Proceedings of the 5th Winona Computer Science Undergraduate Research Symposium

April 20-21, 2005

Table of Contents

Simulation of an Embedded Sniffing Device

Greg Bestland
Student Winona State University
4351 Chester Ct
Webster MN, 55088
507-459-3097

Greg.Bestland@Gmail.com

ABSTRACT

This paper describes the first phase of a two phase project to implement an IMAP protocol analyzer on an embedded device. The first phase is the implementation and testing of a simulated embedded email sniffer, as a proof of concept. The test bed for the simulated embedded device is modeled after the Digi© Connectsp platform. The Connectsp will later be used for the final implementation of the sniffer. We tested the simulated embedded sniffer's ability to detect outgoing, and incoming IMAP messages, in a network environment under various loads ranging from 0Mbs to 79Mbs. We concluded that such a device would be very effective and entirely feasible.

General Terms

Algorithms, Measurement, Performance, Reliability, Experimentation, and Security.

Keywords

IMAP, Embedded, Email, Network Sniffing.

1. INTRODUCTION

The idea of monitoring the activities of people via email is not a new concept. The FBI's Carnivore Project was one of the first that got national attention [5]. Clearly the task of reading unencrypted email is trivial and not worthy of study. This is of course given the condition that you have access to the network traffic carrying the email message. This can be a difficult task if the network traffic is solely internal.

In the instance of large corporations, or other organizations such as a university, a large portion of the network traffic can be internal. This includes email coming from and destined to only internal users. That makes this traffic very difficult to intercept, unless you have physical access to the internal network. While setting up a laptop or desktop computer on a secure internal network is one solution, it is certainly not ideal if a covert solution is necessary. This is where an embedded sniffer's usefulness becomes apparent.

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

The goal of this paper was to complete the first phase of a two phase project. This project's objective is to develop and test the effectiveness of an IMAP protocol analyzer implemented in an embedded device. The embedded sniffer could be a miniature computer, smaller than a deck of cards, which could be placed on an internal network between an IMAP client and IMAP server. Here it would be capable of capturing emails that are sent to the server from the client and to the client from the server. The first phase of the project will be covered in this study. It involves the implementation and testing, of a simulated embedded IMAP sniffer.

In this paper we will explain the underlying network protocols relevant to sniffing IMAP traffic and how these protocols build upon each other to form network traffic. We will present the basic function and design involved in developing a sniffer and the additional implementation details and obstacles that had to be considered when developing a simulated embedded sniffer. We will discuss how we set up our test bed and tests for the embedded device. Lastly we will examine the results of these tests and discuss them.

2. BACKGROUND

In this Section we will go over how network traffic is formed, how sniffers work, why an embedded sniffer requires a slightly different approach than is typically taken with most sniffers and a little bit about the IMAP protocol.

2.1 Network Layers

Network traffic is made up of many different layers. The current working model for most applications on the Internet today is the IP model [13]. The IP model is made up of four layers, as follows: host to network, internet/network, transport and application [1]. The host to network layer is used to transport bits from point to point [1]. The internet/network layer is used to route information [13]. The transport layer is used to manage end to end flow and error control [13]. The application layer is used to turn the bits one party receives from another party into something useful [13]. Each layer has protocols associated with it. While it is beyond the scope of this paper to cover even the most common protocols in their entirety, we will briefly go over the ones we used in our project.

In the first layer, the host to network layer, there is overwhelmingly one standard in use today, Ethernet (IEEE 802.3) [1]. The Ethernet header consists of a source and destination address (each is 48 bits long), a type code, data, and a CRC checksum [13]. You can see the layout of the Ethernet header in Figure 1. Enclosed within this frame's data section is the protocol information of the other layers.

Preamble	Destination Address	Source Address	Frame Type	Frame Data	CRC
64 bits	48 bits	48 bits	16 bits	368-12000 bits IP Packet	32 bits

Figure 1: Ethernet Frame [13]

The internet/network layer is dominated today by the IP protocol. There are two versions of IP currently in use, IPv4, and IPv6. While IPv6 looks to be the new and upcoming standard, most people are still using IPv4. The IPv4 header consists of 13 different fields spread over 196 bits. Following the IPv4 header fields is the data that is encapsulated within the IP packet [1,13]. For the purposes of this paper, only three fields of the header are relevant. The source and destination addresses (32 bits each), and a protocol indicator consisting of 8 bits, this field is bolded in Figure 2 [1, 13]. The addresses on this level are end-to-end, as opposed to the Ethernet addresses which are point to point.

IP Packet Format

0		16	31
Version (0100)	Length (header)	Type of Service	Total Length
Identification (sequence number)		Flags	Fragment Offset
Time to Live (TTL)	Protocol (upper level)	Header Checksum	
Source IP Address			
Destination IP Address			
Options			Padding
Data			
...			

Figure 2: IPv4 Packet [13]

The next level of abstraction is the transport layer. Unlike the first two protocols, there are several different transport protocols in use today. TCP is the only one relevant to our project. The TCP header consists of 13 fields; these are shown in Figure 3. This protocol handles end to end flow control of data and error checking. Its function is to ensure data transmitted from the source is received in order and error free by the destination [1, 13].

Source Port			Destination Port
Sequence Number			
Acknowledgement Number			
HLEN	Reserved	Code Bits	Window
Checksum			Urgent Pointer
Options (If any)			Padding
Data			
...			

Figure 3: TCP Header [13]

The top level of the IP model is the application layer. The application layer protocol we will be sniffing is IMAP, or the Internet Message Access Protocol [7]. The IMAP protocol is used in conjunction with SMTP. SMTP messages arrive to the IMAP server, the server then stores those messages for users to retrieve at a later time. The IMAP server is not responsible for generating outbound SMTP traffic, that task is left up to the client. IMAP uses simple plain text messages to communicate and retrieve email from a central storage server, this process is show in figure 4 [7]. While we will not go into the logistics of exactly what commands are used, the only pertinent information about this protocol is that the client and server use a series of command strings in order to authenticate and exchange plaintext messages. The two command strings that can contain the body of an email message are the FETCH, and APPEND messages. The FETCH method is used to retrieve a specific message from the server [7]. The APPEND message is used to add a message to the server for storage [7].

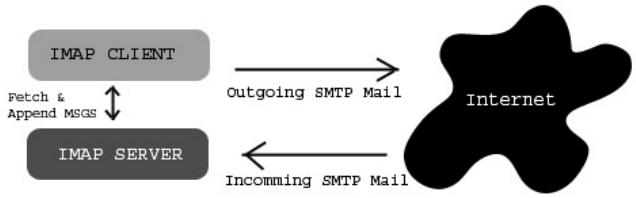


Figure 4 : IMAP Protocol

IMAP messages must be encapsulated in TCP, then IP and then Ethernet in order to transmit the message successfully to its destination [1, 13, 7]. In Figure 5 an overview of this encapsulation process is presented, with the highest level protocol, IMAP, at the center, and the lowest level protocol, Ethernet, on the outside.

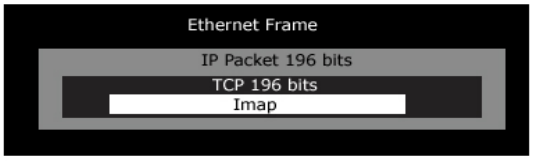


Figure 5: IMAP Encapsulation

2.2 Sniffers

Sniffers, also known as protocol analyzers, come in many different varieties and serve a multitude of purposes. From the network administrator trouble shooting network problems, to an identity thief trying to get your social security number, all network sniffers share two common components, a tap and a filter [14].

The tap is a mechanism that allows the sniffer to receive all traffic. This is most commonly done by setting the NIC (network interface card) to a promiscuous mode [14]. Promiscuous mode is a mode in which the card disregards the destination portion of the Ethernet header and accepts all traffic. In normal operation, a NIC will only accept traffic that is destined to its particular physical address. The destination physical address is contained in the Ethernet header, and is highlighted in Figure 1. In Windows, Linux and UNIX environments, setting the NIC to promiscuous mode is commonly accomplished with the use of a packet capture library (PCAP). PCAP, originally developed for UNIX, was ported to Windows and its name changed to WINPCAP [11]. This was further extended to Java with the JPCAP library [12]. We used a WINPCAP base with JPCAP extensions in order to capture all frames for our experiments.

Another consideration for the sniffer is it needs to be in a position such that it is able to intercept the target traffic. A tap does no good if there is no network traffic to tap into. This is not a trivial thing on a switched network where a node generally only receives traffic that is intended for it. This means the sniffer must be located on a layer one hub between the target and the target's destination or on a switch that is mirroring a port carrying the data between client and server; this can be seen in Figure 5. Once all of the traffic has been captured, a filter needs to be employed.

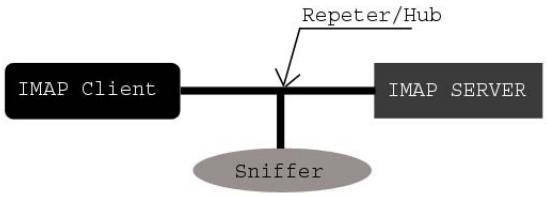


Figure 5: Sniffer Location

The filter takes the fire hose of data flowing into a computer and reduces it to data pertinent to the task that is to be accomplished. Depending on the application, this data stream can be quite large or quite small. There are two main approaches to filtering [14]. The first is to collect all the traffic and store it for later processing. The second is to process everything on the fly and store only relevant information for later.

The first approach is the most popular. One of the most infamous sniffers was the FBI's Carnivore. This was implemented in the form of a laptop sitting at an ISP (Internet Service Provider) capturing bit for bit the information destined for and coming from a subject's computer. This data was stored for later processing [5]. This method has the advantage of capturing all network traffic, but the disadvantage of taking up a tremendous amount of space.

The second approach is much more suitable to sniffing with an embedded device. Instead of saving all the traffic, all data that is not relevant to application is filtered out and discarded entirely. An embedded sniffer requires the second approach. The primary reason for this is the very limited storage space. The embedded device that was used for our simulated model has only 2MB of flash memory [4]. While this is sufficient to store quite a sizeable amount of plaintext email, it is not enough to store one minute of the total traffic that a user may generate. This requires the device to filter out quite a bit of traffic. The filtering process requires far less processing power than one may think. Most applications running over TCP have a designated port number associated with them [13]. Since the headers of all layers are known, and the destination port of the application is identified, sorting out relevant traffic becomes trivial. We can look at the protocol section of the IPv4 header to determine if the upper level protocol is TCP, bolded in Figure 2, then the destination port in the TCP protocol header is examined, highlighted in figure 3, to determine if the traffic is pertinent information to the user. Since the location of this data is known, the comparisons can be done with bit masks, making them non-processor intensive [13]. Now that there is an understanding of how sniffers obtain and filter traffic, the next step is to examine our evaluation of the simulated embedded sniffer.

3. METHODS

In order to test the sniffer's effectiveness, four different elements are needed, a sniffer, to capture and sift through the traffic, an IMAP client to send and receive messages from a server, an IMAP server to serve the messages to a client, and a mechanism to generate network traffic in a controlled manner. We implemented a sniffer, and IMAP client. The IMAP server and network traffic generator are third party applications.

3.1 Experiment Setup

The hardware used during the test consisted of two desktop computers, one laptop and a layer one hub, or a layer two switch. The first desktop served as an automated IMAP client and FTP client. The processor for this computer was an AMD 64 3200+ with 1GB of memory. The second desktop served as an IMAP server and FTP server. Its processor was an AMD XP 2700 with 1 GB of memory. The laptop was a Pentium 4 clocked at 1.4 Ghz and contained 768 MB of memory. The layer one hub was 10 base T for the test that involved network speeds between 0 and 10Mbs. A 100 base T switch was used for tests that had network speeds exceeding 10Mbs.

The testing was set up in such a way that the sniffer would intercept all traffic which passed between the IMAP client and server. This was done on an isolated network through the use of a physical repeater (dumb hub), or at line speeds above 10Mbs through a switch that was using port mirroring. This simulates a wiretap tool that would be used to hook the embedded device into the network. This layout is depicted in Figure 5.

In order to be repeatable, all the elements of the experiment must be controlled. Obviously simply hooking the sniffer up to a network segment and having a person fetch and append IMAP messages via an IMAP client would not only be tedious but also inconsistent. In order to automate the process we implemented an IMAP test client. The client was written in Java and utilizes the JavaMail API [10].

The test client is designed to send and receive IMAP traffic at fixed timed intervals. Our tests consisted of the sending and receiving of 100 IMAP paired messages on 5 second intervals. Each paired messaged was made up of a FETCH message and an APPEND message [7]. Twenty of these tests were preformed. Each test was performed at a different level of network saturation. The first test was performed with no significant extraneous network traffic and the last with maximum network traffic we were able to produce 79.2Mbs.

The IMAP server we choose to use was Mailtraq [9]. It is commercially available and has a freeware version accessible with a limited number of clients. It fully complies with the IMAP protocol (RFC, 3501) and was chosen due to ease of configuration and our funding constraints. The traffic generator we used to generate the desired line speeds was a simple FTP client and server combination. The FTP server used was BulletProof FTP, and the client was Smart FTP [16, 17]. Again these were chosen due to our funding constraints and its ease of configuration. The network load generated by the FTP client and server was verified with a network load utility called Du Meter [15].

3.2 Simulated Embedded Sniffer Implementation

The simulated embedded sniffer was implemented using Java and a combination of the JPCAP and WINPCAP libraries [11, 12]. The simulator was based on the development board manufactured by Digi called the Connectsp. The Connectsp will be used in the

second phase of our project as a test bed on which to implement the sniffer. This device is very small, about the size of a pack of cards. It has built-in Ethernet, TCP/IP functionality, a simulated file system using flash memory, and file input/output capabilities. Its processor is an ARM7 running at 55.7 MHz. The long-term storage capacity of the device is very limited, around 2MB of flash memory. Its main memory capacity is 16mb [4].

In order to effectively simulate the embedded device we started with a laptop running at 1.4Ghz. In order to make the laptop more accurately reflect the performance of the embedded device we had to limit both the memory usage and CPU speed. The CPU speed was restricted by an application called CPU grabber [18]. This application utilizes a certain percentage of the available CPU resources, which can be determined by the user. In order to better simulate the embedded device we chose to make use of 97% of the CPU, leaving only 3% free for the simulated embedded device. This simulated a device running at 42Mhz. We were unable to get a clock speed closer to that of the actual device, 55.7Mhz, due to the limitations of CPU grabber. We decided to use the closet setting that was at a performance equal to or less than that of the device. This clock speed was verified by monitoring the Windows task manager. After running a few initial tests we noted that the program itself never consumed over 11Mb of memory. Due to the fact that the simulator ran well within the bound of the embedded device, no action was taken to limit its memory, as none was needed. (CPU Grabber).

As with all sniffers we needed to implement both a tap and filter. The tap was implemented using the JPCAP library [12]. This allowed us to choose a NIC, and set it to promiscuous mode. It also allowed us to set up a filter. Due to its limited processor power and storage capacity, it is imperative that no unwanted information be captured by the device. In order to ensure this we implemented a two stage filter.

The first stage of the filter had to filter out the maximum amount of traffic as computationally efficient as possible. This was accomplished through the use of bit masks. Since the headers of the protocols are fixed, and known ahead of time, a mask can be applied to all incoming traffic [13]. This allows us to use the non-computationally intense xor operations to filter out the bulk of the traffic. In this manner we can separate out all traffic that is not TCP and destined to or originating from port 143. You can see that all the fields where we applied bit masks are bolded in Figures 2 and 3.

The second stage of the filter was applied to all the remaining IMAP traffic. Since the only IMAP commands that can trigger the transfer of an email message are FETCH and APPEND it would make sense to use bit masking to separate out all extraneous IMAP messages. We attempted this but found that due to inconstancies in the placement of the FETCH and APPEND command strings, among various clients and servers, it was not feasible. We instead used string pattern matching to search for the initial FETCH and APPEND command strings. If either FETCH or APPEND was located we then looked for a particular email address within the body of the message. If the pattern matching algorithm found the email in the FETCH or APPEND message the body was saved and written to a text file.

It is important to note that the first stage of the filter must be given a higher priority than the second stage of the filter or potential email messages may be lost. This is because the first stage of the filter deals with a much larger volume of traffic, and if traffic is not dealt with in a timely manner, some will be disregarded if the Ethernet queue overflows. This is less important with the second stage of the filter. While the second stage requires more computational time per message analyzed, it is less time sensitive, and can be processed with comparative leisure. We accomplished this prioritization through the use of threads, placing the first portion of the filter on a different thread than the second portion.

4. Results

After running the twenty initial capture tests at different line speeds we came up with the following results.

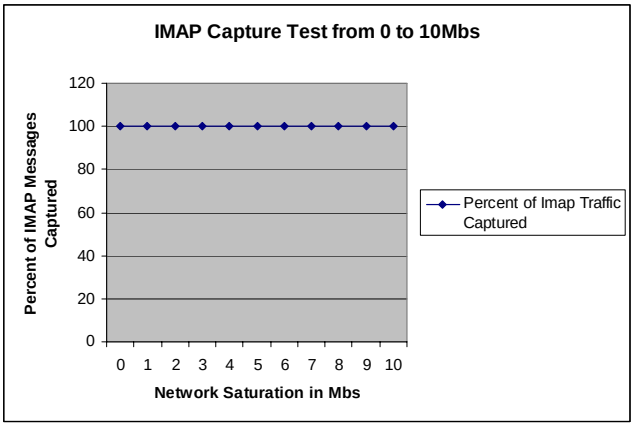


Figure 7: Sniffer Results 0 to 10Mbs

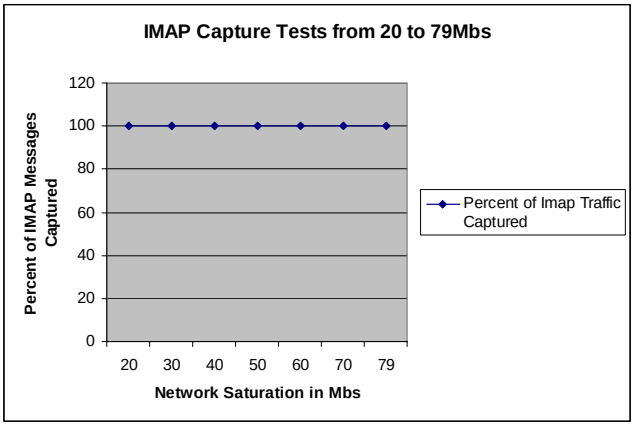


Figure 8: Sniffer Results 20 to 79Mbs

As you can see from the Figures above, the sniffer performed much better than expected. It managed to capture and store all IMAP messages regardless of other network traffic. This exceeded our initial estimates. One thing we did notice however was that the higher the network saturation the more time it took the sniffer to process IMAP messages that made it past the initial bit mask filter.

Although we were satisfied with the results of our tests, we wanted to see exactly how the sniffer would perform in even less ideal circumstances. To accomplish this we removed the timing function of our test client; this caused all email messages to be sent and received at once.

