12-2014

GPS-MIV: The General Purpose System for Multi-display Interactive Visualization

Irving A. Gonzalez Garza
University of Texas-Pan American

GPS-MIV: THE GENERAL PURPOSE SYSTEM FOR MULTI-DISPLAY INTERACTIVE VISUALIZATION

A Thesis

by

IRVING A. GONZALEZ GARZA

Submitted to the Graduate School of
The University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2014

Major Subject: Computer Science

GPS-MIV: THE GENERAL PURPOSE SYSTEM

FOR MULTI-DISPLAY INTERACTIVE

VISUALIZATION

A Thesis
by
IRVING A. GONZALEZ GARZA

COMMITTEE MEMBERS

Dr. Richard H. Fowler
Chair of Committee

Dr. Wendy A. Lawrence-Fowler
Committee Member

Dr. Zhixiang Chen
Committee Member

December 2014

ABSTRACT

Gonzalez Garza, Irving A., GPS-MIV: The General Purpose System for Multi-display Interactive Visualization. Master of Science (MS), December, 2014, 53 pp., 20 figures, references, 45 titles.

The new age of information has created opportunities for inventions like the internet. These inventions allow us access to tremendous quantities of data. But, with the increase in information there is need to make sense of such vast quantities of information by manipulating that information to reveal hidden patterns to aid in making sense of it. Data visualization systems provide the tools to reveal patterns and filter information, aiding the processes of insight and decision making. The purpose of this thesis is to develop and test a data visualization system, **The General Purpose System for Multi-display Interactive Visualization (**GPS-MIV**).** GPS-MIV is a software system allowing the user to visualize data graphically and interact with it. At the core of the system is a graphics system that displays different computer generated scenes from multiple perspectives and with multiple views. Additionally, GSP-MIV provides interaction for the user to explore the scene.

ACKNOWLEDGMENTS

First and foremost, praise thanks goes to my savior Jesus Christ for the many blessings I have received. I would like to thank my parents, Rosa Leticia Garza and Jorge Gonzalez, for all their support during my undergraduate and graduate years. Their love inspired me throughout my life to dream big and work harder. I would like to express my gratitude to my committee chair, Dr. Richard H. Fowler, for giving me the opportunity to work next to him in my graduate years. Finally the love and support of my friends turned any fear of failure into desires to succeed. Thank you.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

In 1962 Richard Wesley Hamming wrote, “The purpose of computing is insight, not numbers.” The electronic computers available then, as well as today, are not an end in themselves, rather, they are only useful to the extent that they help us gain insight into problems and domains described with numbers. The most important things about computers arise in their use to understand, or gain insight, and create. For example much of the economic advance of the past generation is due to the use of computers throughout all the business process. This has had a profound impact on our quality of life. To build upon Hamming's proposition, what we need are computing techniques and systems that convert numbers into insight and guarantee the relevance of those numbers.[4] Visualization of data, or numbers has proven to be one such technique.

Some virtual reality systems, based on immersive displays have proven to be very effective in data visualization. Immersive displays led to the creation of the CAVE (from Computer Augmented virtual Environment) system. The CAVE is a system normally composed of three displays with walls acting as displays surfaces and utilizing stereoscopic projections on the walls to represent a virtual environment. The cost of such display technology has decreased to the point where a system like the original CAVE's first development in the 1990's can be constructed with a dramatically lower price. At this lower cost a larger user base is

possible and a wide range of applications is possible. Additionally, there have also been contributions in the efficiency of how such immersive systems are programmed. When the CAVE was first introduced, it required very significant programming effort specific to the tasks to be performed. Now, there are public domain tools that allow us to program immersive systems at a relatively high level. All these improvements speed up development time and reduce cost, making CAVE systems an option for a wider audience. But despite all this improvements, the CAVE is still quite expensive for some small projects and applications. Also, it requires a significant amount of space for the projection walls to be set up, making a CAVE system impractical for many circumstances. Nevertheless, the features provided by the CAVE in terms of immersion and presence can improve the quality of small applications.[4]

This research deals with creating a low cost alternative to a CAVE system called **the General Purpose System for Multi-display Interactive Visualization (**GPS-MIV**)** that can be run on a single machine providing many of the advantages found with the CAVE. With GPS-MIV a user can visualize data graphically and interact with it. In this study we will show how the system can display different computer generated scenes that have the capability of being seen from different perspectives and multiple views, as well as providing interaction mechanisms for the user to explore the scene.

Motivation

The new age of information has created opportunities for great inventions like the internet; such inventions allow us to have access to tremendous quantities of data. But with the increase in information there is also an increasing need to manipulate that information, revealing unnoticed patterns, especially in large data sets. Data visualization provides the tools to filter

information, make sense of it, and aid in the process of decision making. All this allows us to reduce the cost to access knowledge, as it whether it is in access time, or resource cost.

As predicted by **Gordon Earle Moore,** the price to performance ratio of technology associated with virtual reality has decreased significantly compared to the technology two decades ago. This decrease in price has been most significant in the areas of computing and graphics equipment. Most of the advances in displays and graphics equipment have been achieved through the demand for better gaming experiences. The demand on the consumer side has proven to be a strong force for technologies to grow, Researches have to push new technologies to the public so they can grow.[35]

By creating a lower cost alternative to the CAVE, virtual reality technology can be accessible to wider audience. All this in its turn pushes VR technology forward by allowing more developers to have a taste of what are the benefits of such technology, and making them more likely to invest in more VR technology. It is one of the main motivations of this research to provide with a system that is functional and that provides developers tools to visualize virtual environments, but this is in the short term. In the long term it is a main concern to make VR technology more available so that developers can experiment and discover the advantages of using such technology as it is there already. It is the author's opinion that the reason that VR technology is not being used outside of the research labs as much as it should be is because there hasn't been enough effort in part of researches to promote it, to convince developers of why they should be interested.[35]

Gaining insight is a primary goal in computer visualization. In order to enhance insight and hidden knowledge in a collection of data, we will show that computer visualization coupled with interaction to manipulate views of the data improves the probability of finding insight, as

well as user satisfaction. Insight is about finding patterns and relations that many times are hidden in large data sets. By providing tools to interact and visualize the information these patterns become more apparent and provide the means to facilitate the decision making process.[35]

Goals

The General Purpose System for Multi-display Interactive Visualization should support data visualization, user interaction, and reduce the cost associated with the creation of computer generated environments. The system should be portable and capable of being run on one machine that supports multi-screen configurations. Interaction using the system should be fluid and natural providing conventional means of interaction, as well as, other means of interaction that utilize the human body as part of the interface to facilitate interaction.

The system will provide the ability to be reconfigured in both the hardware and software areas to suit the needs of the project hand and to guarantee portability. The system should be stable to allow researches too use it as the basis for future work in the field of VR and human computer interaction. Most of all, the system should allow the user to create applications that maximize the amount of insight that is revealed through means of interaction and interface creation.

Organization of the Thesis

The next sections describe the background, hardware, and implementation used to create the system. First, background information on the areas of immersive systems, particularly the CAVE VR system, is provided. Then a brief introduction is provided, covering the background of virtual reality, its hardware and software requirements, and the current state of relevant technologies. Next, Stereoscopic 3D is described and an overview is provided of increases in productivity related to multi-screen display. In the following section, The General Purpose System for Multi-display Interactive Visualization is described in detail, listing components and best practices. Finally, the conclusion provides an overview of where the system currently stands and what future work can help extend the capabilities of the virtual reality applications supported.

CHAPTER II

BACKGROUND

The existent literature does not have a unique definition of what virtual reality means and almost everyone has a different understanding. One definition is “Virtual reality is the place where humans and computers make contact." according to William Bricken. We take this definition, and with it we explore the concept of the CAVE and what it means for virtual reality. Though there are many definitions for VR, most agree that it started with Ivan Sutherland’s proposition of the ultimate display. He greatly influenced the future of VR with other contributions like the Sword of Damocles thought to be the very first head mounted display.[4]

When virtual reality started there was enormous hype about how it would change the way we interact with digital devices, but somehow it went away. Nonetheless, Sutherlands initial account of virtual reality provided a window to new worlds and new ways of interaction. Ivan Sutherland said:

“Don’t think of that thing as a screen, think of it as a window, a window through which one looks into a virtual world. "...The challenge to computer graphics is to make that virtual world look real, sound real, move and respond to interaction in real time, and even feel real.”

This vision of what virtual reality should be has motivated scientist to continue researching the field. But what happened? Why did it seem like virtual reality was here, but nothing happened? Frederick P. Brooks, from University of North Carolina at Chapel Hill said,

“our discipline stood on Mount Pisgah looking into the Promised Land”.[4] Currently VR is starting to transition from the technology that almost worked to that one that barely works.

There are some technologies that may be responsible for determining the success of VR in the future. Some of them are the visual display that is the window to that immersion the user is striving for, the system in charge of rendering the frames, the tracking system that reports the position and orientation of the user, and a database that maintains the virtual world.

One concern of VR system has always been cost; we have experienced the evolution of speed and size of CPUs and GPUs, naturally following from Moore's law. This allows for VR configurations to now be assembled from mass market components. There used to be a time where the generation of the images was the main factor of cost, but now the factors dominating cost are the screens or displays and tracking systems. Even though the cost for components has decreased, a CAVE system can very expensive and is one of the reasons we look for alternatives to it.[7]

Tracking the position and orientation of the user’s head used to be one of the major problems faced by virtual reality researchers; it was difficult to deal with the tracking range and distortion due to metal objects. But this is no longer a challenge; now we have devices like the commodity Microsoft Kinect that no longer has problems with magnetic interference and allows for a wide range tracking. The system is not perfect, but currently it is a major area of research.

