminated. Virtual lights sources can be ambient or directional, and how they illuminate the surface is determined mathematically. Properties of the polygons govern how the light affects the surface, giving a better sense of the shape of the object. One of the most important properties is the face normals. Among these details, face normals tell which side of the polygon is the outward facing side. The outward facing side, or face, is visible and lit. The side of the polygon opposite the face is not visible. Making sure the face normals face outward is important for accurately displaying a model.

Also important to computer graphics is performance. Every polygon in a surface, regardless of size, requires the same amount of processing. As a surface becomes more detailed, it contains an increasing number of smaller polygons. The number of polygons is known as polygon-count. As the polygon count increases, so does the time required to render out the image. This time can become an issue when considering the rendering of sequential frames of a constantly changing surface or varying point of view as part of a virtual environment. As the time to render increases, less frames can be rendered each second. The number of frames rendered in each second is knows as frame-rate. If the frame rate drops too low, the visualization will become choppy, causing loss of immersion and difficulty with interaction.

2.3 Data Integration in Virtual Reality

All data follows a similar integration path into the VR Environment. The steps in this path ensure that the data can be displayed in the environment, both accurately and efficiently. While the data type and visualization platform may change, all data follows the same path of conversion, adjustment, importation into the visualization platform, and finally an evaluation, which could lead back to a re-conversion or a readjustment.

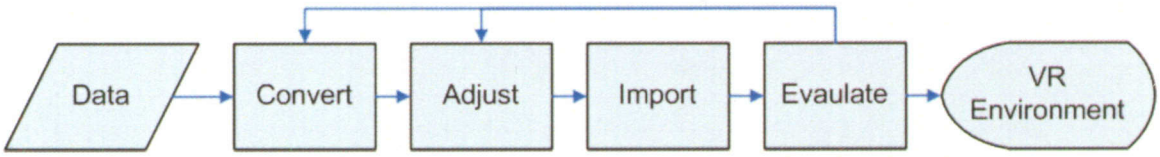


Figure 2-7: Integration of data into virtual reality.

2.3.1 Conversion

Initially, data is usually not in a format that can be read into the visualization platform. The data could be in the form of a computer file that simply needs to be converted to another format or a blueprint that needs to be digitized. A variety of software is used to handle the variety of data types. Tables of numeric data tend to be the easiest and at most require a simple file conversion (e.g., Excel to CSV.) However, much of the data integrated into a VR environment are three-dimensional objects. Because each 3D object format behaves differently in different 3D modeling software and visualization platform, converting 3D data is often not as simple as a file conversion.

There is a variety of 3D object formats. The most basic contain only data pertaining to the vertices and faces of the object. Formats that are more complex contain extra information, such as lighting, textures, animation or extra detail. There is no universal format, although some are more common than others are. The only way to determine which format is the best for each situation is trial and error. Still, the best format can cause problems with accuracy and efficiency of the object in the VR environment. In this case, adjustment must be made to the data.

2.3.2 Adjustment

The adjustment of converted data pertains mostly to 3D object data. Even with the best option for conversion, the 3D object may not conform to the requirements for visualization in the VR environment. The most common issues are too little/much data, poorly stored/interpreted data and redundant data. The most common adjustment comes when the normals of the faces are inverted, giving the object an inside-out appearance. Another common issue is when a model comes from a source that was not intended for 3D visualization, such as a 2D blueprint converted to a 3D model or data gathered from a scanning device. The result is a 3D model that either is distorted visually or is designed in such a way that makes inefficient use of computing power. The goal of the adjustment phase is to make the 3D object as visually appealing and computationally efficient as possible while not altering the data stored within the object.

2.3.3 Importation

The converted and adjusted data must be imported into the visualization platform in order for it to be viewed. The procedure varies depending on the type of data, how it is being used and the visualization platform to which it is being imported. Numeric data can be tied to a plotting function or to 3D objects in a more complex visualization. Static 3D objects tend to follow a simple import procedure, while objects with dynamic functions require more effort. Some visualization platforms are better in terms of how they can display data, while others are limited in what they can visualize.

2.3.4 Evaluation

The only way to know if the data will be displayed accurately and efficiently is to import it into the visualization environment. Once in the environment the visualization can be evaluated. If the data or object is not displayable, the data may need to be reconverted to a more compatible format. If the data is inaccurate or the visualization is running poorly, chances are the data needs to be adjusted. The final phase of the data integration process is a fine-tuning of conversion and adjustment until the data is visualized in the best possible manner.

Chapter 3 - Approach

Virtual reality allows for inspection and analysis of multiple complex datasets simultaneously. The purpose of combining these datasets is to extract new data and to gather information that may have been overlooked by analysis of the individual datasets. A virtual environment is a possible platform in which these overlaying datasets can be combined. The advantage of virtual reality is an environment that is true to reality, in which the user can view, navigate and interact with the data in an intuitive manner. The goal of this thesis is to investigate methods and platforms to integrate many types of data, from many sources, minimizing data loss and corruption, while increasing interaction, immersion and navigability.

For the purpose of this thesis, virtual reality data is separated into three data types: measurement, graphical and functional [8]. Measurement data refers to digitized numerical data originating from raw data gathered by inspection or sensors. Graphical data refers to data that represents the physical worlds, such as landscape, structures and other physical objects. It is used as a setting for the virtual environment or as a reference between the virtual and physical worlds. Functional data is the result of the processing and analysis of measurement data. This may include the decisions made by a neural network. Compared side-by-side with the original measurement data, new information could be revealed to the user. The integration of these three types of data into one virtual environment is one of the main goals of virtual reality

However, while this categorization of data is ideal for the analysis of data, it is less important to the process of integrating the data into the virtual environment. When data is recorded (measurement), calculated (functional) and the physical world is digitized (graphical) it is stored in a data format. When performing the process of integrating the data into virtual reality, the method in which the data is stored (data format) is more important than the type of data. Therefore, methods must be implemented for integrating data based on format, rather than type.

3.1 Integration of Data Types into Virtual Reality

All data, measurement, graphical and functional, must be processed before it can be integrated into a virtual reality environment. How the processing occurs depends on the source of the data, the functionality required and the virtual environment platform to be used for creating the virtual environment. While there are differences in these processes, all data follows the basic path shown in Figure 3-1.

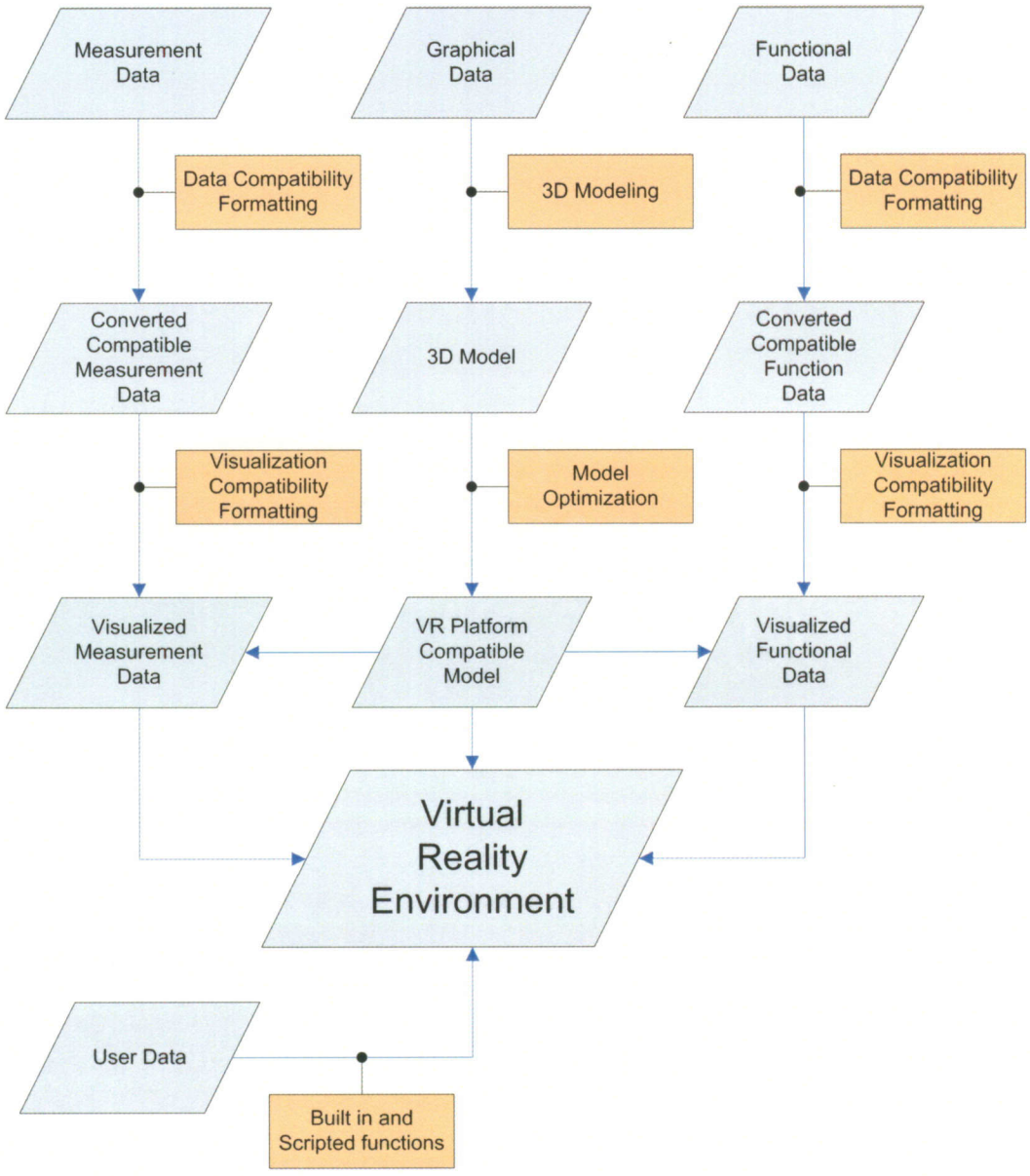


Figure 3-1: Overall integration of graphical, measurement and functional data.

3.1.1 Graphical Data

Graphical data is a visual representation of a real life object. The first step for integration is to convert the data to a digitized 3D model. In general, this can be done by modeling a real world source or by converting from another digitized source. Once this is complete, the data is now a 3D model. While this model is now digitized, it may not be compatible or optimized for the virtual environment platform.

To prepare the data for the virtual environment platform it must first be adjusted and optimized. This can include a variety of operations, such as fixing errors, removing redundancies and excess, scaling and positioning. When the 3D model has been adjusted, it is then imported into the virtual environment platform, using a platform specific method. If the model has errors or does not meet certain requirements, it is passed back to the model optimization step to be further adjusted. If the graphical model has any connection to other data, the data is registered, as discussed in the next section. Once this is complete, the data is imported into the virtual environment platform to be visualized.

3.1.2 Measurement and Functional Data

Measurement and functional data both follow a similar process. Incoming data must be organized into a standard format. This first step is to covert the data, whether measurement or functional, into a format recognized by the virtual environment platform. This conversion usually consists of properly separating and organizing the data. An example of this is separating the data in the form of a Comma-Separated Values (CSV) file and organizing the data into columns, with each column corresponding to each sensor.

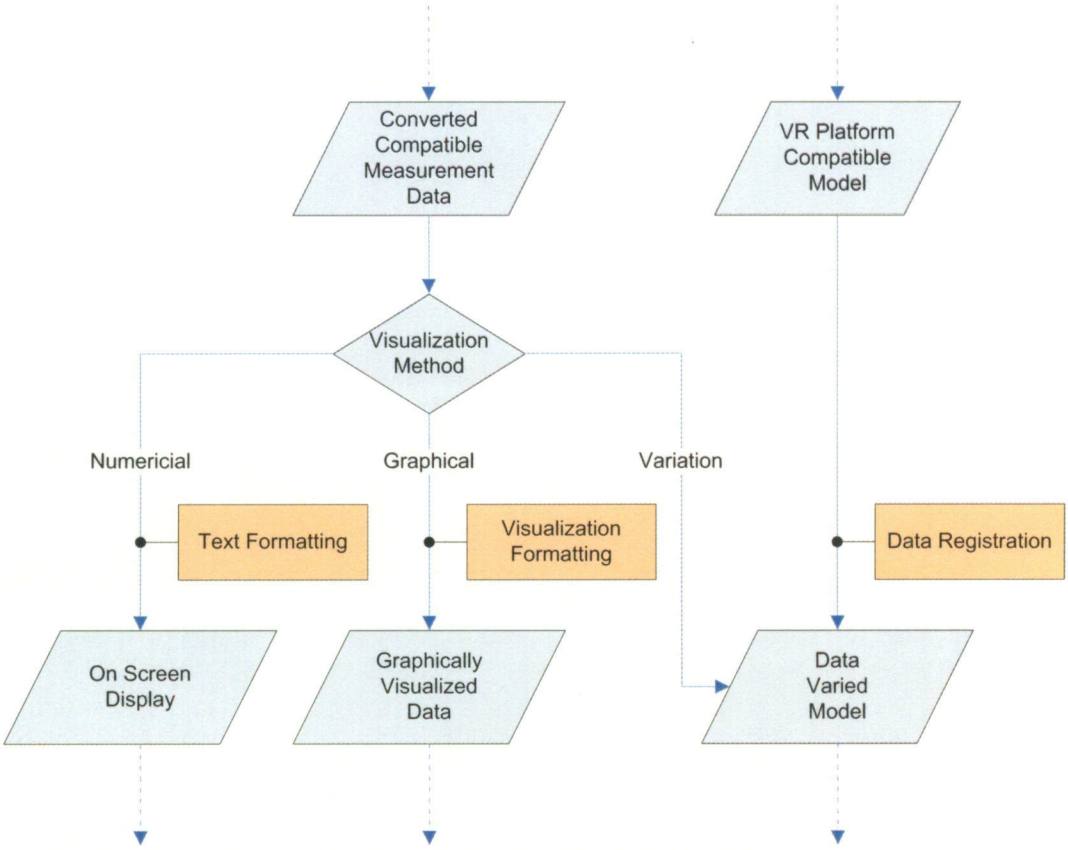


Figure 3-2: Visualization compatibility formatting.

Once the data is in the correct format, it is passed into the virtual environment platform for visualization. There are three general methods of visualizing the data: numerical, graphical and variation. Each visualization method follows a separate path, as shown in Figure 3-2. Numerical visualization consists of data simply being displayed on the screen as numbers or text, and only requires formatting and placement in the virtual environment. Graphical visualization displays the data visually, in a visualization method such as a surface plot or a vector field. The data must be formatted in the adjustment step to allow it to be passed into the graphical model. That data is then formatted so that it displays as the visualization of choice. The exact process in which the data is taken from a set of numbers to visualization is dependant on the virtual environment platform.

The final visualization method is the use of data to cause a variation in a graphical model. There are many examples of this, including controlling visual indicators, such as gauges, causing an object to vary in size or color, or triggering an action. Like the numerical and graphical data, the data is first formatted so that it matches the requirements for importing into the specific virtual environment platform. The formatted data is then registered to a graphical model in a platform specific method. This may require simply importing the data or the scripting of an action. Once the formatting is complete, the data registered model is imported into the virtual environment.

3.1.3 User Data

User data is the run-time data passed into the virtual environment by some type of input device. User data is key to the virtual environment, as it allows the environment to be navigable and interactive, ultimately making the environment more immersive. It can be motion-tracking data, input from buttons or floor sensors. Data from these sources is formatted and imported, either into built-in functions of the virtual environment or into the inputs of a user-generated script. From there, the action is handled by the virtual environment platform.

3.2 Data Formats

When data is obtained for use in virtual reality, it is organized into a specific format. This format depends on the source of the data and the method in which it was obtained, as well as other restrictions. The simplest restriction on file format is the dimensionality of the data. One-dimensional data can only be organized into certain formats, and likewise with two and three-dimensional data. Other restrictions may be influenced by the data itself or the users requirements for the virtual environment. The method in which the data will eventually be displayed also has influence on the choice of data format. The general formats of the data used in virtual environments and their corresponding data integration paths are as follows.

3.2.1 List Data

List data is simply a series of one dimensional data points. Commonly, list data contains a series of sensor readings taken over time, but could also be time stamps, locations or other data. Lists are often stored as a 2D table of multiple one-dimensional sources. This is done to consolidate similar data and is different from the next format, 2D data. An example of list data is shown in Table 3-1: Example of list data

Table 3-1: Example of list data.

Time	Temperature 1	Pressure 1	Temperature 2	Pressure 2
0	34.3	0.7	40.1	1.1
1	34.5	0.7	40.2	1.1
2	35.2	0.7	41.4	1.1
3	35.4	0.7	42.6	1.0
4	34.6	0.8	40.9	0.9
5	35.0	0.8	40.8	1.0

List data is either measurement or functional data and is treated as such. There are two steps for integrating list data in a virtual environment: loading the data into the virtual environment platform and visualizing the data. List data is first converted to the required format, as shown in Figure 3-3. There are two general formats used for 2D data. The first format is a spreadsheet format. This formatting is usually occurs as an Excel file (.xls) being converted to a comma-separated value file (.csv). The second type of formatting is a platform specific format. This usually consists of the adding of data points to the script or configuration file on which the actual environment runs.

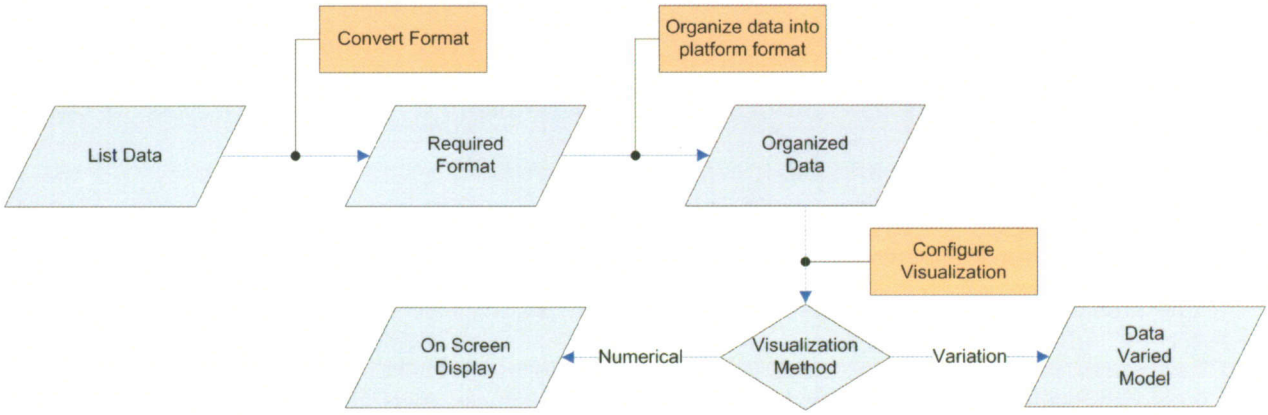


Figure 3-3: Integration of list data.

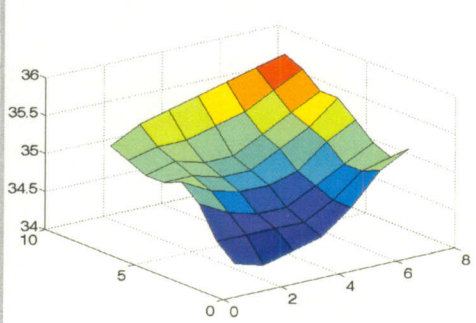
The data must also be organized into a column format recognized by the virtual environment platform, similar to what is shown in Table 3-1. The organization of the data is dependant on virtual environment platform used. Usually the data is organized in columns, each pertaining to a particular sensor. The virtual world is then programmed to expect data from a certain column. The virtual world either displays the data as a numerical display, or registers the data to an object, as the input of a function, as shown in Figure 3-3. This function is tied to and event, causing a variation in the virtual world.

3.2.2 2D Data

2D data is a set of data points over a two dimensional region. This is usually data from an array of sensors spread over a physical region. An example of this is the data obtained from temperature sensors spread over a geographic region. Another possible source is the results of a 2D scan, such as an ultrasonic scanner. 2D data can also change over time and be stored in a three dimensional structure of data. This is different from the 3D data format mentioned below.

Table 3-2: Example of 2D data and surface plot visualization.

	1	2	3	4	5	6	7
1	34.3	34.2	34.3	34.4	34.7	35.1	35.3
2	34.5	34.5	34.5	34.5	34.7	35.0	35.1
3	34.9	34.7	34.6	34.6	34.8	35.0	35.2
4	35.1	34.9	34.8	34.8	34.9	35.2	35.4
5	35.0	35.0	34.9	34.9	35.1	35.3	35.6
6	35.0	35.1	35.0	35.0	35.2	35.4	35.6
7	35.1	35.2	35.2	35.3	35.4	35.6	35.8
8	35.2	35.4	35.5	35.6	35.7	35.8	35.9



Like list data, 2D data is either measurement or functional data, and must be loaded into the virtual environment platform and then visualized. Typically, 2D data is displayed graphically. Otherwise, it follows a path similar to list data, starting with format conversion, as shown in Figure 3-4. The data is either converted from one standard to another, or inputted into a virtual environment platform specific file. Next, the data is organized before being configured for visualization. The possible forms of graphical visualization are limited by the platform. The data is passed into a built-in or scripted function, which converts and displays the data in a scientific visualization, such

as a surface plot (as in Table 3-2) or a vector field. These visualizations are combined with other to make a virtual environment.

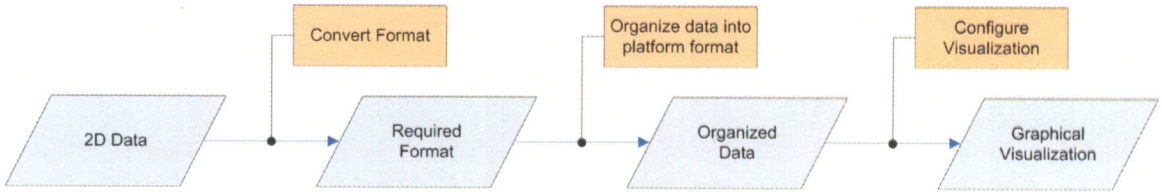


Figure 3-4: Integration of 2D data.

3.2.3 2D Image Data

2D image data is simply a digital photograph. This data can be used as a 2D model or as textures. Image data can make the virtual world more realistic, resulting in better immersion. As a model, it can be used for a direct comparison between the real image and the modeled environment or as a background image, as in Figure 3-5a. Like 2D data, a 2D image can also be the result of a sensor scan. If 2D image data varies over time, a series of 2D images can be integrated into a virtual environment as a movie.