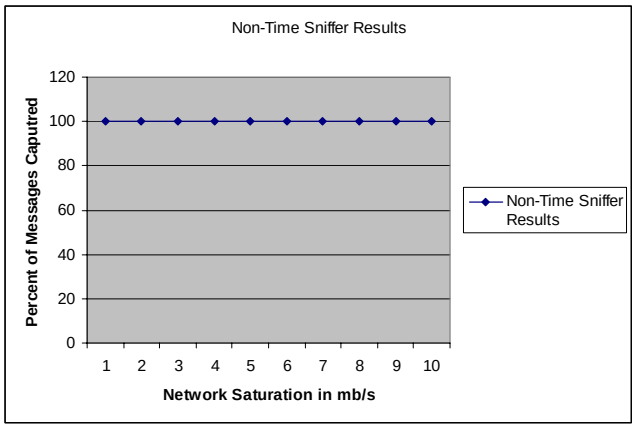


Figure 9 Non-time Sniffer Results

As you can see in Figure 9 the results were surprising. The sniffer managed to capture all relevant messages, even if they were sent with no time interval between them. We also ran one test with the full network saturation of 79Mbs and no timing mechanism. This had the same result as those above. The protocol analyzer captured all 200 email messages. We noted that the sniffer was unable to process all the messages as quickly as they were sent; instead it buffered them and processed them later.

5. CONCLUSION

From these results we have concluded that such a device is entirely feasible to implement. Through the course of testing we have also learned much about such a device's limitations. Our tests have indicated what environment the embedded sniffer would perform best in and what situation is less than ideal for such a device.

While our sniffer managed to capture all traffic in our tests we noticed one interesting trend. The higher the network saturation the more time it took for the sniffer to process IMAP messages that made it past the first stage of the filter. This is because the first stage of the filter runs in a separate thread and is given higher priority than the second stage. In order to keep up with the increased network load the first stage of the filter uses much more of the processor, leaving less for the second stage, which in turn causes the second stage of the filter to run slower. If there was a heavy and consistent stream of IMAP messages in the traffic we were sniffing, it would be entirely possible to overload the second stage of filter by providing it with IMAP messages faster than it could process them. From this we can conclude that the ideal environment for our sniffer is one where there is not a heavy and consistent stream of IMAP traffic. A better environment for the sniffer would be monitoring a limited number of users. This would result in only bursts of heavy IMAP messages, or less intensive consistent IMAP traffic.

This observation does not interfere with our initial concept of an embedded email sniffer. It was never our goal produce a device that would monitor a large number of users email. The embedded sniffer simply does not have the storage capacity, and processor power for such an endeavor. It does however succeed in the task it was designed for, monitoring a limited number of users on an internal network. From these tests we have concluded that our simulated embedded sniffer was a total success and that stage two of our project is entirely feasible.

6. FUTURE WORK

From the results we see here the next step to follow is clear. With a basic prototype developed, the next step would be to implement this application on the Digi ConnectSP platform and see how it performs. As the Digi ConnectSP currently has no ethernet driver capable of promiscuity mode, one would need to be developed.

7. REFRENCES

1. Douglas E. Comer, Internetworking with TCP/IP Principles, Protocols, and Architectures, Alan Apt, 2000.
2. Jonathon B. Postel. RFC 821 Simple Mail Transfer Protocol [online] 2000; Available from: http://www.faqs.org/rfcs/rfc821.html. Accessed 2005 Feb 11.
3. T. Socolofsky, C, Cale. RFC 1180 A TCP/IP Tutorial T. [online] 1991; Available from http://www.faqs.org/rfcs/rfc1180.html. Accessed 2005 Feb 11.
4. Digi International inc, Digi Connect So Hardware Reference 2003.
5. Matt Blaze, Steven M. Bellovin. Inside risks: Tapping on my network door. Communications of the ACM Volume 43, Number 10, Page 136, 2000
6. McClure, Scambray, Kurtz Hacking Exposed, Network Security Secrets and Solutions, Brandon A. Nordin, 1999.
7. M. Crispin. RFC 3501 Internet Message Access Protocol [online] 2003; Available from http://www.faqs.org/rfcs/rfc3501.html. Accessed 2005 Feb 11.
8. J. Myers, M. Rose RFC 1939 Post Office Protocol v3 [online] 1996; Available From http://www.faqs.org/rfcs/rfc1939.html, Accessed 2005 Feb 12.
9. MailTraq Mail Server Software. MailTraq http://www.mailtraq.com/. Accessed 2005 Feb 11.
10. JavaMail API. Sun Microsystems http://java.sun.com/products/javamail/. Accessed 2005 Feb 11.
11. WinPcap Open Source Libaray. WinPcap http://winpcap.polito.it. Accessed 2005 Feb 11.
12. Java Package for Packet Capture. JPCAP. http://netresearch.ics.uci.edu/kfujii/jpcap/doc/index.html, Accessed 2005 Feb 13
13. Gerald Cichanowski, CS 413 course notes, Jan 2005-May 2005.
14. Bob Barr, Sung Yoo, Tom Cheatham. Network Monitoring System Design. Department of Computer Science, Middle Tennessee University.
15. DU Meter Software. DU Meter. http://www.dumeter.com/ Accessed 2005 Feb 14.
16. BulletProof Software. BulletProof FTP Server. http://www.bpftp.com/ Accessed 2005 Feb 17.
17. SmartFTP Software. SmartFTP. http://www.smartftp.com/ Accessed 2005 Feb 17.
18. Microsoft Software. CPU Grabber. www.microsoft.com/ Accessed 2005 Feb 13.

Hierarchical Manipulation of Mathematical Expressions for Visually Impaired Students

M. Fayezur Rahman
Department of Computer Science
Winona State University
Winona, MN 55987
webplots@yahoo.com

ABSTRACT
Students with visual impairments face unique challenges in reading, writing, and manipulating mathematics. This paper describes a tool that lets students organize mathematical expressions in tree structures and manipulate the tree towards solving a mathematical problem. This study shows that visually impaired students can organize mathematical expressions independently, can easily find mistakes in a previously worked problem, and are less likely to misinterpret an expression while solving a problem using this tool.

Categories and Subject Descriptors
G.4 [Mathematical Software]: Documentation, User interface

General Terms
Human Factors, Documentation

Keywords
Math accessibility, visually impaired, navigation, hierarchical structures

1. INTRODUCTION
Mathematics is an inherently visual form of communication. In most languages, speech is the preferred from. A message can be successfully conveyed from one individual to another without the help of the written form. The written form serves the purpose of documentation that can later be understood with the knowledge of pronunciation and prosody. Speech to text and vice versa, quite often, is a one-to-one mapping [1]. This property of common languages makes text-to-speech possible and very successful.

Writing is the preferred form for mathematics. Often times, it is difficult to discuss mathematics in a linear fashion without the high risk of misinterpretation. A simple example can demonstrate this. If one says, "x raised to the power p plus one" - this can be interpreted in two different ways, as shown in Figure 1.

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

(a) **(b)**

Figure 1: Different interpretation of same content.

This confusion can be overcome by saying "x raised to the power quantity p plus one" for Figure 1a or "quantity x raised to the power p plus one" for Figure 1b or by changing different attributes of the audio, for example, pause or pitch. However, this requires the text-to-speech system to understand the content it is reading, an attribute not necessary for ordinary text-to-speech systems.

Over the years many solutions have been imposed on this problem. A successful attempt to read mathematics non-linearly was made by T. V. Raman in his doctorate dissertation AsTeR or Audio System for Technical Reading [2]. AsTeR supports audio formatting that can speak a mathematical expression (ME) as a person with math knowledge would read it. It also supports navigation and abstraction of MEs, which facilitate understanding. UMA or Universal Mathematics Access [3] enables a reader to convert a ME in a preferred format such as Nemeth Code. The difficulty of writing mathematics in word processors was addressed by Knuth in project TeX [4]. LaTeX [5] (an extension of TeX) and MathML are used as a required input for AsTeR and UMA. The developments of these tools have made mathematics easier to read and understand for the visually impaired professionals. However, the technologies of manipulating mathematics for visually impaired students are still primitive. More often than not, visually impaired students have to rely on word processors, or word-processor-like tools to work on their math problems.

Word processors are linear in nature. This makes it difficult for visually impaired students to write and manipulate mathematics, a multi-dimensional language. Use of an intelligent text-to-speech system such as AsTeR requires students to type in or provide the input in LaTeX. This can be a learning over head. Further, "audio formatting makes much information accessible, but not necessarily digestible." [1]

This study proposes an environment that lets a student input MEs in a hierarchy, navigate the hierarchy in an efficient and accessible way, manipulate the content and structure of the hierarchy that leads towards solving a problem, and record each step of the process with a facility for fixing errors or changing strategies. The tool is intended for visually impaired students in high school and college. But anyone with basic understanding of a tree structure can be benefited from this. This environment should

help students in dealing with mathematics in their academic and professional life.

For this research, visually impaired students taking math courses were interviewed to learn about current techniques and tools in use. Based on this, a list of functional requirements for the tool was made. Two mathematics professors assessed the desired functional requirements for fairness in a learning environment. Following the edited list, a prototype was developed for the hierarchical manipulation of mathematics. The prototype was then tested by high school and college students for applicability. Two visually impaired college students participated in a later study. A first version of the tool has been developed based on these studies. The following is an elaborate description of prior research, functional requirements, implementation, usability studies, study results, and future works.

2. PRIOR RESEARCH

Hierarchical architectures to present ME have been considered and used in many different tools. One such tool is AsTeR. "AsTeR is a comprehensive speech synthesis system for higher mathematics." [14] It uses "attributed tree structure" "to represent mathematical content." [2] Therefore a user can navigate and read different logical parts of the content by moving among sibling nodes, parent node to child node, and vice versa. These operations offer the user a choice of abstraction and control of information flow. AsTeR requires input to be in "TeX family of markup languages, i.e., TeX, LaTeX, and AMSTeX." [15] TeX is primarily intended for the printing end. Visually impaired students often use different coding schemes, such as Nemeth Code, for working on mathematics problems. Learning TeX family languages for better accessibility or for producing more visually appealing output can be a learning overhead. This issue was addressed by Universal Mathematics Access project in their program Math Genie. [3]

Math Genie is capable of conversion among different formats, such as LaTeX to Nemeth Code or vice versa. It also takes advantage of the hierarchical structure of an ME to give the user a more interactive listening experience. The reading experience is further enhanced by different levels of abstraction and color coded visual output "to accommodate forms of Dyslexia." [3] It has an audio rendering module that supports more natural reading. It allows a student to convert the output of his/her work in a visually appealing format. Therefore, it neither requires a sighted instructor to learn Nemeth Code, nor requires a visually impaired student to learn LaTeX.

Manipulation of mathematical content, as it is required from a student's perspective, was addressed by Win-Triangle [16]. Win-Triangle eliminates the need for the user to know LaTeX or Nemeth Code. It allows a user to give the input in standard Windows Symbols font, including MathType and MT Extra. [16] Some special symbols (e.g. 🖫 is used to indicate beginning of a fraction) are used to markup contents. It supports printing both in visual format and Braille, therefore eliminating any need of conversion. However, the navigation of the content is limited to a linear structure.

Giving a high level view or "glance" of the complex information using "algebra earcons" was attempted by MathTalk project. [9] MathTalk also offers different levels of abstraction. "Any object that groups more than one term together, by either a parsing mark or spatial location is folded-up and referred to only by its name during browsing." [8] Therefore the system creates different levels of abstraction by predefined rules and the user has to explicitly explore each level for details. The attempt is to present the benefits of a hierarchical structure in a simpler linear structure without requiring user to learn tree structure.

The hierarchical presentation of information has become more popular over the years. It has become an integral part of using computers. For basic need such as file management in Windows to communication or entertainment such as managing "buddy lists" in instant messenger, the tree representation is becoming a natural choice. The tree structure is also used in Integrated Development Environments (IDEs) to represent program structure. The Computer Science Curriculum Accessibility Program (CSCAP) investigated accessibility issues in such representation of data. Functional requirements for a tool to navigate hierarchical structures were laid out in "Nonvisual Tool for Navigating Hierarchical Structure." [7] In making an accessible tool for manipulating mathematical expressions, this project chose a hierarchical structure and follows research done at CSCAP to make navigating hierarchical structures accessible.

3. FUNCTIONAL REQUIREMENTS

The fundamental requirement for an accessible mathematics manipulation tool for visually impaired students is that a ME can be entered and edited to a desirable form with the use of a keyboard and screen reader only. However a good accessible tool should support working on a wide range of problems effectively and efficiently, reduce memorizing, and minimize third party dependence, whether human or software. Moreover a learning tool should also avoid massive automation such as computation, auto error correction, etc. In other words, it should allow users to make any mistakes that their sighted peers can make using paper and pencil.

Visually impaired students currently taking mathematics classes were interviewed for existing difficulties and desired features in a Math tool. Two mathematics instructors with prior experience in teaching visually impaired students have also contributed to the list. The following section describes the list of requirements for an accessible Math tool and some possible solutions.

1. *Learning Environment*
   The tool should allow the user to manipulate contents in any way the user intends, whether right or wrong. It should minimize automation and maximize calculation facilities required for learning. It should also provide a common platform for sighted and non-sighted students to share work simultaneously.

2. *Active Reading*
   Sighted students play an active role while reading printed material. The browsing capabilities of the human eyes allow readers to move to any part of a ME very quickly. These movements are not random and often not linear [10]. On the contrary, a screen reader user plays a passive role in reading, as the system speaks and the user listens with very little facility to move among different part of the expression in any

order but linear. However, the facility to control the content to be spoken and to move among different parts of the problem can help the screen reader user to have a more active rule.

3. *Beginning and ending of expressions and sub-expressions recognition*
   Most often written or printed mathematics works are spatially formatted for ease of reading and understanding. The spatial formatting helps sighted students to visually parse the expression into sub-expressions. However, collapsing multidimensional spatial formatting into linear format results in many extra parenthesis and braces. Given no error took place in transcribing, it is still significantly hard to pair up braces and parentheses. A similar problem in writing computer programs is addressed by some popular IDEs, such as Eclipse [11] and NetBeans [12] by highlighting the corresponding pair of a selected one. Mathematica [13], a popular math computation software, provides feedback on missing braces or parenthesis by changing the color of its possible pair. This technique can be mimicked in an auditory interface to help non-sighted students to find the beginning and ending of a sub-expression contained inside braces or parentheses in initial linear format and provide feedback on missing brackets.

4. *Complex structure and contents can be parsed and documented into smaller or preferred logical pieces.*
   Breaking complex problems into smaller tasks or modularity is a well used strategy in mathematics. Sighted students can do this by reorganizing an expression or breaking down an expression into sub-expressions. The process is also documented on paper, which reduces pressure on the user's memory and the risk of error. These reorganizations and rewritings enable visual students to see and read a problem in the way they want to. Non sighted students often do reading by listening. In a linear format, the listening reader's role is passive and they have very little to no control over how they want to listen to it. This may cause frustrating re-listening to the same content over and over to get the structure and details of the expression. Moreover a structurally complex expression may exhaust the listener's mental resources [9]. This is due to the fact that linear format has very little facilities for navigation and reorganization. The tool should provide an environment where the listener can reorganize the structure for a preferred way of listening and use the reorganized phase of the expression as an external memory for a reference at any time.

5. *Association*
   The tool should allow the user to associate logical pieces of a problem easily and efficiently. This is particularly important to aid Requirement 4 and while solving for two or more variables.

6. *Abstraction*
   Often times, complex information is easier to comprehend with the help of abstraction. The user should be allowed to create different levels of abstraction for a problem in a preferred way.

7. *All intermediate steps towards the solution should be recordable.*
   Sighted students record steps towards the solution using a paper and pencil. This allows them to get partial credit on their work in case of an error. This also works well for future references while doing a similar problem or as a review before an exam. However non-sighted students often miss these advantages if much of the problem was worked on mentally and intermediate steps were not recorded as part of the solution.

8. *Any intermediate steps should be accessible at anytime to change strategy or fix an error.*

   Sighted students can do so by reviewing previously written steps and correcting them by erasing or overwriting. Due to the advantage of visual parsing and multidimensional documentation on paper, it takes less effort for sighted students than it does for non sighted students. If the steps

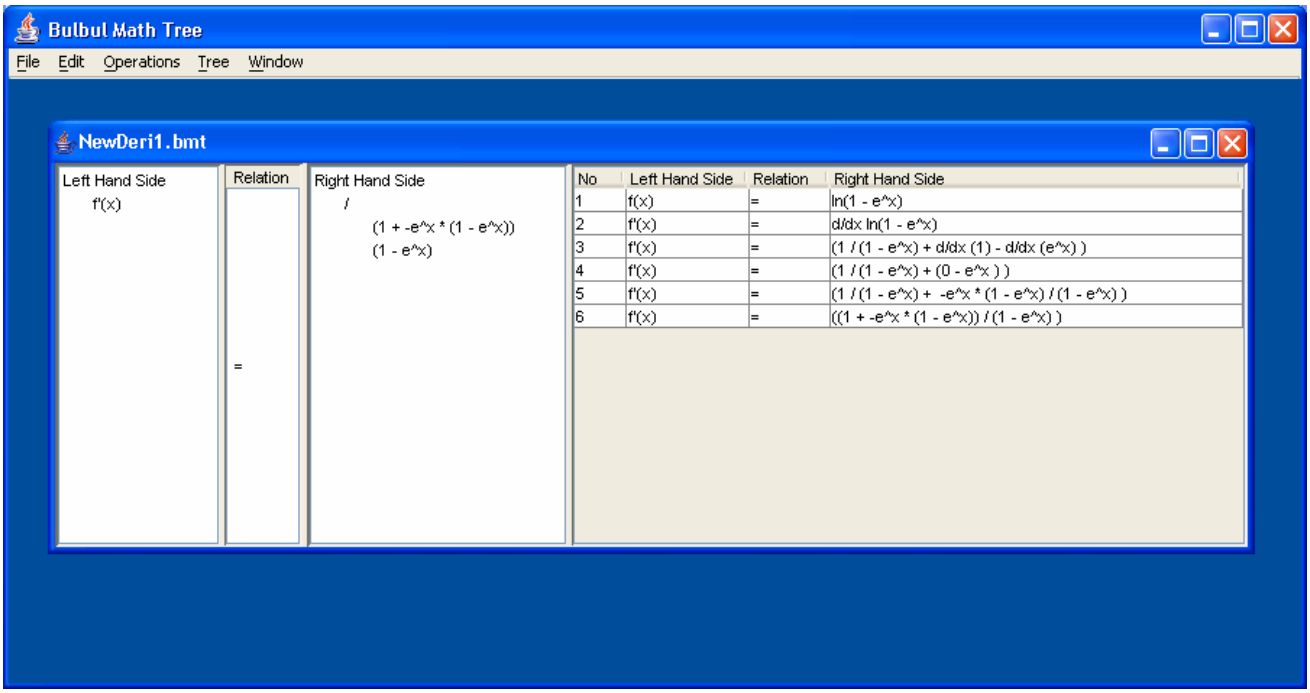


Figure 2: BMT user interface

were not documented, a non-sighted student will have to start all over. If the steps were documented linearly, they would have to create a mental image of those steps over again which may consume time, frustrate the student, and yield more errors. So, non-sighted students should have corresponding parsed pieces of the steps when reviewing.