One of the major challenges that VR faces since its start in 1994 is how to increase the adoption of VR systems in the consumer side; not how to advance the technology but how to increase its adoption. This has to do with how VR is perceived by the public in terms of what increases in productivity can be achieved, improvements in team communication, and reduced costs through this type of technology. Some places where VR is more accepted are in

engineering organizations where VR walkthroughs are used as part of design evaluations and reviews. Some companies surprisingly have not benefited in their design practices but in the way they communicate by using VR. Such is the case of Brown and Root where the painters from the maintenance force figured out that certain fixtures on an oil platform should be made of extra heavy steal because it was not able to be repainted when installed. Thanks to the VR simulations they were able to communicate ideas and find more appropriate methods. I think we will see an exponential growth in VR technologies during the next years, and, perhaps, this time we may be able to reach the Promised Land.[12]

CAVE

The CAVE is a system that creates a projection of the images on large screens surrounding the user. The CAVE was designed and implemented in 1991 at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago. The CAVE enables for data analysis in a visual manner, giving insight into the relationships that very complex data can have. It has the flexibility of being projected in the three dimensions in different views at the same time. The CAVE is constructed as surrounding projection walls, the walls made up of rear projection screens or flat panel displays. The projection equipment is in general high-resolution, since the user is close to the displays and this requires higher densities of pixels to maintain the illusion of a real environment. Also, the user typically has to wear 3D glasses to experience the stereoscopic display produced by the system. In some systems users can walk in the environment, and so walk around objects and visualize them from different angles. Initially, user tracking was done with electromagnetic sensors but more recently infrared cameras are used. The computer tracks the position of the user and displays images and may produce sound

accordingly. The CAVE usually may have multiple speakers placed at different angles allowing for 3D sound along with the 3D images. [3]

The CAVE provides a promising field of research. What makes it interesting are the future areas of research, like adaptation of methods for three-dimensional projections, Adapting the CAVE and its interface to allow different methods for generating sequences. Development of navigation tools that allow keeping track of the location of the current projection in the three dimensional space, together with relative positions of projections. Other areas of research include: how much gain in intuition and understanding of high-dimensional structures do we get by using 3 dimensional projections over 2 dimensional projections, and coordinating the use of other tools such as interactive brushing, painting, isolation, identification of data. The CAVE is a very different method of data visualization that what is commonly used. Users see the difference that it makes when analyzing data and the potential that it has for much more.[3]

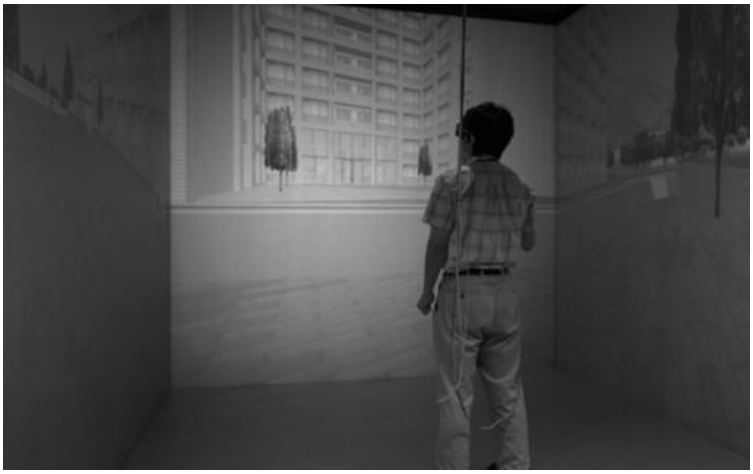

Figure 1 CAVE Program Based on OpenGL[3]

CAVELib

CAVELib was the first software used to program the CAVE created to run only on a specific set of hardware. CAVELib provides support for multi-display, cluster architecture, and device integration. When it was first released it provided developers with an alternative to commercial tools that where expensive, its open source format made it a very popular option. Later on, the software was extended to support different platforms and architecture types. Unfortunately, the original developers chose to license the software, sold that license, and now the software is prohibitively expensive for most university use.

Hardware

Workstation

Currently, the General Purpose System for Multi-display Interactive Visualization runs on an HP workstation with two NVidia graphics cards configured in SLI. The workstation supports the system running up to 4 monitors. This configuration provides great flexibility and portability. Whereas the CAVE is usually composed of specialized equipment to control individual screens, the system developed in this thesis can be extended to control projectors, providing a very large sized display.

Tracking Hardware

Technologies based on depth imagining have advanced tremendously in very short time. The commodity Microsoft Kinect provides all the depth imaging technology needed for powerful application and is affordable and practical for the general public. The pixels that

compose a depth image represent a calibrated depth in the environment, versus an image formed of intensity or color. The Kinect produces a "depth image" with a resolution of 640x480 and a frame rate of 30 frames per second. Such a depth camera has the advantages compared traditional sensors, capable of functioning in low light settings, providing a calibrated scale, being unaffected by color or texture, and distinguishing body ambiguities in a pose. Depth cameras facilitate the process of eliminating a background to create green screen like effects. Importantly, such depth cameras allow to synthesize realistic images of people and build data sets to train to recognize those shapes. The system developed in this thesis is based on the creation of a system capable of analyzing one depth image, and from there, extracting relevant features like body joints, head position, and head orientation.

Our bodies can move in many different ways and assume many different poses that are hard to recreate by simulations and models. The Kinect provides tools to track many of these poses and movements; it also allows to track over 60 points relevant to an individual's face.[16]

Displays

There are many different configurations available. Single surface display that are an alternative to multi-surface displays because of their lower price while still providing elements of multi-surface displays needed for immersion. Some of the approaches used are front projected displays or even tiled displays made of an array of common monitors.

Two Surface Displays provide more immersion for the user but have disadvantages in terms of cost and complexity to operate. In general this displays work as if the projection was on the corner of a room. This type of displays has the disadvantage that the viewer does not have complete freedom to move around.

There are many solutions for dealing with systems that need to provide more than two surface displays. One of the main benefits of three or more Surface displays is the user mobility that leads to feeling more immersed, all this in general achieved by rear projection.[15]

Multi-Screen

Since the invention of windows 98, computers are able to support configurations involving more than one monitor. At first multi-screen set ups where being used for gaming, the area of gaming being the driving force for any of the improvements in computer displays, graphics, and video performance. With the increase in computing power and drop in cost, more and more places like the office and the household can support more applications, so multi-tasking has become very important to take advantage of the new found computing power.[2]

Multi-screen solutions are display configurations of more than one monitor connected to the same computer. The computer can treat the screens as a boundary space, connected space, or extended. For instance, a program can be maximized to occupy the space of one screen or it can also be maximized to occupy the space on all available screens in that particular configuration. Multi-screens let the user move applications between the displays or spread the applications across all displays, depending on what the user sees fit. The real benefit of having multiple displays is in how it improves efficiency.

If we consider scenarios like transferring data from one text processor to another, with a single screen we have to sacrifice visibility to have more than one document open at a time or we sacrifice time by minimizing one and maximizing the other. There are significant issues with these as losing the point in the document that is supposed to be changed. This type of problems means more time going between the documents to find the specific place to be modified.

Task improvements in efficiency are more obvious in cases where data is large or benefits from opening multiple windows. This particularly applies to the area of computer graphics, in which designers benefit from looking at models from different angles or looking at the source material for their model at the same time. Displaying multiple angles of a 3D model clearly cuts down the time that it takes to build a model by not having to rotate or translate the model to accommodate for the use of only one display.

Cognitively, physical placement of the digital content has an effect on the mental processing. There are two main cognitive abilities used in the task of interacting with multiple windows simultaneously: the ability to differentiate between the windows and the ability to track locations that are both individual and relative to each window. One screen forces a disassociation of data by replacement of material or reduced views. This means that the cognitive process is freed from processing the relocation of the information.[2]

Productivity testing is usually measured by reproducing a common work scenario. Participants then are asked to complete a task or series of task using different configurations. the time for completing the task, response time, and satisfaction are measured. Following the data analysis, the participants answer a questioner that rates the display configuration and briefly detail their experience.[2]

Stereoscopic Graphics

A stereoscopic image is created by simultaneously presenting the left and right eye different images, i.e., those images as viewed from the viewpoint of each eye, images that are slightly different. The brain creates an image with depth by combining the two viewpoints. That is the reason why you can only notice one image instead of one superimposed on the other.

Computer generated images usually are created by computing the image from one perspective point, but a stereographic image is different in that it must be generated from two viewpoints.[43]

Monocular depth cues are what allow us to perceive depth in images, even without stereoscopic viewing, i.e., different images for each eye. Monocular depth cues include light and shade, relative size, interposition, textural gradient, aerial perspective, motion parallax, and perspective. Light and shade allows the graphics modeler to create objects with very defined edges or rounded by manipulating shading, bright objects are perceived as been closer to the observer opposed to dim objects. Relative size has to do with how objects look larger when they are closer to the observer and smaller when they are far away. Interposition is when for example you hold a book and you know it's closer than the desk where you are seating because you cannot see through the book. Textural gradient like a grassy lawn or a tweed jacket provides depth cues because the closer you are to an object the better you can see the texture. Aerial perspective is the reduction in vision of objects at the distance. Motion parallax is like when you are in your car and you can see how the telephone poles move faster than say the mountains, not every virtual scene can use motion parallax but if it can, it will decently help to the depth perception. Perspective is among the most important depth cues, and this is because it scales the depth cue. A strong perspective cue will make everything look deeper and easier to perceive.[43]

In addition to monocular cues there are binocular cues. Stereoscopy is one, and it is created by the disparity in the left and right eye images received at the retinas. The brain fuses them into one. A stereoscopic display provides the two images. A simple experiment that can be conducted is holding your finger into your face, when you look at it your eyes will converge on it. If you continue to focus on the finger while at the same time noticing the background, you will see the background image doubles. If you reverse the experiment and focus on the background while

paying attention to your finger, your finger will appear to double. If we could extract the image that each eye receives and superimpose them, we would get two almost overlapping images of the same. This is what is known as disparity, better defined as the distance horizontally of the two viewpoints. [43]

Rendering

Rendering is a series of actions and transformations that generates an image from a model by using software. The end result of such process is called a rendering. A computer scene is stored in a defined data structure or format; it holds texture, lighting, viewpoint, and geometry information that describe in detail the scene that will be generated. The information contained in the data structure is read by the application and transformed to a digital image. Rendering can be seen as analogous to the challenge faced by a painter trying to produce a 2D image from a 3D source; even though, the method to achieve an image varies from a painting to a virtual scene, the transformation steps are relative.