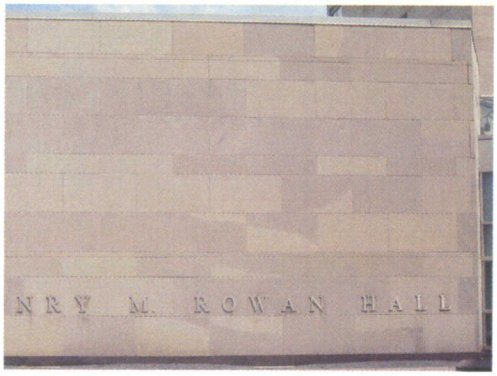
a)  b) 

Figure 3-5: Examples of 2D image data. a) background data, b) texture data.

2D image data follows one of two paths. If the data is used as a texture data, as in Figure 3.5b, it is applied as a texture in the 3D modeling process. Texture data follows the integration path of 3D model data shown in Figure 3-9. The data can also be visualized as a 2D model. When integrated into the virtual environment, it must be converted into a compatible format. The data is then visualized either as a model to be compared with other data or as a background to the virtual environment.

2D image data can also the reference for a 3D model. It requires artist and technical skill to take a real world reference and convert it into a digitized 3D model. Sometimes the modeling is as simple as making a CAD drawing from a blueprint. Sometimes it takes a more artistic approach, requiring the modeling of the world from reference images. Once the modeling is complete, it is treated as 3D model data and imported into the virtual environment platform in the same manner.

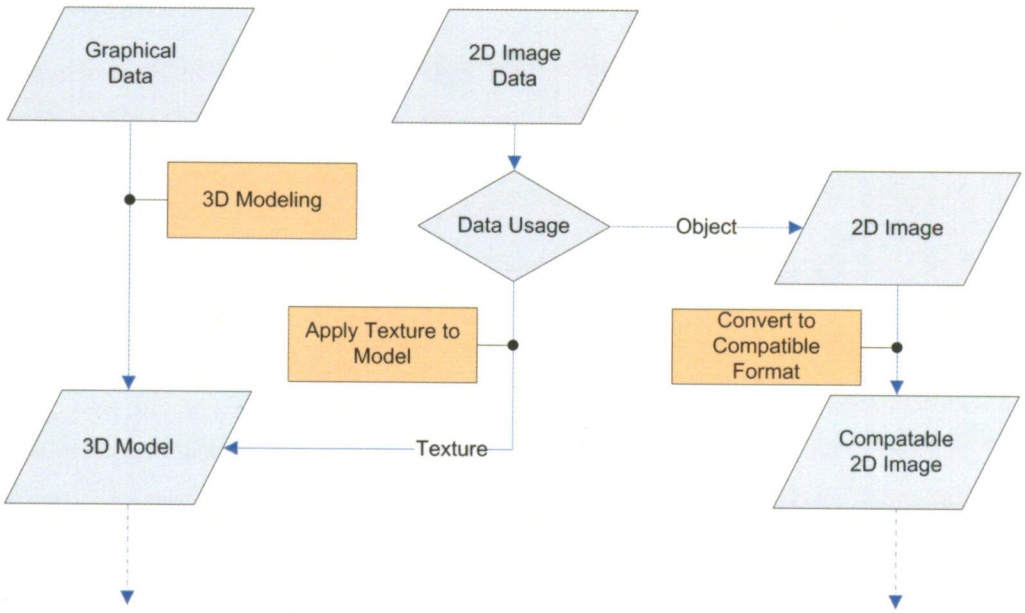


Figure 3-6: Integration of 2D image data.

3.2.4 3D Data

3D data is a set of data points arranged in three dimensions. It is similar to 2D data and usually is the form of measurement or functional data. An example of this is temperature readings over a geographic region at various altitudes. 3D data could also be set of vectors, displaying both wind speed and direction over the same range. Like other data sources, 3D can vary over time.

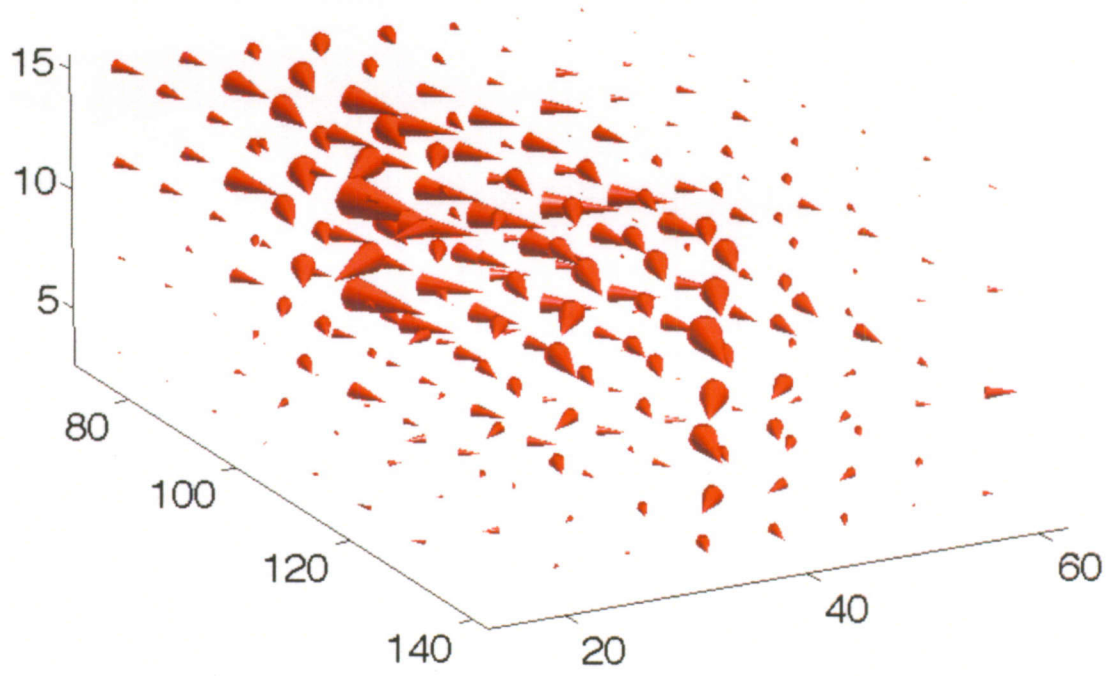


Figure 3-7: Example of 3D data, vector field.

3D data follows a path similar to 2D data, as shown in Figure 3-8. The data first converted to the required format, whether it is from one standard to another or to a platform specific file. 3D data can be visualized with a variety of methods, including vector fields, as shown in Figure 3-7, or an isosurface. The visualization methods available are limited by the virtual reality platform used. For the data to be visualized, it must first be configured. The configured data is passed into a built-in or scripted function

that converts and displays the data as visualizations. These visualizations are combined with others to make a virtual environment.

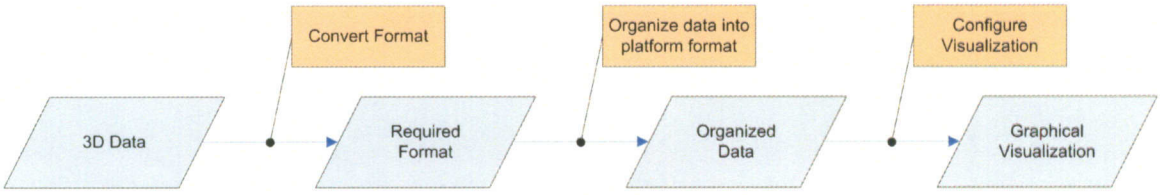


Figure 3-8: Integration of 3D data.

3.2.5 3D Model Data

3D model data is very important to the virtual environment. This data consists of not only the vertices and faces that make up the representation of the physical world in the virtual environment, but also contains, among other things, data for texturing, lighting, reflections and animation data. The representations are what make the virtual world appear more realistic, allowing the user to experience better immersion and interaction. 3D model data can also be used as a real world reference point to compare with the data.

3D Model data can be scan data as well. Data from a three dimensional scanner, such as a CT scanner can be processed and outputted as raw vertices and faces. This data can then be integrated into the virtual environment in the same way as the graphical data is integrated. The raw data is converted to a format compatible with the virtual environment. This process is done in a way to ensure that the actual data is not lost because of aesthetic improvements, such as smoothing and texturing. Therefore, the integrity of the data must be taken into consideration when integrating raw 3D data.

Integration of 3D model data is a file format compatibility problem. Unlike 2D and 3D data, 3D model data has already been visualized. The goal for integration of a 3D model is to transfer the visualization from one format into the virtual environment platform. However, when this data is transferred from one format to another, data is often lost or corrupted. The integration problem then becomes a matter of converting the 3D model to a file format recognized by the particular virtual environment platform, while avoiding data loss and corruption. Within the scope of this thesis, the following 3D model file formats are used.

VRML (.wrl) – The Virtual Reality Modeling Language (VRML) was originally designed for 3D visualization on the World Wide Web [26]. The format is a text file with support for basic 3D models, textures, lighting and animation. This format is outdated, as it was last updated in 1997. However, it is still a standard in research and education, because of its open specifications.

STL (.stl) – The STL format was originally designed for stereo-lithography rapid prototyping machines [27]. It is a very basic format, consisting only of vertex and face data, and lacking in common features such as color and texture data. Its simplicity makes it an ideal format for the output of 3D scanning hardware. However, its simplicity often leads to errors when converted to a more complicated format. The most common error is inverted normals, which make the object appear inside out.

Object file (.obj /.mtl) – The Object file was originally developed as the native format for Wavefront Technologies' *Advanced Visualizer*® [28]. It has since been adopted by many of the major 3D modeling platforms as a standard format. Its wide acceptance gives it an advantage, as it is easily imported, converted and exported. The .obj file on its own is similar to the STL format, containing only basic information about the 3D model. However, .obj files can also store object-smoothing data (vertex normals) and have the ability to group objects. Textures can be added to an Object file model, stored separately in an .mtl file.

OpenSceneGraph .osg/.ive - OpenSceneGraph is an open 3D graphics toolkit used for a variety purposes [29]. The toolkit has two native file formats: the binary .osg and the ASCII .ive. It has the advantage of being nearly as advanced as the propriety formats while being an open standard like VRML. This gives the format higher compatibility for transferring more complex 3D model data. However, it is still not a universal format.

3Ds Max® (.3ds) – 3Ds Max is an industry standard 3D modeling platform. The .3ds file format is the older of the proprietary file formats for 3Ds Max, succeeded by the .max file. While the .3ds does not include all the updates of the .max file, the .3ds file still contains more detail than most standard formats. Because it has been around for a while, the .3ds file is accepted by many platforms. However, because it is not an open standard, it is rarely fully compatible, leading to data and feature loss in the transfer process.

Maya® (.ma/.mb) – Maya is also an industry standard 3D modeling platform, often used for artistic applications [30]. The Maya proprietary file format comes in two types: the

.ma and .mb, which are ASCII and binary. The Maya file format stores large amount of data about the model, including a history of how it was created. However, these file formats are not standard, and therefore not compatible with most platforms. Any model designed in Maya must be converted to another format before integration into a virtual environment platform.

Filmbox (.fbx) - Filmbox was a 3D modeling platform developed by Kaydara. AutoDesk, the same company that now owns 3Ds Max and Maya, has since purchased it [31]. The .fbx file format, originally designed as the native Filmbox format is now used as a data exchange format for transferring complex 3D model data between platforms. For platforms that support the .fbx format, it is the ideal method of file conversions, as it greatly minimizes data loss and corruption.

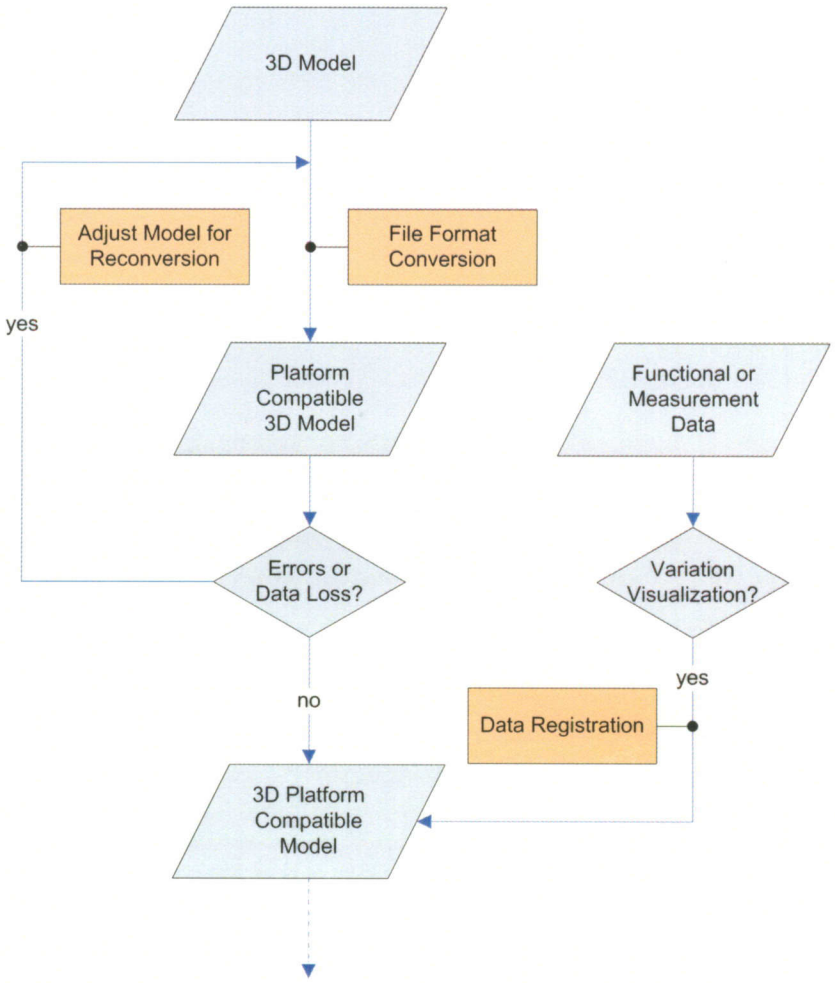


Figure 3-9: Integration of 3D model data.

Integration of 3D model data follows the convert, adjust, import and evaluate process flow. The first goal is to get the 3D model from its original file format to a format recognized by the virtual environment platform. The initial format is determined by a variety of factors, including original software used, scanner output and external sources. The final format of the data is determined by the 3D visualization platform used. The limitations of which format can be used include the required features available in the chosen file format as well level of compatibility for those features.

Once the formats have been determined and the data converted, it is imported into the virtual environment. If the visualization is missing data or if the data is corrupt, there are two possible outcomes. The original data may be converted again, possibly to a different format or with a different set of parameters. Otherwise, the convert model is adjusted to correct for these errors. This cycle continues until the visualized model matches the intended original model. However, due to limitation in file formats and the virtual environment platforms, as well as computer processing power, the intended model may not possible.

3.3 Virtual Environment Platforms

All data, regardless of type or source, is ultimately visualized in a virtual environment platform. These platforms differ in how they accept, interpret, visualize and interact with data. Each platform has a different set of file formats it accepts. Even when a format is accepted, how the data is interpreted varies. Data loss or corruption can often occur. Once the data is accepted and interpreted, the platform has a limited number of methods to visualize the data. This includes visualization of datasets, as well as 3D models. Once visualized, the platform also determines how the data can vary in the world and how the user can interact with the data. Within the scope of this thesis, two virtual environment platforms are used: VRco's vGeo® and WorldViz's Vizard®.

3.3.1 vGeo®

vGeo® is a data visualization platform. 3D models and simple data visualizations, such surface plots and vector fields, can be displayed in immersive (stereoscopic and head tracking) and navigable (fly mode and drive mode with Wanda) virtual reality. Interaction is limited in vGeo® to turning objects off and on and the moving of objects.

```
#---melt_sand_1_100_zoom.vGeo---#

#---Time Defs---#
TIME_DEF allTime 0 1 DATA_RANGE 0 1 USER_TIME

#---Coord Defs---#
COORD_DEF melt_sand_1_100_zoom_COORD_DEF 0 1 USER_UNIT 0 1 USER_UNIT 1 1
USER_UNIT GENERIC TRANSFORM melt_sand_1_100_zoom_Xform OUTLINE_OFF

#---Object Defs---#
STATIC_MODEL melt_sand_1_100_zoom_MODEL melt_sand_1_100_zoom_COORD_DEF
allTime melt_sand_1_100_zoom.wrl Dummy_Xform

#---Transforming---#
CREATE_TRANSFORM melt_sand_1_100_zoom_Xform        #     ÿ    x    y    z
#       TRANSFORM melt_sand_1_100_zoom_Xform SCALE            1    1    1
#       TRANSFORM melt_sand_1_100_zoom_Xform POSITION         0    0    0
#       TRANSFORM melt_sand_1_100_zoom_Xform ROTATE       0   0    0    0

CREATE_TRANSFORM Dummy_Xform

#---Gobs on---#
GOBS_ON melt_sand_1_100_zoom_MODEL
```

Figure 3-10: Example of a vGeo® configuration file.

In vGeo®, the virtual world is configured in an ASCII text file, as shown in Figure 3-10. These configuration files set the static values of the world. These may include world variables, like layout, menus and controls, as well as the data and models involved. Furthermore, the vGeo® configuration files may call other vGeo® configuration files, creating a hierarchy. This allows for a main configuration file that calls all of the world variables first, and then calls individual configuration files for each dataset and model.

vGeo® is not set up for displaying list data. However, vGeo® can accept data as 2D data in the format of comma-separated values (CSV.) A vGeo® specific header is added to the file. The header determines how the data is displayed as well as formatting options. A mesh of points is first described to determine how the data will be mapped. This mesh may be simple plane or may be stretched, skewed and wrapped in all three dimensions. A common use for this is data collection from unevenly spread sensors or to wrap data around a cylinder. The file also contains formatting options, determining position, scale, color (solid and gradient) and the type of plotting function. A series of 2D data can also be used to create animated data.

vGeo® also has support for 2D images. Like any other data, a vGeo® configuration file is required. The configuration file includes information on where the file is located and how it is scaled. The file then refers to the location of the file on the computer. Once loaded, the image is displayed as a 2D plane within the 3D environment. 2D images may also be used as textures, if applied to the modeling stage.

In vGeo®, 3D data is loaded in the same fashion as 2D data. A header to the configuration file contains all of the formatting data such as color and the type of visualization. The configuration file also includes a 3D mesh for the scaling and positioning of the data, as well as the 3D data formatted as CSV. Common visualizations of 3D data in vGeo® include isosurfaces or vector fields, like in Figure 3-7.

3D models in vGeo® are limited to the VRML format. Graphics are converted from the original file format to a VRML file using 3D development software, usually Deep Exploration or 3D Studios Max. The VRML model is then checked for any obvious problems, such as redundant and excess features or inverted normals. When the model is

ready, a vGeo® configuration file, similar to the one used for 2D images, points to the VRML file on the computer. Likewise, this configuration file also includes positioning and scaling information.

The vGeo® configuration file is similar to a basic HTML file. The configuration file includes settings, graphics and data, as well as reference to other configuration files. Objects then load in order and do not interact with each other once loaded. The user can navigate the world, turn data on and off, and start and stop animations. Interaction is limited to the moving of data, within the world, something that must be set in that particular data's configuration file.

The completed vGeo® configuration file is loaded in vGeo® and if it is not displaying what is expected, the graphics are adjusted and the data is reformatted. If the model runs slow, the graphics can be cleaned or trimmed or the amount of data is reduced. Visual and performance checks are repeated until the VR environment is acceptable; much like is shown in Figure 2-7.

3.3.2 Vizard®

Vizard® is a general visualization platform, allowing it to be used as a platform for games and 3D worlds, as well as scientific visualization. 3D Models can be displayed and manipulated in real time by data or user input. Data can be tied to any 3D model, imported as a prebuilt 3D model or displayed with a user-built function, such as a surface or vector field plot. Vizard® allows for a higher level of interaction in the VR environment, however, most of this interaction is not built-in and must be scripted for every new VR environment.

Open-ended scripting in Vizard® allows for an array of interaction, immersion and navigation abilities not available in vGeo®. Stereoscopic viewing and head tracking are built into Vizard® and scripted functions can add more immersion, such as sound effects. Basic navigation via the mouse is available, but further navigation, such as the Wanda or floor pad must be scripted. Vizard® also allows for interaction with data and models directly or through user interfaces. User input from a keyboard, mouse, Wanda or other input device can be used to move and modify both data and models, navigate either manually or automatically, or change the overall look and feel of the environment, all in real time.

Vizard®'s scripting library includes a variety of functions. Models can be loaded and manipulated with just a few lines of code. Loops and programmable logic coupled with customizable user interaction and accessible data points create a 3D environment with a far greater level of interaction. Every element of the Vizard® virtual environment can be augmented with user generated scripts.

Yet, while this open ended scripting allows for an enhanced virtual environment, it creates a more complex data integration processes. For example, vGeo® has built-in ability to visualize 2D data a surface plot. While this function may not be as advanced as what Vizard® could accomplish, this ability is not available by default and must be scripted by the user. Similarly, vGeo® allows interaction with a 3D tracked navigation device. While Vizard® can integrate a larger selection of devices, the functionality must be programmed from the raw data obtained from the device. This carries over to the integration of all data, making it a less straightforward process than vGeo®.

3.3.2.1 Data

Vizard® has the built-in ability to load data in as comma separated values. List, 2D and 3D data can all be stored and read this way. Once loaded into the environment, they are stored as an array variable and can be read by the environment at any time. This differs for vGeo®, which only accessed the data at initialization. The data is then visualized in one of three ways, as shown in Figure 3-2. The simplest is an on-screen display showing the numeric values. For this, the data is passed to a built-in text display function. The data can also be tied to a variation in an object imported from a 3D model, such as a change in color or position. This requires a greater level of scripting to map the values of the data stored in a variable, to the input of another function. Finally, the data can be used an in input to a visualization function, such as a surface plot or vector field. These functions, unlike vGeo®, are not built-in to Vizard®, and must be scripted. However, once the user has scripted such a function, it can be copied, modified and reused in other virtual environments.

3.3.2.2 3D Models

A 3D model is any visual element in the VR environment that was designed outside of the visualization platform. Models are from outside sources or designed by the user. User designed model are models designed with the VR environment in mind, making the transition of the modeling software to the VR environment easier. Models from outside sources often come in incompatible formats or not optimized for real time rendering.

Vizard® accepts a larger array of 3D model formats, including VRML, Object File, and 3Ds Max, with a preference for OpenSceneGraph. Any new 3D model is

converted into one of these Vizard® compatible format using 3D modeling software, usually Deep Exploration or Maya. The file format used depends on the original format and the inherent features of the format. In the same 3D modeling software, the model is checked for redundant or excess features. The completed files can then be dropped into or scripted into the Vizard® VR environment very simply. If the object in the model is static, it is scaled and positioned in the VR environment. If it is a dynamic object, it is tied to a data input via a script, as mentioned in the section on data.

3.3.2.3 Data as a 3D model

Data can also be converted to a 3D model outside of the Vizard® VR environment, and then imported. This is best for static data, as it never varies inside the environment. Data can be converted from its raw form to a 3D visualization using software like Matlab® or the X-ray CT reconstruction software. The new 3D model can then be converted to the correct format and then used like any other 3D model.