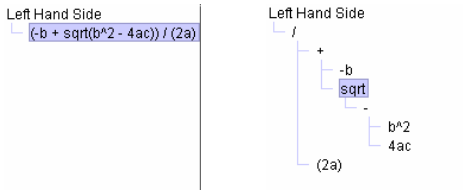
9. *Multiple Work book*
   Solving some math problems requires working on multiple expressions or equations at the same time. Sighted students can do so by dividing the paper spatially or using multiple pages. This gives them the opportunity to compare. Non-sighted students should be able to have multiple problems open at the same time and seamlessly move among them.

10. *Exporting work for submission*
    The tool should be able to provide all the work done into a printable format.

4. IMPLEMENTATION

Following the standards set in Section 3, a tool was implemented in Java. The tool has been named Bulbul Math Tree (BMT). The tool has a user interface accessible through the keyboard and an external screen reader, such as JAWS® [6]. As shown in Figure 2, the tool has two trees, a text box between the trees, and a table. The trees are used to input and manipulate mathematical expressions. For an equation, the left-hand-side of the equation is entered into the left tree and right hand part of the equation is entered in the right tree. The text box between the trees contains the relation. The table is used for recording desired steps of the process. Since it's a learning tool, it has very limited automation and computation facilities. The tool also provides support for exporting results and the steps taken in deriving the results in a printable format that can be used to submit homeworks and exams.

Students can enter a mathematical expression using plain text. Then they can create the architecture of the expression in successive steps by breaking the expression into smaller logical pieces. For example, (-b + sqrt(b^2 - 4ac))/(2a) [Exp 1] can be arranged in the picture of the tree Figure 2.1a. This is very similar to that of a parsing tree.

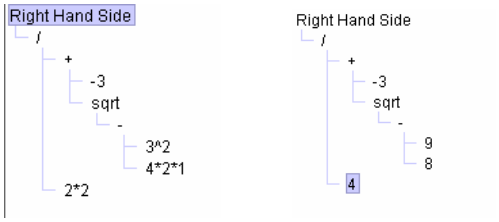


(a) **(b)**

Figure 3: Entering and Expanding an ME.

Going from figure 3a to 3b can be done through the "Expand" operation. In this operation, students select a segment of the expression, and then select an operator. After the operation, the operator becomes a new parent node, and the operands become its children nodes. This operation can be done repeatedly on an expression for logical simplification and abstraction. Four "Expand" operations were performed in the process of converting the linear text in Figure 3a to the hierarchical expression in Figure 3b. We hypothesize that this allows students to arrange the mathematical expression in a preferred structure for better understanding, making subsequent operations easier and lowering the chances of misinterpretation.

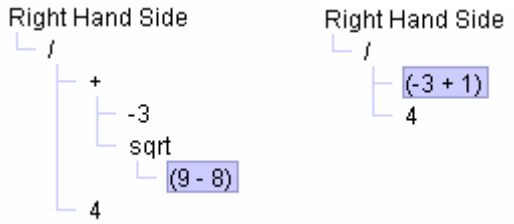
The tool allows students to substitute the value of a variable. For example, in Exp 1, if a = 2, b = 3, and c = 1, the variables in the tree can be substituted using "substitute" operation. For the sake of learning, the substitution process is limited to one node at a time, and not "substitute all the b in the tree by 3". "Substitute" operation can be used for either to substitute a variable or to simplify an expression. For example, a node containing 4*2*1 can be substituted for 8. In Figure 4a and 4b, we show the result of substitution of the mentioned variable on Figure 2.1b. We hypothesize that substitution and simplification in a tree structure is easier and has lower risk of error than that of a linear structure.



(a) **(b)**

Figure 4: Substituting variables

The tool allows students to put logical pieces together through the "Collapse" operation. As in Figure 4b, the child nodes containing 9 and 8 with a parent node containing minus can be collapsed in a single node containing "9 - 8". It can be simplified then using the "Substitute" operation. The resulting node can be collapsed again to form the node sqrt (1) and such. Successive application of the "Collapse" and "Substitute" operations can be used to simplify the expression in 4b and therefore solve the problem in Exp 1. The result is shown in Figure 5a and 5b. We hypothesize that this will help students in working on smaller part of the problem towards solving a big and complex problem.



(a) **(b)**

Figure 5: Simplifying Expression

The tool allows documentation of each step of the problem solving, frequently a requirement for the students when submitting their work as homework or exam. This can be done using the "Record step" operation after any change made in the

tree, if desired by the student. Figure 6 shows a table of recorded steps for the previous problem. This helps students in two ways. First, it helps student document their work and secondly, it helps them to go back to a step if a mistake was made. Going back to a step can be done using the "Revert" operation. In this operation, visually impaired students can check all the steps they have done towards solving the problem, correct a mistake in a step, and continue the work from there again, just like sighted students can do using paper, pencil and eraser. Going back a step reconstructs the corresponding phase of the tree at that step. This allows students not to start all over, rather continue from where they left off or where an error took place. We hypothesize that this will increase efficiency in working error prone problems by finding the error in the process of solving problems faster and fixing the problem easier than linear architecture in use.

No	Left Hand Side	Right Hand Side
1	x	(-3 + sqrt(3^2 - 4ac)) / (2a)
2	x	(-3 + sqrt(3^2 - 4*2*c)) / 2*2
3	x	(-3 + sqrt(3^2 - 4*2*1)) / 2*2
4	x	(-3 + sqrt(9 - 4*2*1)) / 2*2
5	x	(-3 + sqrt(9 - 8)) / 2*2
6	x	(-3 + sqrt(9 - 8)) / 4
7	x	(-3 + sqrt1) / 4
8	x	(-3 + 1) / 4
9	x	-2 / 4
10	x	-1/2

Figure 6: Recorded steps of a problem.

The navigation strategies of the tool closely adapt earlier research on navigating hierarchical structures. [7] The navigation rules for BMT are as follows:

1. All the branches in the tree are open at all times.
2. The spatial orientation of the tree is standard side ways rather than top-down. All the child nodes are located to the right of the parent node.
3. An error sound is generated if a user tries to make an illogical move.
4. The list of siblings is presented as a non-circular list (i.e., an attempt to move to the next sibling from the bottom most sibling is considered illogical.)
5. Bookmarks can be set to a node and later can be visited by the most recent first.

The tool also supports saving a problem which gives students the flexibility of saving a problem to work on later. It allows students to export a printable version of their work containing all steps. Other standard operations include, opening an existing file, cutting, copying, and pasting of nodes, deleting a node, and undo and redo operations. More operations are being evaluated for "fairness in learning environment" by a study group of mathematics instructors for the project. This includes, finding a variable in the tree, automatic substitution, applying distributive law through a template, etc.

5. USABILITY STUDY

The usability study of the BMT was done in two different phases: Applicability Test and Accessibility Test. In the first phase, to assess the applicability of the test, 18 undergraduate students and 2 high school PSO (Post Secondary Option) students were recruited by asking for volunteers in 7 different classes. All the students were fully sighted. Sighted students were primarily recruited for availability and the fact that the applicability and not accessibility of the tool was in question. Among the 20 students, 14 were male and 6 were female. The survey subjects were composed of many different majors: 3 computer science, 1 physics, 1 chemistry, 3 engineering, 6 mathematics, 3 nursing and 3 others. The survey subjects varied in age from 16 to 24. The length of the study was one hour and was done in six different sessions with 3-4 students each session. Students were thanked with candies for their contribution.

Students had no prior exposure to the software. For the first part of the survey, they were handed a tutorial. The tutorial contained a sample problem and step-by-step instructions on what operation they needed to perform to solve the problem. The first occurrence of an operation was given in detail and in the subsequent ones they had to remember how they did it before. The students were allowed to ask questions, think aloud, and talk among themselves about strategies. The average time to go through the tutorial was 30 minutes.

For Test 1, to check students' understanding of a mathematical expression laid out in hierarchy, students were given pictures of problems expanded in BMT with an average of 4 levels height and 4 levels depths and asked to write down the expression on paper.

For Test 2, they were give an expression of higher complexities and two pictures of that expression expanded in BMT, with one of them being correct and the other one with right content but wrong structure of logic. Their first job was to decide which picture correctly represented the expression in question. After that they were asked to use BMT to expand the expression in the similar manner to support their answer and comment on why the other picture was wrong.

For Test 3, students were asked to solve a problem by showing every step of the problem solving process. The problems assigned varied from pre-calculus, calculus, and discrete math depending on the highest level of math class they had taken. The problems were suggested by mathematics instructors as representative of homework or test questions. Sample problems are given in the Appendix. The students were allowed to ask questions and comment while working on their individual problem.

Ten participants were assigned problems from pre-calculus. The problems were simplifying expressions or solving equations that had a moderately large structure. Six participants were assigned calculus problem. The problems were either taking the derivative using the product rule or integration. Since the problems in this section were much harder than the one demonstrated in the tutorial, participants were given some problem specific tips in the beginning of the problem. Four students were assigned problems from discrete math. Problems in this section were in general larger than the others and involved verifying arguments. The tool does not yet support special symbols used in mathematics such as, √ (square root), ∫ (integrate), subscript, superscript, etc. Therefore, students were asked to type in the name of any such symbol.

A smaller usability test was conducted afterwards to assess the accessibility, usefulness and efficiency of the tool as well as applicability. Two visually impaired students participated in the usability test. They are both experienced JAWS user, and had limited exposure to the tool before testing.

Both visually impaired student participants were male. Participant one (P1) is a senior and participant two (P2) is a freshman. They are both computer science majors. P1 had prior experience with tree structures. He was enrolled in Calculus II at the time of the study. P2 had limited experience with tree structure. Pre-calculus was the highest level of mathematics class he had finished and was enrolled in Discrete Mathematics at the time of the survey. Both participants took the same tutorial as sighted students. P2 was briefed about the tree structure prior and during the tutorial.

P1 did two pre-calculus, two differentiations, and one integration problem. P2 did two pre-calculus problems.

6. STUDY RESULTS

The hierarchical representations of the MEs were found to be very intuitive by sighted participants. 17 out of 20 students wrote down MEs from their hierarchical layout correctly in the first attempt. The other three students had a common mistake of flipping the numerator and denominator of a fraction. When asked about it, their general response was that they were not aware of the fact that order of siblings would matter. Participants also demonstrated understanding of the layout through identifying errors in a hierarchical layout of MEs. The results are given in Table 1.

Participants' Answer	Number of participants
Identified correctly	16
Expanded correctly	17
Changed answer after expansion	2
Justified correctly	18
Total correct answer	18

Table 1: Cross matching MEs to hierarchical layout

Category	Total number of participants	Participants with correct answer
Pre-cal	10	9
Calculus	6	5
Discrete	4	4

Table 2: Number of participants with correct answer

Statements	P1	P2
The tool gives more control over information flow.	8	9
The tool helps to break down problems into manageable pieces.	9	8
Abstraction achieved by the tool helped to understand the structure of a problem.	7	8
The tool is easy to learn.	7	7.5
The numbers of steps in solving a problem is manageable.	8	7.5
I would use this tool in future.	9	9

Table 3: Non-sighted students survey results

Most of the students successfully completed the task of solving individual problem in Test 3. Table 2 shows the result of the test.

Both non-sighted student participants completed their tasks correctly. Table 3 shows some of their responses after test. The participants answered questions in a scale from 1 to 9 with 1 being negative, 5 neutral, and 9 being positive.

Observations:

1. BMT is Accessible

Both visually impaired student participants were able to use the tool with a keyboard and screen reader only. Both users were able to learn the hot-keys after a very short period. Users were able to work on each part of their individual problem independently without getting lost.

2. BMT is easy to learn.

All the participants finished reading the tutorial including doing a sample problem in under half an hour. After that, they worked on their individual problem. Most participants were able to work on the individual problem without any help. Few students referred back to the tutorial when not sure how to do something. In general, participants were able to learn the commands and patterns in a relatively short time. Figure 7 shows the "learning experience" ratings by the participants.

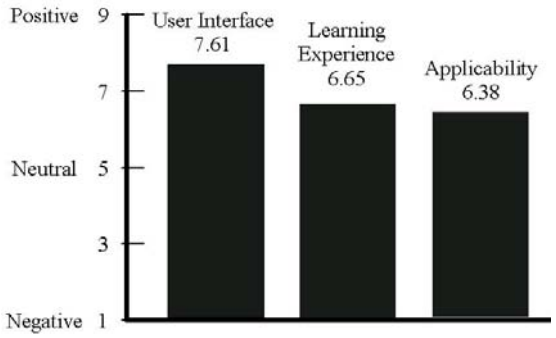


Figure 7: Overall user rating of BMT

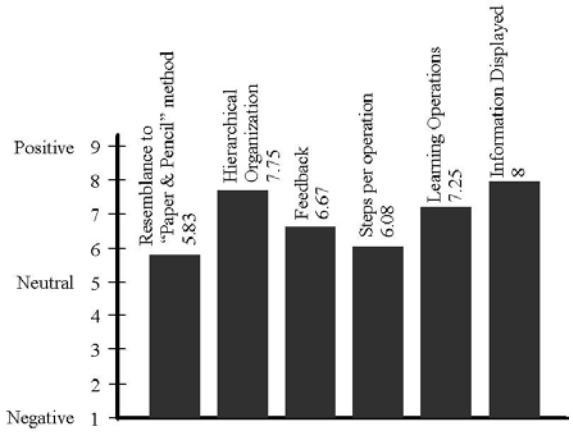


Figure 8: Overall user ratings of particular issues.

3. Hierarchical representation of mathematics is intuitive.

Most of the participants didn't need any help in reading a ME from the tree. When participants were asked to write down a ME represented in the hierarchy, 17 out of 20 students were able to write down the expression correctly. The other three students had a problem understanding the order of sibling nodes and had related errors in the expression such as flipping a numerator and denominator or flipping the operands of a subtract operation. One of them commented that "I didn't know the order of these [nodes] would matter." The others said that the layout of the problem was different but they figured it out after few minutes looking at it.

4. The "Expand" operations is effective for organizing MEs.
The "Expand" operation was the most liked and often used operation by the participants. One participants said, "I liked the expand most of all because it breaks the problem down into components." Another participant commented, "All the steps of solving a problem are easily visible [by expanding]". The instructors in general predicted that it will help student better organize their problems.

5. BMT requires more support for "undo" and "redo".
Participants used "undo" and "redo" to fix an immediate error while "Revert" was used to go back to a previous state. Participants disliked the fact that the tool would allow them to "undo" and "redo" a step only once. Many participants suggested that the tool should allow them to "undo" until the very first step. "Revert" was used only by a few participants and was much praised.

6. Auto resizing of windows are necessary for sighted students.
User complained that they had to frequently resize the panels in order to have a full view of the trees. This is due to the fact that, whenever any branch of the tree grew out of the panel size allocated for it, either the user would have to use the scrollbars or resize the panels. Auto resize of the panels were expected by most users. This is particularly important in a group study environment where sighted and non-sighted students may use the tool as a common platform.

7. Voluntary recording of steps found to be insufficient.
Most of the participants didn't record the process as often as they should. When asked about it, most of the participants answered that they forgot, followed by they thought the program would record for them automatically. Some participants suggested that an automatic record or a reminder for record would have been helpful.

8. Spacing between operator and operands are considered overhead.
The tutorial suggested participants to put a space between an operator and operands. Participants in general didn't like this. Often times they forgot, to put spaces. When asked about it, the most common answer was they forgot followed by it was awkward and didn't make sense.

9. Feedback messages.
The tool provides immediate feedback with specific messages if the user skips a required step of the process instead of simply disabling the "Ok" or "Next" button. Data in Figure 8 shows that this feature was much appreciated by the participants.
10. Steps per operation.
Participants in general had very mixed reactions on the number of steps it took to perform an operation. The absence of the use of right-click on mouse and icons upset many participants. However participants taking the calculus part of the survey were showed how to reduce the number of operations and steps, such as doing "substitution" and "expand" in a single "expand" operation. Few of them also found the hot-keys very useful. Two participants from the calculus group significantly used less steps and operations just by expanding the expression differently.

6. CONCLUSION AND FUTURE WORK

BMT is a mathematics learning tool that is easy to learn yet efficient in managing large MEs. It does not require a user to type in any special coding or markup languages, such as Nemeth Code or LaTeX. The tool is applicable to problems that involve algebraic and numeric manipulation. It also provides a common platform to discuss mathematics for sighted and non-sighted students.

The development of BMT is far from complete. Future work will include a mechanism for entering special symbols in an efficient yet accessible way. Some popular mathematics tools are currently being investigated for such mechanisms. This will be useful particularly when letters from alphabets other than Roman-English are used. The tool will also use MathML to present a more visually appealing form for recorded steps and printing.

7. ACKNOWLEDGEMENTS

I would like to thank Joan Francioni for making this project possible with her help, support and guidance in every step of the process. My deepest gratitude to Ann Smith, Gary Bunce, and Felino Pascual for giving me time in their busy schedules. Thanks to all the students who volunteered for the usability testing, in particular Denny Schwab and Mohamed Abdel-Magid. Last but not the least, thanks to my family for their continuous support and encouragement.

REFERENCES

[1] Hayes, B. 1996. "Speaking of Mathematics", *American Scientist*

[2] Raman, T. V. 1996. "AsTeR – Towards Modality-Independent Electronic Documents." Available at http://www.research.digital.com/CRL/personal/raman/home.html

[3] Karshmer, A. I., Gupta, G., Pontelli, E., Miesenberger, K., Ammalai, N., Gopal, D., Batusic, M., Stöger, B., Palmer, B., Guo, H-F. 2004. "UMA: A System for Universal Mathematics Accessibility." In *Proceedings of ASSETS 2004*, Atlanta, Georgia, October 2004, 55 - 62.

[4] Knuth, D. E. 1984. *The TeXbook*. Reading. Addison-Wesley, Mass.

[5] Lamport, L. 1986. *LaTeX: A Document Preparation System.* Reading. Addison-Wesley, Mass.

[6] JAWS for Windows, Freedom Scientific, http://www.freedomscientific.com

[7] Smith, A., Cook, J., Francioni, J., Hossain, A., Anwar, M., Rahman, M. 2004. "Nonvisual Tool for Navigating Hierarchical Structures." In *Proceedings of ASSETS 2004*, Atlanta, Georgia, October 2004, 133 - 139.

[8] Stevens, R. D., Edwards, A. D. N., Harling, P.A. 1997. "Access to Mathematics for Visually Disabled Students through Multi-Modal Interaction." *Human-Computer Interaction (Special issue on Multimodal Interfaces)*. 47 – 92

[9] Stevens, R. D., Wright, P. C., Edwards, A. D. N., Brewster, S. A. 1996. "An Audio Glance at Syntactic Structure Based on Spoken Form." *ICCHP '1996: Interdisciplinary Aspects on Computers Helping People with Special Needs*, Oldenbough Wien Munchen, 1996, 627 – 635.

[10] Raman, T. V., Gries, D. 1994. "Interactive Audio Documents." In *Proceeding of the First Annual ACM Conference on Assistive Technology*, 1994, 62-68.