Rendering deals with the process of creating polygons and giving particular characteristics to polygon faces. Polygon faces can be textured by applying images to their surfaces, these textures can create an illusion where the polygons seem to have more detail than what is actually defined by the underlying geometry. Unfortunately, texture have the disadvantage of requiring video RAM memory which is at the time of this writing limited to no more than 6 GB for commercial computers running high end graphics cards. Computer scientists have come up with ways to minimize texture memory use in order to increase the performance of applications. A very well known technique for optimizing textures is using MIP mapping, this means that pre-calculated optimized textures are attached to the main texture, this textures are a

lower resolution representation of the same texture. This works because as the camera moves away from a particular object, the texture attached to that object will be reduced since the detail on the object is not visible. Similar to the MIP approach, geometry on the scene can be adjusted to accommodate for the camera placement. A method called level of detail (LOD) changes the number of polygons in an object based on how far they are from the camera.[21]

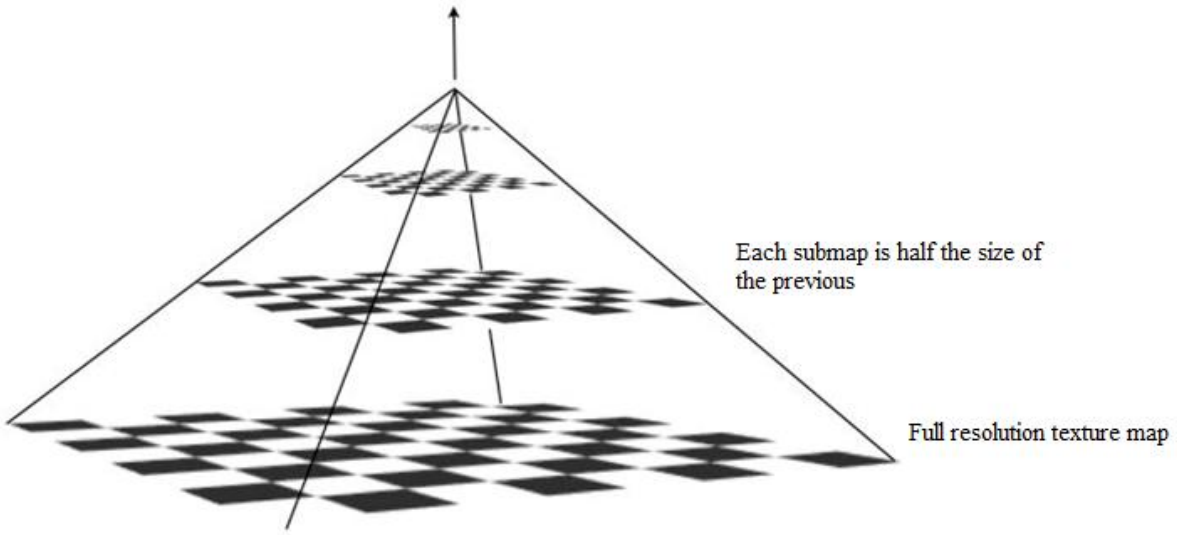

Figure 2 MIP Mapping

In computer graphics there are two types of lighting models known as static and dynamic lighting. Static lighting does not change position on the object. It is very computationally efficient for rendering scenes that are not in motion. Since the scene does not move, the lighting can be calculated in advance, which increases the efficiency drastically. The resulting directional light, shadows, and brightness are combined with the texture to create the final effect.

Dynamic lighting is that which is in constant movement within the environment, for that reason it cannot be pre-calculated since it is dependent on the actions performed in the scene. Once all the parts described before combine, the last process is the rendering. This process

consists of generating the display that will be drawn on the screen. In this last step the scene is rendered in real time which allows the user to interact with the world seamlessly.[21]

The Cost of Knowledge

Computers are becoming more and more to an information access point. Naturally this will continue as information grows and memory cost comes down. In 1991 Tennant and Heilmeier predicted that by 1995 the information would have increased 10,000 greater than it was, and they were right.[6]

The challenge since the 90's has been to come up with innovative ways to deal with the increasing amount of information, how to access and process it. It was observed that information that is part of a system has a cost structure. The cost structure is a cost value for the information depending on what part of the system it is. The cost of knowledge is not only applicable to digital information but any kind, getting a document from a stand in a library or a filing cabinet and placing it in your work area will reduce the cost associated with accessing it multiple times.

A key point on designing an information system is to work with the cost of knowledge in mind and design it as efficient as possible. The paper argues that in a world with such amount of information the real task is not to find information, but to allocate the limited time a person has on gaining useful information.

The idea is to increase the benefits of information for each established unit of cost, so we need to figure out how fast the information is growing for each unit of time spent looking. This is defined as the Cost of knowledge Characteristic Function. In any system like a library we have information arranged as a hierarchy where small chunks of information are accessed at low cost and larger chucks cost more to access. The way we can improve such scenario is by reducing the time cost for each different level in the hierarchy or by keeping the number of documents the

same, which reduces the access cost. The Cost of Knowledge function helps us have a better idea of what system runs more efficiently when compare cost performance.

In the context of this thesis, the information search deals with using multi-screen configurations to reduce the Cost of Knowledge on direct walk tasks, which are tasks where a user has to navigate from a start to an end by a series of direct manipulation methods like mouse clicks. Some examples of these tasks can be a series of button presses and mouse clicks to operate a hierarchical file system or a help system. In essence this file systems display the file structure and the user points to a file resulting in a new display, and then the process is repeated.[6]

CHAPTER III

THE GENERAL PURPOSE SYSTEM FOR MULTI-DISPLAY INTERACTIVE VISUALIZATION

When creating a system like the one reported in this work, there are several considerations in how to approach the creation of the system. The goals of the system have to be clear, the hardware to be used, especially screen and tracking technology, and peripheral device support. Usually, the best way to have flexibility in wide customization options in the development phase is by taking advantage of open source software. But sometimes this approach is difficult, as documentation for this type of software tends to be nonexistent or not comprehensive enough.

The General Purpose System for Multi-display Interactive Visualization takes the approach of combining open source and custom developed software to speed up the development process. The system uses the Windows API for handling the windowing system and display adjustment. OpenGL provides model rendering, scene creation, and stereoscopic projection, as well as, low level control over the graphics hardware. The Microsoft Kinect SDK provides device integration for the Kinect device. Though there are open source alternatives to the Microsoft's proprietary software, the SDK provides enough flexibility to customize applications as needed while at the same time providing optimized skeleton and face point recognition.

The following figure represents the overall software layers used in The General Purpose System for Multi-display Interactive Visualization.

Figure 3 The GPS-MIV Software Architecture

OpenGL

OpenGL provides an environment where interactive graphics can be developed; it is the most widely used graphics standard. Since its creation OpenGL has allowed developers to program for a wide range of computer platforms. OpenGL speeds up the development of software applications by providing rendering, texture mapping, special effects, and very versatile graphic functions. OpenGL provides support for all desktop and workstation platforms allowing for a vast array of applications in all platforms. Any application that requires performance, from CAD modeling to 3D animation can utilize OpenGL to minimize overhead associated with the use of higher level languages. OpenGL was specifically chosen for this project as it has been proven to deliver consistent visual display result on any compliant hardware. Another reason OpenGL was chosen is for its forward thinking design. The design allows to incorporate new hardware innovation through extensions of the OpenGL API. All this is in contrast to graphics programming direct competitor, DirectX. Since it is owned by Microsoft all new updates have to be approved by them, which can take a long time. OpenGL is easy to use thanks to its intuitive design and state machine model.

As noted before, open source software tends to be poorly documented and difficult to work with, but in the case of OpenGL there is a great deal of available code, multiple books and tutorials are available, as well as wealth of forums, making the access of OpenGL information relatively inexpensive and easy to obtain.

OpenGL Pipeline

As it can be seen from the diagram below, the architecture of OpenGL is designed as a stage pipeline. The following picture represents a somewhat simplified version of the pipeline.

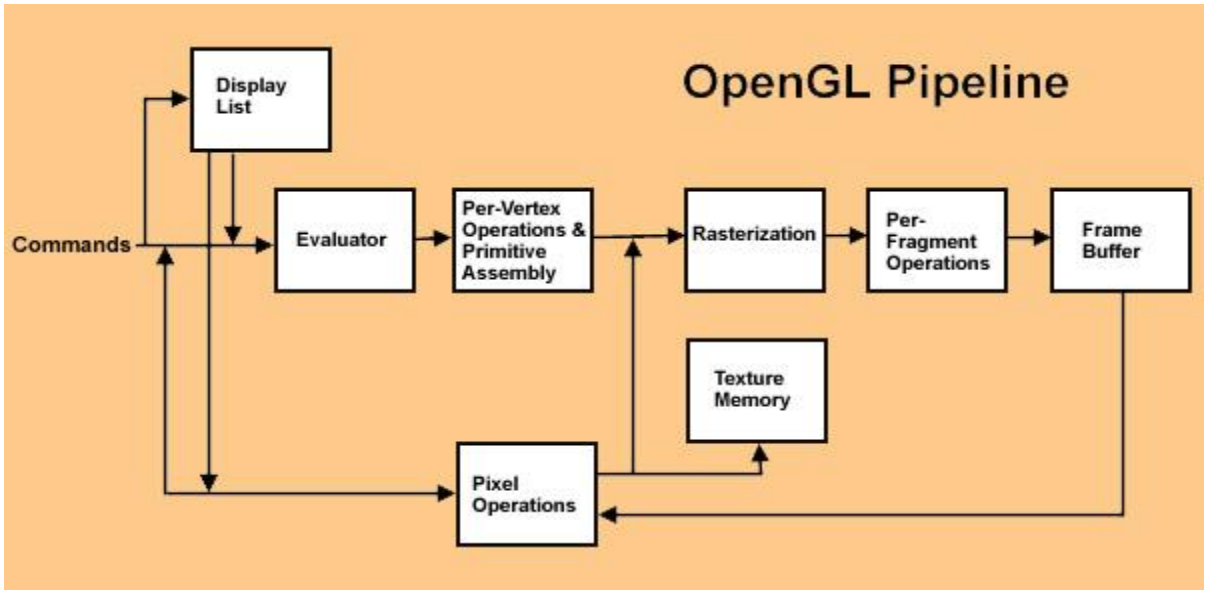

Figure 4 OpenGL Pipeline

From left to right, commands are fed into the pipeline. The commands can be accumulated in a display list or sent down the pipeline. A display list can improve the performance of the application since it allows commands to be stored and executed at a later time. In general, it is a good practice to store commands if you are planning on redrawing the scene multiple times, or have internal stages that will be re-applied.

The first stage following the display list is the pipeline evaluator. The evaluator examines the commands and relates them to the corresponding vertex and attribute commands. The evaluator serves as a translator converting surfaces or parametric curves that maybe be described by functions or control points. The evaluator provides a way to derive the vertices from the functions or control points.[22]

The second stage is known as the per-vertex operations, this stage converts vertices into OpenGL primitives. The vertex data is transformed by 4x4 floating point matrices. Special coordinates have to be transformed from a 3D position to a position on the display. This stage does a lot of computation and becomes quite busy as this is where texture coordinates generated, lightning calculations performed using the transformed vertex data, as well as considering light source position, material properties, and other information to produce color values.[22]

The third stage is rasterization, which is the conversion of pixel and geometric data into fragments. These fragments are a series of frame buffer addresses and their corresponding values. Calculations like lines, polygon stipples, and antialiasing are important in this stage as vertices connect with lines and the interior polygons need filled. Each fragment also has an associated color and depth value that helps in calculations relative to the perpendicular plane of the camera.