3.3.2.4 User Data

The user provides user data through input devices at the run time of the virtual environment. User data is very important in Vizard® and can come from a variety of sources, including traditional devices, like keyboard and mouse, as well as more advanced sources, like motion tracking hardware and floor sensors. Beyond just navigation, input devices can be used to reposition scale and change the display of data, resulting in visualization with the potential of revealing more information to the user. In Vizard®, user data can also modify the data in ways not possible in vGeo®.

3.3.2.5 Finalization

Models, data and functionality are all scripted into Vizard®. Visualizations and scripted functions are adjusted until they display and perform, as they should. This is done following a process similar to Figure 2-7. Like any other virtual environment, graphical data is loaded, checked, adjusted and reloaded until it displays correctly. A common issue with Vizard® VR environments comes when the complexity of the world surpasses the computing power. When this happens, the graphics, data and functionality must be cut back or optimized. This may include lowing the complexity of models in a 3D modeling platform, re-scripting more efficient functions or an overall reduction in the functionality of the virtual environment.

3.4 Comparison Criteria

When evaluating multiple pathways for the development virtual environments, a set of criteria must be defined. The goal of this evaluation is to find information that could lead to virtual environments with a greater chance of extracting data. To do this, higher quality environment are a goal. However, higher quality environments tend to be more complex creating higher development costs. Therefore, it is also a goal to lower the development cost of creating such an environment.

One of the difficulties of evaluating virtual reality is the inability to predict outcomes. It is not possible to give a set up inputs, and compare a set of outputs. In this thesis, the pathways are evaluated not on the performance and outcome, but on how the pathways meet a set of predefined goals and criteria. A matrix diagram, Table 3-3, is used to show which evaluation criterion applies to each goal. The diagram is separated into

two parts. Development characteristics are the criteria important to the building of the virtual environment, while run-time characteristics correspond to the criteria important to the actually execution of the visualization. The highlighted areas show which of the criteria apply to the goals at the left side of the table. The highlighted areas show the criteria that do not affect the goals.

Table 3-3: Relationship between pathway goals and evaluation criteria.

	Development Criteria				Run-Time Criteria		
Goals \ Criteria	Ease of Design	Accuracy	Efficiency	I/O Frame-work	Efficiency	Interaction with User	Software Integration
Lower Development Costs							
High Quality Environments							
Extracting Information							

3.4.1 Ease of Design

Certain pathways and software may impede the design of a virtual environment. The ease of design criterion measures a pathway on how intuitively and quickly a result can be reached. It does not take into account the quality of the result, but simply the step required to get from data to visualization. Ease of design is important to lowering the cost of developing virtual environments, and a quality pathway will look to make the process as simple as possible, while considering other criteria.

3.4.2 Accuracy

In this thesis, accuracy is defined as how exact a virtual environment can visualize the original raw data. Data can be lost in the process of designing, developing and deploying an environment. This could be the result of software error, conversion error or designer error. Software error and conversion errors are similar, in that part of the pathway unintentionally changes the data. Designing error come when the method for visualizing the data compromises the integrity of the data. This is usually caused by modifying data to increase the visual appeal of the environment. In the end, an accurate model ensures high quality environments with a better chance of extracting information.

3.4.3 Efficiency

Efficiency is the combination of ease of design and accuracy. Essentially, it describes how easy it is to make an accurate visualization. It also describes how easily it is to create an environment with enhanced immersion, interaction and navigation. This could include visually pleasing environments, integration of user interaction and extra environmental features. Efficiency can also refer to how well the environment operates when executed. This is essentially a measure of how well all the required data is visualize, versus limitations, such as frame-rate.

3.4.4 I/O Framework

When creating a virtual environment, the pathway chosen may not allow for the desired output. In this thesis, the I/O framework criterion measures how well a set of inputs return the desired output. This can be compared to the versatility of the pathway to

perform under different sets of requirements. The ideal pathway will have an I/O framework that does not impede the development of the virtual environment.

3.4.5 Interaction with User

The ability for immersion, interaction and navigability is what sets virtual reality apart from other scientific visualizations. These abilities add to the quality of the virtual environment and ultimately increase the chance of extracting information from the data. The interaction with user criterion measures how a virtual environment takes advantage of these abilities to increase the users connection with the data. A quality pathway leads to a virtual environment that has intuitive user interaction.

3.4.6 Software Integration

A variety of software is used in the development of a virtual environment. The software integration criterion evaluates the functionality of the software with the functionality the pathway itself. Pathways that rely greatly on the software used are limited by the quality and versatility of the software. A quality virtual environment takes advantage of the most powerful and versatile software features.

Chapter 4 - Results

4.1 Visualization of Data as a Surface Plot

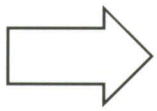
One of the simplest forms of 3D scientific visualization is a surface plot. Surface plots can be used to display a system with two inputs and one output. The result is a plot like the one shown as part of Table 3-2. Unlike a simple 2D line graph, a surface plot can be viewed from any angle, requiring a 3D viewing method. Virtual Reality is an ideal method for viewing the visualization, as it will allow the user to intuitively navigate the data. Therefore, the visualization of 2D Data as a surface plot is used the compare the different processes for vGeo® and Vizard®.

4.1.1 Visualization a Surface Plot in vGeo®

vGeo® has been set up to handle basic visualization, such as surface plots and follows a flow similar to the one described in Figure 3-4. However, for 2D data, not much needs to be done in terms of converting the format. vGeo® cannot import commercial spreadsheet formats, like Microsoft Excel. Instead, the data is contained with the vGeo® virtual environment configuration file. This step simply involves adding the data, as text, to the configuration file.

Organization of data in vGeo® involves a few steps. The first is to set up the delimiters of the data. Each element is separated by a space, while each new line is separated by a space, resulting in a long, one dimensional stream, as shown in Figure 4-1.

342	464	278	753	236
523	239	348	402	230
453	705	134	527	847
234	220	138	347	235



```
342 464 278 753 236 523
239 348 402 230 453 705
134 527 847 234 220 138
347 235
```

Figure 4-1: Conversion of 2D data to a vGeo® configuration file format.

In order to organize the data back into its original 2D format, a grid is configured. This is accomplished by defining a grid in the configuration file. In the case of Figure 4-1, the grid is 5x4x1. This grid can be an evenly spaced square grid or something more complex. When the configuration file is executed, the data points will be mapped, in order, to this grid. When the end of one row is reached, the next data point is mapped at the beginning of the next row. It is the responsibility of the designer to ensure the outputted data matches the original grid. Otherwise, the data will be displayed incorrectly.

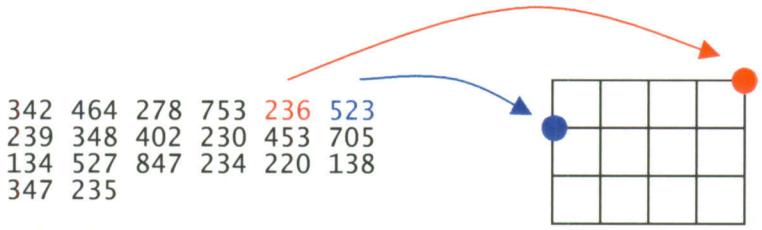


Figure 4-2: Mapping of 2D vGeo® data to configured grid.

Once the data has been converted and formatted, vGeo® must be configured to the visualization needed. The vGeo® virtual environment configuration file has a number of required fields. These include version number, world and visualization configurations, as well as the grid configuration and the actual data. To visualize the data as a surface

plot, the required field is set to surface plot. Other configuration fields include color or color map, scale and position. Once the configuration is complete, it can be loaded with the vGeo® software.

4.1.2 Visualization of a Surface Plot in Vizard®

Importing data into Vizard® for visualization as a surface plot also follows the flow in Figure 3-4. The required format for Vizard® is comma separated values (CSV.) This conversion can be made with any spreadsheet program. The CSV file is stored with the Python script file that controls the virtual environment.

A script command is used to load the data from the CSV file to a variable in the environment. This command is a built-in CSV read command and requires no extra scripting. This variable can then be used for any function within the Vizard® virtual environment.

Unlike vGeo®, Vizard® does not have a built-in surface plot function. To display 2D data in Vizard®, a script to do so must be created. This script takes each data point, draws a corresponding vertex and then draws faces based on those vertices. This step is much more involved than the vGeo® method. However, once the function has been designed, it can be reused for future environments. Furthermore, this function can be modified, giving more flexibility than vGeo®. Once completed, the Vizard® scripted virtual environment file is executed.

4.2 Visualization of 3D scans of Synthesized Particle Aggregates

Visualization of 3D data gives a view of the data from all angles. Virtual reality allows a user to intuitively navigate to those angles. In this case, virtual reality is being used to compare two different scans of one particle. The first scan is optical tomography, essentially a 3D model synthesized from photographs taken at set angles. The second scan is x-ray CT scan of the same particle, considered the "gold standard." The particular demonstration of virtual reality has two parts. The first is getting the raw data of the scan into the virtual environment. The second step requires setting up the interaction between the user and the particle, allowing the user to intuitively compare particles.

4.2.1 Conversion of Raw Data to 3D Model

Particle data starts out as a set of vertices and faces. This may be the STL format or a simple collection of vertices and faces. These formats must be converted to virtual environment compatible format. However, when this conversion occurs, the new formats contain fields for data not contained within the raw shape data. These fields therefore are filled with a placeholder, and sometime lead to erroneous models. A common error is inverted face normals.

Face normals give information about the faces of a 3D model. Most importantly, these faces give the direction in which the face is facing. A face is only visible from the side the face normals are pointing. Therefore, in order to see a particle, all the face normals must be pointing outward. However, because the direction of the face normals was arbitrary set as a placeholder, they may not be facing outward. If the face normals are pointing inward, the front of the particle will be invisible, and the inside of the back will

be shown. If this is the case, the normals must be inverted using a 3D modeling platform, in this case, Deep Exploration.

The converted and corrected file is now loaded into the virtual environment in the processes described in section 3.3. For vGeo® this file must be VRML, while Vizard® can handle a wider array of files. In this case, an Object file is used for Vizard®. With the normal corrected, the model should look correct. If the model has a texture, differences in how the 3D modeling program and the virtual environment platform handle the texture may result in data loss. Therefore, if the texture is displaying incorrectly, it must be adjusted in the 3D modeling platform and reloaded into the virtual environment until it displays correctly.

Next is the finalization step of integrating the data. In vGeo® the appropriate configuration file fields are set to scan and position the object. For Vizard® the scaling and position script functions are used to set the object. If the object is dynamic, as it is with this demonstration, the interaction must be configured. This is all shown in Figure 4-3.

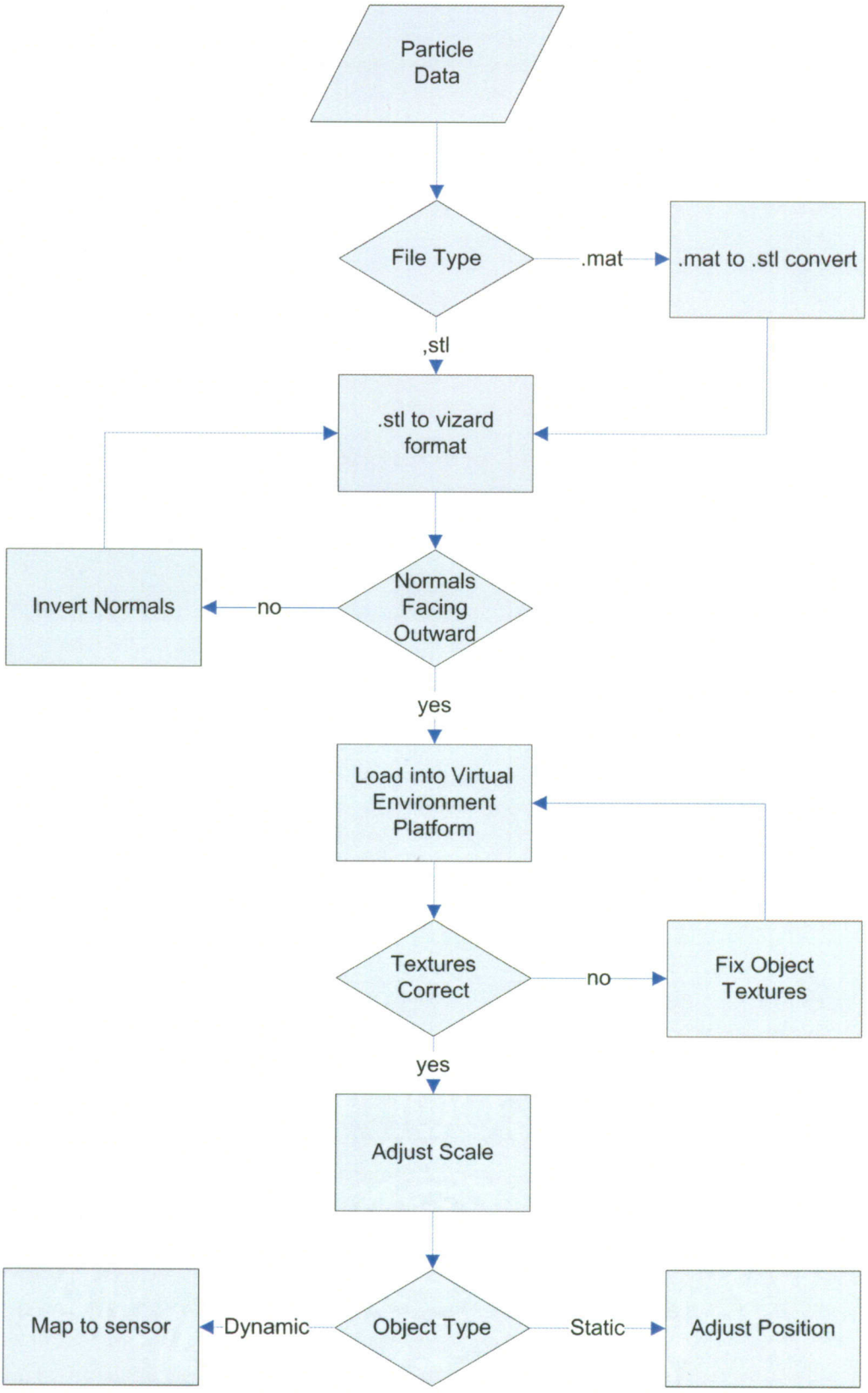


Figure 4-3: Integration of synthesized particle arrogates.

4.2.1.1 Conversion of X-Ray CT to 3D Model

The result of the X-Ray CT scanner is raw data. This data is passed to a reconstruction computer where the output is an STL file. vGeo® and Vizard® do not support STL files. The file is then transfer to a computer with 3D modeling software. The file is checked for error and converted to an Object file for Vizard® or a VRML file for vGeo®. The files are then transferred to the visualization computer to be loaded into the virtual environment.

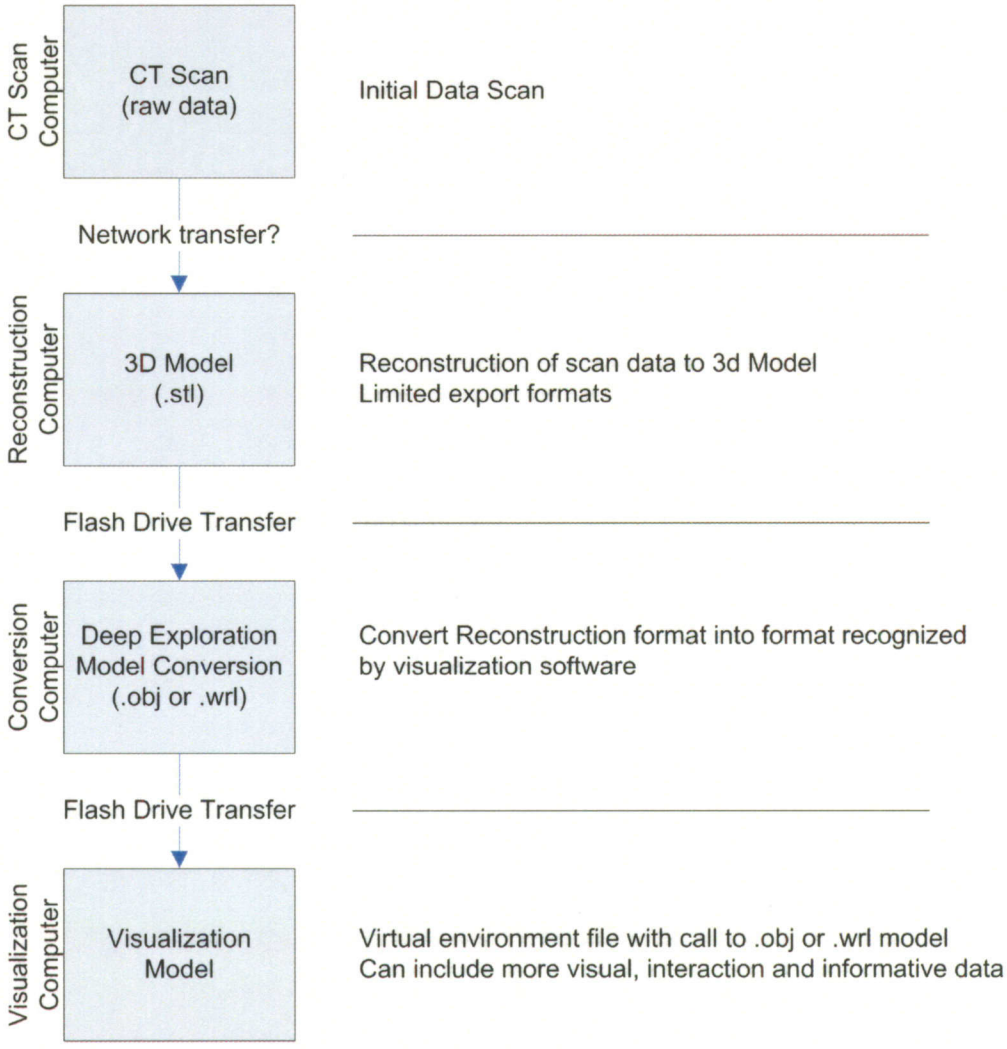


Figure 4-4: Integration of x-ray CT data.

4.2.1.2 Conversion of Optical Tomography to 3D Model

Optical Tomography (OT) data is reconstructed in Matlab® and therefore the output is a .mat file. The .mat file is a proprietary file that stores the model in a very simple format. Matlab® is required to convert the data into a basic 3D model. Matlab® is also need to convert this model to an STL file. From this point, the process is the same as the X-Ray CT process. The file is converted to a platform compatible format. Because the Matlab® data was so basic, these models are more likely to have inverted normals. The adjusted and completed models at loaded onto the visualization computer.

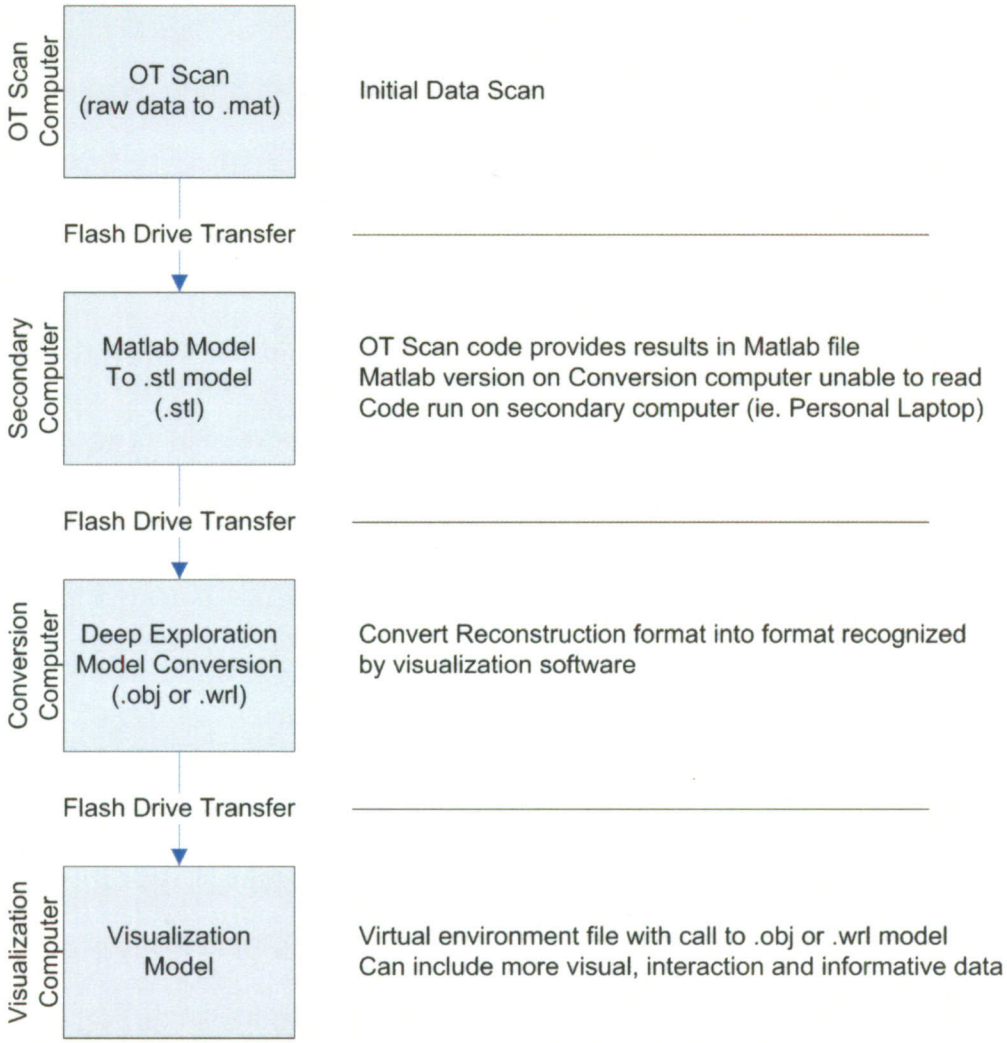


Figure 4-5: Integration of optical tomography data.

4.2.2 Configuration of User Interaction

In order for the user to compare the two particle scans, they must be able to interact with the models. This is where the processes for vGeo® and Vizard® differ greatly. The configuration of vGeo® was relatively simple. One of the fields the configuration file sets the ability to move an object. With this set, the user can "spear" the particle with the 3D cursor and move it around, as if it were on the end of a stick.

For Vizard®, the particle must be tied to the motion sensors. A plug-in was developed to import the motion sensor data into Vizard® in real time. This acts as the input to a set of position script commands. The user holds a sensor in each hand and moves them around as if they were holding the particles. This ends up being much more fluid and intuitive than the vGeo® "spear" method. In case the particles are mismatched in size, the Vizard® environment takes things a step further. Two floor sensors are tied to scaling script commands, allowing the used to increase or decrease the size of the particles, in real time, with the push of a button. Two more sensors are tied to scripts that allow the user to change the current scans being compared. A background and sound effects were added to add the environment. The resulting environment can be seen in Figure 4-8.