[11] Eclipse, Eclipse Foundation. http://www.eclipse.org

[12] NetBeans, Sun Microsystems. http://www.netbeans.org

[13] Mathematica, Wolfram Research. http://www.wolfram.com.

[14] Jones, R. 1994. "Math and Science Symposium At Recording for The Blind", Available at http://www.rit.edu/~easi/itd/itdv01n4/article1.htm

[15] Raman, T. V. 1994. "AsTeR: Audio System for Technical Readings". Available at http://www.informotions.com/serials/itd/itd-v-1n04-raman-aster.txt

[16] Gardner, J. A., Stewart, R., Francioni, J., Smith, A. 2002. "Tiger, AGC, and Win-Triangle, Removing the Barrier to Sem Education." In *Proceedings of the 2002 CSUN International Conference on Technology and Persons with Disabilities*, Los Angeles, CA, March 2002.

Appendix:

Sample problems from usability test:

Pre-calculus:

Simplify the expressions
1. (x + 2) / (x - 1) – x^2 / (x - 1)^2
2. a / (a + b) – b/ (a - b)

Calculus:

Find the derivative
3. r^2 / (2*r + 1)
4. x^2 * Sin (x) + 2 * x * Cos(x) – 2 * Sin (x)

Find the integrals
5. Integrate (x^2 + 5x + 8) dt
6. Integrate (4 / x^2 – 3 / x^3)

Discrete Math:

Represent the following expression in a binary tree
7. A * ((B + (C / (D^2))) - E)

Assuming that p and r are false and that q and s are true, find the truth value of the following proposition.
8. (s -> (p & ~r)) & ((p -> (r or q)) & s)

3D Graphics Then and Now: From the CPU to the GPU

Joseph M. Mahoney
Computer Science Department
Winona State University
Winona, MN
mahoneyxp@gmail.com

ABSTRACT
Since the late 1990s, developers have been pushing graphics hardware to its limits, trying to create photorealistic environments in their games similar to those seen in recent blockbuster movies such as The Incredibles™. The need for graphics accelerators is becoming more apparent every day, as millions of calculations need to be done to render the complex scenes within games. There have been four generations of graphics accelerators, with the prospect of many more in the future. We found that the latest two generations of graphics accelerators speed performance are more dependant on the CPU speed of the system they are operating on than the previous two generations. Furthermore, newer versions of Direct3D offload more work to the Graphics Accelerator, leading to increased performance with Direct3D software on slower systems.

General Terms
Performance; Experimentation.

Keywords
Direct3D, DirectX, Graphics Accelerators, Graphics Processing Unit, Shaders, Vertex, Video Cards.

1. INTRODUCTION
In a time where video gaming is a billion dollar industry, it is no wonder that there is a large demand for high performance graphic accelerator technologies. Developers are continually trying to make an immersive world that is more beautiful and photorealistic [1]. While these graphics can be accomplished by large graphic "render-farms," as seen in motion pictures such as Lord of the Rings™, currently it is not feasible in real-time using current consumer graphic accelerator technology [1]. Each scene in a game takes millions of floating point calculations and matrix manipulations, and to become more immersive and realistic requires even more calculations. Video game developers are pushing the hardware more and more every day and, because of this, there have been leaps and bounds in graphic card performance and features.

Since the mid 1990's, there have been four generations [2] of consumer level video cards available, each generation having significant improvements over the last. One of the main improvements has been an increase of speed, shown mainly by the frames per second achieved in games. Each generation carries faster memory and graphics processing units (GPU). Some of the GPUs on the market today actually perform operations at speeds higher than today's fastest CPUs [2]. This is a good trend since the CPU can only do so many calculations per second. If the CPU had to deal with graphics as well as general computations, performance would lag and the game would be unrealistic. Besides speed increases, successive versions of Microsoft's API for graphics development, Direct3D, provide developers new ways to interact with the GPU so they can create a more realistic and complete graphic experience for gamers [4]. It is also known that the system you operate on has a large impact on the graphics card you are running. Newer versions of Direct3D allow the GPU to do more computation itself with its own local memory and information streams, which frees up many of the old performance bottlenecks. Since it is clear that video card technology is crucial to the future of game development, there is an ever-present need to know what the next big thing in graphic hardware will be. A review of the history and current trends in graphic accelerators is needed in order to predict where the technology will go next.

With each generation of hardware, the GPU seemed to do more work independent from the CPU. While this appeared to be the trend, we wanted to find out if this was actually the case by comparing the four generations of cards in varying CPU speed environments. It was predicted that the performance of the first and second generations of GPUs would have a strong correlation to the power of the CPU that they ran on. We predicted performance would increase linearly, proportional to the speed of the CPU. Furthermore, while we predicted that the performance of the newer generations of GPUs would have some correlation to the power of the CPU they ran on, we did not predict it would be as much as the older ones, leading to a slower degradation of performance on slower CPUs and leveled performance on faster CPUs. More formally, when analyzing 3D benchmark and frames-per-second tests, CPU speeds affect speed performance of third and fourth generation graphics accelerators less than the previous generation cards.

We also tested the performance differences on the machines with varying Direct3D versions. The Direct3D versions tested in our study were 7.0, 8.1, and 9.0. Direct3D versions 8.1 and 9.0 allow programmers to take better advantage of the GPU compared to older versions. Developers can write advanced shader programs and manipulate the GPU more than ever before, reducing the amount of work the CPU must perform [3]. Because of this, we predicted that the software using newer Direct3D instructions

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

would depend more readily on the GPU than the CPU, leading to less performance degradation with the new graphics cards on the older CPU computers.

2. BACKGROUND

2.1 Graphics Hardware

On the timeline of computer history, 3D graphics have been around for a relatively short period of time. There have been four generations of consumer-level video cards since the mid 1990's. Each generation brought about a unique feature or advancement that constituted the leap to a new generation.

2.1.1 First Generation

Graphics accelerators in the first generation were developed in the mid to late 1990's. The most notable graphics cards in this generation were the 3dfx Voodoo Series, Rendition Verité, NVIDIA Riva TNT2, and the ATI Rage. These cards provided a way for the programmers to do simple manipulations such as blending of textures, but the cards still required the CPU to do most of the rendering work. There was an average of 10 million transistors on this generation of cards [2]. As shown in Figure 1, this was the first step from 2D environments to 3D environments. The figure shows a scene in the game Quake II rendered using two different methods, one using software to draw the scene (no 3D hardware used) and the other utilizing the 3D hardware. As you can see, the left side of the image is drawn using software, relying on simple shapes. The textures are very jagged, especially on the wall in the distance. The right side of the scene was rendered using 3D hardware. The image is smoother and more detailed. Also, the software rendered scene lacks dynamic lighting and coloring effects, as shown by the green light at the top of the picture not casting color on the left side.

2.1.2 Second Generation

In 1999, the second generation video cards came to the market. Representing this generation were NVIDIA's GeForce256 and GeForce 2, ATI Radeon 7500, and the S3 Savage3D, carrying an average of over 25 million transistors [2]. The main improvement in this generation was the inclusion of the transformation and lighting feature (T&L) [2]. T&L makes it so the GPU handles the transformation of matrices and lighting calculations that the CPU used to be responsible for. These calculations are done many times per scene, resulting in a reduction of transfers between the graphics card and the CPU that greatly increases overall performance capability. [2]

2.1.3 Third Generation

In the 2001 and 2002, the third generation of video cards were developed, mainly by the two competing companies: NVIDIA and ATI. The cards in this generation are the GeForce3-4 and the ATI Radeon 8500, averaging around 60 million transistors. This generation also made the first significant step towards programming directly for the GPU. Developers could now do vertex computations [3] and other operations at the GPU level. The programs written for the GPUs were very simple and limited in the scope of what they could perform, but it was a vast improvement over the previous generation, as now the GPU could run the programs independent of the CPU. [2,3] At this time, graphics cards were also beginning to show the characteristics of parallel processing. Before this, rendering was handled by serial processing managed by the CPU. Furthermore, "rendering primitives were handed off to a GPU, the CPU went off and did a little work, then fed the GPU a bit more." [1] This advancement allowed for faster performance and less redundant communication between the CPU and the GPU.

2.1.4 Fourth Generation

First released in 2003, the fourth generation of video cards represents the current state of graphics technology. The main cards in this generation are the ATI Radeon 9700-x800 series, and the NVIDIA GeForce FX and GeForce 6 series, both carrying around 100 million transistors [2]. Cards in this generation can perform 370 vertices calculations per second and allow for multiple calculations per pixel for each rendering pass. This allows for highly complex and realistic scenes to be generated with fewer passes and memory accesses. Like the third generation of cards, these are also programmable. However, the instruction sets for these cards were improved and now programs can be thousands of instructions long compared to hundreds of instructions long in the second generation. GPU programming languages such as Cg were developed for the purpose of developing complex shaders for the chipset, allowing programmers to create scenes unimaginable in the late 1990's [4]. Furthermore, each pixel now has floating point precision, allowing for more accurate calculations and precision of the rendered scene.

Figure 1: Step from 2D to 3D

Figure2: Rendering Using D3D 7

Figure 4: Rendering Using D3D 8.1

Figure 5: Rendering Using D3D 9

2.2 DirectX Overview

DirectX is a set of APIs that "act as a kind of bridge for the hardware and the software to 'talk' to each other." [6] The main function of DirectX is to allow programs to access the advanced features of the hardware on a system. Direct3D (D3D), in particular, allows programs to interact with the advanced features of graphics cards. DirectX simplifies development by controlling the low-level functions so that programmers do not have to develop different code for each set of hardware. [6] D3D slowly has been gaining support from developers, and there have been many versions throughout its history. The versions whose features we will be discussing are Direct3D 7, 8.1, and 9.

2.2.1 Direct3D 7

This version of D3D is often referred to as the first legitimate contender to the OpenGL API, and was the first D3D version accepted by game developers. It was released by Microsoft in 1999. One of the main features that D3D 7 provided was the enhancement of transformation and lighting support through the 3D hardware, which removed much of the computation burden off the CPU so it could do other tasks such as game physics or artificial intelligence [7]. This was a big step in real-time realistic game scenes. D3D 7 also allowed for more sophisticated reflection and lighting effects and realistic texture transformations. Figure 2 shows what a current game, Half-Life 2, looks like rendered using D3D 7. As you can see, the water is rendered very blue and you can see right through it. There are no reflections in the water, and the water's edge is harsh and easily visible. To render more complex reflections in this version of DirectX would require many passes, and often is not a viable option because of the performance hit it would create.

2.2.2 Direct3D 8.1

Released in 2000, DirectX version 8.1 brought about many new features to the D3D API. It provided programmable shaders for vertex and pixel operations, which "provides the framework for real-time programmable effects that rival movie quality." [8] D3D 8.1 shaders are pieces of code written in assembly language that execute on every vertex calculation that occurs in a scene. These shaders are used instead of the T&L pipeline for vertex calculations,[9] as shown in Figure 3. With this new ability, developers could implement new and innovative effects that were never imaginable before because they could program exactly how they want their calculations to be handled. For example, a shader could be used to calculate the correct way that light should reflect off different surfaces. It would run these calculations and the results would be part of the scene [1]. These programs were written in assembly language and limited to 128 lines in length [9]. In addition to this, D3D 8.1 brought about the concept of point sprites that allow for more detailed particle effects such as explosions or sparks [8]. These and many more features were added to D3D 8.1, which was a large step in bringing the cinematic experience to our computer screens. Figure 4 shows the enhancements this version of D3D brought over D3D 7. The textures are much larger, sharper, and the overall scene looks more realistic. In addition, the shader technology enhances the scene, allowing for more realistic water effects. The water now has ripples on it and is no longer blue and transparent. It now provides some reflection and looks more realistic.

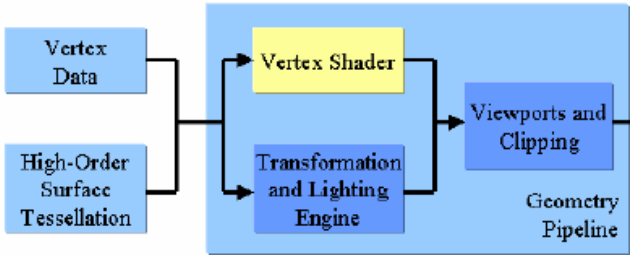


Figure 3: DirectX 8 Geometry Pipeline [9]

2.2.3 Direct3D 9

The latest version of Direct3D, completed in 2002, brings realistic gaming worlds to life. D3D 9 is very similar to D3D 8.1, but it provides many enhancements that improve performance and quality. One of the major enhancements was the improvement of the pixel and vertex shader technology [10]. Programmers are now able to develop shaders using a high-level shader language, DirectX 9 High Level Shading Language [1]. Because of this, more people can develop custom shaders, creating a wider variety of rendering techniques and effects in games. This is a vast improvement over the assembly-based shaders in D3D 8.1[3], as few people were able to develop them. Some of the other improvements in this version were the increase to 32-bit texture formats and improved support by the development community. As shown in Figure 5, the scenes rendered in D3D 9 are leaps and bounds ahead of the previous generations of D3D. The scene shows drastic improvements in the water effects. The advanced

pixel shaders allow for the water to accurately reflect the building. Furthermore, you can see some improvement of shading, reflections, and crispness of the rendered scene between the two, particularly in the details in the background of the screenshot. Between D3D 9 and D3D 7, the difference can be seen like night and day. Games are finally becoming photorealistic like their cinematic siblings.

3. METHODS

3.1 Systems' Setup

To see the difference between the four generations of cards and their performance correlation to processor speed, we set up a number of test systems. First of all, we collected one graphics card from each of the four generations of graphics accelerators. From the first generation, we chose a NVIDIA Riva TNT2 32mB. For the second generation, we chose a NVIDIA GeForce 2 MX 64mB. In the third generation, we chose a NVIDIA GeForce ti4200 128mB. Finally, in the fourth generation, we chose a NVIDIA GeForce 6600GT 128mB. We chose to use NVIDIA cards because NVIDIA still supports all of their video cards and current drivers are available. As shown in Table 1, we tested these cards on two PCs of varying speeds from 900mhz to 2100mhz. To achieve the differing CPU clock speeds on a single PC we used the system bios to change the CPU clock settings. We could set the clock incrementally from 1200mhz to 2100mhz. This ensured that other factors in the systems such as memory and architecture did not affect that particular set of scores. The processor on Computer 1 is an AMD AthlonXP, thus it uses a different numbering scheme for its CPUs. In attempts to compete with Intel, AMD uses numbers that represent what their CPU is comparable to in the Intel line. For example, if the AMD CPU is a 2600XP, it is comparable to an Intel 2.6ghz processor, even if it is running at only 2100mhz. In Table 1 we show both the actual speed and the "marketing" speed of the processor. We will use the XP number throughout the rest of this paper. All systems are running Windows XP Service Pack 2 with the latest video card drivers available for each of the boards, specifically the NVIDIA ForceWare version 71.84.

Computer Name	CPU Speed	RAM
Computer 1	1600XP (1200 mhz)	512mB
	2100XP (1600 mhz)	512mB
	2600XP (2100 mhz)	512mB
Computer 2	900 mhz	256mB
	1200 mhz	256mB

Table 1: Test Systems' Configuration

3.2 Test Suite

To perform our tests we used various 3D benchmarking programs and multiple gaming tests to calculate metrics such as frames per second (FPS), and overall benchmark numbers. We used two main types of benchmarks for our analysis, real-game tests and synthetic benchmarks. The main difference between the two is that the real-game tests are measured by taking average frames per second (fps) in games that are on the market today, while the synthetic benchmarks are software programs developed to create stress tests for the different features in video cards [5]. Halo, Half-Life, and Half-Life 2 are the real-game performance benchmarks we ran. The synthetic benchmarks we ran are 3dMark2000 and 3dMark2001SE. The benchmarks were performed on all supporting hardware and the results recorded. We needed at least one software title for benchmarking that every generation would be able to run and we chose Half-Life and Half-Life 2 running Direct3D 7. Each test was performed three times and we recorded the average score.

We also tested to see if the new Direct3D versions, which give programmers the ability to utilize advanced shader instructions [4] on the GPU, had an impact on the performance of the newer cards on the slower CPU speeds. We used the latest two generations of video cards and tried the different Direct3D versions of Half-Life 2. We then analyzed the results to see if there was any correlation between the speed of the system and the performance with the different Direct3D software. [5]

4. RESULTS

4.1 Generations' Performance

To test the correlation between CPU speed and the four generations of graphics cards we ran a multitude of tests. We used both gaming and synthetic tests to show the correlation between the generations.

4.1.1 Gaming Performance

We used the original Half-Life and also Half-Life 2 running Direct3D 7 for a test that every video card could perform. In Half-Life we ran the blowup1104 benchmark at a resolution of 1028X768. In Half-Life 2 each card ran the HardwareOC Coast benchmark at a resolution of 1028X768 with all the details turned up [5]. In this scene, a fight between aliens and the main character is played out, producing many scenes that require shading and water effects. We chose this scene because it should thoroughly stress the cards. Besides Half-Life 2, we also tested each card using the video game Halo running Direct3D 8.1.

4.1.1.1 Half-Life

When testing the original Half-Life we found some interesting results. First of all, every video card scored very similar to each other card, regardless of generation. We believe that this occurred because of how simple the rendering was to draw this game. There was no shading and the calculations were so simple that every card was limited to what the CPU could send to the card. All the graphics card had to do was convert what the CPU gave it to output for the screen.

4.1.1.2 Half-Life 2

Half-Life 2 created more interesting results, as this game is newer and more stressful on the cards. Rendering this scene involves more computations than the original Half-Life, as every aspect of the scene is shaded and drawn in greater detail. As shown in Figure 7, this stress is depicted in the benchmark results. The latest two generations of cards, the 6600GT and ti4200 performed equally well, increasing at a rate of around 20% with each increment of CPU speed. On the other hand, the performance of the first two generations of video cards, the Riva TNT2 and Geforce2, leveled off in both cases. The cards' performance did not increase significantly even with a great increase of CPU speed. This could mean they reached a point where they could not calculate any more information, creating a bottleneck in

performance. The latest two generations were not maxed out in this set of tests and the performance was tied closely to the CPU speed.

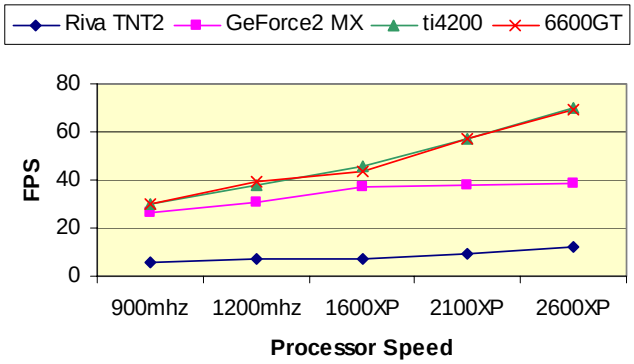


Figure 7: Half-Life D3D7 Generations Performance

4.1.1.3 Halo

To see if this trend would hold across other games, we tried the Halo benchmark. This test rendered using Direct3D 8.1 and was even more visually demanding. We could not run this test on the Riva TNT2 for it is unable to render Direct3D 8.1 applications. As you can see from Figure 8, the trend shown in section 4.1.1.2 continues here; the lower the generation of video card, the less increase of speed with each increment of CPU speed. This test even shows the 6600GT pull away from the ti4200 at a higher rate with each jump in CPU speed.