The fourth stage is the per fragment operations. For values to be stored in the frame buffer, there are a series of operations that can alter or remove fragments. OpenGL allows the programmer to decide if these operations should be enabled or disabled. One of these operations is texturing, in this operation a texture element is generated from the texture memory associated with each fragment. Next, calculations like fog, alpha test, stencil test, and depth-buffer test may be applied. These tests help optimize the system by not doing some unnecessary

computations like not drawing polygons that are out sight or obscured by other objects. Lastly, the completely processed fragment is drawn into the corresponding buffer where it is converted into a pixel with a final display place.[22]

Installation and Configuration

Most of the support libraries used in this particular system are open source meaning that the source code for compilation is provided. Following is a list that provides a brief overview of the required dependencies for building The General Purpose System for Multi-display Interactive Visualization:

gl.h Library provides support for calling functions defined in the OpenGL API. Most popular operating systems like Linux and Windows come with some form of this library. Windows comes with the version 1.1 of OpenGL, so you will need to make sure that you download the latest drivers for your graphics card to support some of the features included in the system.

glm.h is a header library for c/c++ that provides support for the OpenGL shading Language (GLSL). Glm simplifies mathematical computation programming by providing functions that resemble GLSL conventions. Glm was built with interoperability in mind allowing communication with any third party sdk.

Immersion and Presence: A Study of Stereoscopic 3D

Stereo 3D is perceived by the general public as a relatively new invention, mainly because of its newly found integration in movie theaters, but it was actually first introduced in 1838. One may ask the reason why it has taken so long for 3D to be integrated in obvious

environments like a cinema. One strong reason for the recent resurgence of 3D technology is the switch from analog to digital. Errors and discrepancies that can appear in the two images required for 3D can destroy the visual effect, but with new digital film and editing tools this errors can be corrected, whereas with analog the problems could not be easily corrected.

Everything indicates that it may not be long before 3D is adopted they way it was intended many years ago. But there are still many consumers that do not have a good opinion or are just not yet convinced that the 3D technology is a way to enhance the experience. Some have the impression that the technology is just a way that studios try for increasing profit. But 3D used appropriately can allow an artist to immerse the audience and extend what the audience can feel.[15]

Stereoscopic 3D in Application

OpenGL has support for quad-buffered stereo. Quad-buffer allows to render into four buffers independently: left and right(for the two images required for stereoscopic images), and front and back (allowing double buffering required for smooth movement perception). This allows the buffers to sync shutter glasses for the image of the front left and front right buffers that display the stereo image while at the same time the back left and back right buffers can be updated without interference. A solid understanding of how OpenGL geometry works is essential for achieving correct manipulation of the camera and viewports. When trying to render scene in stereo there are two techniques that are common Toed-in stereo and Asymmetric frustum parallel axis projection stereo. [44]

Toed-in stereo is easily implemented but does give room to some unwanted visual effects. Keystone distortion is such effect, it occurs when using this technique; this effect is produced by trying to project a scene on the screen with the viewpoint at an angle, very similar

to how an image looks when a projector is not well centered and the image appears larger on the sides. [44]

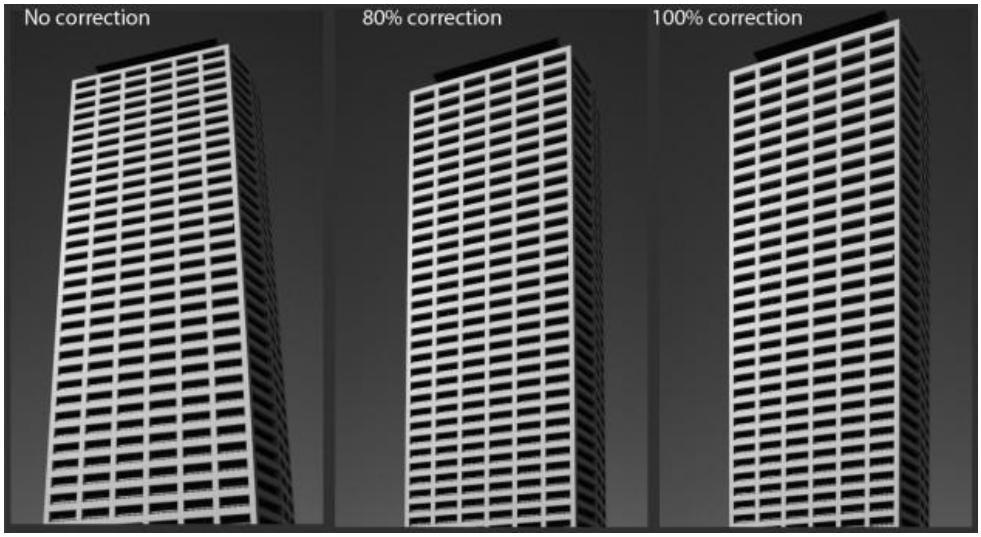

Figure 5 Keystone Effect[44]

Before explaining how this type of projection is set up on OpenGL, the concept of viewing frustum should be explained. The frustum is a region of space in the virtual world that is projected onto the screen; it is analogous to the field of view of a conventional camera. The frustum requires to specify the near and far clipping planes and aspect ratio. Just as with a camera the field of view specifies how much of the virtual environment is visible at a given moment. Given all this information the process of setting up a scene for rendering is similar to taking a photo. You start by setting a tripod and pointing the camera, which is equivalent to a viewing transformation, arrange the objects in the scene (modeling transformation), and choose a camera lens or adjust the zoom. the final step of choosing a camera lens relates to our viewing frustum.

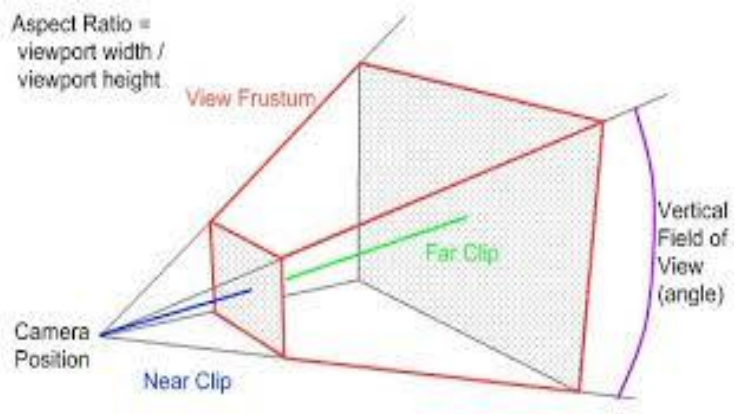

Figure 6 World Camera vs. OpenGL Projection[19]

The next diagram represents how the Toed -in projection is composed using a frustum and right and left eye positioning.

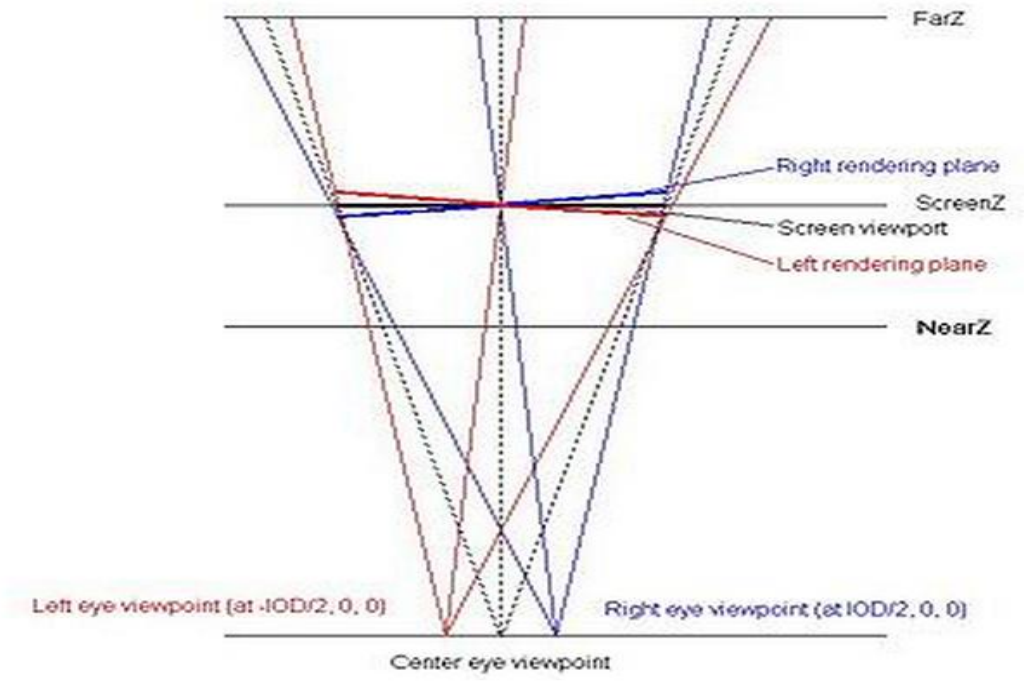

Figure 7 Toed-in Projection[44]

One frustum is created for both eyes that is represented by the middle triangle in the diagram, the left and rights are shifted right or left accordingly. The displacement of each eye with respect to the viewing frustum creates independent rendering planes for each eye. Now the keystone effect can be seen as a distortion such that shapes like squares appear to be trapezoidal.

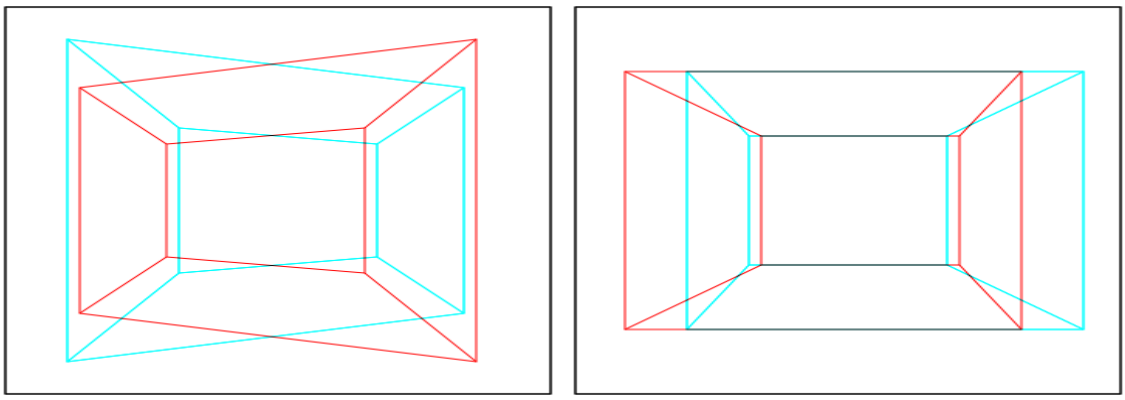

Figure 8 Keystone Effect vs. Compensation for the Keystone Effect.[44]

The left side of the image above shows the left and right images over imposed as to create a 3D image, this particular image shows how the keystone effect can degrade or alter an image. The brain will not be able to correctly merge the views. In contrast the right image provides an example of using the Asymmetric frustum parallel axis projection.

This approach is quiet more complex, since two asymmetric frustums need to be set up. Using OpenGL the asymmetric frustum is set up at the point (0,0,0), and then it is translated by the intraocular distance divided by two to the left or right accordingly.