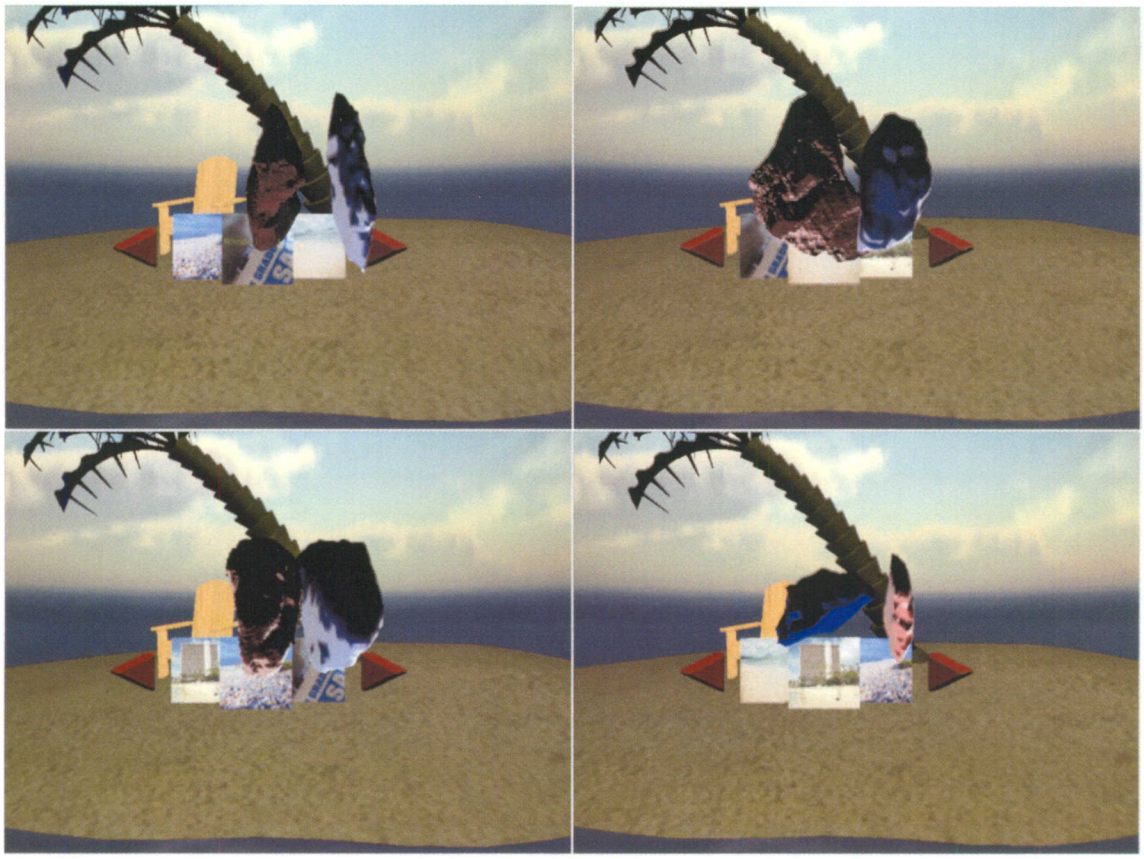


Figure 4-6: Vizard® particle visualization.

In the end, the vGeo® model was easier to set up, but Vizard®s offers a much more interactive and immersive experience. The "spear" method in vGeo® makes aligning the particles very difficult. In Vizard®, the particles move with the users hands, and can be aligned by the user simply bringing his or her hands together. The Vizard® environment is also much more expandable, as shown by the scaling functionality. All of this gives the Vizard® environment a higher quality virtual reality experience, while the reusable motion sensor plug-in and floor sensor functionality, the cost of developing such an environment has been decreased.

4.3 NASA Test Stands

NASA uses a variety of test stands to test their rockets and thrusters. This test stand allows for the test of equipment before it is put into use, to ensure proper operation. If the stands can be visualized in virtual reality, the test can now be replayed in a 3D environment. If logic can be imported, it could also be possible for virtual tests to be run, removing the need to expensive and time consuming real life tests. In the scope of this thesis, two test stands were visualized in virtual reality: the E1 Test stand and the E-3 MTTP Trailer.

4.3.1 E1 Test Stand

The E1 Test Stand is one of many test stands used by NASA. It is shown in Figure 4-7.

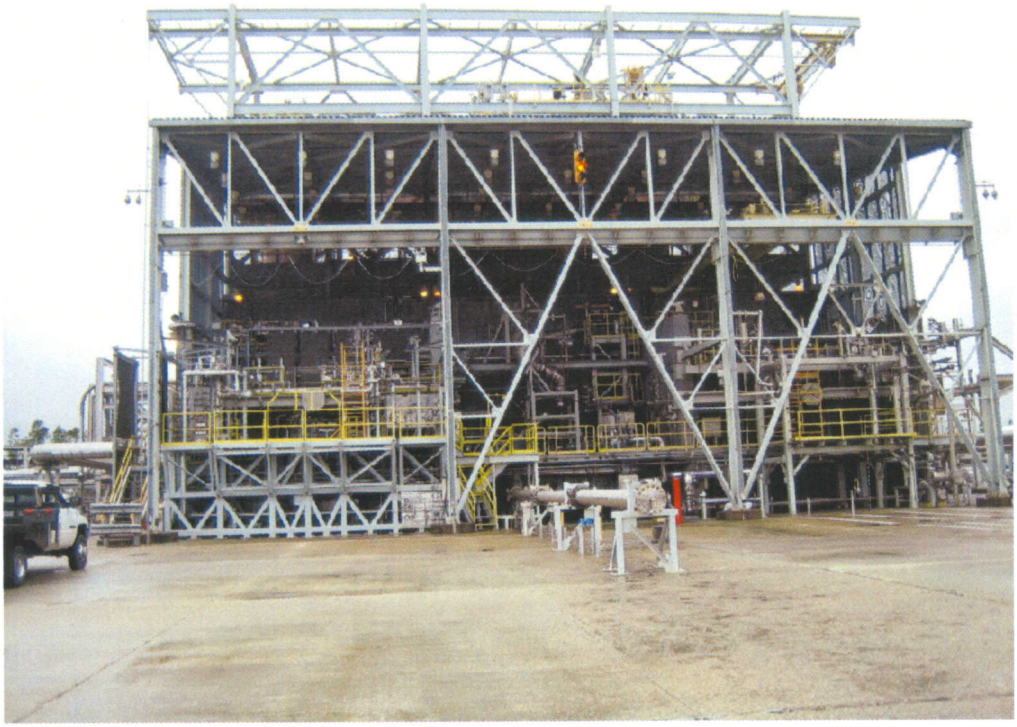


Figure 4-7: E1 test stand.

4.3.1.1 Visualization of the NASA E1 Test Stand in vGeo®

In vGeo®, the E1 Test Stand visualization is simply an integration of 3D models problem. The models were obtained from NASA in the form of Solidworks® files. Deep Exploration was used to convert the files to VRML files for use with vGeo®, as shown in Figure 4-8. These VRML files are then loaded into vGeo® via a configuration file. In this case, the files were found to have many errors and redundancies, caused by either bad initial data or corruption in the conversion process. Therefore, the files are loaded into 3ds Max, adjustments are made to the 3D models. This is repeated until the 3D models display correctly.

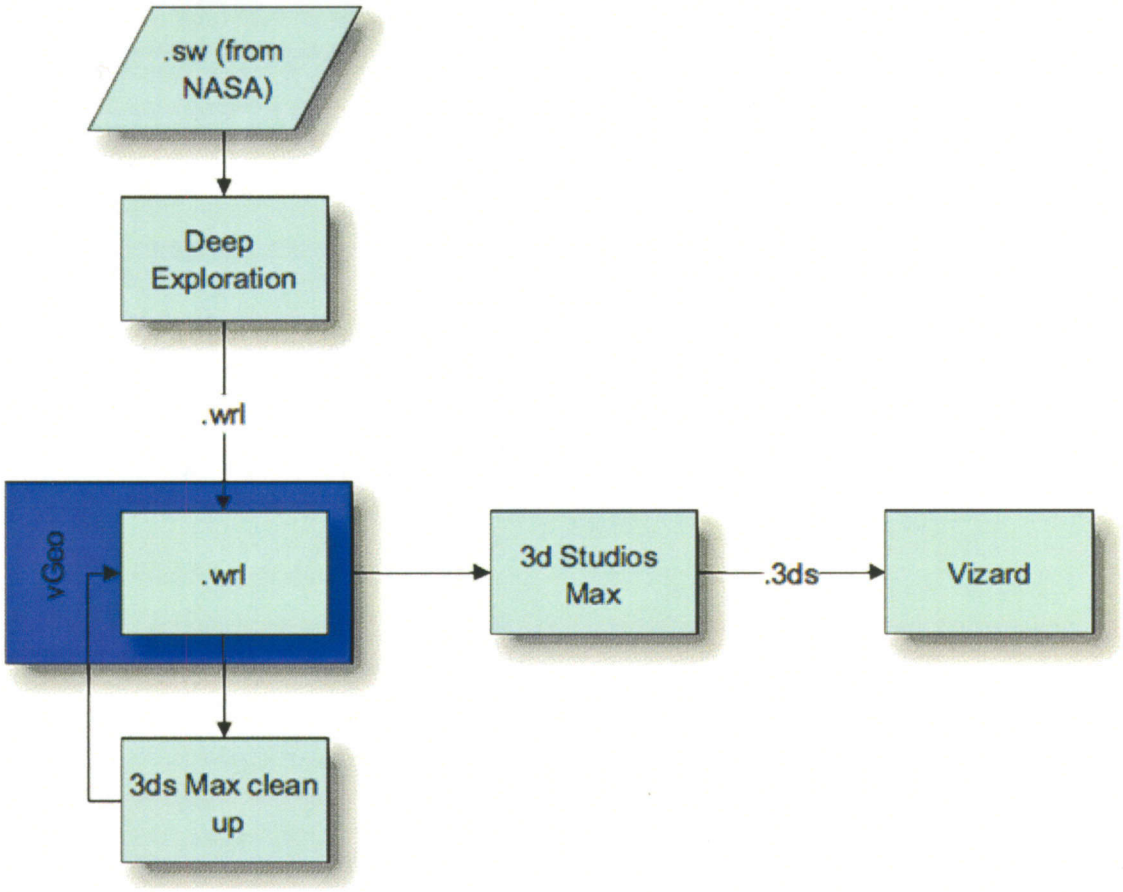


Figure 4-8: Integration of E1 test stand model data into vGeo® and Vizard®.

The integration of the Solidworks® files ran into many problems, the first of which was performance. When the entire model was loaded into vGeo®, the frame-rate dropped sharply. To fix this, the model was separated in to sub-models in 3ds Max. The models were then loaded into vGeo® one at a time, choosing the most important models first, until the frame-rate started to drop. Each model was colored a different color in order to show the different sub-models, as shown in Figure 4-9. This way, each sub-model can be recognized and the more important ones can be left in the environment. The resulting model is much less detailed than the image in

Figure 4-7. The next step was to determine what was causing a simple model to require a greater amount of computing power.

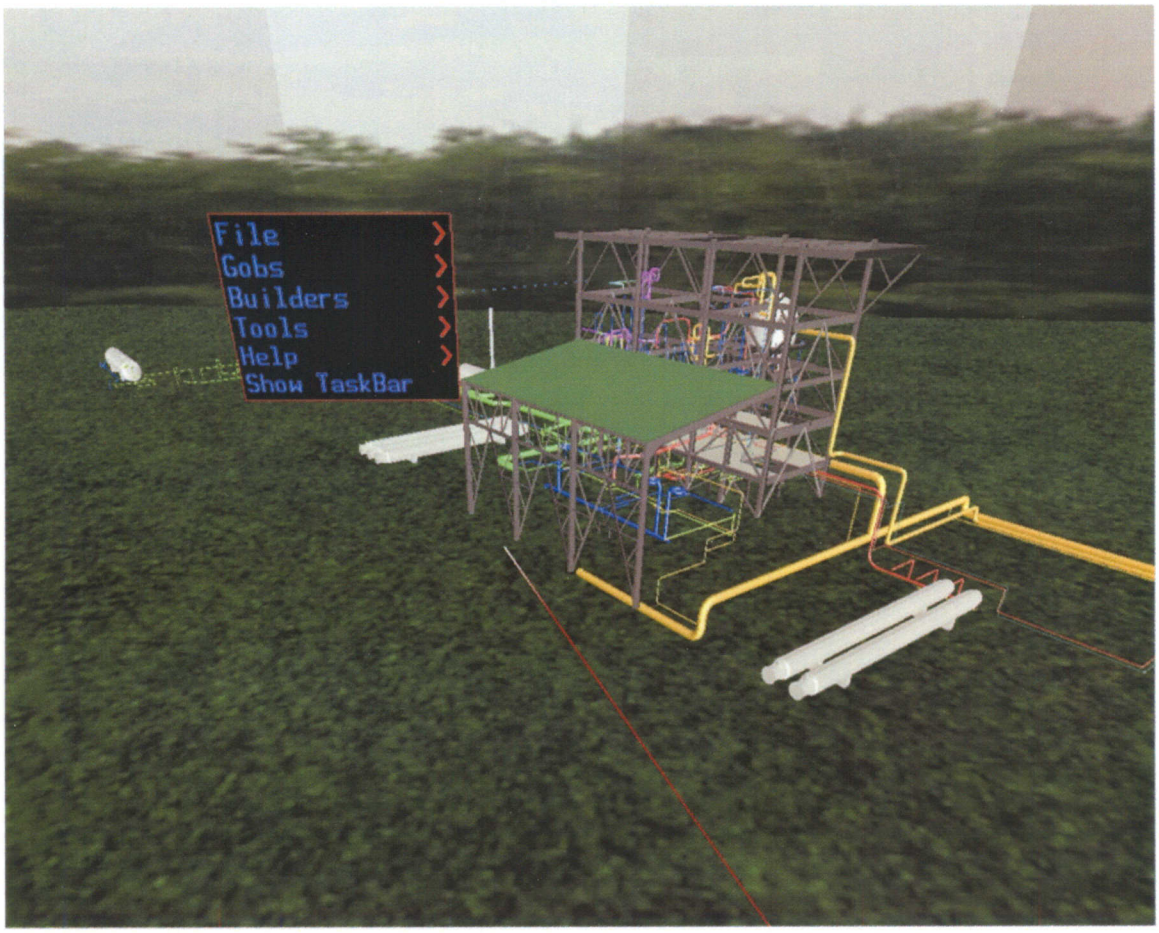


Figure 4-9: Menu system and colored sub-models of the E1 test stand in vGeo®.

Closer inspection reveals a variety of performance killing problems with the model. The model contained some excessive detail. Nuts and bolts had been modeled with extremely high polygon counts. This adds to the detail of the model, making it more immersive, but these small details often had higher polygon counts than the piping systems. Therefore, they were removed from the model to increase the frame-rate.

The 3D graphical data of the test stand also suffers from poor modeling. Figure 4-10 shows three typical pipe elbows from the vGeo® models. Models with a lower number of faces have a low polygon count. A lower polygon count requires less computing power. While these three pipes appear to have relatively low polygon counts,

there is much excess in this model. The ideal model is like the part of the lower pipe marked in green. These faces are orderly and there are no extra faces. The red faces on the middle pipe are an example of excess faces. This part of the pipe could be modeled like the green section. While these extra faces may not seem like much, each additional polygon adds to the computing power required, as problems like this repeated throughout the model can add up quickly.

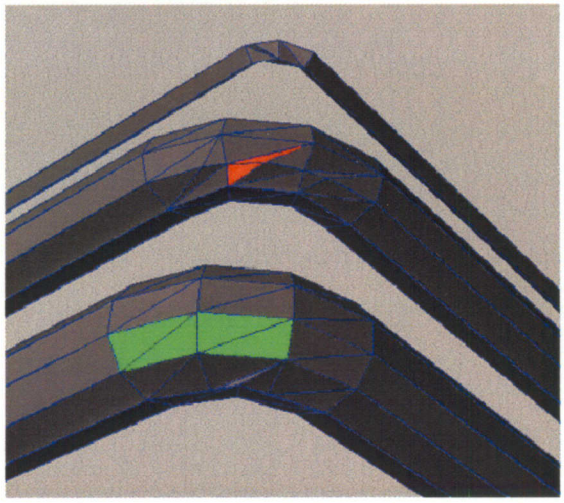


Figure 4-10: Close-up of E1 test stand models in vGeo® [7].

Another issue with this model was redundancy. Redundancy in 3D models occurs when two similar models are drawn directly on top of each other. Processing the same model twice can greatly reduce the frame rate. In Figure 4-11, the green section is a duplicate of part of the larger pipe model in the foreground. This redundant model is a result of a poor Solidworks® model, which was not fixed to the conversion to an Object file. Redundancy causes problems that can be difficult to find within a model, while redundant models effectively require double the processing power.

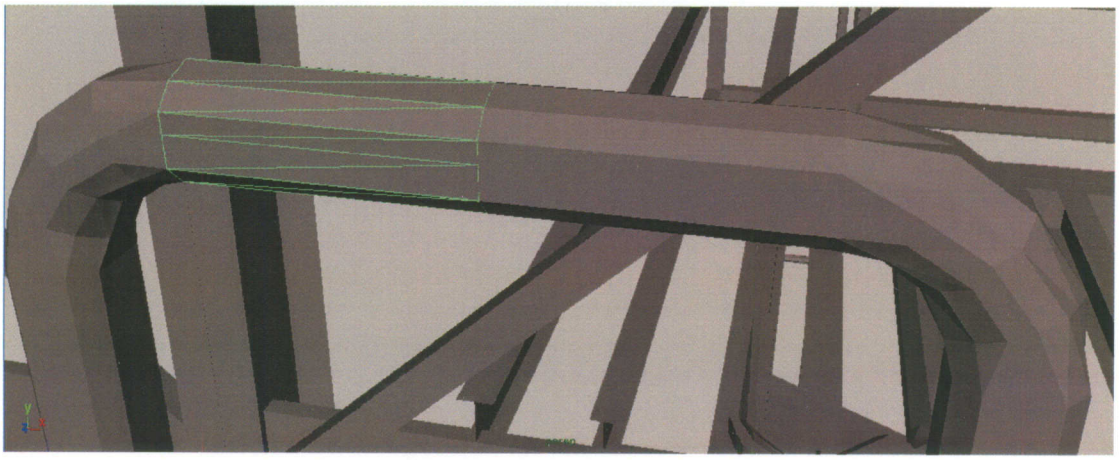


Figure 4-11: Redundancy in the E1 trailer model [7].

Another performance reducing problem in the E1 test stand model is infinitesimally thin, segmented models. Normally a model simulates a three dimensional shape. For example, a cube would consist of twelve polygons, two for each side. Each side also shares common edges with the neighboring sides, making the cube a solid shape. If the cube were to be modeled from infinitesimally thin, segmented models, each side is a separate plane. Not only does this complicate the model, but also each side is actually an infinitesimally flat cube, which increases the polygon count and decreases the frame rate. An infinitesimally thin, segmented model is shown in Figure 4-12.

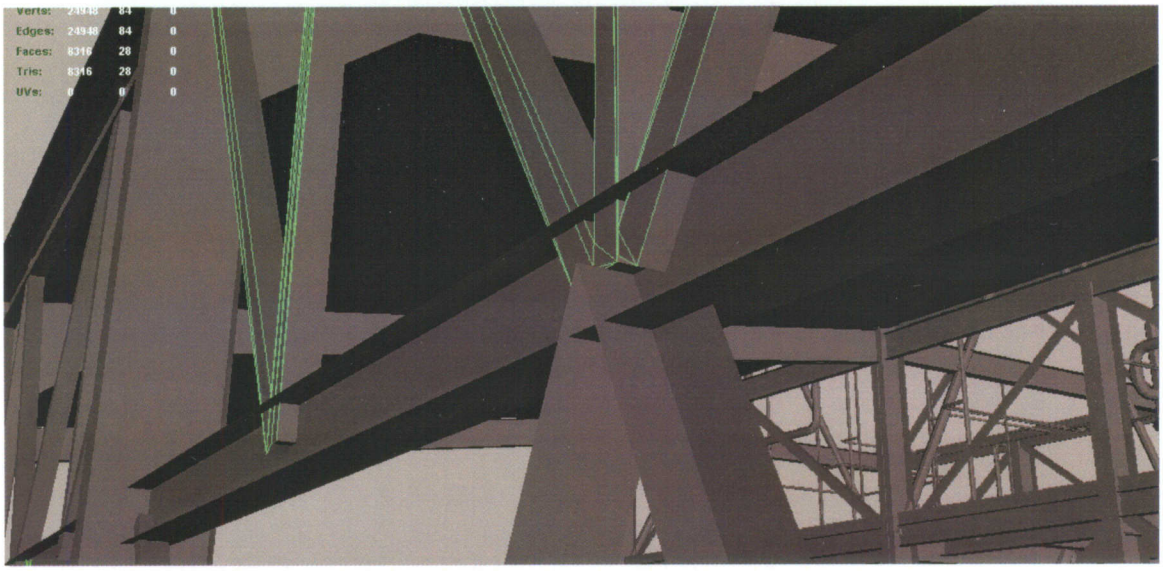


Figure 4-12: Infinitesimally thin, segmented models in the E1 test stand model [7].

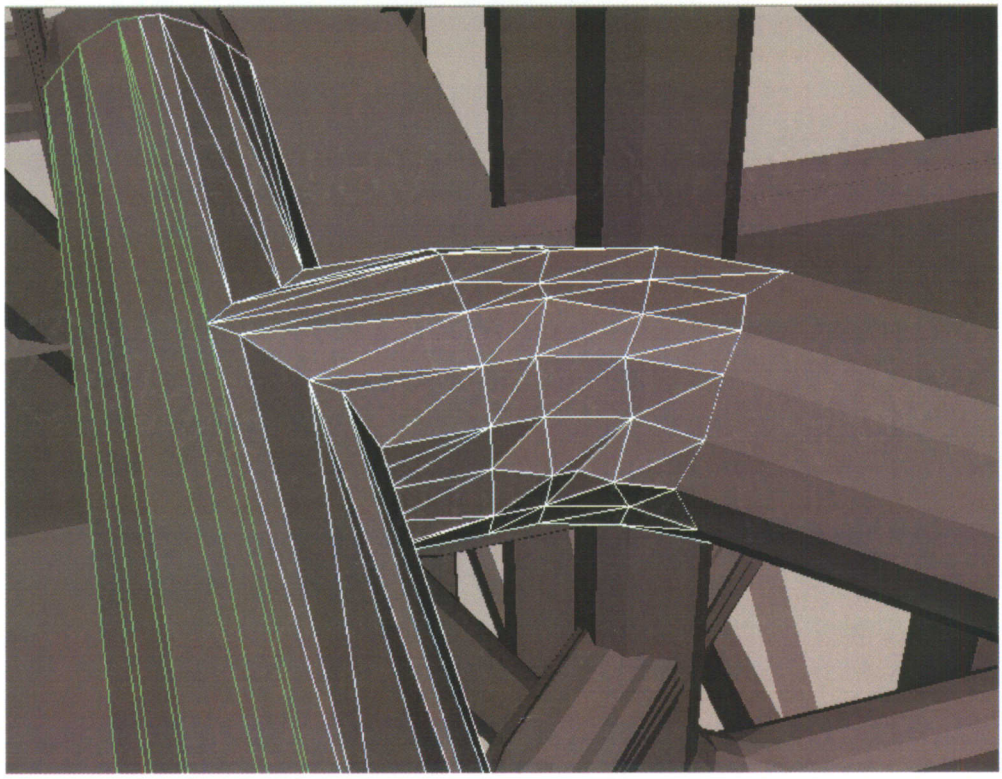


Figure 4-13: "Worse case scenario" of poor modeling in the E1 test stand model [7].