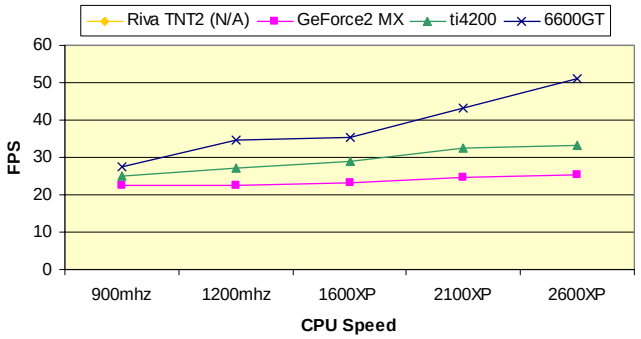


Figure 8: Halo DirectX 8.1 Benchmark

4.1.2 Synthetic Performance

Synthetic tests produce scenes to render that are not in any particular game. They are a suite of tests that are meant to use the latest 3D features available at its release time. The scores that the benchmarks output are a compilation of frames per second averages between four simulations multiplied by a common number which depicts how "important" that test is in the grand scheme of the test. For example, in 3DMark2001, a test which calculates the frames per second average for a DirectX 7 feature is "less important" than the tests for DirectX 8 features, thus would be multiplied by a lower number.

We used 3DMark 2000 and 3DMark2001SE for this test because they test Direct3D 7 and 8.1, which most of the cards support. As with many benchmark suites, these numbers are fairly arbitrary, but are meaningful if we are looking for trends, not raw number performance.

4.1.2.1 3DMark2000

As discussed earlier, 3DMark2000 tests the features available in Direct3D 7. Rendering these scenes is fairly simple by today's standards. As shown in Figure 9, the performance of the top two generations, the 6600GT and the ti4200, are again related closely to each other. An increase of CPU speed creates higher performance. The GeForce2 MX and the Riva TNT2 level off quickly and do not gain any significant performance increases even with the CPU speed doubling over the tests. These results seem very closely related to the Half-Life 2 tests from Section 4.1.1.1.

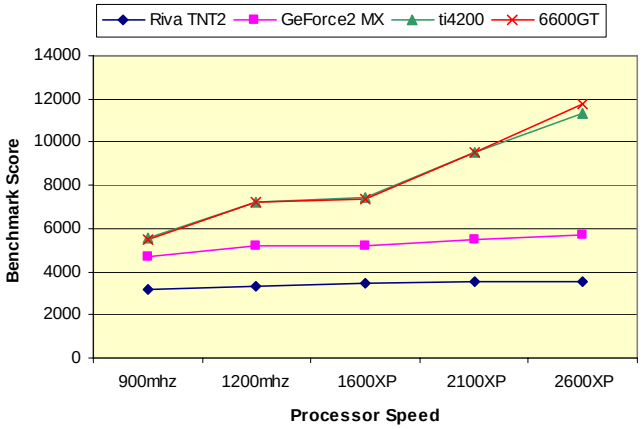


Figure 9: 3DMark2000 Benchmark Results

4.1.2.2 3DMark2001

3DMark2001 calculated a benchmark number based on three tests of Direct3D 8.1 functionality along with one test of Direct3D 7. Figure 10 shows the results from this test run. The Riva TNT2 and Geforce2 MX performance leveled off again at the higher speeds, showing only slight improvements as the CPU speeds increase. The latest two generations again showed great performance increases with each increment of CPU speed similar to that of the Halo test from Section 4.1.1.3.

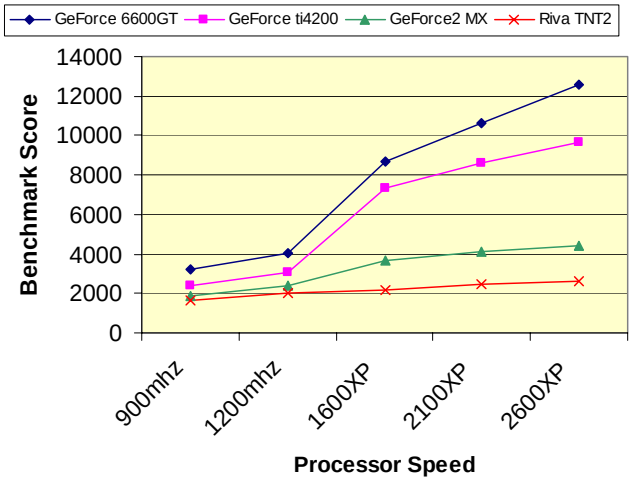


Figure 10: 3DMark2001 Benchmark Results

4.1.3 Analysis

The results found went against what our original hypotheses predicted. We hypothesized that the older graphics cards frames-per-second performance would increase at a greater rate with each increase of CPU speed. Furthermore, we predicted that newer video cards would peak out at their best performance and level off sooner. What we found suggests the opposite is the case. The newer graphics cards jumped in speed performance with the slightest increase of CPU speed while the older cards stayed at the same score regardless of the CPU speed. This could be a factor of the older cards reaching their limits and the newer cards performing well within their limits. It is hard to say what the exact reason is, but both gaming and synthetic tests produced similar results. Given slightly more CPU input, the newer generations of video cards can perform and render many more frames compared to their older siblings.

On another note, while the results may have shown some cards performing equally well in speed, one thing that increased greatly between the generations was image quality. This was the case even on the same game with the same graphical settings between the cards. For example, in the Halo tests both the GeForce2 MX and the GeForce ti4200 scored similar scores, only off by around 10 frames per second. While this may not have been a huge jump in speed performance, the jump in image quality and rendering was huge. The GeForce2 MX drew many textures plain white, and many characters were not drawn in the large battle scenes at all. The GeForce ti4200 rendered the scene as it was meant to be played, without rendering anomalies.

4.2 Direct3D Performance

4.2.1 Gaming Performance

In this set of tests we will be using the latest two video cards running Half-Life 2 benchmarks. We chose this game because Half-Life 2 provides the ability to choose the Direct3D version used for rendering. Both cards ran the HardwareOC Coast Half-Life 2 benchmark as mentioned in Section 4.1.1. Our GeForce 6600GT ran all three versions of Direct3D, but we could only test D3D 7 and 8.1 on our GeForce ti4200 because D3D 9 is not supported by the card.

Our results for the two tests are depicted in Figure 11 and 12. DirectX 7 provided the best frames-per-second performance on both cards. It increased at a high rate with the increase of CPU speed, scoring the same on both of the graphics cards. The frames-per-second speed of DirectX 8.1 moved up at almost the same rate as DirectX 7 on the 6600GT, but increased more slowly on the ti4200. DirectX 9 showed the least increase of frames-per-second performance throughout the different CPU speeds, increasing at a rate around 10-20% compared to the 20-30% increase shown while rendering D3D 7.

4.2.2 Synthetic Performance

We did not do a synthetic performance analysis on the differing DirectX versions, for the benchmark numbers are not recommended to be compared between versions of the tests. In other words, 3dMark2000 scores do not correlate to 3dMark2001SE scores and vice-versa. [5]

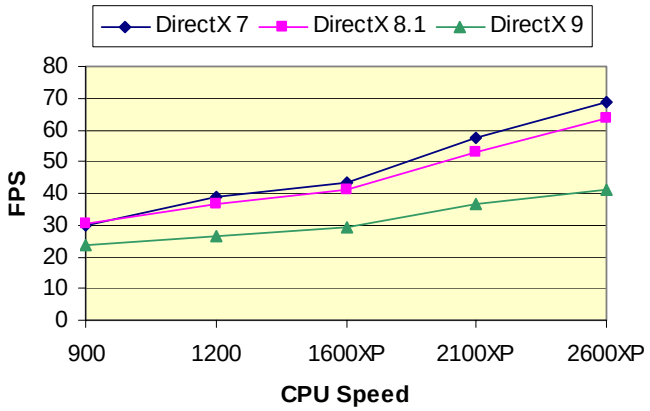


Figure 11: Half-Life 2 Direct3D comparison on 6600GT

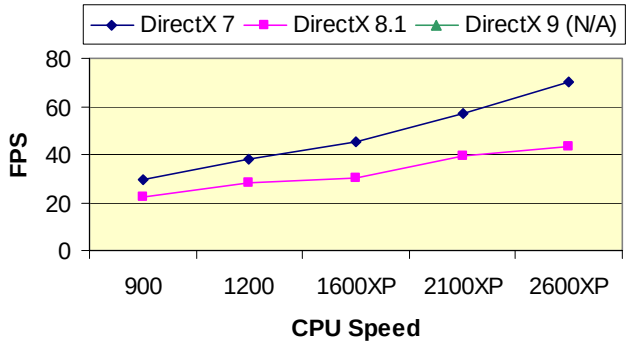


Figure 12: Half-Life 2 Direct3D comparison on ti4200

4.2.3 Analysis

The results found in this test tie in closely to what we believed the expected results would be. First of all, compared to the other versions of Direct3D, D3D 9's speed performance decreased at the slowest rate. This also meant that D3D 9's speed performance increased at the slowest rate as well, and it never exceeded the speed that its predecessors could achieve. However, speed is not everything in the realistic rendering of games. The human eye can only see around 32 frames-per-second, so everything above that just leaves room for frame rate fluctuation in the game. To make a scene realistic, it has to be able to draw scenes efficiently using real-time lighting effects and shadows. Direct3D 7 and 8.1 have limited abilities to do this, leading to the fast performance but simple images. This is why the performance curve increases greatly with each increase of CPU speed. Most of the calculations are done in the CPU, leaving the graphics card to do simple calculations and drawing. Direct3D 9 takes the route of increasing overall image quality while relying more on the graphics card. The speed performance of D3D 9 grows slower with each CPU speed increase, meaning that the CPU is handing more work to the GPU and it must do all the transformations and shader effects before the scene is drawn.

5. CONCLUSION

We addressed two major issues in 3D graphics in our research, the speed and quality of rendering. First of all, we found the trend that the latest two generations of video cards' performance has a much

higher correlation to the speed of the CPU than the previous generations. A slight increase of speed provided a major increase in performance. Although it went against our hypothesis, it is a good trend to see. Newer video cards are generating greater speed performances with the same amount of CPU speed. This needs to be a major trend if we are ever to reach cinema quality rendering in games.

Addressing the quality of rendering, our second hypothesis held true. Compared to Direct3D 7 and 8.1, Direct3D 9's performance was not closely related to the speed of the CPU. Direct3D 9 increased and decreased at the slowest rate, which could mean that the GPU was doing more work independently of the CPU than the other D3D versions. As discussed earlier, the fact that its speed performance increased the slowest is not necessarily a problem. Once graphics reach a speed of 32 frames-per-second, the rest is wasted since the human eye cannot perceive anything faster than this. What the human eye can perceive using Direct3D 9 is a greatly enhanced visual experience.

With each generation's increased ability to render images faster and with more quality, real-time 3D worlds are well within our grasp. We predict that future generations of video cards will continue the trends we found here. Graphics cards will gain significant speed performance increases with each generation and will be able to perform many more of calculations with each increase of CPU speed. Furthermore, future DirectX versions will make the speed of your system less of an issue for overall image quality. If we have made this much progress towards it in just ten years, imagine what ten more could bring.

6. ACKNOWLEDGMENTS

I would like to thank all my colleagues that helped me in developing my research and my paper, especially Dr. Gerald Cichanowski who guided me through the process and offered a lot of helpful advice. I would also like to thank my Dad for lending me his computers, and my roommates for putting up with all my equipment in the living room for the whole semester.

7. REFERENCES

[1] Nick Porcino. *Gaming Graphics: The Road to Revolution*. Queue. 2, 2. April 2004.

[2] J. L. D. Comba, Carlos A Dietrich, Christian A Pagot and Carlos E Scheidegger. *Computation on GPUs: from a programmable pipeline to an efficient stream processor*. Revista de Informatica Teórica e Aplicada. Porto Alegre: v.10, n.3, 2003.

[3] Erik Lindholm, Mark J. Kligard, Henry Moreton. *A User-Programmable Vertex Engine.* International Conference on Computer Graphics and Interactive Techniques. 149-158, 2001.

[4] William Mark, R. Steven Glanville, Kurt Akely, Mark Kilgard. *Cg: a system for programming graphics hardware in a C-like language.* ACM Transactions on Graphics (TOG). v.22, n.1. July 2003.

[5] Robert Richmond. "3D Benchmark Guide." http://www.sysopt.com/articles/3dbench/ July 4, 2001. Accessed on February 9, 2005.

[6] Microsoft Corporation. "DirectX Technology Overview." http://www.microsoft.com/windows/directx/default.aspx?url=/windows/directx/productinfo/overview/default.htm 18 March 2002. Accessed on March 2, 2005.

[7] Microsoft Corporation. "DirectX 7.0 Fact Sheet." http://www.microsoft.com/presspass/features/1999/09-22directx2.asp 1999. Accessed on March 2, 2005.

[8] Microsoft Corporation. "What's New in DirectX Graphics." http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dx81_c/directx_cpp/Graphics/WhatsNew.asp 2001. Accessed on March 2, 2005.

[9] Chris Maughan, Matthias Wloka, NVIDIA Corporation. "Vertex Shader Introduction." http://developer.nvidia.com/attach/6543 NVIDIA White paper. May 2001. Accessed on March 2, 2005.

[10] Microsoft Corporation. "What's New in DirectX Graphics." http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/directx9_c/directx/graphics/whatsnew.asp 2004. Accessed on March 2, 2005.

Visible Consistency of Animation Mechanisms

Padraic McGee
Student
Winona State University
(920)-231-8630
padraicmcgee@gmail.com

ABSTRACT
Computer animation is the task of continuously and rapidly drawing a scene to simulate the movement of objects. The number of times per second the scene is drawn is known as the frame rate. For motion to appear consistent regardless of computer rendering speed, the animation must be driven by a frame rate independent mechanism. Different animation control mechanisms will drive an animation to update or render at different times, based on how they interact with the underlying hardware. Temporal consistency, or smoothness of the animation, is an objective for some kinds of presentation. In this paper we compare the visible consistency of four animation mechanisms.

General Terms
Performance, Experimentation, Human Factors.

Keywords
Live Animation, Visual Consistency, Multithreading, Frame Rate Independence, OpenGL.

1.INTRODUCTION
Animation at its most basic level is the generation of successive images and their eventual playback in order. In this paper we consider live animation, where one image (known as a frame) is generated before the previous one is consumed (unlike a movie where the entire production is prerecorded prior to viewing). Live animation is required if there are dynamic elements to the animation such as user input. However, it requires a sustainable level of performance to meet the needs of the human visual system.

Animation applications written for personal computers have a significant challenge in taking full advantage of the available hardware while working with its limitations. The process can be simplified by using an algorithm that takes advantage of the CPU (central processing unit) and GPU (graphical processing unit) at different times by instructing the animation application when to render and when to update. In this paper we report on a study that compared four pre-existing animation algorithms that drive the underlying hardware in different ways while maintaining a consistent interface for the animation application.

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

We have certain expectations for each of these algorithms, such as a visually acceptable frame rate. It is also expected that rendering should be correct and perform any tasks necessary to ensure visual artifacts such as flashing will not be present. We also expect playback predictability, such that animations should not run at different rates on different computers.

After meeting these expectations, we tested each animation mechanism for the consistency of its visible updates, which we define to be the state of the animation each time it is rendered. The more consistent the visible updates are, the smoother the animation appears. By determining how these animation mechanisms compare with respect to visible update consistency, we can identify which will animate an arbitrary scene the smoothest. We expect an animation mechanism that employs both multithreading and fixed interval updating will provide the most stable animation foundation with respect to rendering and updating consistency as compared to their single threaded and variable interval update counterparts. A complete description of these methods is given in section 3.

2. BACKGROUND RESEARCH

2.1 Frame Rates
Animation is an illusion that seeks to trick the human eye into apprehending a set of disparate frames as the continuous visual flow of information we normally see. The more frames rendered per second, the easier it is to maintain this illusion. Animation begins to appear as motion at around 10 frames per second. There is no accepted limit to what frame rates the human eye can perceive, although it is generally accepted that there are limited returns beyond 60 frames per second under normal viewing circumstances. Some current mediums use significantly less. Movies shown in theaters are usually shown at a rate of 24 frames per second, but employ heavy use of motion blur and are shown under controlled lighting. Motion blur can be used to give the appearance of motion on a single frame. Televisions display around 30 frames of two interlaced images, which is a motion enhancing technique [1, 2].

When working on computer monitors it is more important to present sharp images correctly than give the illusion of motion. For this reason, current LCD and CRT monitors are capable of displaying at least 60 frames per second (also referred to as a refresh rate of 60 Hz). Significantly more expensive devices advertise frame rates of around 250. Animation frame rate is limited by monitor refresh frequency because it's impossible to see more frames than the monitor is capable of displaying. This limitation has a significant impact in defining the animation environment. If the animation does not use motion blurring

techniques, it is highly desirable to achieve frame rates that approach the upper bound of a monitor's capabilities [3].

It is entirely possible to simulate motion blur in computer animation [4]. It is usually done through a process called accumulation, where multiple renders are accumulated over time to form one frame. This is disadvantageous in a computationally critical activity such as live animation because it requires at least twice as much time for each frame (for N accumulations after the Nth frame). Because motion blurring is often impractical, computer animation generally relies on higher frame rates for smoother animation as compared to movies and television, where lower frame rate standards don't leave any choice. For this reason we did not consider motion blurring techniques when examining visual consistency [3].

2.2 Visual Anomalies

An important part of animation is the prevention of visual anomalies that may occur, especially those that may only appear under certain circumstances. Single buffering is when one memory buffer is used simultaneously to render to and display on the screen. Visual anomies such as flashing or partially rendered scenes can appear if single buffering is used. This occurs if the single display memory buffer is written to while the video device refreshes. This should always be prevented using a technique called double buffering, where all drawing is done to a back buffer and swapped with a front buffer which is then displayed [5]. This is done in a windowing system specific manner and will be represented generically as the term SwapBuffers().

Double buffering is necessary, but not sufficient in the prevention of visual anomalies. If the buffers are swapped while a screen refresh takes place then you will see frame tearing, where parts of two or more frames will be visible at once. You can avoid this problem by synchronizing buffer swaps to the time when the screen is preparing for the next refresh (which is known as VSYNC). In fact there is another reason for doing so. If two frames are drawn and the buffers are swapped twice on a single refresh, there will still be only one visible frame. This is guided by the simple fact that you can never see more frames than the physical refresh rate of your monitor. Spending computational time generating frames that will never been seen is a wasted effort [6].

2.3 Frame Rate Independence

You would not expect a movie to play faster on one television set than another. Likewise you wouldn't want live computer animation to run faster on a more capable machine. This is where the need for frame rate independence comes into play. Synchronizing to monitor refresh rates is a good start because it puts an upper bound on the frame rate, which is usually user defined to be of acceptable quality. However, since monitor refresh rates are configurable and there is no guarantee your simulation will always run fast enough to generate a frame for each monitor refresh, there is a definite need for a more sophisticated mechanism.