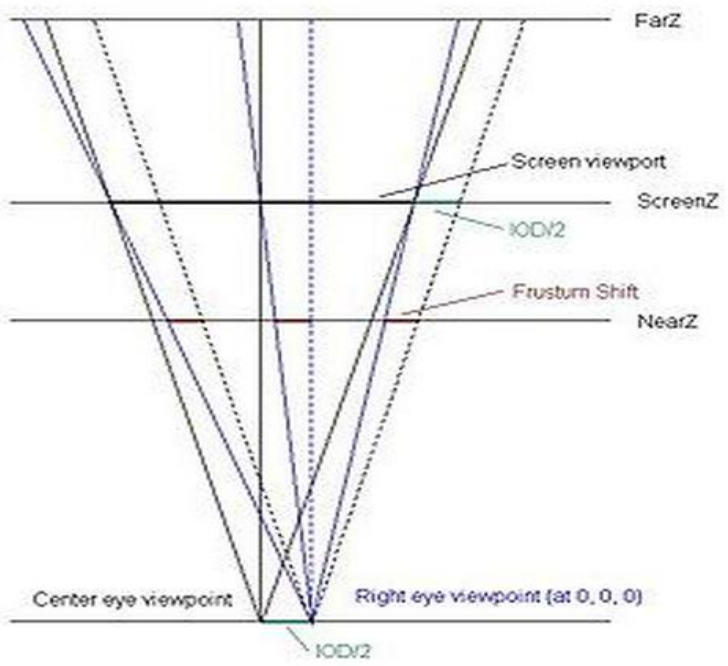

Figure 9 Frustum Shift[44]

As it can be seen from the diagram above in this case the screen view port is parallel to the corresponding eye due to the fact that the viewing frustum is being shifted to correspond with the placement of each eye. This technique provides the correct 3D effect as on the right side of figure 7.

The General Purpose System for Multi-display Interactive Visualization is composed of one workstation that serves as the replacement for a cluster in conventional CAVE set ups. In general, Cave system is composed of a set of slave nodes and one master node. The slave nodes are usually tasked with controlling the rendering of the Virtual Environments, one of the nodes may be used exclusively for computing tracking information obtained from cameras, depth sensors, and other input devices. And the master node is used for syncing the slave nodes, controlling set up operation, configuration files, and very importantly the master node tells each other node what portion of the virtual environment to display in a viewport, so that movement

received from tracking devices appears fluid in the nodes. Touching on the last point a viewport is the area of interest, the area we want to visualize in some coordinate system.

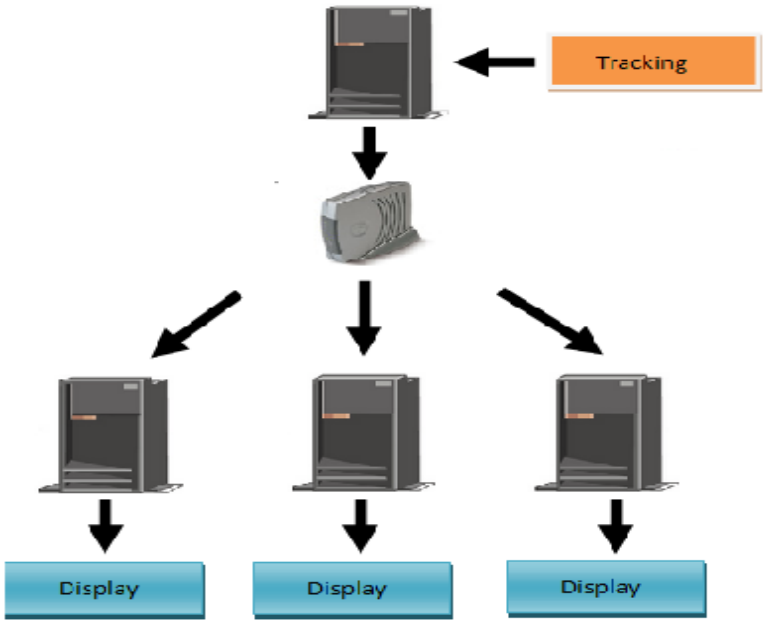

Figure 10 Conventional CAVE Configuration

Given the previous description of what components are required for a CAVE and how they interact with each other, it can be seen that a CAVE requires a large number of components to be put together which is fine for large visualization labs. However, smaller labs or projects can benefit from a single node that works as master and slave while at the same time decreasing cost and facilitating the test of environments. The single node system works by taking advantage of Nvidia's SLI technology. SLI is a revolutionary technology that allows to drastically increase the graphical performance by combining various GPUs in systems that have a certified SLI port. SLI uses Nvidia's proprietary software algorithms that can duplicate or scale the performance depending on the number of graphics cards.

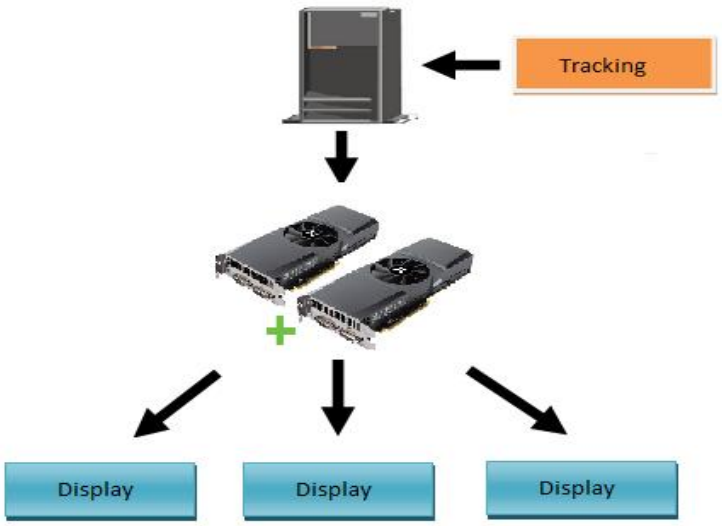

Figure 11 Proposed CAVE Configuration

The use of an SLI eliminates the need of having a router sending packages from the master node to the slave nodes and in-between the slave nodes. Each SLI compatible graphics card can easily support two displays run through display port outputs. This system provides developers with more options when choosing a system and hardware configuration that suits their needs. It does not attempt to replace conventional CAVES, but to be used as a cost effective solution for developing Virtual Environments.

Applications

The General Purpose System for Multi-display Interactive Visualization runs applications that are native to OpenGL. These applications help a developer get familiar with the development of software in the system. In general the elements of the sample applications are combined to create more robust applications. The system easily integrates existing OpenGL code from other applications, allowing these applications to take advantage of the features provided by the system discussed in this paper.

When developers create OpenGL applications, they have to use utilities like glut and freeglut or use plain Windows API function calls and initializations. The system developed abstracts the Windows API calls and simplifies the process of creating windows, setting view ports, and initializing perspective and orthographic projections. This is particularly useful for programmer that are starting to get acquainted with OpenGL as it frees them from having to deal with all the windowing system details and lets them dive right into programming with OpenGL. The system defines the class " `GLContext`". This class handles processes like initializing a window, choosing window formant, enabling stereo, etc. The following code segment shows how easy it is to initialize a window:

```
GLContext         guiContext(0);     // WinAPI Context for the GUI Window
hWnd = CreateMyWindow(hInstance, TEXT("GUIWindow"), 1080, 1000);
guiContext2.Initialize(hWnd);
guiContext2.EnableStereo();
```

Medical Applications

Computer simulations are becoming more and more important, and the development of techniques for obtaining digital data for medical imagining are increasing. Technologies like ultrasound, magnetic resonance imaging, and computer tomography are part of daily medical procedures. These new technologies provide very detailed anatomical models that allow for better data and analysis representation. The simulation of medical operations allows for predictions on the outcome of different medical procedures such as: radiation therapy and neurosurgery. Virtual reality plays a new role in telemedicine for remote diagnostic and even interventions. This past decade medical applications of virtual reality have moved from a curiosity for researchers to invaluable tools.[19]

Especially in radiology visualization provides a much more natural view and analysis of a patient's anatomy. Endoscopy or bronchoscopy techniques based on virtual reconstructions of a patient's anatomy are in fast development. This allows the patient to have the benefits of comfort and cost reduction. One of the most noteworthy developments has been in colonoscopy, which is currently in clinical validation phase.[19]

For a surgery preoperative planning is crucial for the highest probability of success. Traditionally, this preoperative planning has been limited and its implementation in the operation room no guaranteed. Computer assistance and virtual reality technology can substantially contribute to the precise execution of preoperative plans.

For areas such as conformal radiotherapy and stereotactic neurosurgery, virtual reality have become a necessity for treatment and preoperative planning. Other areas, such as craniofacial neurosurgery and open neurosurgery, have also benefit from planning and realistic predictions facilitated by VR technologies.

.OBJ Model Support

Content in 3D exists in various formats and variants, .OBJ being the most popular one. This file type is preferred for its open nature, cross platform use, and simplicity. Software like Blender, Maya, Zbrush and many other graphics packages support .OBJ.

.OBJ is a type of file that defines geometry created by Wavefront Technologies. This type of file is open format and has been adopted by a wide array of 3D graphics programs, it could be said that it is a universally accepted format. The .OBJ format can be expressed as a data format that allows for geometric representation by the points of each vertex, it also defines the UV position for each texture, vertex normals, and the faces that conform each polygon defined as a

list of vertices and texture vertices. It is important to note that vertices are stored in a counter-clockwise order by default, making unnecessary the explicit declaration of the face normals.

The following set of image show how a model can be imported into our system using .OBJ file support. In this case the model was created using Blender and exported to our system.

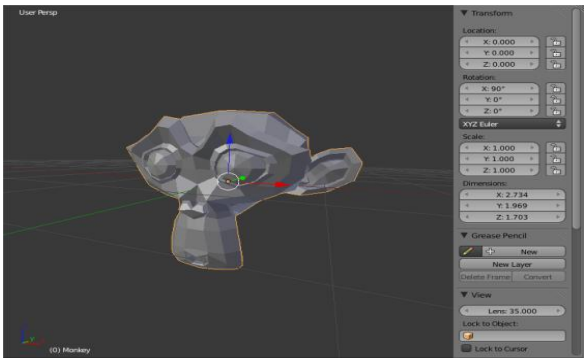

Figure 12 Blender Model

Inside The General Purpose System for Multi-display Interactive Visualization we can show up to four viewports of a scene that is four different perspectives of the world. The system can set on or off the stereoscopic 3D effect individually for each viewport.