The E1 Test Stand model was plagued with problems. Figure 4-13 shows a "worst case scenario" of poor modeling. This tee intersection of two pipes has redundant models, as well as sloppy and excess geometry. These problems were fixed to some extent; however, many of them were difficult to find. Searching for them all would greatly increase the cost of developing. Therefore, the few that were easily found were fixed and only the necessary sub-models were kept in the final model. The results are shown in Figure 4-14. This model shows the main structure, major pipe systems and major tanks. While it gives a visual overview of the entire system, visually, it is far from the actual test stand shown in Figure 4-7.

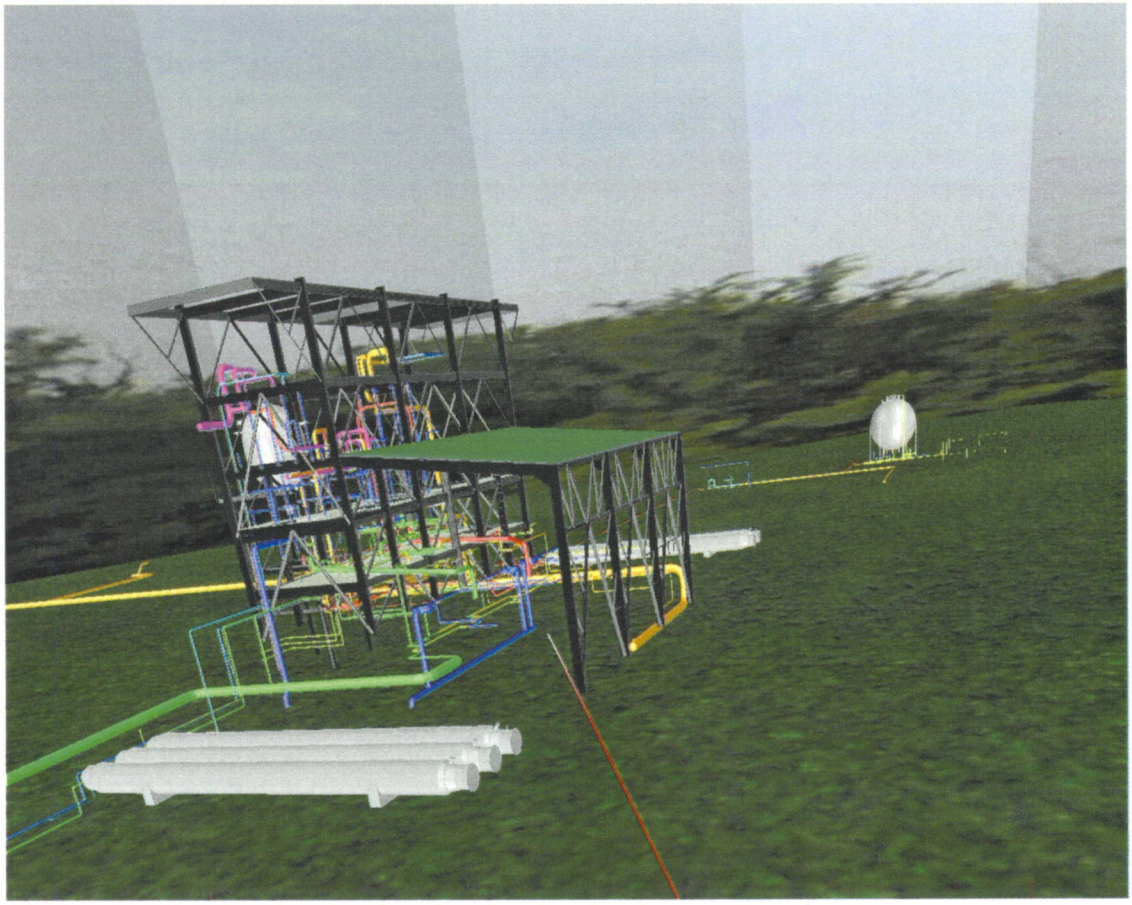


Figure 4-14: vGeo® model of the E1 test stand.

To add to the immersive aspects of the vGeo® E1 Test Stand model, environmental aspects were added. Initially the model was simply floating in space. To give the sense the model is on the ground, a 2D plane with a simulated grass texture was added to the model, as shown in Figure 4-15. This was added to vGeo® like any other 3D model. To give the environment even more realism, a sky dome was added. The sky dome is a model of a textured dome placed over the models to give a sense of horizon and sky above. The texture used is a 2D data in the form of actual pictures of the surrounding scenery in Mississippi. The sky dome is modeled in a 3D modeling platform, exported as a VRML and imported into vGeo® like any other model. The results are shown in Figure 4-16.

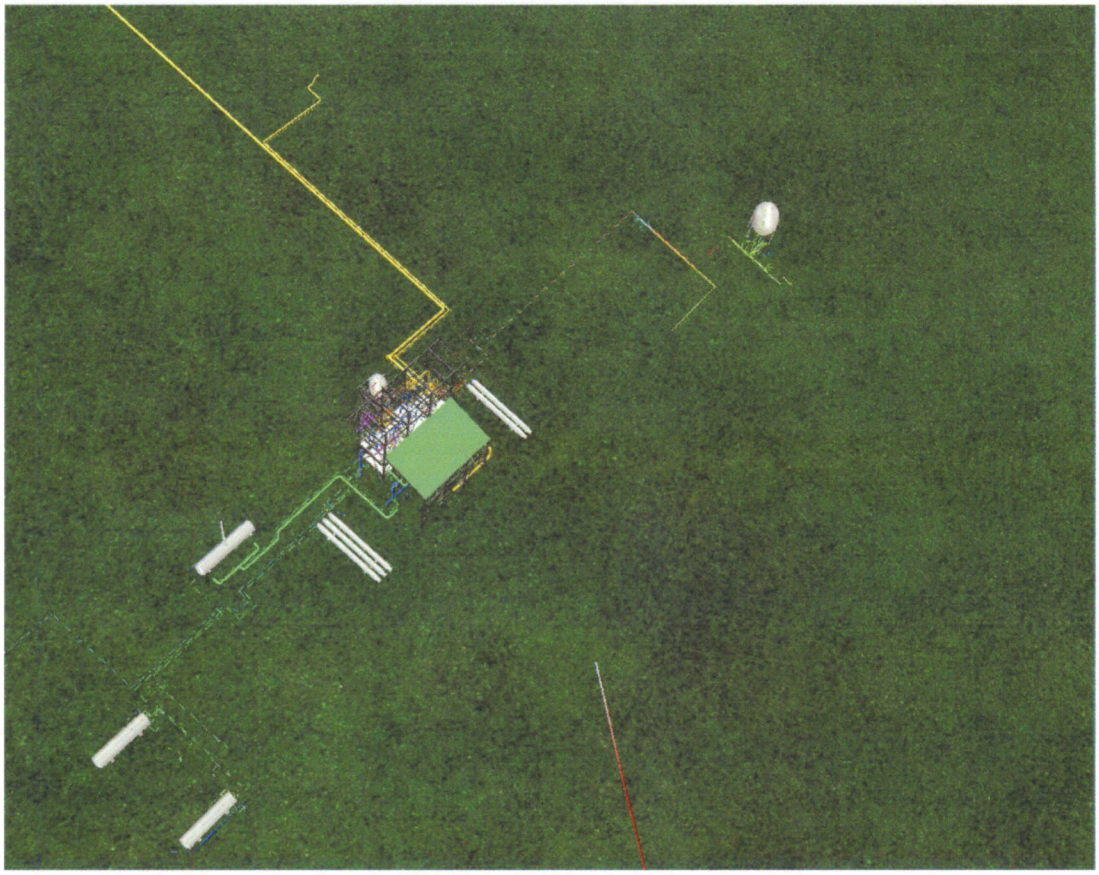


Figure 4-15: Grass textured plane in vGeo® E1 test stand model.

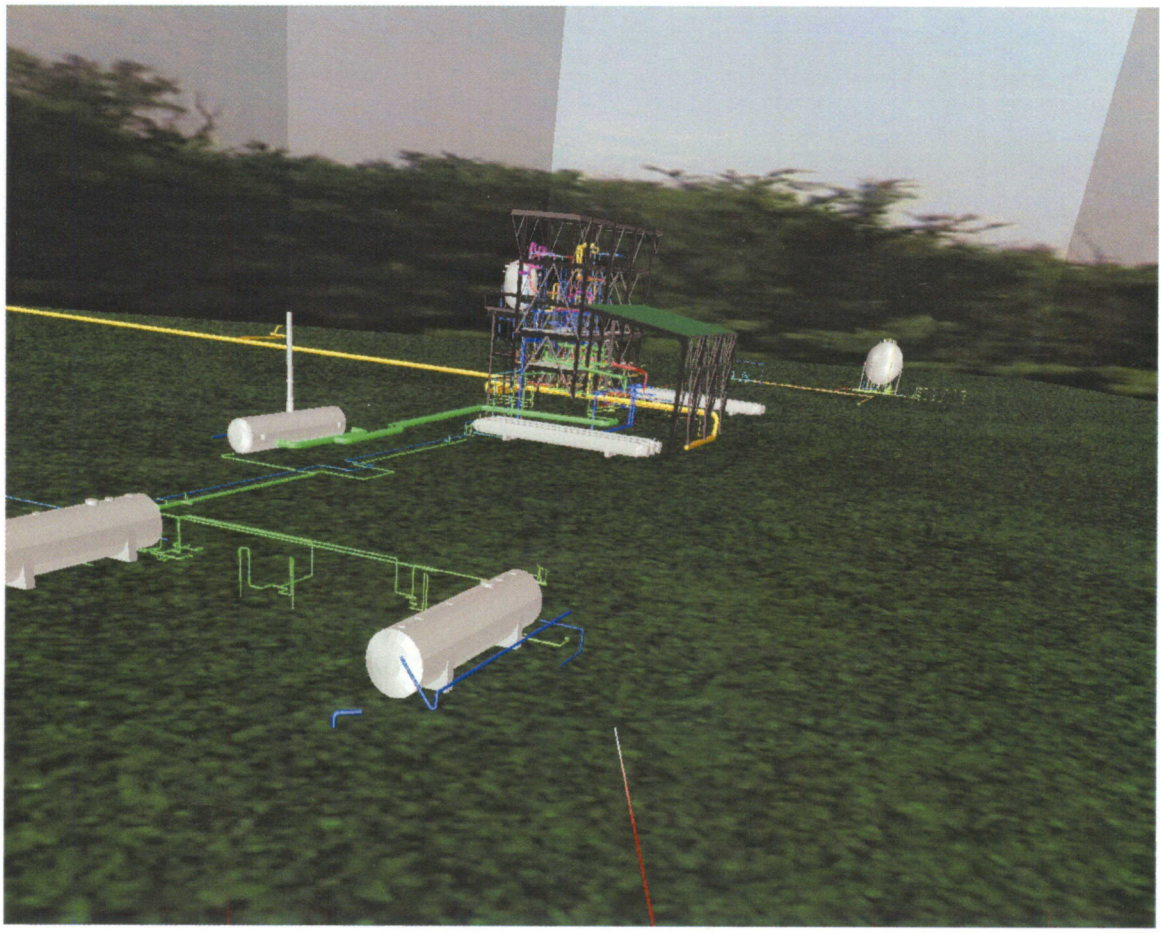


Figure 4-16: Textured sky dome in vGeo® E1 test stand model.

The resulting virtual environment consists only of graphical data, lacking both measurement and functional data. Visually, the model is not very accurate compared to the photographs. The process of transferring the 3D models was relatively simple, but poor models and file formats caused the process to be inefficient. The environment is partly immersive and can be navigated, but lacks in interaction. vGeo® lacked a simple method for implementing the list data of sensor reading, creating a very static environment. Overall, the cost of developing, designing and deploying this virtual environment outweighed the advantages of viewing the stand in virtual reality.

4.3.1.2 Visualization the NASA E1 Test Stand in Vizard®

Because of the limitations of vGeo®, a new virtual environment platform was chosen. The NASA E1 test stand was once again visualized, this time, in Vizard®. Vizard® provides a number of advantages over vGeo®. These include greater support for a variety of file formats, better handling of list data and Python scripted environments. The ability to script environments using the Python scripting language is possibly the greatest advantage of Vizard®, allowing for advanced interaction and advanced navigation.

File conversion

Vizard® supports a greater number of file formats than vGeo®. The formats supported can be seen in Table 4-1. Deciding which format to do could be a difficult process. How well a particular file format performs depends on many factors. Each format handles the common features of 3D model formats differently and may contain additional feature not necessarily found in other formats. While this platform may support a particular format, it may have partial or no support of certain features. An example of this is a 3D model file including the 3D model as well as extensive texture and animation data, but the visualization platform only loading the model data.

Table 4-1: Compatible file formats.

File Type	wrl	Obj+mtl	3ds	Osg/ive	Ma/mb	fbx	stl
vGeo®	X	X	X	X			
Vizard®	X						

There are also points in the process where data loss and corruption can occur, shown in Figure 4-17. When the 3D data is load into the 3D modeling platform, the model may not load as intended, especially if the format is not the platforms native

format. If no information is known about the intended model, this will go unfixed. Next, the model is exported as a virtual environment platform compatible model. This step is also susceptible to data loss and corruption for the same reasons. The final step is loading the data file into the virtual environment platform.

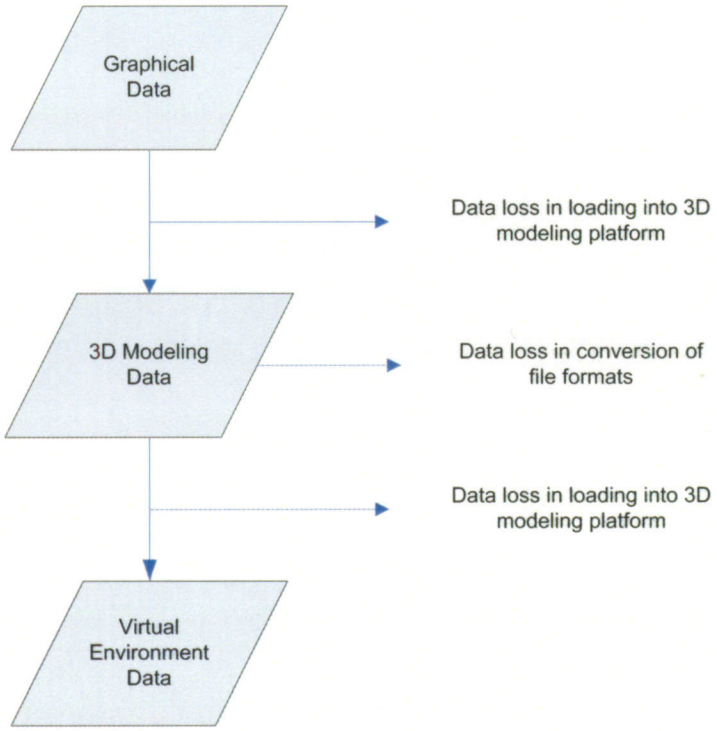


Figure 4-17: Potential locations for data loss or corruption.

For the E1 test stand, only the model data was important. It was not necessary to transfer texture, animation or any other additional data. Using 3ds MAX the files original Solidworks® files, the data was converted to every possible format supported for export by 3ds MAX and for import by Vizard®. Each file was loaded into Vizard® for comparison. Either using the load functionality of the Vizard® interface, or including a load script command can accomplish this. The Object file format was chosen solely on performance, as it ran with the highest frame rate for this particular model. However, the

imported Object files lose the color information, as shown in Figure 4-18. Ultimately, the VRML files were used, even though they contained many errors.

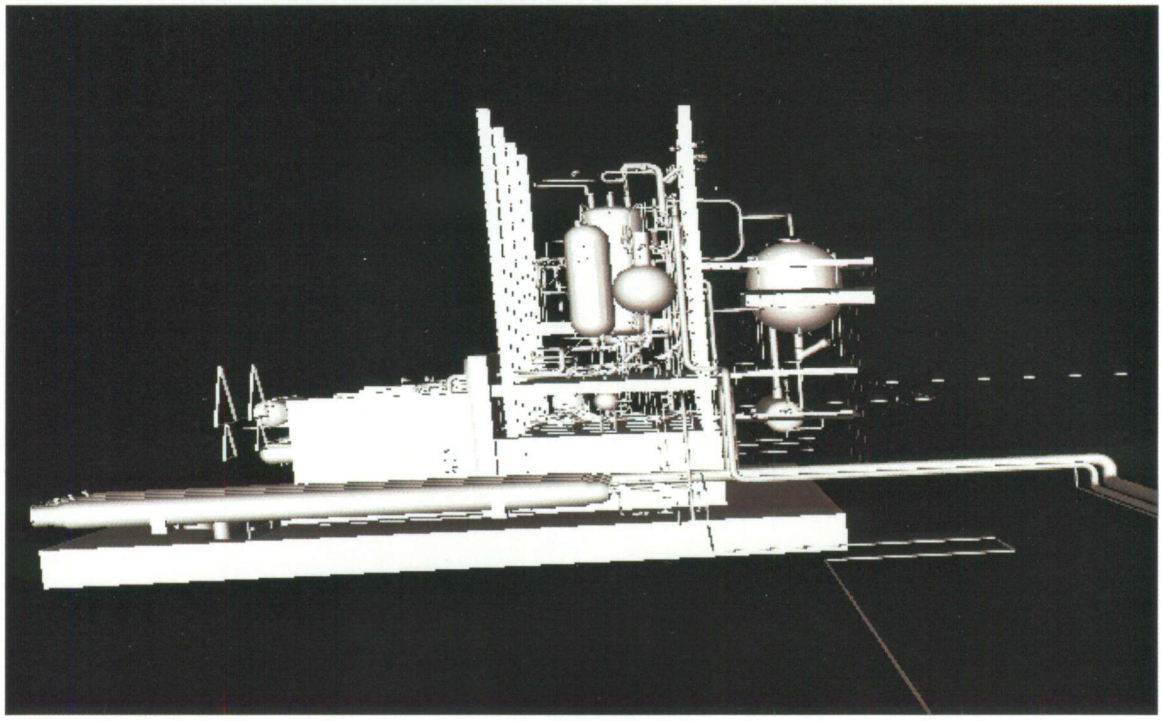


Figure 4-18: Imported object file with missing color data.

Data import

Vizard® also has far better support for list data. List data is very useful for visualizing data such as reading from a thermocouple over time. Vizard® accepts list data as comma separated value (CSV) files. Like 2D data, list data can be converted to the required format with any spreadsheet software. Once in the CSV file, the data can be loaded into a variable in the virtual environment. This is using a built-in CSV read Python script command. Once loaded into the environment, this variable can be used as input to a variety of functions. The data may simply be the input to a numerical display. As the input to model manipulation functions, the data could vary the position, size or color of

an object. As the input to logic and conditional statements, the data could be used to trigger events. The overall process of integrating this list data is shown in Figure 4-19.

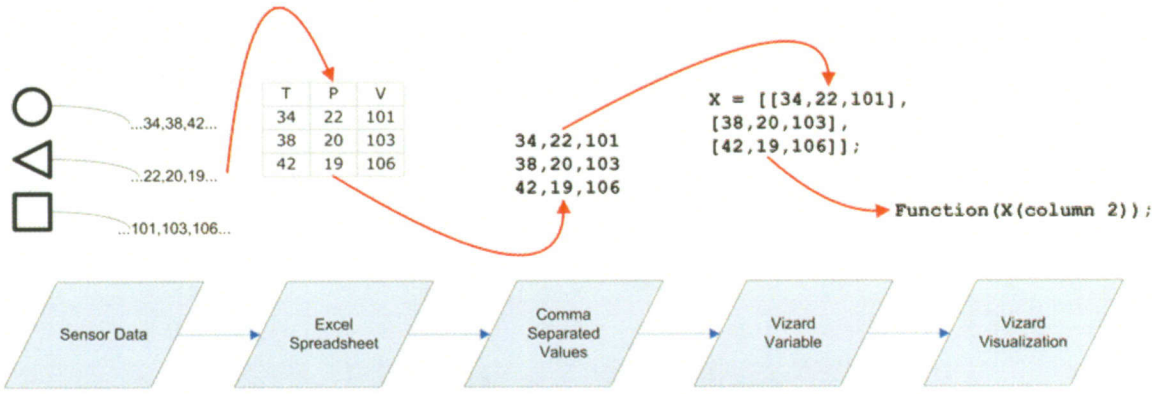


Figure 4-19: Integration of E1 list data into Vizard®.

Data Registration

An important feature of Vizard® is the ability to register data to objects. This allows data to alter the environment in real time. Four types of data were used in the NASA E1 test stand visualization: valve command, valve position, temperature and pressure. This data was collected over a period of time from sensors along the pipes or from control information in the case of the valve. The data was then stored in a CSV file and imported into a variable in Vizard®. The variable is a matrix, in which each column corresponds to a known sensor, and each row is a new time step. The environment runs in a loop, loading a new time step with each iteration. How fast the loop runs is set by the user.

The simplest example of data registration in the E1 test stand visualization is the visualization of temperature. With a change in temperature, the color of the corresponding pipe valve is changed. An intuitive indication of red for hot and blue for cold was used. Vizard® includes a built Python script for changing the color of an object, with input corresponding to the levels of red, green and blue. In this case, green was set

to zero, red was set proportional to the temperature data and blue was set inversely proportional. The color levels were scaled to the values of the input to ensure a full range of red to blue was used, and a noticeable change could be seen. The outcome of this is shown in Figure 4-20.