3. Animation Mechanisms

For each animation frame there are two tasks to accomplish: updating and rendering [10]. When the mechanism tells the application to update, it specifies the number of milliseconds that have passed since the last update. The application would then use this value to adjust all the scene objects. For example, in a physics simulation all objects would be moved according to their velocities and accelerations. When the application is instructed to render, it takes the current state of all scene objects and sends them to the graphics layer (which is OpenGL in our case) where they are transformed into an image in a process called rasterization. At the end of the render phase, the application calls a windowing system specific method to swap the display buffers, which ensures the newly created image will be displayed on screen as soon as the next refresh cycle begins. It's easiest to understand this by seeing an example.

The simplest animation mechanism is a single loop. In each cycle there is one update and one render followed by a SwapBuffers() call, as depicted by Figure 1. In this case, there will be a one to one correspondence of updates to renders. Each update is variable depending on how much time has passed since the last update. At the beginning of each update phase a timestamp is compared to a timestamp from the start of the previous update phase. This time will vary for each frame cycle. We will refer to this method as *single threaded variable interval* [8].

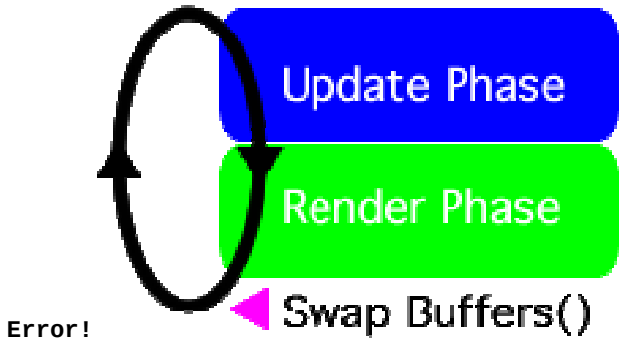


Figure 1. A sample animation loop

We depict a hypothetical timeline in Figure 2, which shows the execution of our single threaded variable interval animation. Blue regions still represent when updates occur and for how long. Green regions correspond to rendering periods and pink triangles show when the buffers are swapped. After each SwapBuffers() call there is a white area where the program idles while it waits for the next monitor refresh interval. Observe that each update time is of a different duration, which is based on the amount of time since the previous frame. These times will be relatively constant because a significant amount of time is spent waiting at the swap buffers call until the next screen refresh.

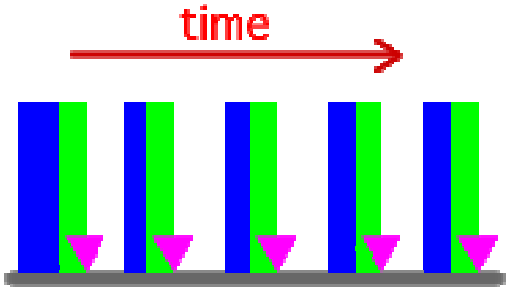


Figure 2. Single threaded variable interval animation

A variation on this method gives the illusion of regular updates to the animation layer. A fixed time increment is chosen and used for all updates. During the update phase a dynamic number of fixed duration updates are triggered based on the amount of time that has passed since the last cycle [8]. This means that if a large amount of time has passed since the previous cycle, multiple updates are triggered sequentially. For example, if we are using an update interval of 8 milliseconds and 17 milliseconds have passed since the last cycle, then two updates would be triggered. This would leave one millisecond unaccounted for, which would be added to the next update cycle's update duration. By comparison, the variable interval update mechanism would trigger a single update of 17 milliseconds.

Regulating the size of updates prevents a number of animation problems. Abnormally large updates could cause scene objects that would have collided to skip past each other. Abnormally small updates can introduce precision errors that accumulate over thousands of frames. This method is called *single threaded fixed interval* [8].

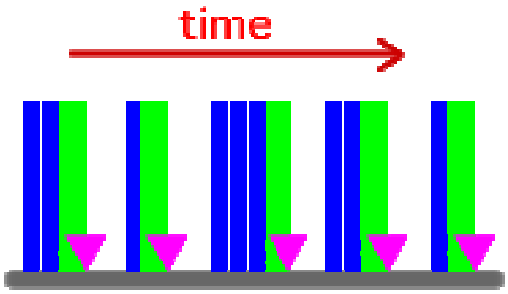


Figure 3. Single Threaded fixed interval animation

Figure 3 depicts an example of single threaded fixed interval animation. If a larger duration of time passes since the previous frame then multiple updates are triggered sequentially. If the fixed interval time is configured to be quite large it is entirely possible that some frames will not update. Since execution is still on a single thread there continue to be periods where the CPU idles waiting for the swap interval.

For performance reasons it is often undesirable to spend any percentage of computation time waiting for the graphics card to finish rendering [6]. This can be avoided by decoupling the rendering and updating tasks, which is commonly accomplished through multithreading. We will do this by using two threads, where all updates will occur in one thread and all rendering will be controlled from the second. By putting them each in their own thread we can continue to calculate updates even when rendering is blocked by the video system, as is depicted in Figure 4. Since both threads will be using the same object state information, use of a memory protection method such as semaphores is necessary. We must protect against an update and a render being triggered simultaneously. Once the SwapBuffers() call occurs we are free to update since the graphics card is working with its own copy of state information [3,6].

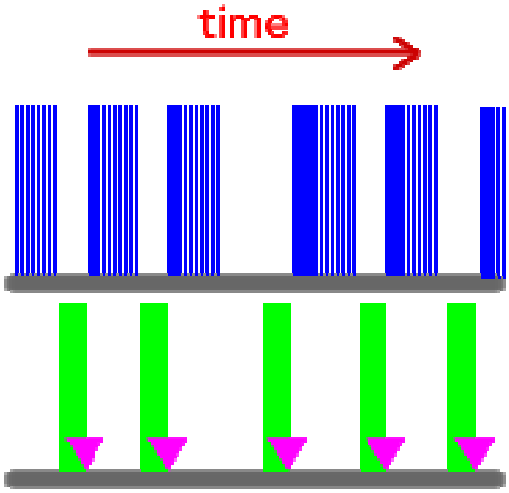


Figure 4. Variable interval multithreaded animation

Just as we could use fixed or variable interval updates with a single thread, either are applicable to multithreading. The differences between the two methods are more exaggerated because previously VSYNC (synchronizing to the monitor's frame rate) put an upper limit on the number of updates each second. Now that all updating is done in its own thread, it is no longer limited by any part of the system and will trigger variable updates at a much increased rate. This means the time intervals in-between updates will also decrease. *Multithreaded variable interval* updating, depicted in Figure 4, was observed to increase the number of updates per second by several orders of magnitude over its single threaded counterpart.

Finally we considered *multithreaded fixed interval*. Similarly to when it was used with a single threaded model, the fixed interval updating checks if sufficient time has passed before triggering an update. Since this is happening in a loop on its own thread there will often be nothing to do since very little time has elapsed since the last iteration. However, as is shown in Figure 5, updates can still be triggered as needed during the swap interval.

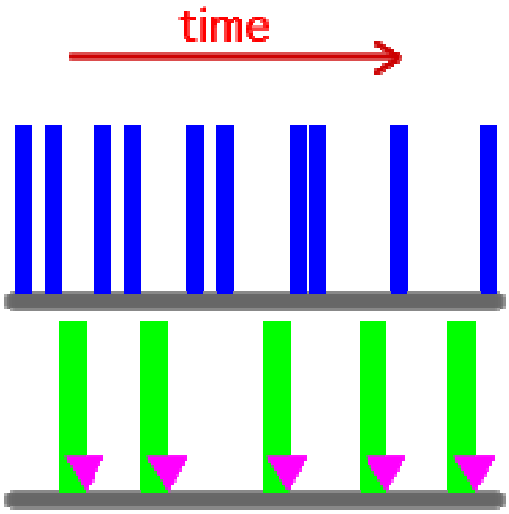


Figure 5. Fixed interval multithreaded animation

4. METHODOLOGY

Each of the four animation mechanisms described in the previous section was implemented and used to drive our test animations. The pseudo-code of the implementations follows.

A. Single threaded variable interval

Repeat:

Compute number of milliseconds between now and the start of the last cycle

Tell animation to update for computed number of milliseconds

Tell animation to render

Swap the buffers

B. Single threaded fixed interval

Fixed interval methods were configured to use a fixed interval of 8 milliseconds per update, which corresponds to 125 updates per second. This value was chosen because it is approximately twice the monitor frame rate of 60 frames per second.

Initialize working time to zero milliseconds.

Repeat:

Compute number of milliseconds between now and the start of the last cycle

Add this number to our working time

Repeat while working time is greater than fixed interval duration of 8 milliseconds:

Tell animation to update for 8 milliseconds

Subtract 8 milliseconds from working time

Tell animation to render

Swap the buffers

C. Multithreaded variable interval

Initialize Semaphore to one

In our update thread:

Repeat:

Wait on Semaphore

Compute number of milliseconds between now and the start of the last cycle

Tell animation to update for computed number of milliseconds

Post Semaphore

In our render thread:

Repeat:

Wait on Semaphore

Tell animation to render

Post Semaphore

Swap the buffers

D. Multithreaded fixed interval

We used the same fixed interval time of 8 milliseconds per update, which corresponds to 125 updates per second. Note that even though there are multiple updates per cycle, they are still all run in the same thread.

Initialize Semaphore to one.

In our update thread:

Initialize working time to zero milliseconds.

Repeat:

Wait on Semaphore

Compute number of milliseconds between now and the start of the last cycle

Add this number to our working time

Repeat while working time is greater than fixed interval duration of 8 milliseconds:

Tell animation to update for 8 milliseconds

Subtract 8 milliseconds from working time

Post Semaphore

In our render thread:

Repeat:

Wait on Semaphore

Tell animation to render

Post Semaphore

Swap the buffers

Profiling information was embedded into each mechanism to record when update and render events were triggered. Double buffering was used and VSYNC was enabled for all methods in order to prevent visual anomalies.

By recording when updates and renders occur relative to each other in a real environment, we can better understand how each theory on animation actually performs. Our main objective is the consistency of visible updates because these are what are actually seen. Doing so discards any rendering or updating that doesn't contribute to the visible output, such as duplicate renders or updates that aren't rendered. Visible updates depend on when rendering actually occurs and also the most recent corresponding update. The secondary metric is that consistently shorter visible update times are better than consistently longer visible update times because these will contribute to a higher frame rate.

Since we are recording when all updates and renders occur, computing visible updates is simply a matter of recording how much update time has passed at each render. For variable interval methods, this will closely resemble the amount of real time that has actually passed while for fixed interval methods this will always be a multiple of the fixed interval duration.

Tests were run against three animation data sets, which were designed to stress different parts of the hardware, as follows:

1. Render a single spinning sphere in OpenGL immediate mode, which consistently ran at the maximum frame rate of 60 frames per second.
2. Render four spinning spheres in OpenGL immediate mode, which consistently ran below the maximum frame rate.
3. Render four spinning spheres using OpenGL's optimized display list mode.

All of the animation data sets were designed to have the same computational needs throughout the entire test, such that any inconsistencies recorded could be attributed solely to the underlying animation mechanisms. Graphical computational consistency is addressed in other research such as [7].

Since all simulations reported exceptionally large times while loading, the first ten cycles were thrown out for all tests. Tests were run for a ten second period which generated approximately one thousand data points. The test computer was an Apple iBook single processor 900 MHz G3 with a 32MB ATI Mobility Radeon 7500 and an LCD running at 60 frames per second.

5. RESULTS AND ANALYSIS

The four charts of Figure 6a–d show the visible update times of Data Set 1 for the four animation models we tested. The x-axis is frame number and the y-axis shows the number of milliseconds in update time that passed since the previous render. Consistent times are best and shorter times are preferred. Initial spikes that occurred during loading were disregarded.

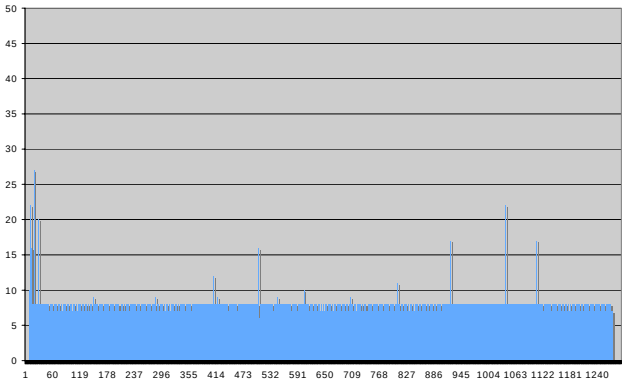


Figure 6a. Single threaded variable interval (Data Set 1)

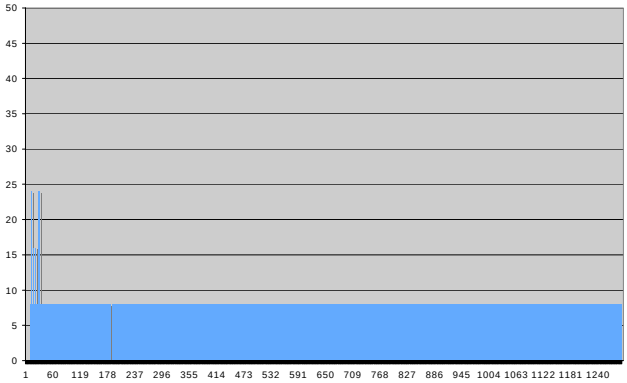


Figure 6b. Single threaded fixed interval (Data Set 1)

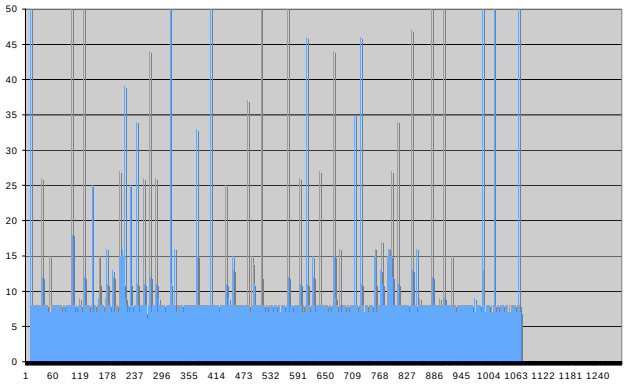


Figure 6c. Multithreaded variable interval (Data Set 1)

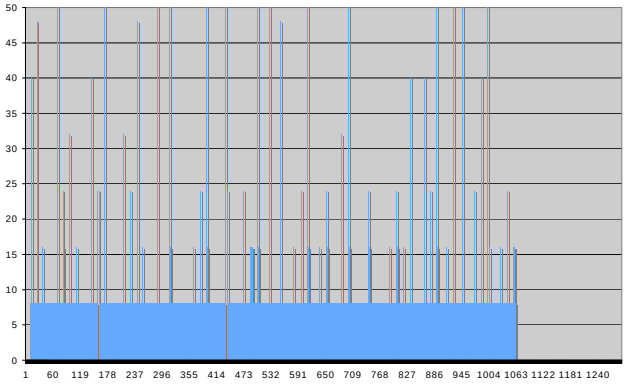


Figure 6d. Multithreaded fixed interval (Data Set 1)

It is obvious that the first two graphs have far fewer spikes, and the last two have multiple visible update times over the 33 millisecond range. This number is important because 33 millisecond frames correspond to animating at 30 frames per second. The multithreaded methods are therefore much more likely to have moments when the animation appears to stutter.

Table 7 shows the standard deviations for our four animation methods over the three data sets. The letters correspond to the methods as previously described at section 3.

Method	Test Run	Visible Update Standard Deviation
A	1	1.18084034
	2	3.90778201
	3	2.75414964
B	1	2.21053741
	2	3.56331835
	3	4.02149695
C	1	9.99542299
	2	13.946042
	3	25.0276482
D	1	9.38461892
	2	13.8106081
	3	24.4218264

Table 7. Visible Update Standard Deviation

The mean squared error for each method, which can be used to compare the relative durations of a visible update times is given in Table 8. The times are in milliseconds and were generated from the first test run. The results are given in the following chart.

Method	Mean Squared Error (ms)
A	57.98
B	61.05
C	182.4
D	168.5

Table 8. Visible Update Mean Squared Error

The mean squared error results show which method had the shortest visible update times on average. We can clearly see that the shorter times were for the single threaded methods, with the variable interval method edging out the fixed interval method. The multithreaded methods had average visible update times of almost three times longer.

6. CONCLUSIONS

The differences between the fixed interval updating and variable interval updating is much less than the differences between single threading and multithreading. Since neither variable nor interval updating was a clear winner in the visible update test, it follows that fixed interval methods are better since they guarantee a constant update time.

The effects of threading are fairly clear. Multithreaded examples had standard deviations around eight to nine times larger than their single threaded counterparts. Thread scheduling mechanisms at work clearly disadvantaged the consistency of updates. This may also be affected by running on a single processor machine. The lower mean squared error for single threaded models means overall time intervals were smaller, which is also desirable.

From these facts we can conclude that an animation system that uses single threaded fixed interval updating will have more consistent or roughly equivalent visual updates than one that uses any of the other three methods.

7. REFERENCES

[1] Amanatides, J. Mitchell, D. "Antialiasing of Interlaced Video Animation." *ACM Computer Graphics*, Vol. 24, No. 4, August 1990.

[2] Ware, C. Balakrishnan, R. "Reaching for Objects in VR Displays: Lag and Frame Rate." *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 4, pp 331-356 December 1994.

[3] McReynolds, T. Blythe, D. Programming with OpenGL: Advanced Rendering. SIGGRAPH 1997 Course. http://www.opengl.org/resources/tutorials/advanced/advanced97/notes/ (16.7 Tuning Animation)

[4] Korein, J. Badler, N. "Temporal Anti-Aliasing in Computer Generated Animation." *ACM Computer Graphics*, Vol. 17, No. 3, July 1983.

[5] Fernando, R. Harris, M. Wloka, M. Zeller, C. Programming Graphics Hardware. EUROGRAPHICS 2004.

[6] Apple Computer, OpenGL Performance Optimization: The Basics. http://developer.apple.com/technotes/tn2004/tn2093.html (Understanding VSYNCH)

[7] Funkhouser, T. Séquin, C. Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. Proceedings of SIGGRAPH 93. pp. 247-254. August 1993.

[8] Diener, A. Time-based animation. http://sacredsoftware.net/tutorials/Animation/TimeBasedAnimation.xhtml.

[9] Brostow, G. Essa, I. “Image-Based Motion Blur for Stop Motion Animation.” *ACM SIGGRAPH 2001*. pp 561-566. August 2001.

[10] Lipscomb, J. “Reversed Apparent Movement and Erratic Motion with many Refreshes per Update.” *Computer Graphics*, Vol. 14, No. 4, pp. 113-118, March 1981

[11] Brostow, G. Essa, I. “Image-Based Motion Blur for Stop Motion Animation.” *ACM SIGGRAPH 2001*. pp 561-566. August 2001.