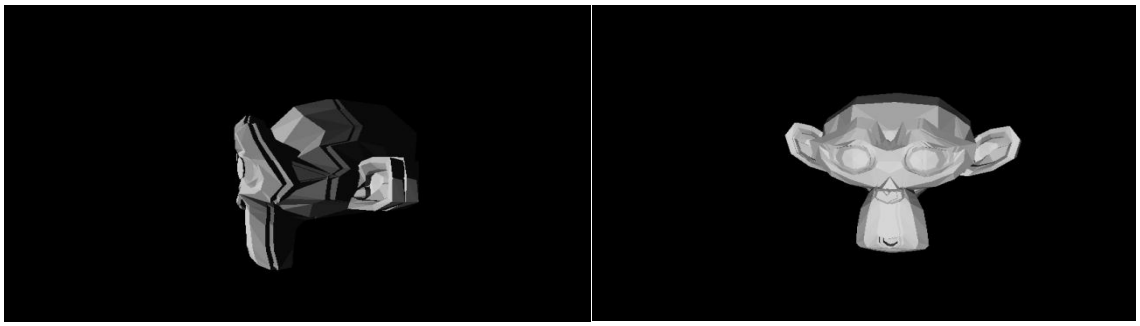

Figure 13 The GPS-MIV Multi-Perspective

FP Camera Control

Included in the system is the option to use what is commonly known in games as a FP (First Person) camera. This type of camera perspective refers to seeing the world through the eyes of a player in the environment. Many times this perspective is used to simulate the feeling of driving a car or some type of vehicle as it allows to show the inside of the cockpit and adds realism. First-person shooter made this type of perspective very popular as the graphical perspective has a very immersive impact on the experience.

The system developed in this work is as noted, an alternative to a conventional CAVE system. One of the applications that a CAVE can have is to aid architects or engineers by providing virtual guided tours of buildings or construction. This provides the designer a spatial feel for what the end product will be like and allows to the designer to make changes based on the experience. Customers that want to build a house can navigate through the digital model and decide if that is what they want.

Microsoft's Kinect

The latest entry in the virtual reality inspired gaming world is Project Natal, now known as Kinect, new technology developed by Microsoft now for the Xbox 360. The technology has a new way of interacting with video games and also with computer systems in general. In their demo videos, a system was proposed that required no keyboard and no controller. A user's voice and motions served as their methods for interacting with the system. The Kinect device works very well, but games have not yet exploited its full potential. Lately, the development of software for Kinect is shifting to the computer. This is exemplified by the release of Kinect for

windows. This new version of Kinect supports a near mode allowing detection of users when located closer to the device and extended depth data giving developers more flexibility to make a choice of using the data beyond 4 meters.[5]

GPS-MIV takes advantage of the Kinect's depth tracking feature to locate a user in the room and track the skeleton position. This is accomplished by using the IR emitter on the Kinect to project an irregular pattern of infrared dots of varying intensities. Then, the Kinect creates a depth image by detecting the distortion in the pattern.

Figure 14 Speckle Pattern[34]

The Kinect has a hard coded pattern of the speckle pattern. Each dot created by the infrared camera has a relative surrounding that facilitates identification on a scene. The algorithm used chooses a dot in the speckle pattern and compares it with the one cast on the environment and then checks for the eight surrounding pixels. Once that is completed, it is relatively easy to check for disparity and adjust using the focal length of the camera to determine the depth of that pixel. The process has to be repeated for each dot in the speckle pattern.

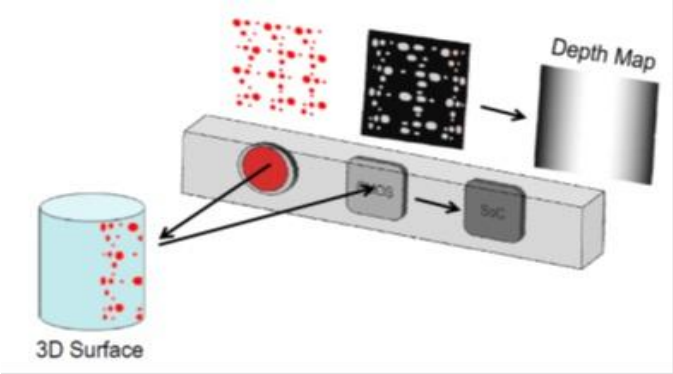

Figure 15 Kinect System Overview[34]

The Kinect offers the flexibility to choose the range at which the sensor should work. This is done through the different sized dots on the speckle pattern. The range can be adjusted to work anywhere in between 1m and 8m. GPS-MIV takes advantage of the infrared sensor in the Kinect by proving skeleton recognition in seated and standing mode, additionally it provides two range modes: near and far.

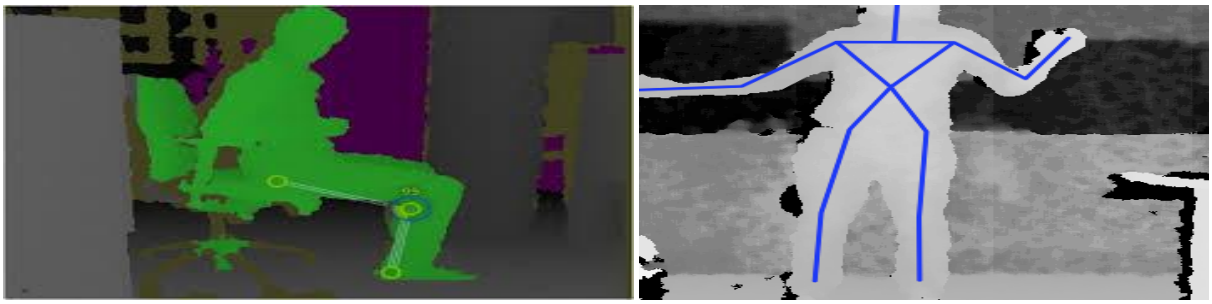

Figure 16 Seated and Standing Tracking

Existing applications on the GPS-MIV can support head rotation on the x-plain to control the camera and look at the environment freely. This level of interaction is intuitive and allows to examine the information presented on the scene faster. For instance, the boids document visualization system has been ported to work with the system described in this paper.

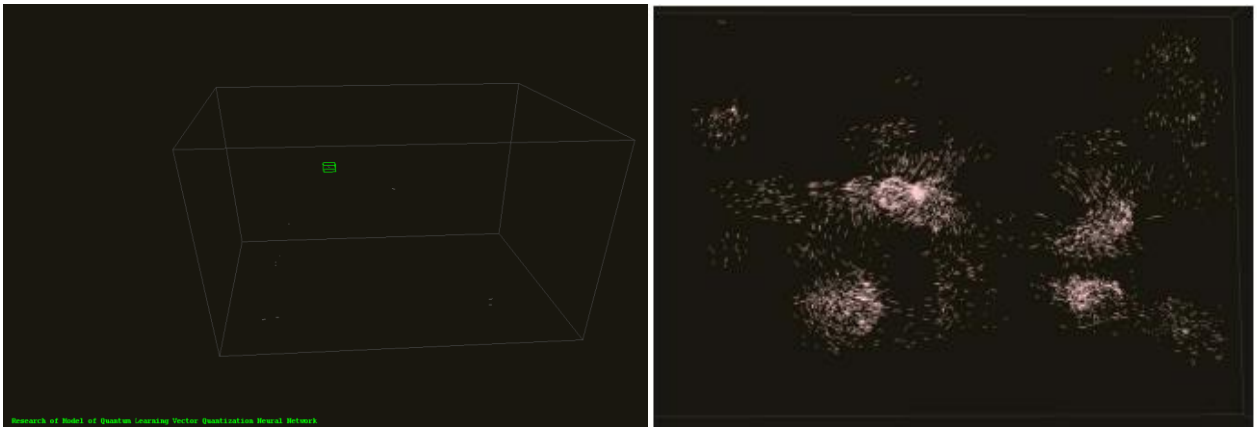

Figure 17 Boids Document Visualization

The boids essentially creates clusters of documents and represents them as flocks of birds. By porting the boids application to the GPS-MIV it can incorporate the motion controls and multi-screen functionality to increase the level of information that a simulation can yield.

CHAPTER IV

CONCLUSION

Just as paper and pencil can help us overcome the limits of our memory and conscious reasoning by enhancing the ability to compute and store information, virtual reality technology can increase productivity beyond what was previously possible.

The goal of virtual reality is to present virtual objects that are as close as possible to their real world counter parts or that are beyond what is found in the real world. Virtual reality attempts to emulate the real world or even go beyond. This notion of going beyond what is normal is the basis of many principles of information visualization. Using computer aided software in the process of analyzing data leads to the discovery of hidden patterns and provides better choices for the analyst.

There are different commercial software solutions for analyzing data sets but they can have large licensing and software cost. This has been one of the main reasons that led independent developers to form an open source community and develop software solutions that are free of charge and have the power and upgradeability to create applications in different fields from realistic environments to information retrieval and decision making support. Open source software at the same time has suffered by commercialization and licensing from companies that change its free distribution to expensive packages based on proprietary software. The current state of the software market has led researches and independent developers in the direction of new open source alternatives for their Virtual reality needs. As with most open source software

out there, VR open source software in general lacks proper documentation. The aim of the system proposed in this paper has been to provide developers with a system that can be tailored to their needs, well documented, and upgradeable. The GPS-MIV provides the tools needed to port existing OpenGL applications to this new system abstracting must of the work. The system proposed in this paper provides an inexpensive alternative to a full sized CAVE, and thanks to the optimizations used it can run on one single machine reducing much of the cost.

REFERENCES

1. J. Shotton, M. Cook, T. Sharp. "Real-Time Human Pose Recognition in Parts from Single Depth Images." *Microsoft* [Online] Available: <http://research.microsoft.com/pubs/145347/bodypartrecognition.pdf>.

2. J. Anderson, J. Colvin, N. Tobler. " Productivity and Multi-Screen Displays ." *University of Utah* [Online] Available: <http://www.ergotron.com/Portals/0/literature/whitePapers/english/Multi-Mon-Report.pdf>.