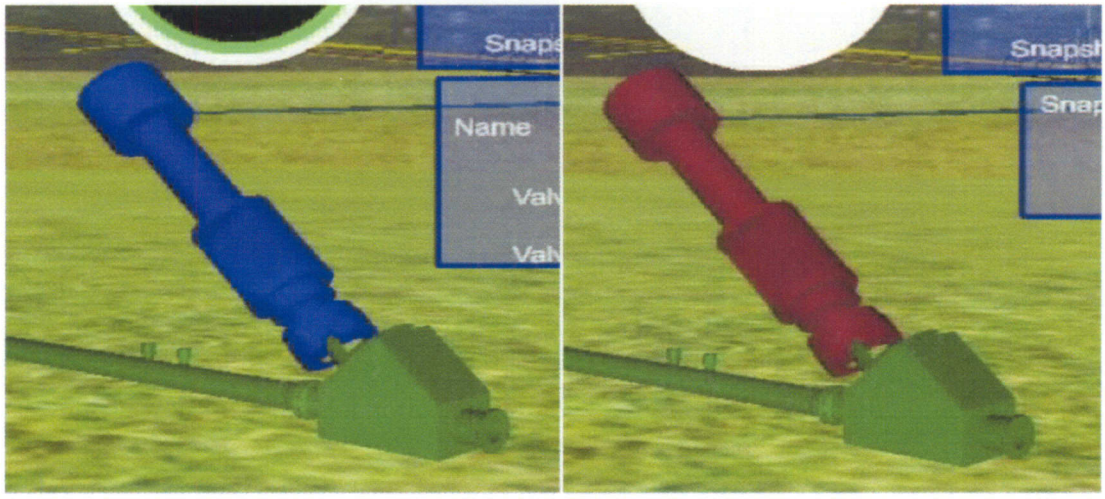


Figure 4-20: Temperature visualized by color change.

The visualization of the valve command and value position was registered to an indication model. This model was designed in a 3D modeling platform separate from the graphical data and imported into Vizard®. The indication model consists of three spheres. The first is a white sphere that never changes. A black sphere changes size based on value position data. When the valve is closed, the sphere has a size of zero. When the valve is fully open, the black sphere is set to the same size as the white sphere. This is done by registering the data to the input of a model scaling Python script command. A third green sphere, registered to the command data, is scaled to zero for command closed and the size of the white sphere for command open. The models were designed with

inverted normals in a way that all the spheres are never blocked by each and can be viewed from all angles. The resulting indicator model is shown in Figure 2-7.

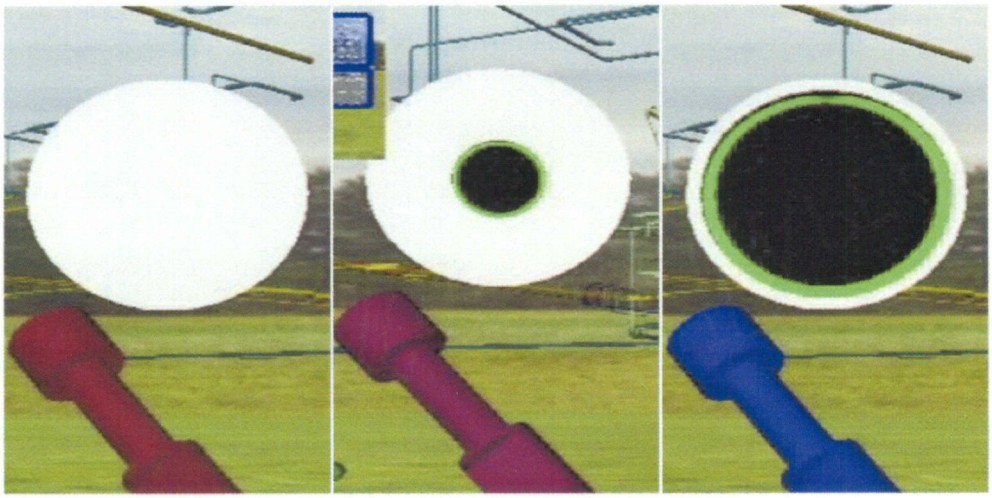


Figure 4-21: Vizard® valve command and position indicator.

The final type of data registered indication is pressure. This was visualized using the data as the input of a more complex scripted function. A Python script was written to make the valve rotate back and forth, within a constrained range of degrees. The pressure data from the sensor is added as in input to the function, changing the rate of rotation and the range of angle. The result is a value that vibrates slowly when the pressure is low and fast when the pressure is high.

Advanced Interaction

Interaction in the E1 test stand visualization was enhanced using Vizard®'s built-in menu functions. This gives the user the ability to have greater control over the environment. The main menu, shown in Figure 4-22, gives the user the ability to check time data, control the playback and access sub-menus.

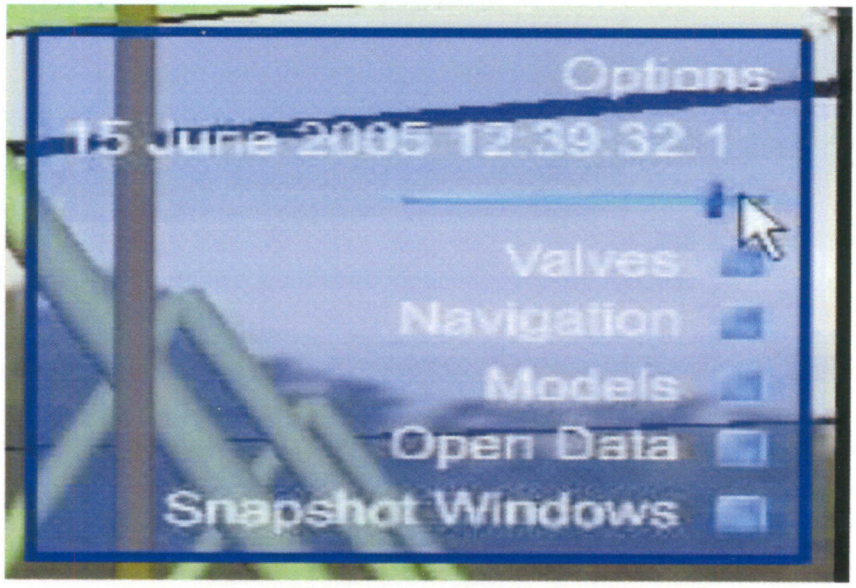


Figure 4-22: Vizard® E1 test stand main menu.

The first sub-menu is the value menu. This allows the user to view numerical data about each valve. This allows for more quantitative viewing of data, compared with qualitative visualization of the visual indicators.

Figure 4-23: Vizard® E1 test stand valve menu.

The snapshot menu allows the user to add up to 3 extra views for viewing multiple points of interest at one time. The snap shot windows show up at the top of the screen, as shown in Figure 4-24. The models menu, shown in Figure 4-25, allows the user to turn parts of the 3D model off and on.

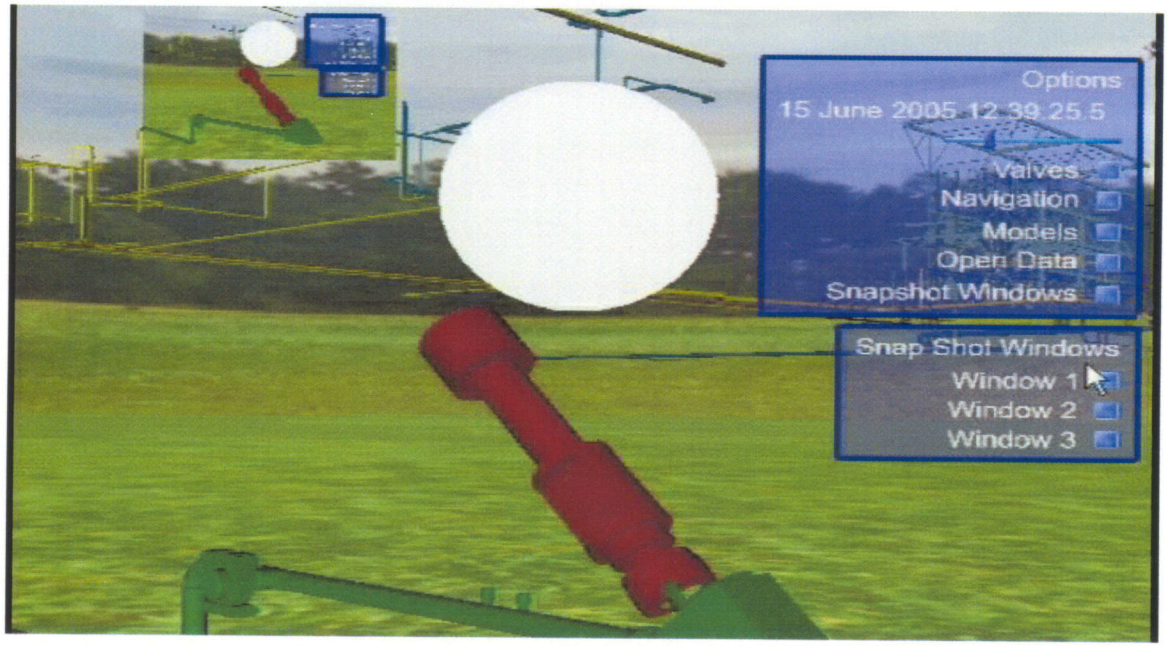


Figure 4-24: Vizard® E1 test stand snap shot menu.

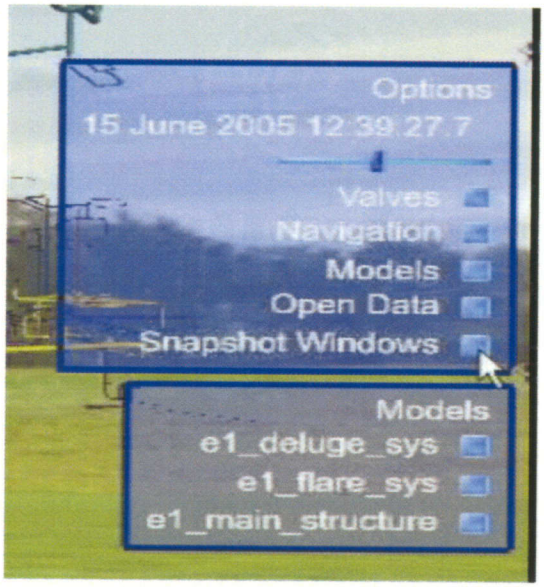


Figure 4-25: Vizard® E1 test stand models menu.

Advanced Navigation

A new feature in the Vizard® visualization is the ability to automatically "fly to" a preset point of interest. This was accomplished using point of view script commands. It is trigger by the check boxes in the valve menu in Figure 4-23. When selected, the view of the world will automatically navigate to a particular valve. This saves the user from having to search for values inside the virtual environment.

4.3.2 E-3 MTTP Trailer

The E-3 MTTP Trailer is another test trailer used by NASA. It is much smaller than the E1 test stand, as can be seen in Figure 4-26. The purpose of visualizing this trailer was to use lessons learned from the previous two E1 test stand virtual environments and create a new virtual environment. Since the E-3 MTTP Trailer is much smaller than the E1 test stand, creating a complete environment is much simpler. Again, this virtual environment includes 3D data, this time taken from real world data and list data, taken from sensor readings.

Figure 4-26: NASA E-3 MTTP trailer. (E1 test stand in background.)

Figure 4-27: Reference images for the NASA E-3 MTTP trailer.

3D Modeling

The previous two NASA visualizations had issues with visualizing 3D model data. The original Solidworks® models had many problems, which lead to poor images and poor performance. In the Vizard® visualization, an attempt was made to fix some of these issues, but only created more issues. While the models looked and performed better, color data was lost. To overcome these problems, the NASA E-3 MTTP Trailer was modeled from reference images, like the ones shown in Figure 4-27. While this requires extra time and skill, it results in a far superior visualization, shown in Figure 4-28. Having this level of control over the model, as well as having the knowledge of how it was modeled, had many advantages.

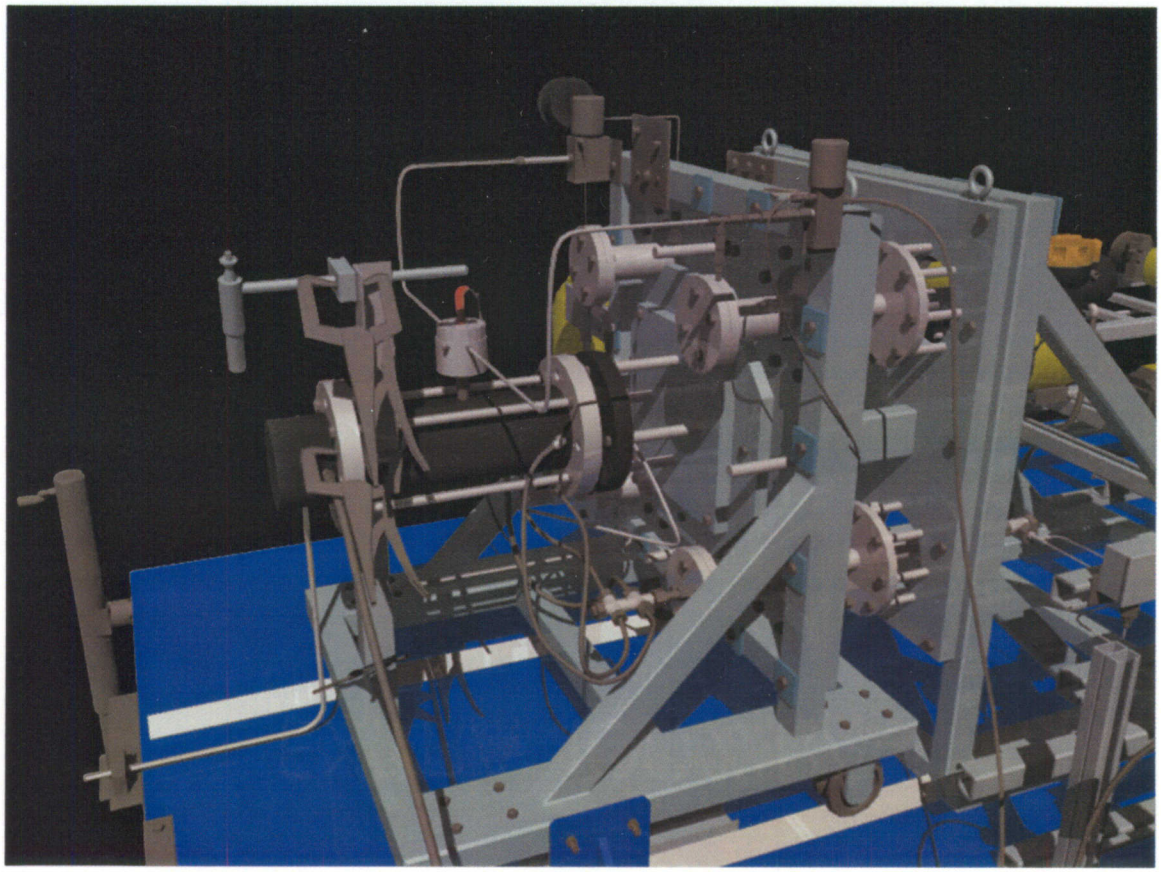


Figure 4-28: Fully rendered NASA E-3 MTTP trailer [7].

Greater control over the modeling process allows the visualization to be optimized. In Figure 4-29, the piping has a high polygon count, giving the full detail of the system, while in Figure 4-30, the tires have a much lower polygon count. This optimized model has the most 3D modeling data where important, while reducing the polygon count on less important models. This differs greatly from the vGeo® models, which had excess 3D modeling in all models.

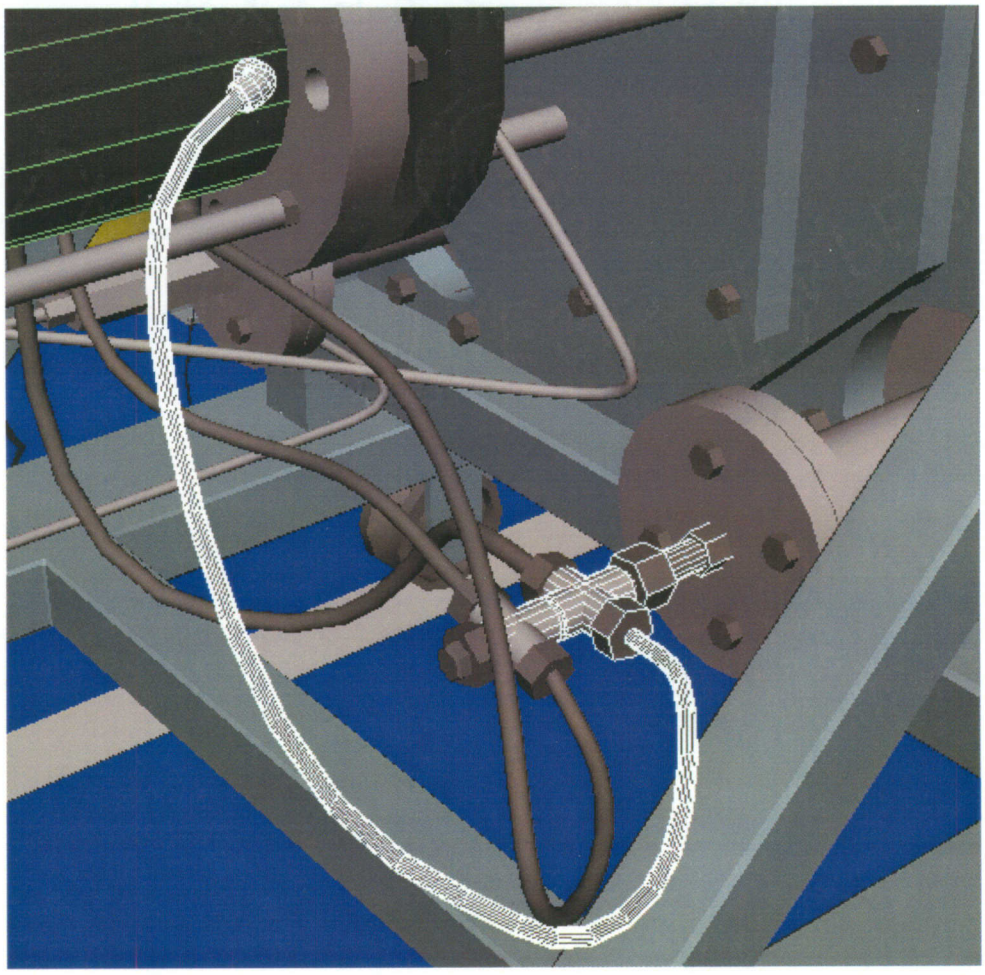


Figure 4-29: High detail piping system, NASA E-3 MTTP trailer in Vizard® [7].

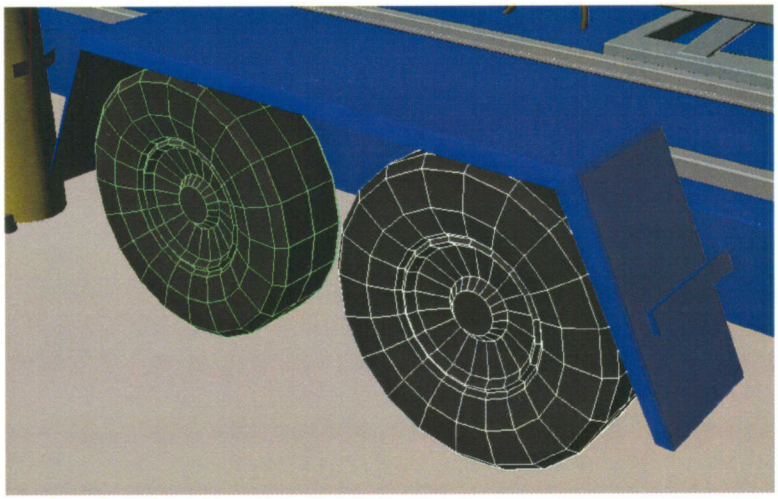


Figure 4-30: Low detail tires, NASA E-3 MTTP trailer in Vizard® [7].

Visualization of List Data

One of the goals of the NASA E-3 MTTP Trailer visualization in Vizard® was to streamline the integration of data. Data identifiers are all called with a common piece of Python script. Only four fields are required to define an identifier. First is the name of the identifier, followed by the type. The type tells which indicator model to load, which in this case, only one model is available. The position field determines where the model will be located, most likely over the valve of which it is displaying data. The final field is the data select field, which tells the indicator which data sources to draw from. Like the E1 test stand visualization, these are drawn from known column in a variable loaded from a CSV file. Because of this, new sensor data visualization can be added quickly and easily.

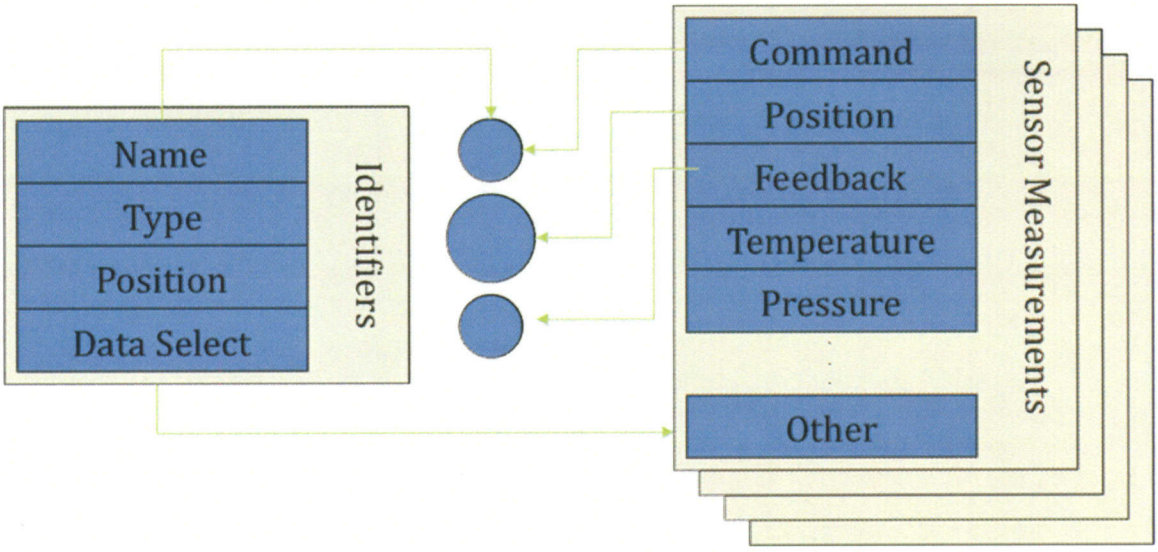


Figure 4-31: Configuration of an identifier, NASA E-3 MTTP trailer in Vizard®.

Communication with NASA allowed for better understanding of the virtual environment being created. Coupled with better knowledge the models allowed for a common framework for component recognition. Once the model was complete, the data had to be registered to the valves. However, it was unknown to which valve the data was supposed to be registered. Because the model was more accurate, an image of the valves could be understood by both the developer and NASA was created. This image, shown in Figure 4-32, opened a clear communication channel for registering the data. With all the data for each valve registered correctly, the NASA E-3 MTTP Trailer visualization is more complete.