Client-Server versus Peer-to-Peer Architecture: Comparisons for Streaming Video

Lisa McGarthwaite
Saint Mary's University of Minnesota
700 Terrace Heights
Winona MN 55987
lcmcga01@smumn.edu

ABSTRACT
Today streaming media is gaining importance as a way to efficiently deliver real-time news and events. Currently the client-server architecture is chiefly used to stream data to end-users. This network design leads to performance impediments for the client when the server becomes overwhelmed with requests. A degradation of quality can also occur if the server is on a lossy network. Because of the transfer protocols used machines are not able to recover the packets that were lost. Recently Peer-to-Peer (P2P) networks have been proposed as an alternative solution. P2P applications allow multiple clients to send desired data to the requester. We hypothesized that the P2P network architecture would use less bandwidth and increase the frame rate of the media streamed. While network Transfer Control Protocol (TCP) traffic did increase for P2P topology, this architecture proved to afford the user with a better frame rate. Most importantly, bandwidth usage for P2P on average was less than client-server architecture.

Categories and Subject Descriptors
C.2.2 [**Network Protocols**]:Application; C.2.5[**Computer-Communication Networks**]: Local and Wide-Area Networks – Internet

General Terms
Measurements, Performance

Keywords
Peer-to-Peer, Client-Server, Streaming media, Networks

1. INTRODUCTION
Streaming media is a recent technology with its earliest usage beginning in 1998 for Internet radio. Today large news corporations, such as MSN, and even local businesses use streaming media to broadcast news and events. Even the ordinary user is beginning to take advantage of this technology by broadcasting radio and video to the world.

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

While streaming media is becoming a very popular way to share information, there are limitations to this technology. If a data source is highly requested, the server becomes overwhelmed with the request load and is unable to deliver or only poorly deliver the streamed media [9]. To overcome this obstacle, corporations are using distributed servers. When requests are made, they are directed to the closest available server. This solution is costly in terms of overhead and machinery needed. To avoid this complication, recent focus has shifted to Peer-to-Peer (P2P) networks. Growing from the increasingly popular file sharing technology, P2P networks allow streamed media to be broadcast from multiple nodes to a requester. This topology would be useful when a media file is in high demand. Once a client receives a stream, it can then broadcast the stream to other clients in the network requesting the same media. Although controversy involving the sharing of copyrighted material is a concern using this type of system, there are definite legal and desirable uses. While research has been conducted to create P2P streamed media applications, *performance comparisons* of traditional client-server versus P2P network topologies are scarce.

We believe that a P2P network will utilize less bandwidth and provide better quality for streaming video. To test the effectiveness of client-server and P2P networks, the two topologies were tested with 5 machines across three networks. Network monitoring tools were used to keep track of available bandwidth and video information. Comparisons of network traffic were also recorded and analyzed.

2. BACKGROUND
In order to understand the nature of our experiment, some important concepts must be known. These areas include background information on streaming media, network protocols, client-server networks, and P2P networks.

2.1 Streaming Media
Streaming media allows users to broadcast media files in real-time or on-demand to the Internet. The actual size of video and audio files is quite large, impeding the use of the file on the Internet. Using lossy compression algorithms, the file size is compressed, which results in a degradation of audio and visual quality. To stream a media file, each compressed video or audio clip is divided into packets, and each packet is sent in sequence to the client. The client is responsible for decompressing the packets and playing the pieces in order as soon as the packets are received. The rate at which each packet is played is called frame rate or frames per second (fps). Since the clip is not sent all at once, file size becomes less of an issue. One concern is the loss

of packets in transit on a network. The more packets that are dropped results in the file suffering a loss in visual or audio quality. Some streaming media broadcasters have tried to overcome this problem with progressive streaming. Packets are placed in a buffer before being played. Once the buffer reaches certain fullness, the clip begins to play. As the file begins to play, the buffer continues to collect packets. Many of the interruptions in service are then not as noticeable.

2.1 Network Protocols

To stream media over the Internet, certain protocols must be followed. Two transport protocols are Transfer Control Protocol (TCP) and User Datagram Protocol (UDP). If a packet is dropped in TCP, the server tries to resend the lost packet before sending the client any further packets, which can cause a lengthy delay in the stream. This protocol emphasizes maximum reliability and data integrity rather than timeliness. This focus has made it more appropriate for sending files and critical information than streamed media [3].

While TCP allows files to be transmitted over IP, it is not generally the best option for streaming media. The adoption of the Internet protocol User Datagram Protocol (UDP) and Real Time Streaming Protocol (RTSP) has made the transmission of data even more efficient. RTSP is a transport protocol layer that provides mechanisms for sending and receiving applications to support streaming media. While RTSP can be carried by either TCP or UDP, it typically runs on top of UDP. In contrast to TCP, when a UDP or RTSP packet is dropped, the server continues sending information and does not try to resend the lost packet. This drop causes only a brief interruption in the stream. These protocols are focused on the timeliness of delivery rather than the reliability of sending packets. However, some streaming media broadcasters choose not to use RTSP or UDP since some firewalls block these protocols, and other protocols offer a more reliable connection.

2.2 Network Architectures

Traditionally client-server architectures have been used to stream media. Figure 1 illustrates this network topology. The server holds the content that users wish to see. When a request is made for a file, the server looks up the file in its directories and begins to stream the media to the client. The stream only goes to one client, and clients do not share the stream with each other.

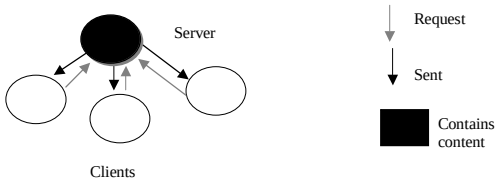


Figure 1. Client-Server structure.

In contrast to client-server, true P2P networks require no central server to provide content or look-up services. Each member of the network can act as client or server (see Figure 2). When a client makes a request for a file, the machines on the network perform a distributed query to find the file. For example, in Gnutell, a file sharing P2P, the client knows of at least one other IP address. It queries this machine, and that same machine then queries other computers on the network, thus performing a ripple effect to locate the desired content. Those nodes that are willing to share content respond by sending the content to the requester. Using this network topology, more than one peer is able to send data to the requester. This is known as *parallel* streaming.

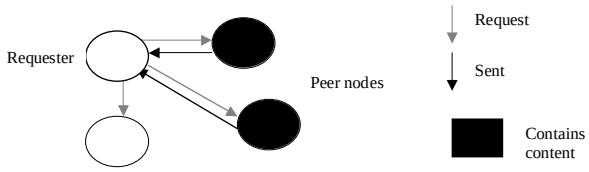


Figure 2. P2P structure.

Another version of P2P networks includes server-mediated networks. A central server is used to keep track of all peers in the network. This provides a more efficient way of determining what peers and content are available in the network. Recent studies have been conducted to find the most efficient means of creating a peer network [2, 4, 7, 8, 9, 11]. These studies are concerned with the underlying architecture of the P2P system. In addition, it has been proven that the more altruistic, e.g. the more willing peers are to share resources the more efficient a P2P system becomes [12].

3. SOFTWARE

To conduct our research, we located two software tools to provide a P2P and a client-server network. PeerCast and Apple's QuickTime server were chosen [5]. PeerCast is an open source P2P application that supports streaming audio and video. A server-mediated P2P, this application allows users to broadcast streams to other peers running a PeerCast client. A PeerCast client can act as both server and client. The user is able to set the number of channels that he/she would like to stream out. Other users can look up the various peer broadcasts on the *yellow pages*, the application's directory of peers. PeerCast also supports private channels, in which the peer is not directly listed on the yellow pages. Other peers must know the IP address of the streaming peer to be able to access the content. By limiting the number of direct streams, the application forces requesting peers to act as the streamer to other peers (see Figure 3). Once the direct stream limit is reached, two in the case of Figure 3, further requesting nodes receive the stream from the peers receiving the direct stream. When a client receives a stream, the client is put in RECEIVE mode. This mode allows other users to connect to the client and listen to the same stream. When the user is done with the file, the PeerCast stream is put into IDLE mode. Other users are still able to use the client as a streamer. To view the file, Windows Media Player was used [10]. This program offers the ability to view the bandwidth, measured in kilobits per second (kbps), and frame per second (fps) of the streamed video.

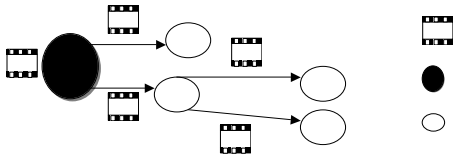


Figure 3. PeerCast streaming structure.

For the client-server network, we used Apple's QuickTime Server to hold our video content. A Sorenson 108 kilobit per second (kbps) encoded file was placed on the server. Apple's QuickTime Player was used to access the file [6]. This program also offers the ability to view fps, kbps, and packet drops of the file while it is being played.

In order to capture the network traffic occurring during the experiment, we used the network monitor tool CommView [1]. By configuring filters for capture and display, we were able to view the various IP protocol traffic specific to our machines and generate graphs of the output.

4. METHODS

Our research methodology was broken up into three phases: preliminary research, experiment mock-up, and experimentation.

4.1 Environment Set-up

Because of the nature of this experiment, the environment ranged across three networks. For a listing of the machines used in this experiment, please refer to Appendix A. The same machines were used for both the client-server and P2P clients. To ensure a correct measurement, services and protocols not needed in the experiment were disabled on the machines whenever possible. For the client-server portion of our experiment, we used a 5.1 MB Sorensen 108 kilobit per second (kbps) encoded video file. To accurately test the P2P structure against this file, we chose a 108 kbps encoded stream already found on PeerCast's yellow pages. We recorded measurements of frame rate and bandwidth of each stream for five minutes, which we felt would give us an accurate idea of the performance of each stream.

4.2 Experiment Mock-Up

Before beginning the actual experiment, a mock-up was performed on both architectures. This test proved that each system was operational and provided the sequence of events that must be followed to conduct each trial. To provide a starting ground for comparison, the video file was played from the desktop of one machine using QuickTime and measurements were taken of the frame rate, bit rate, and packet count. Using the client-server architecture, the file was requested from each machine. The P2P architecture test also used each machine to request the file from PeerCast and play it.

4.3 Experimentation

Two types of tests were performed: access and stress. To conduct the actual experiment, we followed a strict pattern. First we accessed the file and recorded the bit rate, frame rate and number of packets received and dropped. Once all the results were in, we analyzed the results and did comparisons of the two types of architectures.

The first test was concerned with accessing the file from a single machine at different locations on three different networks. These networks included Saint Mary's University of Minnesota, the Computer Science Department of Saint Mary's University, and Winona State University. The goal of this test was to view how performance was affected when the stream was viewed on different networks. During the client-server portion of our testing, the file was accessed on through QuickTime Player by using `File> Open URL` and entering the address of the file. Using `Window> Show Movie Info` and `Movie> Get Movie Properties` options we were able to attain the frame rate, bit rate, and the number of packets dropped.

The P2P stream was viewed with Windows Media Player. The stream's statistics were accessed by `File>Statistics>Advanced`. During the streaming, the highest and lowest bit rates were recorded as well as the highest and lowest frame rates. By recording these highs and lows we would be able to compare which architecture afforded the most benefits to the user. After the file completed streaming, statistics were taken of the TCP and UPD traffic.

The second test was the stress test and concerned with the outcome of multiple machines accessing the same file at the same time. To test this part of the experiment, the same file was accessed simultaneously from four machines. Again, the bit rate, packets received/dropped, and frame rate were recorded using the same methods previously stated.

5. RESULTS

The experimentation of our client-server and P2P networks were quite successful. By examining the amount of generated packets, we were able to determine the impact of the two types of streaming. The statistics of the average number of TCP and UDP packets received were calculated (See Table 1). Because the P2P used http protocol to stream, this topology generated more TCP packets while the average UPD packets were less. A P2P network would experience a greater number of TCP packets than a client-server topology.

Architecture	TCP packets	UDP packets
client-server	272	15,277
P2P	5, 961	14, 056

Table 1: Average TCP and UDP Traffic

As we predicted, P2P streaming afforded a higher frame rate than client-server (See Figure 4). The P2P streaming was able to achieve an average high of 15.85 fps compared to 10.302 fps for the client-server. However, P2P also was more prone to stream interruptions and access problems.

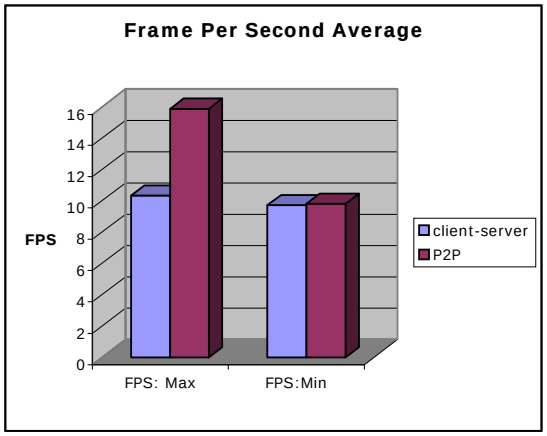


Figure 4. Average Frame Per Second (fps)

The main focus of our interest was the comparisons of bandwidth. As we suspected, P2P on average used less bandwidth than the client-server (P2P maximum/minimum 119/74 kbps versus client-server's maximum/minimum 224/107 kbps). Even during the stress test, P2P on average demanded less bandwidth than the client-server (See Figure 6). Aside from the technical measurements, we observed that the overall quality of the streamed video was greater on the P2P network. This improvement can be attributed the higher frame rate that was attained.

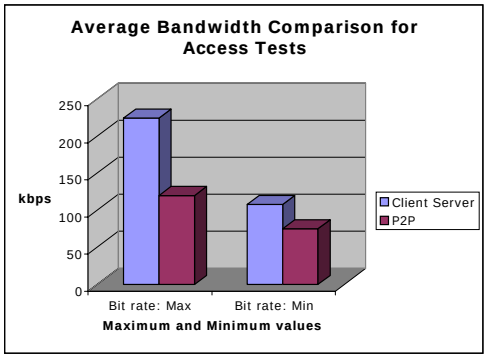


Figure 5. Average Bandwidth for Access tests

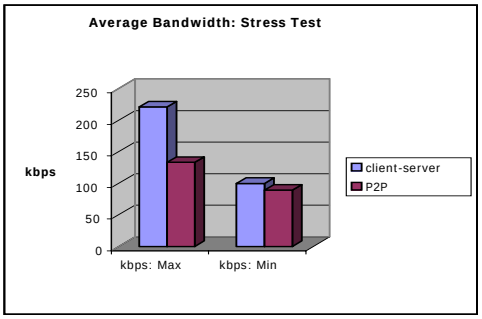


Figure 6. Average Bandwidth during Stress test

6. CONCLUSTIONS

As stated previously, streaming media is becoming an industrial standard way to deliver real-time media to the public. Because of the server's limitations, P2P networks have become a viable alternative to the traditional client-server topology. We believed, and our research has shown that P2P networks offer an increase in frame rate and decrease bandwidth usage. The client-server architecture did, however, prove to provide a more reliable stream.

Future research will attempt the same experiments on a much larger scale. First, we would collect and compare statistics from multiple streams. Next we would recommend using a grid simulator or an available test bed such as PlanetLab to test the streams on a larger network. Other studies will focus on finding the optimal types of machines that would afford the greatest quality of streaming media.

7. ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Cichanowski for all his help during the process of this paper. Also, I would like to thank Dave Hajaglou for letting me pick his brain whenever I had questions. Finally, I have to give a huge thank you to Eric Heukeshoven for allowing me to use the SMU streaming server and providing me with the file that I used to stream.

8. REFERENCES

[1] CommView. Available from http://www.tamos.com/download/main/. Accessed 2005 March 20

[2] Dejan Kosti´c, Adolfo Rodriguez, Jeannine Albrecht, and Amin Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of SOSP* 2003

[3] Josh Beggs and Dylan Thede. *Designing Web Audio*. "Chapter 5 Introduction to Streaming Media" January 2001. Available from: http://www.oreilly.com/catalog/sound/chapter/ch05.html. Accessed 2005 February 9

[4] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. SplitStream: High-Bandwidth Multicast in Cooperative Environments. In *Proceedings of SOSP* 2003

[5] PeerCast. Available from http://www.peercast.org. Accessed 2005 January 30

[6] QuickTime Available from http://www.apple.com/quicktime. Accessed 2005 February 8

[7] Reza Rejaie and Antonio Ortega. PALS: Peer-to-Peer Adaptive Layered Streaming. From *NOSSDAV* 2003

[8] Song Ye and Fillia Makedon. Collaboration-Aware Peer-to-Peer Media Streaming. October 16th, 2004

[9] Venkata Padmanabhan, Helen Wang, Philip Chou and Kunwadee Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Workshop on Network and Operating System Support for Digital Audio and Video*, Miami Beach, Florida, 2002.

[10] Windows Media Player. Available from http://www.microsoft.com/windows/windowsmedia/9series/player.aspx. Accessed 2005 February 8

[11] Yang Guo, Kyoungwon Suh, Jim Kurose, and Don Towsley. P2Pcast: Peer-to-Peer Patching Scheme for VoD Service. *WWW2003* May 20-24, 2003

[12] [Yang-hua Chu and Hui Zhang. Considering Altruism in Peer-to-Peer Internet Streaming Broadcast. From *NOSSDAV'04*, June 16-18th, 2004

Appendix A

Computers used:

Computer type	OS	Processor	RAM
Laptop	Windows XP	1.4 GHz Intel Pentium M Processor	512 MB
Desktop	Windows 2000 (NT)	X86 Family 6 Model 8 Stepping	254.5 MB
Desktop	Windows 2000 (NT)	X86 Family 6 Model 8 Stepping	254.5 MB
Desktop	Windows 2000 (NT)	X86 Family 6 Model 8 Stepping	254.5 MB
Desktop	Windows 2000 (NT)	2.53 GHz Intel (R) Pentium	510 MB

A Comparison of Firewall Performance in Distributed Systems

Logan Twedt
Saint Mary's University of Minnesota
700 Terrace Heights #955
Winona, MN 55987
1- (507) 494-6042
Latwed01@smumn.edu

ABSTRACT
As the amount of data being transferred over networks increases, the firewalls used to protect private networks must process traffic both faster and with greater reliability. A distributed firewall is a firewall that enforces each hosts policy from the host itself. In this paper, we show that distributed firewalls may provide faster access and higher data throughput than conventional firewalls, which only reside on the entry points of networks. We conclude that this can, to a certain extent, be attributed to its separation of policy enforcement to each network endpoint.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – security *and protection, firewalls.*

K.6.5 [**Management of Computer and Information Systems**]: Security and Protection – *authentication, unauthorized access*

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *distributed networks*

C.2.2 [**Computer-Communication Networks**]: Network Protocols – *applications (FTP).*

General Terms
Measurement, Performance, Design, Reliability, Experimentation, Security.