3. J. Symanzik, D. Cook. (20011). "Dynamic Statistical Graphics in the CAVE Virtual Reality Environment," Iowa Center for Emerging Manufacturing Technology, Iowa State University, Ames, IA, 50011

4. F. P. Brooks, Jr. (1999, DEc.). "What's Real About Virtual Reality?" *IEEE Xplore.* [Online]. 19. (6), 16-27. Available: < http://faculty.utpa.edu/fowler/VR-papers/Brooks_1999_WhatRealAboutVR_Computer.pdf>

5. J. Yi, Y. Kang, J. Stasko, "Toward a Deeper Understanding of the Role of Interaction in Information Visualization" presented at IEEE Symposium on Information Visualization (InfoVis '05), pp. 111-117, 2007. [Online] Available: <http://www.cc.gatech.edu/~stasko/papers/infovis07-interaction.pdf>

6. S. Card, P. Pirolli, J. Mackinlay. (1994, April.)."The Cost-of-Knowledge Characteristic Function: Display Evaluation for Direct-Walk Dynamic Information Visualizations" *ACM Conference on Human Factors in Computing Systems.* [Online]. Available: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.2797&rep=rep1&type=pdf>

7. A. D. Mauro. (2009.). "Virtual Reality Based Rehabilitation and Game Technology" *eHealth & Biomedical Applications Vicomtech* [Online] Available: http://ceur-ws.org/Vol-727/eics4med9.pdf

8. M. C. Rinard. (2012, Dec.). "Technical Perspective: Example-Driven Program Synthesis for End-User Programming." *Communications of the ACM.* [Online]. 55. (8), 96. Available: <http://ezhost.utpa.edu:2845/magazines/2012/8/153802-technical-perspective-example-driven-program-synthesis-for-end-user-programming/fulltext>

9. M. Slater, V. Linakis, M. Usoh, R. Kooper. (2006). "Immersion, Presence, and Performance in Virtual Environments:An Experiment with Tri-Dimensional Chess" *Department of Computer*

Science University College London, [Online]: Available: <http://reference.kfupm.edu.sa/content/i/m/immersion__presence__and_performance_in__388082.pdf>

10. University of North Carolina (1997). "StereoGraphics Developers' Handbook", *North Carolina at Chapel Hill* [Online]: Available: <http://www.cs.unc.edu/Research/stc/FAQs/Stereo/stereo-handbook.pdf>

11. Bowman, D. A. et al. (2006). "3D user interface: New Directions and Perspectives", International Journal of Virtual Reality. [Online]: Available: <http://faculty.utpa.edu/fowler/VR-papers/Bowman_2006_New-Directions-in-3D-User-Interfaces_IJVR.pdf>

12. D.. Bowman, E. Kruijff, Joseph J. LaViola, Jr., I. Poupyrev et al. (2006). "3D user interfaces Theory and practice", Addison-Wesley [Online]: Available: < http://ptgmedia.pearsoncmg.com/images/9780201758672/samplepages/0201758679.pdf>

13 J. Derry. (1995, Oct.). "Virtual and Real Realities". *Fiction-Mediated Communication:* [Online]. Available: < http://cas.illinoisstate.edu/english/mediations/deery.html>

14. J. Suler. (2004). "Computer and cyberspace addiction". *International Journal of Applied Psychoanalytic* [Online], 359-362. Available: <http://users.rider.edu/~suler/psycyber/cybaddict.html>

15. G. C. Burdea, "Output devices: Graphics, Three-dimensional sound, and haptic displays," *in Virtual Reality Technology, 2*th ed. USA: Wiley, 2003, ch. *3*, pp. *58–111.*

16. D. Rowan. (2010, Oct.). "Kinect for Xbox 360: The inside story of Microsoft's secret 'Project Natal' " *Wired Magazine.* [Online]. Available: http://www.wired.co.uk/magazine/archive/2010/11/features/the-game-changer

17. R. M. Wham. (2012, Jun.) "Three-Dimensional Kinematic Analysis Using the Xbox Kinect". *Tennessee Research and Creative Exchange.* [Online]Available: <http://trace.tennessee.edu/cgi/viewcontent.cgi?article=2555&context=utk_chanhonoproj>

18. Tracy Samantha Schmidt. (20011). "Is the Wii Really Good for Your Health?". *Time Magazine.* [Online] Available: http://www.time.com/time/business/article/0,8599,1584697,00.html

19. R. M. Satava. (1999, Feb.). "MEDICAL APPLICATIONS OF VIRTUAL REALITY". *Medical IVR.* [Online] Available: <http://www.neurovr.org/pdf/papers/VR_Clinical/MedicalVR.pdf>

20. A. Shendarkar. (2006, Jun.). "CROWD SIMULATION FOR EMERGENCY RESPONSE USING BDI AGENT BASED ON VIRTUAL REALITY". *Winter Simulation Conference.* [Online] Available: <http://www.informs-sim.org/wsc06papers/067.pdf>

21. ADAM GORLICK . (2011, April). “New virtual reality research”. *Sanford News.* [Online] Available: <http://news.stanford.edu/news/2011/april/virtual-reality-trees-040811.html>

22. "The Trusted Leader in High Performance Computing." *SGI*. N.p., n.d. Web. 16 Nov. 2014.

23. Fanelli, Gabriele. "Random Forests for Real Time Head Pose Estimation." *Random Forests for Real Time Head Pose Estimation* . [Online] Available: <http://www.vision.ee.ethz.ch/~gfanelli/head_pose/head_forest.html>.

24. Shotton, Jamie, Mat Cook, and Toby Sharp. "Real-Time Human Pose Recognition in Parts from Single Depth Images." *Microsoft*. [Online] Available: <http://research.microsoft.com/pubs/145347/bodypartrecognition.pdf>.

25. Newcombe, Richard A., ShahramIzadi, and OtmarHilliges. "KinectFusion: Real-Time Dense Surface Mapping and Tracking∗." *Microsoft*. . [Online] Available: <http://research.microsoft.com/pubs/155378/ismar2011.pdf>.

26. Inthanayothin, Chanjira, NonlapasWongwaen, and WisarutBholsithi. "Skeleton Tracking Using Kinect Sensor & Displaying in 3D Virtual Scene." *AICIT* . [Online] Available: <http://www.aicit.org/IJACT/ppl/IJACTVol4No11_Part23.pdf>.

27. Sko, Torben, Henry Gardner, and Michael Martin. "Studying a Head Tracking Technique for First-Person-Shooter Games in a Home Setting." *IRIT*. . [Online] Available: <http://www.irit.fr/recherches/ICS/events/conferences/interact2013/papers/8120247.pdf>.

28. Erichsen, Martin H., and Peter H. Poulsen. "Virtual Reality: A Study On Perception." 01 June 2012. [Online] Available:. <http://image.diku.dk/sporring/files/poulsen.erichsen120619.pdf>.

29. Li, Songnan, King NgiNgan, and Lu Sheng. "A Head Pose Tracking System Using RGB-D Camera. [Online] Available:. <http://www.ee.cuhk.edu.hk/~snli/ICVS2013.pdf>.

30. Bowman, Doug A., Sabine Coquillart, and Bernd Froehlich. "3D User Interfaces: New Directions and Perspectives." N.p., Nov.-Dec. 2008. Web. <http://faculty.utpa.edu/fowler/VR-papers/Bowman_2008_3dUIs_NewDirectionsPerspectives_Computer.pdf>.

31. Wingrave, Chadwick A., Brian Williamson, and Paul Varcholik. "The Wiimote and Beyond: Spatially Convenient Devices for 3D User Interfaces."

32. Xia, Lu, Chia-Chih Chen, and J. K. Aggarwal. "Human Detection Using Depth Information by Kinect." . [Online] Available: <http://cvrc.ece.utexas.edu/Publications/HAU3D11_Xia.pdf>.

33. Choppin, Simon. "Kinect Biomechanics: Part 1." Engineering Sport Blog. The Centre for Sports Engineering Research, 9 May 2011. Web. [Online] Available: <http://engineeringsport.co.uk/2011/05/09/kinect-biomechanics-part-1/>.

34. "Skeletal Tracking Fundamentals (Beta 2 SDK)." Channel 9 - MSDN. Microsoft. Web. 15 Feb. 2012. [Online] Available: <http://channel9.msdn.com/Series/KinectSDKQuickstarts/Skeletal-Tracking-Fundamentals>.

35 Moore, G.e. "Cramming More Components Onto Integrated Circuits." *Proceedings of the IEEE* 86.1 (1998): 82-85. Print.

36. Schmalstieg, D.; Wagner, Daniel, "Experiences with Handheld Augmented Reality," *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* , vol., no., pp.3,18, 13-16 Nov. 2007 doi: 10.1109/ISMAR.2007.4538819

37. N. Polys, S. Kim and D. Bowman, "Effects of information layout, screen size, and field of view on user performance in information-rich virtual environments," *Proc. ACM Symp. Virtual Reality Software and Technology*, pp. 46-55, 2005.

38. D. Pape, J. Anstey and G. Dawe, "A low-cost projection based virtual reality display," *Proc. SPIE (International Society for Optical Engineering),* vol. 4660, pp. 483-491, 2002.

39. C. Cruz-Neira, D. Sandin and T. DeFanti, **"**Surround-screen projection-based virtual reality: The design and implementation of the CAVE," *SIGGRAPH '93 Proc. 20th Ann. Conf. on Computer Graphics and Interactive Techniques*, pp. 135-142, 1993.

40. Henderson, D. A., Jr. and Card, S. K. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. ACM Transactions on Graphics 5 (3, July 1986)., 211- 243.

41. A. Buja, J. A. McDonald, J. Michalak, and W. Stuetzle, "Interactive data visualization using focusing and linking," presented at IEEE Conference on Visualization (Visualization '91), San Diego, California, pp. 156-163, 1991.

42. A. Dix, J. Finlay, G. D. Abowd, and R. Beale, "Human-computer interaction", 3rd ed: Pearson Prentice Hall, 2004.

43. N. A. Dodgson, J. R. Moore, S. R. Lang, "MULTI–VIEW AUTOSTEREOSCOPIC 3D DISPLAY" , University of Cambrige Computer Laboratory [Online] Available: <http://www.cl.cam.ac.uk/~nad10/pubs/IBC99-Dodgson.pdf>

44. G. R. Little, S. C. Gustafson & V. E. Nikolaou, 1994, "Multiperspective autostereoscopic display", Proc. SPIE, 2219, 388–394.

45. F. Guimbretiere, M. Stone, and T. Winograd, "Fluid Interaction with High-resolution Wall-size Displays," Proceedings of ACM Symposium on User Interface Software and Technology, Orlando, FL, 2001, pp. 21-30.

APPENDIX A

APPENDIX A

API Overview and Tutorial

The General Purpose System for Multi-display Interactive Visualization provides its own API to abstract many of the steps required to set up a virtual environment such as: registering a window, setting up a projection type, setting up transformations for the openGL pipeline, and more. **Setting up windowed mode**

We begin by looking at the class GLContext, this class is fundamental for the system discussed in this paper.

```
GLContext        guiContext(0);
```

We have the option to choose between windowed or full screen mode. by initializing a class instance with either **0** for windowed or **1** for full screen. In the example above we have created an application in windowed mode. Next, we need to register our class with windows by using the command **RegisterMyClass(hInstance)**, where hInstance is of type HINSTANCE.

Then, we create the specified window using the following command:

hWnd = CreateMyWindow(hInstance, TEXT("GUIWindow"), 1080, 1000) and hand it over to our GLContext by using the next command:

```
guiContext.Initialize(hWnd)
```

In summary, we need four commands to set up or window. They are as follows:

```
GLContext        guiContext(0);
```

RegisterMyClass(hInstance);

hWnd = CreateMyWindow(hInstance, TEXT("GUIWindow"), 1080, 1000);

guiContext.Initialize(hWnd);

Setting up full screen mode

The real strength of the system comes from creating multiple full screen viewports of the system and setting them up in the desired virtual location and direction. This is analogous to having a set of cameras and getting ready for a photo shoot by placing them around the studio.