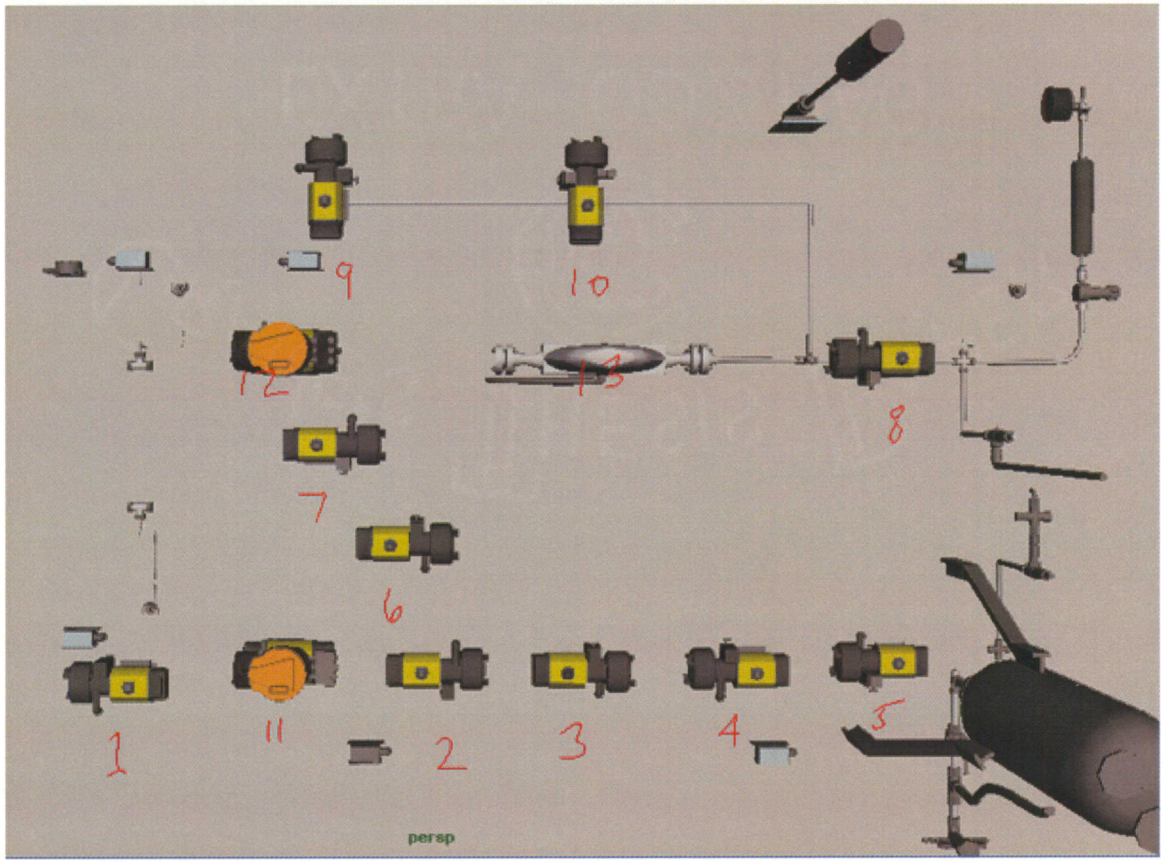


Figure 4-32: Identification of valves, NASA E-3 MTTP trailer in Vizard®.

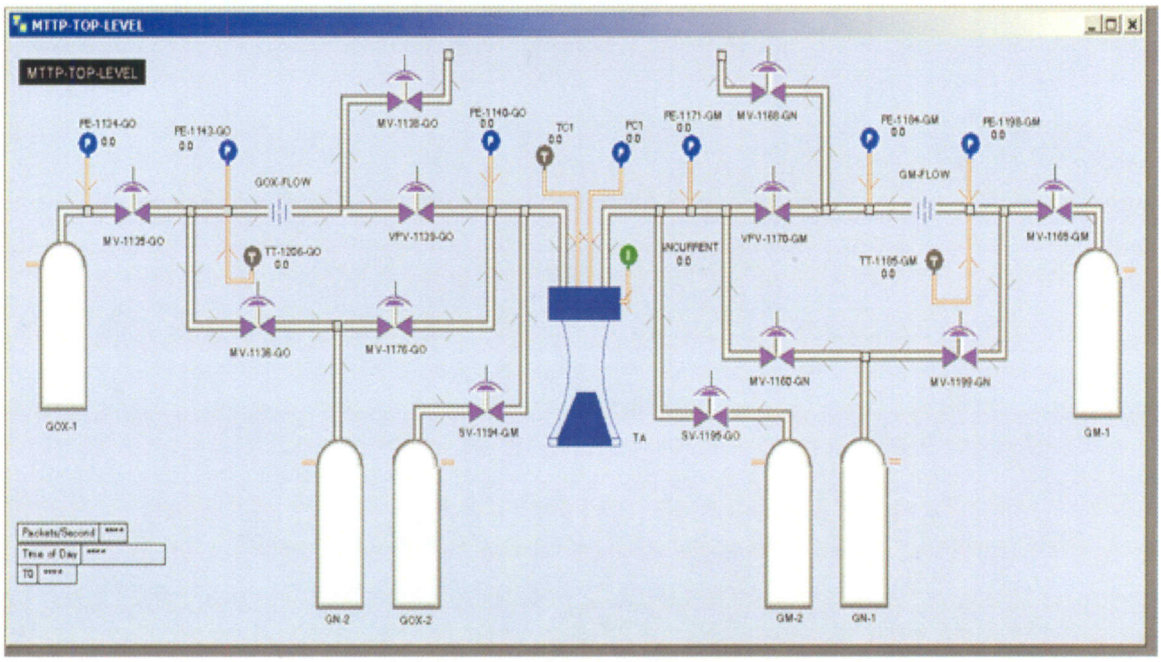


Figure 4-33: Example of data from NASA.

For this virtual environment, a data indicator for each was modeled, shown in Figure 4-34. For each valve, a new instance of the model loaded. The indicator consists of three spheres, which change in color, from red to green. The largest sphere in the middle indicates the position of the valve, where green is open and red is closed. If the valve is between those states, the sphere is a shade between red and green and the outer rings rotate. These rotating rings give an easy to recognize indication that the valve is in motion. The top sphere indicates that the command has been given to the valve: red if commanded to close, green if commanded to open. Some valves do not report position, but instead, trip a limit switch when the valve is fully open or fully closed. This is indicated on the bottom sphere, as red for closed and green for open. The indicators can be seen in action in Figure 4-35.

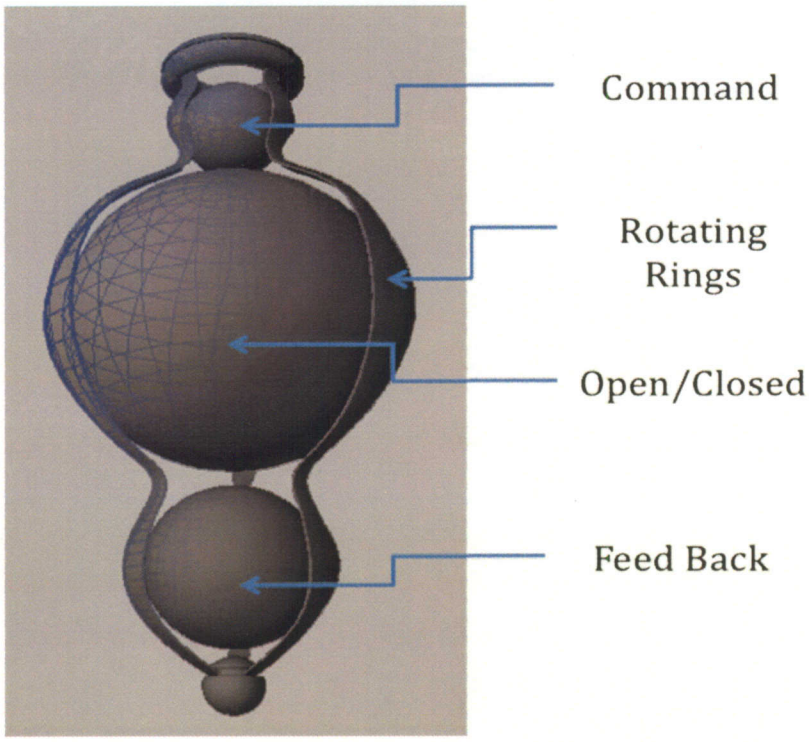


Figure 4-34: NASA E-3 MTTP trailer data indicator [7].

Figure 4-35: NASA E-3 MTTP trailer data indicators in action [7].

4.4 Pathway Comparison

In this thesis, three pathways are implemented using two different virtual environment platforms. The pathways implemented are 2D data to surface plot visualization, 3D data to particle visualization and a realistic environment for showing both 3D models and sensor list data. These pathways are implemented in both vGeo® and Vizard®. The pathways are evaluated on the criteria of ease of design, accuracy, efficiency, I/O framework, user interaction, and software integration.

The evaluation criteria were broken into sub-criteria. For each pathway and software combination, is given the numerical value. A valve of zero means criterion cannot be met, while a value of ten means the pathway meets the criterion perfectly. A value of one to three corresponds to below average. Values between four and six correspond to an average ability to meet the criterion. Finally, a value of seven though nine corresponds to a pathway with above average ability to meet the criterion. The designers of the virtual environments assigned these values.

4.4.1 Ease of Design

The ease of design criterion was broken down into five parts: learning curve, user interface, design structure, setup and built-in functions. The learning curve evaluates how easy a new user can create an environment using the pathway. The interface criterion measures the quality of the user interaction with the design pathway. The design structure is a measure of the complexity of the design processes inherent to the pathway. Setup refers to the initial costs of the pathway. Finally, the built-in functionality criterion measures the availability of 'right out of the box' functionality.

Table 4-2: Evaluation of ease of design of the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Easy Learning Curve	6	4
Interface	5	6
Design Structure	4	5
Setup	4	4
Built-in Functionality	9	1
Average	**5.6**	**4**

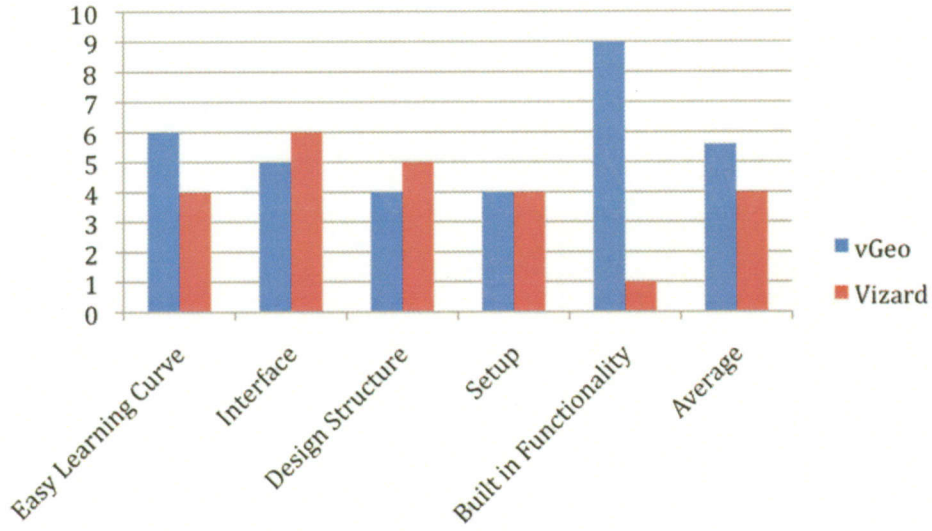


Figure 4-36: Evaluation of ease of design of the surface plot pathway.

vGeo® was the pathway of choice for easy visualization of 2D data as a surface plot. The main reason for this is the built-in surface plot functionality in vGeo®. vGeo® required advanced scripting to register the data to the vertices on a grid. vGeo® also easier learning curve in this pathway, as a surface plot is one of the simplest functions of vGeo®. Vizard® is better in terms of interface and design structure. The graphical interface and Python scripting of Vizard® is easier to use than vGeo®'s text configuration file, however in this case it does not add much to this pathway. Setup of each pathway is similar.

Table 4-3: Evaluation of Ease of Design of the Particle Visualization Pathway

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Easy Learning Curve	4	6
Interface	2	8
Design Structure	4	8
Setup	4	4
Built-in Functionality	5	5
Average	**3.8**	**6.2**

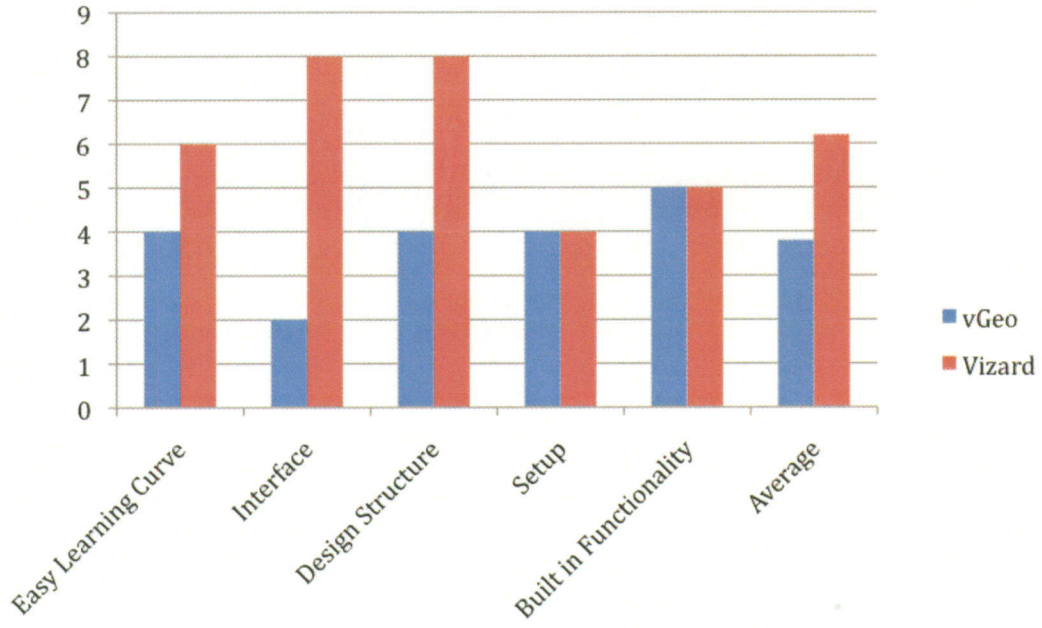


Figure 4-37: Evaluation of ease of design of the particle visualization pathway.

Vizard® was the easier to design pathway for the visualization of particle aggregates. The main reason for this was the graphical interface and the Python scripting. Both pathways required the processing of raw data to a platform compatible format. However, Vizard® model loading is simpler than vGeo®'s configuration file. Both paths had similar set up and the built-in functionality did not add much ease to the pathway.

Table 4-4: Evaluation of ease of use of the NASA pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Easy Learning Curve	5	8	3
Interface	2	8	8
Design Structure	2	8	8
Setup	4	4	4
Built-in Functionality	2	7	7
Average	**3**	**7**	**6**

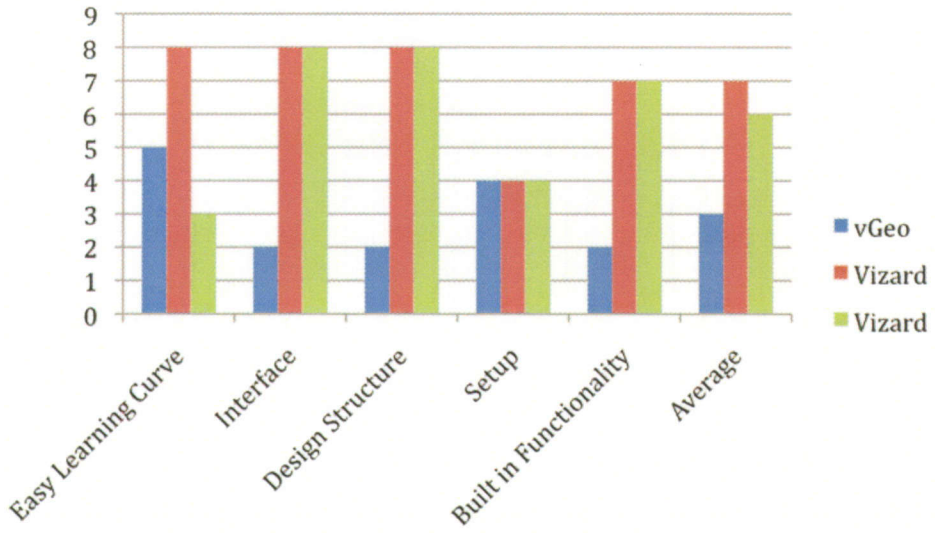


Figure 4-38: Evaluation of ease of design of the NASA pathways.

Overall, Vizard® was the choice for easy visualization of NASA test stands. The user interface allowed for easy importing of models, and the scripting ability allowed an easy implementation of data visualization and user interaction. The built-in functionality of Vizard®, such as a single line of script to change the color of an object, made the overall pathway much easier. However, the need to model the MTTP trailer from images required someone with 3D modeling skills, greatly increasing the learning curve of such a pathway.

4.4.2 Accuracy

The accuracy criterion was divided into two sub-criteria: data retention and quality of the environment. Data retention is a measure of the comparison of data in to data out and is measured by two criteria: data loss and data corruption. Quality of the environment is defined by three criteria: visually appealing, ease of understanding and ease of interpreting data. Visually appealing environments are both aesthetically pleasing and display data in high detail. Ease of understanding measures how intuitive the environment is and how easily the user can be immersed. Ease of understanding data is a measure of how clearly the raw data has been visualized.

Table 4-5: Evaluation of accuracy of the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Data Retention	9	9
-Low Data Loss	9	9
-Low Data Corruption	9	9
Environment Quality	5	5
-Visually Appealing	5	5
-Ease of Understanding	5	5
-Ease of Interpreting Data	5	5
Average	**6.6**	**6.6**

Accuracy did not play a factor in the surface plot pathway. No data was lost or corrupted in either process. In terms of quality, this visualization is average. It shows what it has to show simply and easily, therefore the visual appeal, ease of understanding and interpreting data are given an average score. The surface plot visualization is very straight forward, and does not put much strain on the system.

Table 4-6: Evaluation of accuracy of the particle visualization pathway,

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Data Retention	8	8
-Low Data Loss	8	8
-Low Data Corruption	8	8
Environment Quality	4.67	6.33
-Visually Appealing	5	7
-Ease of Understanding	5	6
-Ease of Interpreting Data	4	6
Average	**6**	**7**

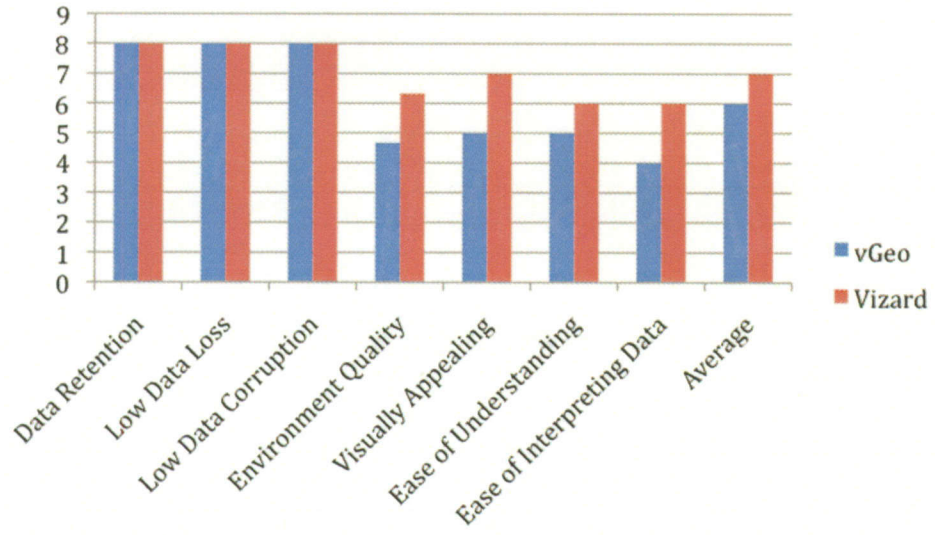


Figure 4-39: Evaluation of accuracy of the particle visualization pathway.

Vizard® was slightly more accurate than vGeo® for the particle visualization pathway. Both did equally as well in terms of data retention. Vizard® allowed for a visual pleasing word, with backgrounds, animations and sound effects. Vizard®s interaction and model selection was superior to vGeo®, which lead to easier interpretation of the data.

Table 4-7: Evaluation of accuracy of the NASA model pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Data Retention	3*	5*	8*
-Low Data Loss	1	5	8
-Low Data Corruption	5	5	8
Environment Quality	4	4	4
-Visually Appealing	4	5	8
-Ease of Understanding	4	6	7
-Ease of Interpreting Data	2	4	7
Average	**3.2**	**5**	**7.6**

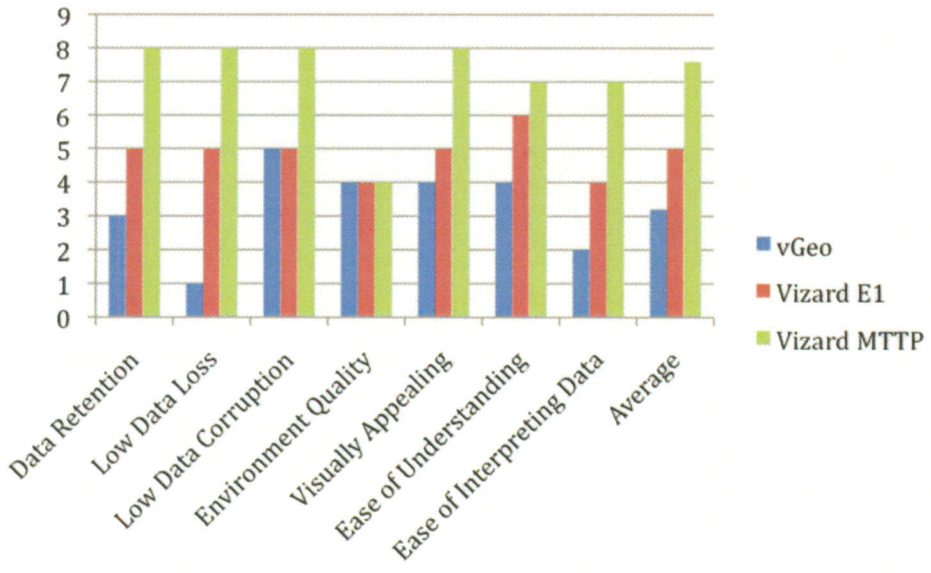


Figure 4-40: Evaluation of accuracy of the NASA pathways.

The MTTP pathway was superior E1 in terms of accuracy. vGeo® was unable to display the sensor data, essentially losing it in the pathway. Both Vizard® pathways retained the sensor data. For the E1 test stand, in both vGeo® and Vizard®, data was lost or corrupted in the modeling and conversion process. Specific data retained by file formats is shown in Table 4-8. Because the modeling was done 'in house' for the MTTP trailer, much more of the intended data was retained.

Table 4-8: Evaluation of file formats for data retention.

	vGeo®	Vizard®	flat model	texture	material	advanced geometry	retaining data	face normals	vertex normals	animation
wrl	x	x	5	2	2	4	4	4	2	2
obj+mtl		x	5	4	3	3	2	3	3	0
3ds		x	5	5	4	4	2	4	5	0
osg/ive		x	5	3	4	5	4	4	5	3
ma/mb			5	5	5	5	5	5	5	5
fbx			5	5	5	4	3	5	5	3
stl			5	0	1	5	5	1	0	0

Table 4-9: Scale used for evaluation of file formats for data retention.

0: Not at all
1: sometimes/ not reliable
2: limited ability
3: strong ability
4: full ability/ limited reliability
5: full ability

Each file format was evaluated on a variety of criteria. The flat model is how well the format stores the basic model. Texture refers to formats ability to use texture mapping and how well it transfers from one platform to another. Materials include such things as lighting and reflections. Sometimes tricks can be done in a 3D modeling platform to make complex models for efficiently. However, not all file formats understand such modeling. Advanced geometry is a measure of how well a format handles this. Retain information refers to how well the file format retains all information stored in the model. Face normals give information about the faces of the polygons, such as the direction they are facing. Vertex normals give information for smoothing edges. These measures tell how well a format understands and retains this information. Animation refers to if the format can store animations.

The Maya format works well in all area, however it is not compatible with either virtual environment platform. Of the others, no one format is best for everything. VRML is an older format and has not been updated since 1997. This tends to make VRML unacceptable for newer applications. The trouble with the E1 test stand is proof of this. Object and 3Ds work well for most applications. OSG is the closest thing to a native format in Vizard®, making it ideal. However, not all modeling platforms support the exporting of OSG files. A drawback of the greater file support of Vizard® is the increased design cost of determining which file format to use.

4.4.3 Efficiency

Efficiency was measured during the design and at run time. Design efficiency was broken down into how a pathway could be used to design complex model, enhanced interaction and advanced data display methods. Runtime efficiency measured the amount of data and the amount of detail that could be integrated into the world before some limitation, like frame-rate, took over.

Table 4-10: Evaluation of efficiency of the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Design	n/a	n/a
-Complex Models	n/a	n/a
-Enhanced Interaction	n/a	n/a
-Advanced Data Display	n/a	n/a
Run-time	5	5
-Amount of Data	5	5
-Amount of Detail	5	5
Average	**2**	**2**

Efficiency is not much of an issue in the visualization of surface plots. A surface plot alone has little design complexity. Therefore, this pathway cannot be used to display complex models, enhanced interaction or advanced data display. In terms of run-time, not much is displayed in terms of data or detail. The visualization puts an average amount of stress on the system. However, 2D data as surface plots could be part of a more complex visualization, for which another evaluation is necessary.

Table 4-11: Evaluation of efficiency of the particle visualization pathway.

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Design	2.67	8.67
-Complex Models	4	8
-Enhanced Interaction	2	9
-Advanced Data Display	2	9
Run-time	5	5
-Amount of Data	5	5
-Amount of Detail	5	7
Average	**3.6**	**7.6**

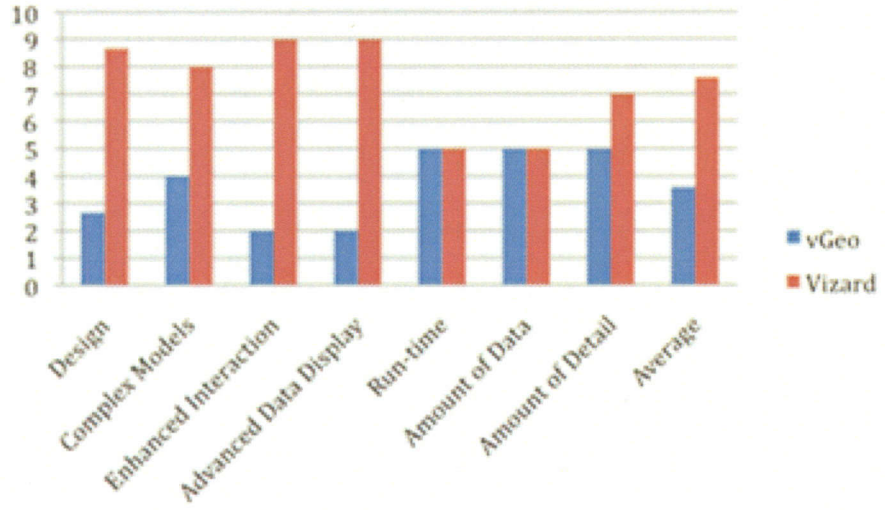


Figure 4-41: Evaluation of efficiency of the particle visualization pathway.

The use of Vizard® in the particle visualization pathway is far more efficient. Vizard®s user interface and scripting abilities allow for more complex model and the design of much better interaction and data display. Vizard® allowed for many enhancements over vGeo® including, motion sensor control of particles, onscreen particle selection and scale manipulation with the use of additional inputs. At run-time, both pathways were similar. The amount of data displayed in each was the same, but Vizard® allowed for extra detail, both functional and aesthetic.

Table 4-12: Evaluation of efficiency of the NASA model pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Design	3.33	7	8
-Complex Models	5	5	8
-Enhanced Interaction	2	8	8
-Advanced Data Display	3	8	8
Run-time	2.5	6.5	7.5
-Amount of Data	3	7	8
-Amount of Detail	2	6	7
Average	**3**	**6.8**	**7.8**

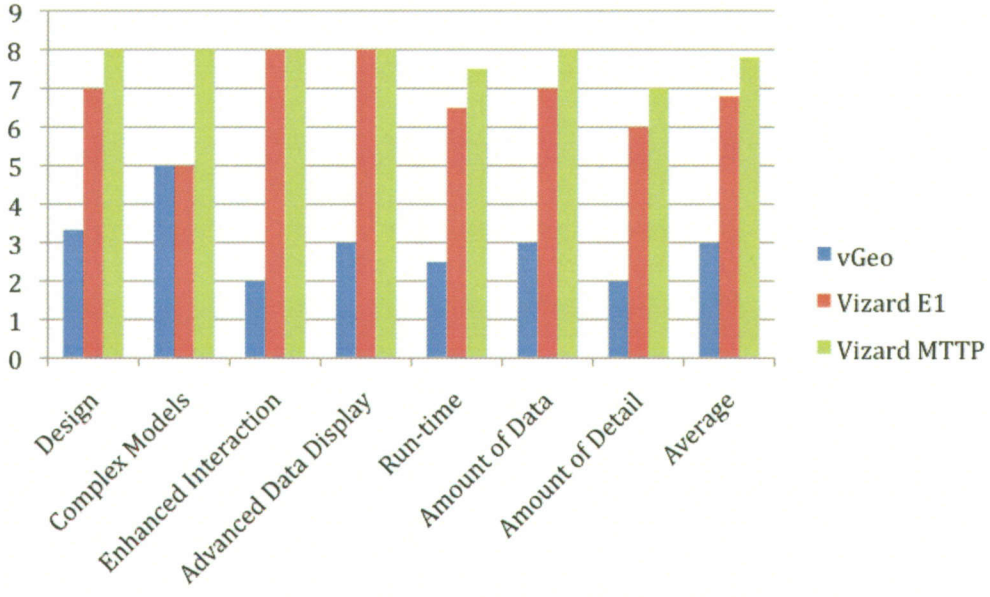


Figure 4-42: Evaluation of efficiency of the NASA pathways.

Vizard® was able to make a much more complex environment more efficiently than vGeo®. For the E1 test stand, both vGeo® and Vizard® have average VRML models, while the 'in house' modeled MTTP environment had a more complex model. vGeo® had no interaction and no data display, while Vizard® allowed for onscreen menus and Python scripted visualization. vGeo® reliance on VRML lead to a low detailed world and the simple data display allowed nothing more the model data on this pathway. The E1 test stand allowed for greater detail, even with the use of VRML, when the Object format did not work. Vizard® was also able to use the scripting allowed for the sensor data to be visualized. The more efficiently designed MTTP trailer allowed for more data and detail than the Vizard® E1 test stand.

4.4.4 I/O Framework

The I/O Framework has been broken down into two criteria: Concept to Environment and Lack of Restrictions. Concept to Environment is a measurement of how easily a concept for an environment can be implemented. Lack of Restrictions refers to how little the pathway hinders the development process.

Table 4-13: Evaluation of I/O framework of the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Concept to Environment	5	5
Lack of Restriction	5	4
Average	**5**	**4.5**

Again, the visualization of a surface plot was simple. The pathway from concept to environment was average for both software packages. Vizard® had a restriction, requiring the surface plot to be programmed. Otherwise, the framework of visualizing a surface plot was very straightforward.

Table 4-14: Evaluation of I/O framework of the particle visualization pathway.

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Concept to Environment	2	6
Lack of Restriction	2	7
Average	**2**	**6.5**

Vizard® allowed for much more design options than vGeo®. The goal was an immersive world, in which the user could intuitively compare two particles. vGeo® did lead to an immersive world, and the method for moving the particles was clumsy. Vizard®, on the other hand, allowed for a more immersive environment. The Vizard® based environment had the desired intuitive controls. The vGeo® pathway had many restrictions. The platform is limited to just one method of moving objects in real-time. Vizard® has the ability to script any interaction required. It was also not possible to create the intuitive on-screen particle selection in vGeo®. Vizard®s scripting ability and open-ended design structure allowed an environment to be created as intended.

Table 4-15: Evaluation of I/O framework of the NASA model pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Concept to Environment	2	6	7
Lack of Restriction	2	7	8
Average	**2**	**6.5**	**7.5**

Again, Vizard® allowed for much more design options than vGeo®. The goal was an immersive model of the test stand with visualization of sensor measurement throughout. Restrictions in vGeo® allowed only for a visualization of the test stand. The main restriction causing this was the inability to change object during run-time. Vizard®'s scripting ability allowed for this data visualization. Knowledge of the model in the pathway for the visualization of the Vizard® E1 was lacking, leading to an inaccurate data visualization. The 'in house' modeling of the MTTP Trailer allowed the data to be displayed accurately and in the correct location.

4.4.5 Interaction with User

The user interaction was divided into three parts. These are a measure of how well the environment meets the interaction, immersion and navigability criteria.

Table 4-16: Evaluation of interaction with user for the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Immersion	5	5
Interaction	2	2
Navigation	5	5
Average	**4**	**4**

The surface plot visualization is a simple visualization. There is little interaction in this environment. Immersion and navigation was average in this pathway. Navigation is the key to a virtual environment such as this, allowing the user to easily view the data from any angle. The required navigation was there, but nothing special was implemented.

Table 4-17: Evaluation of interaction with user, particle visualization pathway.

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Immersion	5	6
Interaction	4	7
Navigation	5	7
Average	**4.7**	**6.7**

The Vizard® pathway produced a more immersive, interactive and navigable environment. Vizard®'s greatest improvement was in interaction. The two sensor controls and the floor sensor abilities added interaction which vGeo® lacked. The enhanced environmental elements added to the immersion of the Vizard® visualization.

Table 4-18: Evaluation of I/O framework of the NASA model pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Immersion	4	7	7
Interaction	2	8	8
Navigation	3	8	7
Average	**3**	**7.7**	**7.3**

Again, the Vizard® pathway produced a more immersive, interactive and navigable environment. The restrictions of vGeo® created only a static model of the E1 test stand, lacking any interaction with the sensor data. The vGeo® pathway also lacks the ability to create interactive menus. The graphical abilities of Vizard® allowed much more aesthetically pleasing and immersive environments. The automatic navigation of the E1 test stand visualization was not repeated in the MTTP trailer visualization, giving the E1 test stand slightly better navigation.

4.4.6 Software Integration

The software integration criteria were divided into two parts. Some pathways include software that can be easily updated. However, such updates tend to break previous models. Software integration is measured on the criteria of available updates, and lack of broken environments.

Table 4-19: Evaluation of software integration for the surface plot pathway.

	Surface Plot	
Criterion	**vGeo®**	**Vizard®**
Available Updates	5	5
Lack of Broken Environments	5	5
Average	**5**	**5**

Table 4-20: Evaluation of software integration for particle visualization pathway.

	Particle Visualization	
Criterion	**vGeo®**	**Vizard®**
Available Updates	4	7
Lack of Broken Environments	4	3
Average	**4**	**5**

Table 4-21: Evaluation of software integration of the NASA model pathways.

	NASA E1		NASA MTTP
Criterion	**vGeo®**	**Vizard®**	**Vizard®**
Available Updates	4	7	7
Lack of Broken Environments	4	5	5
Average	**4**	**6**	**6**

Software updates had little to do with the surface plot and did not lead to any broken environments. For the particle and NASA visualization, vGeo® did not include much in terms of updates. However, Vizard® is often updated, giving fixes and new functionality to the environments. However, sometimes these updates break previous environments and require more development to fix. An update to Vizard® completely

broke the particle visualization and the environment script had to be adjusted. Similar results happen to the NASA visualization, but much less severe.

4.5 Summary of Results

In nearly every case Vizard® was the better platform. However, a notable exception is the surface plot visualization, in which vGeo® performed better overall because of the built-in functionality. vGeo® also suffered less from broken visualization because of software updates. Overall, the open-end nature of Vizard® was better for most situations, especially as the environment become more complex and interactive.

File format became a big issue with the integration of 3D model data. With many possibilities for data loss and corruption along the path, the best file format for the situation was evaluated. No one format was perfect for every situation, making file format selection an extra layer of the 3D model pathway. As result of this, the MTTP trailer visualization was developed with models designed in-house. The extra-time required for modeling the trailer led to higher quality visualization in the end.

Chapter 5 - Conclusions

This focus of this thesis has been to study and recommend optimal techniques for developing immersive virtual environments for a variety of applications. The overarching objective is to ensure that virtual environments can be created and deployed, rapidly and accurately using commercial off-the-shelf software. Specific subjective and objective criteria have been employed to determine trade-offs between multiple pathways for designing such environments and specific recommendations are made.

5.1 Summary of Accomplishments

The major accomplishments of the research work described in this thesis are:

1. *Examination of current algorithms and software used to design, develop and deploy virtual environments.*

This thesis has addressed the evaluation of the best overall approaches to create virtual environments. The value of open, object-oriented systems has been identified, with the ability to easily integrate a variety data, inputs and hardware. Although it could be argued that dedicated custom-built software has some advantages over open-ended systems that require much more programming to reach the same goal, generic data for visualization is much more suitable for visualization using the methods described in this thesis.

2. *Definition of a general method for developing virtual environments.*

A method for integrating graphical, measurement and functional data into a generic virtual environment has been developed. Graphical data were treated as 3D models needing to be processed and incorporated into the virtual environment. Measurement and functional data were treated like raw data, which required formatting for visualization.

3. *Expansion of on these general methods to methods applicable to common inputs and situations.*

Data formats were list data, 2D data, 2D images, 3D data, 3D models and real life data. Each data format proved to have different requirements for integration. 2D and 3D data were treated like raw data for visualization. 3D models and 2D images were loaded into the platforms. List data had a more diverse pathway. It was treated similarly to raw data, but could be applied to a variety of new visualizations. The most interesting of these was the use of list data as the input to scripted functions. This allowed sensor data to be displayed in an environment model at the point at which it was recorded.

4. *Application these methods to actual examples, including different types of datasets and applications.*

These methods were exercised on three different applications, covering all of the data formats. The development pathway for incorporating 2D data in a virtual environment used surface plots. The development pathway for visualizing 3D data was demonstrated with reconstructions of geomaterial (sand) particles. The simulation of the NASA test stand demonstrated the application of the visualization of the following data types – 1D

sensor data, 2D image data and 3D model data. 3D model data turned out to be the most complex of all data formats, relying greatly on file formats. Each 3D modeling and 3D visualization platform handled the file formats differently. Different source data also affected the outcome. All compatible file formats were tried for each model. A comparison of the formats was recorded for different application based on experience. The formats were compared on their perceived reliability. No one format was ideal for every situation.

5. *Application of measurement criteria to evaluate the effectiveness of these methods in increasing the value of VR while reducing development cost.*

In most cases, the use of Vizard® was the best choice. There were only two cases in which vGeo® was the better choice. vGeo®'s built-in surface plot functionality was much more efficient than Vizard®'s open end design. However, once the environments got more complex, vGeo® was not able to keep up. Using Vizard® to create an interactive environment for the E1 test stand took less than half the time of the model only visualization in vGeo®. The other case was in the area of software updates. Vizard® had the advantage of regular updates, but these updates could lead to loss of functionality in visualization. vGeo® on the other hand was less susceptible to problems caused by these updates.

5.2 Recommendations for Future Work

The approaches described in this thesis could be further expanded and refined by applying them to applications that are more diverse, such as computational fluid dynamics, biocomputing, architecture and design. These approaches and methods could also be expended to handle fully immersive environments. Deployment of virtual environments for CAVEs adds many new development complexities to be analyzed. Most of all, these methods could be applied to other virtual environment platforms. If these approaches could be changed to include other platforms, results that are more diverse could be acquired. Ultimately, the expansion of these methods and the resulting evaluations could be used to develop a framework for an ideal virtual environment design platform.

Works Cited

[1] T. Hong, A Virtual Environment for Displaying Gas Transmission Pipeline NDE Data. Iowa State University M.S. Thesis. 1997.

[2] S. Papson, J. Oagaro, R. Polikar, J. C. Chen, J. L. Schmalzel. "A virtual reality environment for multi-sensor data integration," *Sensors for Industry Conference*. January 27-29, 2004.

[3] G. Stephen. M. Handzic, "Knowledge discovery through visualising using virtual reality," *Proceedings of the 37th Annual Hawaii International Conference on System Sciences, Honolulu, Hawaii*, 2004.

[4] K. Mania, D. Wooldridge, M. Coxon, A. Robinson, "The effect of visual and interaction fidelity on spatial cognition immersive virtual environments," *IEEE Transactions On Visualization And Computer Graphics*, Issue 3, Vol. 12, pp. 396-404, May/June 2006.

[5] A. van Dam, A. Forsberg, D. Laidlaw, J. La Viola and R. Simpson, "Immersive VR for scientific visualization: A progress report," *IEEE Computer Graphics and Applications*, pp. 26-52, November-December 2000.

[6] P. Giordano Jr., D. Barrot, P. Mease, K. Garrison, S. Mandayam, B. Sukummaran., "An optical tomography system for characterizing 3D shapes of particle aggregates." *Proceedings of the 2006 IEEE Sensors Applications Symposium*, pp. 125-127. 2006.

[7] G. Lecakes Jr, J. Morris, J. L. Schmalzel, S. Mandayam. "Virtual reality platforms for integrated systems health management in a portable rocket engine test stand," *IEEE International Instrumentation and Measurement Technology Conference*, Victoria, Vancouver Island, British Columbia, Canada, May 2008.

[8] S. Papson, An Investigation of Multi-Dimensional Evolutionary Algorithms for Virtual Reality Scenario Development. Rowan University M.S. Thesis, 2004.

[9] J. Bram, A "Divide-and-Conquer" Strategy for NDE Signal Inversion in Gas Transmission Pipelines. University M.S. Thesis, 2005.

[10] D. A. Bowman, D. B. Johnson, L. F. Hodges, "Testbed evaluation of virtual environment interaction techniques," *Presence*, Vol. 10 Issue 1, : The MIT Press, Cambridge, Massachusetts, pp. 75-95. 2001.

[11] D. Bowman, A. Chadwick, "Design and evaluation of menu systems for immersive virtual environments," *IEEE Virtual Reality Conference*, Yokohama, Japan, pp. 149-156, 2001.

[12] D. Bowman, L. Hodges, "Formalizing the design, evaluation, and application of interaction techniques for immersive virtual environments," *Journal of Visual Languages and Computing*, Issue 1, Vol. 10, Elsevier, pp. 37-53. 1999.

[13] C. Just, A. Bierbaum, A. Baker, C. Cruz-Neira. "VR Juggler: A framework for virtual reality development." *2nd Immersive Projection Technology Workshop,* Ames, Iowa, 1998.

[14] S. Ziegelr, R. J. Moorhead, P. J. Croft, D. J. Lu, "The MetVR case study: meteorological visualization in an immersive virtual environment," *IEEE Visualization 2001 Proceedings*, San Diego, California, pp. 489-492, 2001.

[15] J. L. Gabbard, D. Hix, J. E. Swan, "User-centered design and evaluation of virtual environments." IEEE *Computer Graphics and Applications*, Issue 6, Vol. 19, pp. 51-59. November-December 1999.

[16] S. Bryson, "Virtual Reality in Scientific Visualization." *Communications of the ACM*, Vol. 39, 5, May 1996.

[17] A. van Dam, D. H. Laidlaw, R. M. Simpson, "Experiments in immersive virtual reality for scientific visualization," *Computer & Graphics*, Vol. 26. Elsevier Science, 2002.

[18] L. G. Shappiro, G. C. Stockman, *Computer Vision*, Prentice Hall, Upper Saddle River, New Jersey, 2001.

[19] P. Figueroa, M. Green, and B. Watson, "A framework for 3d interaction techniques," *CAD and Graphics*, Kunming, China, August 22-24, 2001.

[20] K. J. Kuchenbecker, J. Fiene, G. Niemeyer, "Improving contact realism through event-based haptic feedback," *IEEE Transactions on Visualization and Computer Graphics*, Issue 2, Vol. 12, pp. 219-230, March-April 2006.

[21] C. A. Zanbaka, B. C. Lok, S. V. Babu, A. C. Ulinski, L. F. Hodges. "Comparison of path visualizations and cognitive measures relative to travel technique in a virtual environment," *IEEE Transactions on Visualization and Computer Graphics*, Issue 6, Vol. 11. November-December 2006.

[22] D. A. Bowman, E. Kruijff, J. J, LaViola Jr., I Poupyrev, "An introduction to 3-D user interface design," *Presence*, Issue 1, Vol. 10, The MIT Press, Cambridge, Massachusetts, pp. 96-108. February 2001.

[23] G. Lawton, "Making virtual reality more accessible," *Computer*, Vol. 39, 6, June 2006.

[24] VRCO Inc., "vGeo® User's Guide Version 1.7," VRCO Inc, Virginia Beach, Virginia, 2002.

[25] M. P. Baker, D. Hearn, *Computer Graphics with OpenGL*. 3rd ed., Prentice Hall, Upper Saddle River , New Jersey: 2003.

[26] G. Bell, A. Parisi, M Pesce, "VRML 1.0C Specification," [Online] http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html. 1996.

[27] M, Burns, *Automated Fabrication*, Prentice Hall, Section 6.5, 1992.

[28] Object File. [Online] http://local.wasp.uwa.edu.au/~pbourke/dataformats/obj/.

[29] OpenSceneGraph. [Online] http://www.openscenegraph.org/projects/osg/wiki.

[30] Autodesk Maya Help. [Document] San Rafael, California, United States of America: Autodesk Inc, 2007.

[31] Autodesk FBX. [Online] http://www.autodesk.com/fbx.