GLContext guiContext2(1);

RegisterMyClass(hInstance);

int counter= MonitorCount();

hWnd2= CreateFullScreenWindow (hInstance,0);

guiContext2.Initialize(hWnd2);

guiContext2.EnableStereo();

guiContext.Draw(draw);

The first change compared to the windowed initialization is that we set the initialization parameter for GLContext to **1,** this will prepare our GLContext to work in full screen mode. The second like will stay the same as before. Next, we use MonitorCount() to query the operating system for the number of monitors and the size of each in pixels. The fourth line creates the window and requires an HINSTANCE parameter and the monitor number. In this particular example we set the full screen to be on monitor 1 represented by 0, So if we have 4 monitor they

start from 0 to 3. The fifth like starts the rendering of graphics in or full screen. The line **guiContext2.EnableStereo()** is optional, it enables the stereo 3D in that particular screen. The last line takes as a parameter a function, this function sent as argument should contain all the OpenGL commands needed to create a virtual environment. This is particularly helpful since the developer can use the start up application provided with the system and just work within the draw function without having to worry about any other code, but the one that pertains to that particular application and OpenGL. This capability allows for easily translating OpenGL code form other existing applications to this system taking advantages of all the added functionality,

Note: stereo 3D will not work in windowed mode.

.OBJ loader

Virtual environments are composed of 3D models. These models are usually created using specialized modeling software such as Maya or 3ds max. It results impossible to attempt to model complex geometry using only OpenGL commands, of immersive environments are to be supported by the system presented in this paper; integration with .obj files is necessary as it is the most commonly supported 3d format.

The system provides a specialized class for importing models from .obj format to OpenGL applications with a couple lines of code. The special class is named **Model**, it holds information about vertices, normals, elements. The following segment of code shows how to initialize an instance of the Model class:

```
Model myModel;
myModel.load_obj(“myfile.obj");
```

myModel.draw();

As can be seen from the previous example it is easy to import a model from any software with .obj support. The first line declares and instance of the class Model, the second line reads the information from the file and populates the variables in our instance, and the last line can be call anywhere to draw our model.

It is important to note that this functionality is one of the main advantages of the system, since it provides the means to incorporate a plethora of already existing 3D models from a wide range of applications, from AutoCAD files to blender designs.

As of this writing the system supports flat shading when lighting the models, this is a basic type of shading. Shading is implemented in drawing a scene and depicting levels of light and darkness on screen, in computer science in particular shading refers to changing the color of a 3d model in the scene. This is done by calculating the angle of intersection of the object and the source of light. The shading in general is computed in the rendering process of the application called a shader.

The system has been created with upgradability in mind, and with some more development time it can be updated to incorporate different types of shading. The next image shows some of the better known shading models.

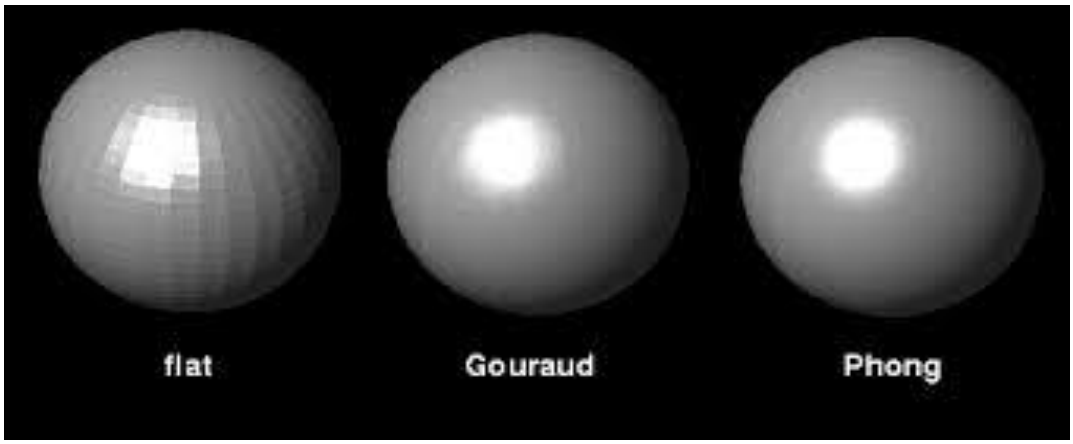

Figure 18 Shading Models

The flat shading technique is used to light each polygon based on its normal and the angle between the normal and the light source. It is used for efficiency and high speed rendering where other methods are too computationally expensive.

Smooth shading rather than assuming that light changes per polygon it changes it per pixel, it also assumes that a surface is curves and uses interpolation to calculate the value for a pixel between the vertices.

Gouraud shading determines the normals at the vertices of each polygon, applies an illumination model to all vertices to calculate vertex intensity, and interpolates the vertex intensity over the surface polygon.

Camera

To interface and coordinate the multi-screen support a camera control system has been implemented, this control allows for each glContext to have control of where the camera is placed and where it is pointing to. Because the camera move in first person view extensive

computation is done to coordinate depending on the number of current screens and tracking input. The Camera rotates all the view ports based on the input received be it through a depth sensor or mouse and keyboard. This Camera class is hidden from the user to facilitate the use of the API. The camera functions are linked to methods that are part of the glContext class. They can be accessed by calling **"glcontext.cam."** for instance to rotate a particular camera you would use the command "glcontext.cam.Rotate(angle)" following is a list of the methods available as of this writing:

```
void MoveFront();
 void MoveBack();
  void MoveLeft();
  void MoveRight();
  void RotateLeft();
  void RotateRight();
  void Rotate(int r);
  void Setoffset(int r);
  void Rotate( float a);
```

Vertex Rendering

When OpenGL was first created in1998 by Khronos Group, it utilized a pipeline based on fixed-functions that permitted the developers to interact with the graphics hardware at very low level by typing OpenGL functions that were predefined. This first approach introduced was not very flexible and constricted what could be achieved with the API. The next iteration of OpenGL added a programmable shader pipeline, this meant that the programmer now had the flexibility to

write specialized shaders that were more suited to meet the needs of the program in question. With the arrival of OpenGL 3.2 specification a new shader was added, this shader lets the programmer introduce shaders into the rendering pipeline on the fly and to perform that add complex operations like approximating better curves. Given the new control provided by some of the latest specifications the programmer has to ways of rendering geometry: immediate mode and vertex arrays.

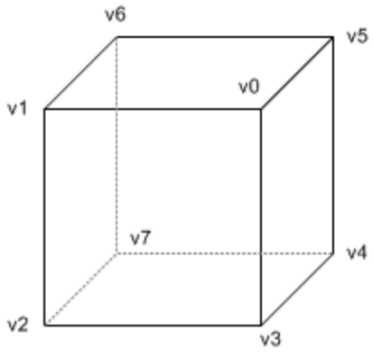

Figure 19 OpenGL Cube

When a program is rendered in immediate mode, all the rendering commands have to be called between a matching glBegin and glEnd function calls, this mode is easy to implement and is very transparent to beginner programmers. The problem with this method is that to create geometry in the scene, redundant function calls are made. This can greatly affect the performance of your application when drawing millions of polygons on screen. If for instance a cube is drawn in immediate, six faces are needed to create the cube. For each face we need three vertex, so in all we need 36 calls to glVertex() function. If a vertex normal and color is applied to the particular face, this furthermore increases the number of function calls. Something to notice on the figure above is that vertices are shared among faces, looking closely at vertex v1 we see

that it is shared with 3 adjacent faces. But because we are in immediate mode that same vertex has to be provided 6 times, greatly increasing the computational cost.

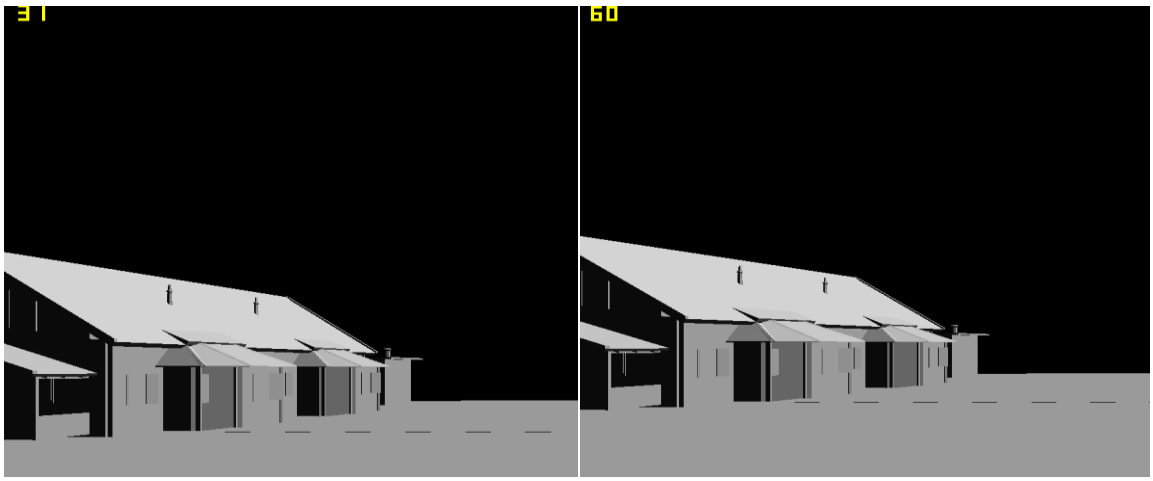

Figure 20 Rendering Models Immediate vs. Vertex Array

The system described in this paper uses an alternate method known as vertex arrays which reduces the number of times functions and vertices are called. Eliminating redundant function calls can greatly increase the frame rate of the program in question creating a smoother experience for the user. Since the storage of vertex arrays is done on the client side and OpenGL gets access to them from the server side, we have to use different functions as to glBegin and glEnd. The next figure shows the frame rate of the same virtual environment being render in immediate mode vs. vertex arrays.

BIOGRAPHICAL SKETCH

Irving Alan Gonzalez earned his Bachelor of Science in Computer Engineering degree from The University of Texas-Pan American in 2012. He received his Master of Science degree in Computer Science in 2014 from The University of Texas-Pan American. Irving Alan Gonzalez has been recipient of numerous honors and awards including the Xerox Scholarship and awarded the Dean's honor list multiple occasions. During his time as a graduate student, he worked as a Teacher Assistant and Research Assistant for the Department of Computer Science. After he graduates in December 2014, Irving Alan Gonzalez will move to Austin to work as a Software Developer. You may contact Irving Alan Gonzalez at iagonzalez2@broncs.utpa.edu.