Keywords
Distributed Firewall, Netfilter, FTP, Network Security

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

1. INTRODUCTION
A firewall is traditionally thought of as a system residing between a public and private network (Figure 1) that controls access to that private network. A conventional firewall, also known as a Single Entry Point (SEP) firewall, consists of a machine enforcing a policy from the edge, or entry point, of a network. For example, a private network containing two servers, one web server and one mail server, would have a firewall at the network's entry point letting through only the mail traffic destined for the mail server and HTTP traffic destined only to the web server. The policy is enforced at the entry point node, by the SEP firewall. Contrast this with a distributed firewall, where the policy enforcement process is separated so that each server's individual policy is "enforced at each individual network endpoint" [3], or at the server itself. Any policy enforced for all nodes is still enforced by the entry point node.

Conventional firewalls require more CPU usage than distributed firewalls for equal amounts of traffic, because all of the policies for all nodes are enforced by one machine, instead of splitting the time to process traffic over the network's nodes. This difference in processing power is more evident at higher volumes of network traffic.

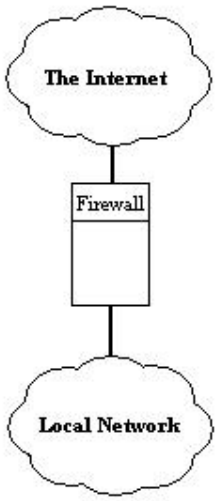


Figure 1. A Firewall residing between two networks

In networks having multiple entry points (i.e. networks with wireless access points), use of a conventional firewall would

require another firewall (enforcing all of the individual policies for the entire network) to be inserted at each point of entry. The network employing the distributed firewall does not need another complex firewall at each point of entry. The firewall at that point of entry would enforce a much smaller set of rules, providing connected hosts greater throughput to and from the private network. Showing that a distributed firewall performs better gives system and network administrators a reason to switch to the distributed firewall scheme, trading the money it takes to switch for faster and more reliable connections to their servers.

The goal of this research was to show that the average file download time through a network utilizing a distributed firewall was faster compared to the same downloads from the same servers with a conventional firewall. In addition, the network with the distributed firewall is able to process a larger number of downloads, meaning there is a higher data transfer throughput. At higher transfer rates the number of packets resent will increase with a greater momentum, due to the firewall node's inability to process traffic at those rates of transfer. Greater CPU usage needs to be dedicated to traffic processing from the servers with the distributed firewall, but ultimately there is less network congestion and greater throughput due to the lack of a bottleneck at the entry point.

2. METHODOLOGY

2.1 Test Bed

Figure 2 shows the test bed that was set up for comparing a conventional firewall versus a distributed firewall. One physical network was constructed with flexible software configurations at each node. The test bed contained four "client" nodes ranging from 550 MHz to 1.2 GHz, six "server" nodes all rated near 266 MHz, and one entry point (SEP) firewall node, also rated at 266 MHz. The server nodes were connected to one of the firewall node's two network interfaces through a layer 2 switch. The client nodes were connected to the firewall node's other network interface through a different layer 2 switch (Figure 2).

All nodes, both client and server, ran a stripped down (telnetd, X, etc., removed) version of Linux [4], called Linux From Scratch [6]. The servers had the FTP server Vsftpd [7] installed for serving a 5 MB test file during testing. The clients had the FTP client Wget [8] installed on them for requesting files from server nodes.

All firewalls fall into either the stateful inspection or filtering (stateless) categories. Stateless firewalls treat all packets as individual packets in deciding whether or not to accept, deny, or reject the packet. In contrast, stateful inspection firewalls track existing connections (such as TCP streams) [2]. The use of a stateful firewall amplifies the firewall's need for processing power. Using Netfilter/IPTABLES [9], a stateful firewall was constructed on the firewall node and on each server node.

During testing of the conventional firewall, the firewall software on each server node was set to accept all traffic, as was the firewall node's software during control tests. During testing of the distributed firewall the configuration previously enforced by the firewall node was split based on each individual server's policy. A general firewall still existed on the firewall node.

A set of scripts ran the automated FTP client, Wget, a given number of times in the background in order to simulate downloads as concurrently as possible. The head client used SSH to communicate with the other clients, notifying them of the proper time to execute their scripts. The traffic was scaled up in an attempt to saturate, or overload, the firewall.

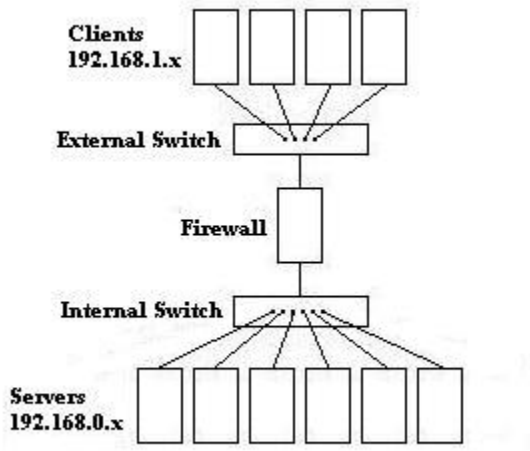


Figure 2. Test Bed Setup

2.2 Test Plan

Each test set increased in size by a multiple of the first test set. In the first test set, each client downloads the file 10 times from each server, for a total of 60 files downloaded per client. With four clients, this makes 240 downloads across the four clients for the first test set. The second test set doubled the first, having each client download the file 20 times from each server, for a total of 480 downloads. The third test set tripled the first test set, and the fourth test set quadrupled the first test set. The point at which the firewall became overloaded was easier to determine using sequential increases in the number of downloads. The number of packets resent was monitored on a test set-to-test set basis to gauge system load behavior. As firewalls are challenged to handle higher traffic loads, resending due to lost packets increases. Therefore, we monitored packet resends in order to get a sense of overload threshold for our conventional firewall configuration.

Both authorized and unauthorized access attempts were generated so that the firewall had traffic that was, to some extent, similar to a real world network. The server nodes accept anonymous local requests, authenticated local requests, anonymous remote requests, and authenticated remote requests. Some server nodes had different policies about accepting some of these types of connections, thus some traffic was rejected at the server's level, and some at the firewall's level. Because both Vsftpd and IPTABLES reject different types of unauthorized connections, any traffic that passed through the firewall that should be blocked was logged by the FTP server, with all other blocked traffic logged by the firewall. The log

files generated by Wget showed an average download speed in KB/sec for each download that took place. Dividing total file size by these speeds gave a download time accurate to within 0.1 seconds. During each test set, the CPU usage was logged through the use of the UNIX command, 'top'. Comparison of CPU usage for the firewall node and server nodes showed the load level at which the nodes became overloaded. Usage was observed for all test sets for both firewall configurations. Deciding whether or not the distributed firewall has a performance advantage over the conventional firewall is straightforward given these metrics.

3. RESULTS AND ANALYSIS

From the graphed results in Figure 3, it is apparent that both the conventional and distributed firewall became somewhat overloaded during testing, in particular test set 4, having 960 simultaneous downloads. However, this is at least partially due to hardware constraints, as is shown through comparison with the control group (no firewall). All systems had an average download time of 40 seconds in the first test set, with only 60 downloads initiated by each of the four clients. Both firewalls download times increased significantly between the second and third test sets and leveled out at 131.68 seconds (Distributed) and 134.43 seconds (Conventional) in the fourth test set. From these results it can be inferred that the hardware was reaching a maximum rate of data transfer.

Figure 4 shows the amounts of resent packets during each of the four test sets. If the firewall was being overrun, the graph would show a sudden increase, signifying the point at which the firewall becomes overloaded with traffic – the point at which it would stop letting traffic through (thus forcing it to be resent). Instead, there is little difference across the graph, only a slight increase in resent packets. The conventional firewall showed a gradual increase from 167 to 283 packets resent, where the distributed firewall showed a gradual increase from 147 to 240. The client and server nodes simply could not generate enough traffic to overwhelm the firewall node.

Figure 5 shows CPU usage statistics for the firewall node. Results from averaging server node CPU usage proved inconclusive, as the results spanned a very broad range, and did not hold a particular pattern during any tests. Simply using the firewall node as a bridge between networks used an average of 56% of available CPU time for the first test set. Adding the distributed firewall to the network gave the firewall node a more difficult time and boosted the usage to 67%, while the conventional firewall made the firewall node process even harder, at 70% of its capacity CPU usage. The conventional firewall's firewall node rose to 82% CPU usage by the fourth test set, showing that it was becoming fairly overloaded (although not completely) with the traffic it was processing. The distributed firewall's firewall node showed only 76% CPU usage during the fourth test set. It would be better to rate the distributed firewall's CPU usage using statistics from all nodes running firewall software, but results from this were fairly skewed.

4. CONCLUSIONS

Our results show that the hardware we used was not able to saturate the network enough to overload the firewall. Even though this overload did not occur, our results indicate that the distributed firewall configuration can handle more traffic than the conventional firewall. The distributed firewall nearly always outperformed the conventional firewall, on most test sets, and in all metrics recorded. Our observations are qualified by two things. First, FTP was the only protocol tested. It is possible that a group of servers offering different services would not saturate the firewall in the same way that FTP requests did in these tests. Second, the firewall was configured to track connections. Packet filtering firewalls, as opposed to the stateful firewall used, may not be saturated as easily because of the connection tracking employed by the Netfilter/IPTABLES [9] firewall. This may prevent the firewall from becoming saturated at relatively low connection amounts, but it could be assumed that if the traffic was scaled up to levels that our hardware could not reach in these tests the results would be relatively similar to the ideas put forth by Bellovin [1].

5. FUTURE WORK

As prescribed in concluding remarks, testing must now be done with other services, and possibly multiple services on each machine within the server side of the network. As the authors of [1] and [2] suggest, the use of IPSec in the firewall for public key authentication forces greater CPU usage in the firewall, making the effects of an overloaded firewall easier to see. Also, different FTP server and firewall software should be tested, along with different operating systems, to eliminate the variable of software capabilities. Larger network topologies should be tested, with faster machines, to test the affects of firewalls on more modern and widely used hardware.

6. ACKNOWLEDGMENTS

Many thanks to the faculty and staff that aided in the design, development, and execution of this research, including Gerald W. Cichanowski, Ann C. Smith, Joan M. Francioni, Chris M. Johnson, and David E. Hajoglou.

7. REFERENCES

[1] S. M. Bellovin. Distributed Firewalls. *;login: magazine, special issue on security*, November, 1999.

[2] C. P. Pfleeger & S. L. Pfleeger. *Security in Computing, Third Edition.* Prentice Hall, 2003.

[3] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. *Proceedings of the 7th ACM conference on Computer and Communications Security*, 2000.

[4] M. Miller and J. Morris. Centralized administration of Distributed Firewalls. *Proceedings of the Tenth USENIX System Administration Conference*, September, 1996.

[5] W. R. Cheswick & S. M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 1994.

[6] Linux From Scratch. http://www.linuxfromscratch.org/

[7] Vsftpd. http://vsftpd.beasts.org/

[8] GNU wget.
http://www.gnu.org/software/wget/wget.html

[9] The Netfilter/IPTABLES project.
http://www.netfilter.org

8. APPENDIX

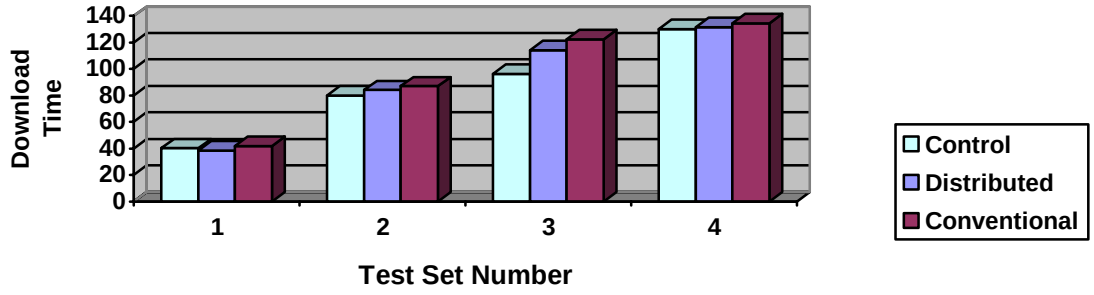


Figure 3. Download Times

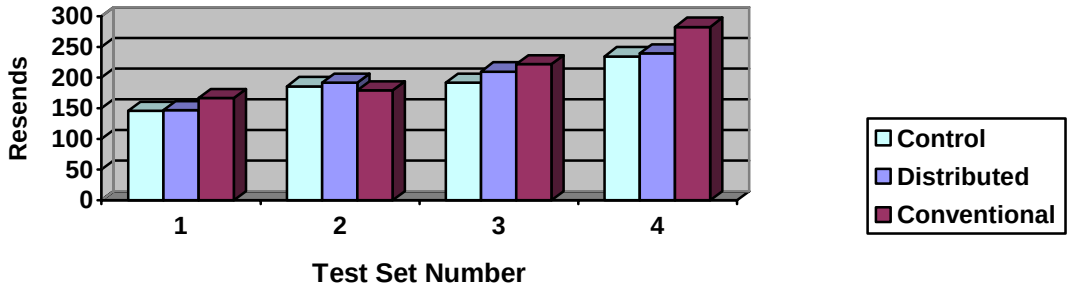


Figure 4. Resends per Download for each Session

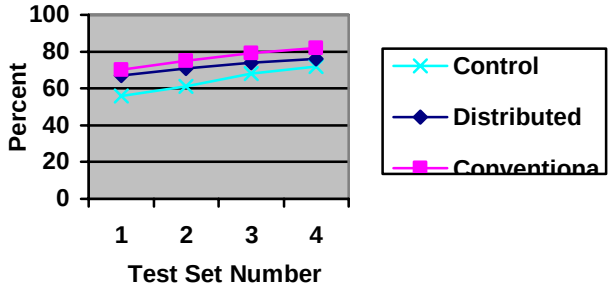


Figure 5. Firewall Node CPU Usage

Initialization Settings of a Force Feedback Device: Analysis of User Preferences

Michele Clarke
Saint Mary's University of Minnesota
700 Terrace Heights #413
Winona, MN 55987
1-563-581-1157
meclarke@gmail.com

ABSTRACT
Blind and visually impaired users often find themselves limited by the availability and accessibility of usable maps. A digital 3D software environment designed for spatial exploration with a force feedback device will enhance map usage for such users. One such device, the Logitech Force Feedback Mouse, can be programmed to provide a user with different levels of navigational guidance. This paper explores the effects of different initialization settings for navigational forces. Since such programmed settings are subject to the needs of individual users, usability tests were designed to determine the average force settings preferred by a set of tested users. Results are based on both the analysis of the usability tests and reaction surveys. The complied results define a default standard that is recommended for the initialization settings of forces programmed specifically for Logitech Force Feedback Mouse.

General Terms
Documentation, Design, Human Factors

Keywords
Orientation aids, blind users, force feedback devices, usability

1. INTRODUCTION
Most people do not think twice when reading a map. However, traditional maps require the ability of its users to see and thus are unusable for the blind and visually impaired. Currently the blind and visually impaired rely on audio and tactile aids for map reading.

A tactile aid by definition is an aid perceptible to the sense of touch. Blind people wishing to have a paper-based map may utilize a Braille printer to print a map from a computer. The printer will convert the lines of the map into a set of raised dots that can then be traced with a finger to feel the lines of the tactile map. This method of map production limits the ability to apply additional information to maps, such as directional and geographic labels. For this reason, researchers have been working on the production of a more digitized map environment that utilizes the computer to express map information.

Proceedings of the 5th Winona Computer Science Undergraduate Research Seminar, April 20–21, 2005, Winona, MN, US.

Research has led to a new trend that is emerging for audio tactile aids. A remarkable example of this trend is the Talking Tactile Atlas series [5], currently composed of 44 maps embossed on tactile sheets. Each sheet contains raised lines and textures that express map information. Once the embossed sheets are placed on the T3, a peripheral desktop computer device with a touch screen, user-computer interaction is possible. When various symbols, icons, and regions on the tactile surface are pressed, audio information on what the user is feeling comes from a connected computer (see Figure 1).

Figure 1: Talking Tactile Tablet [5]

Unfortunately these two map reading methods, like almost all currently employed maps for the blind and visually impaired, have one major limitation. These maps for the blind are not readily available due to the fact that they are made by hand and thus accrue higher costs and less accessibility. There is no database like Mapquest that blind users can access. Rather they must request the production of each individual map for the method they employ as the need arises. Additionally, there is no way for a user to enlarge or shrink a map without producing a completely new one. For these reasons and numerous others, it is obvious that a new computer based navigation aid for tactile maps for the blind is currently needed [4].

Our proposed solution to this problem is to create an interactive digital 3D environment that simulates currently used static maps. We intend to determine how to apply a layer of forces on top of a digital map, thus making it interactive when it is explored using a force feedback device. The aim of the interaction is to create an experience of being in a computer-generated environment that feels realistic [7] and thus provide the user the ability to fully understand the environment or map.

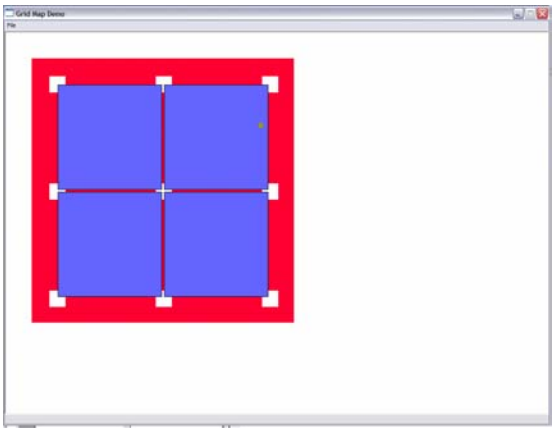


Figure 2: GUI with underlying forces visible

1.1 The 3D Digital Environment

The currently employed environment we developed is composed of a grid system of squares equally spaced far enough apart to provide room for the traversal of a mouse pointer (see Figure 2). These squares are then programmed to be semi-impenetrable by a force feedback device (meaning the user moving the mouse will feel as if they are passing through mud when moving the mouse pointer across a square). At each intersection there are small boxes that are programmed to feel like attraction points, thus alerting the user of their position at such intersections. In order to ensure that the user remains within the designated area, a boundary box surrounds the grid. The mouse pointer can only cross over this boundary box once substantial force is applied. Finally, there can be a grid system of raised lines placed on a layer below the previously mentioned objects. These lines are equally spaced to provide a sense of distance traveled as the pointer crosses a set of lines.

Users navigating this environment with a force feedback device will thus feel restricted to move along the line spaces (grooves) created between the squares and around the grid of squares. In terms of a map environment, these grooves represent the traversable roads upon which a user travels. Further information specific to the programming tools used to create this environment will be explored later in this paper.

1.2 The Force Feedback Device

Many computer users today are familiar with modern gaming equipment and the new controllers (i.e., force joysticks and rumble pads) used to provide force sensations that mimic the actions occurring in the game. The forces give the impression of recoil, vibration, impact, and many other sensations. These function under the direction of a computer, which instructs the force feedback device to transfer forces to a user's hand or fingers. The availability of various tactile effects depends on the device hardware in combination with software that directs what kinds of effects the device should play. [7] For example, the Logitech Force Feedback Mouse (see figure 3) used in this study can be programmed to emit the following forces: periodic, texture, enclosure, ellipse, spring, grid, constant, ramp, damper, friction, and inertia. The way in which the forces are initiated in the software determines the strength and intensity in which device will emit the various forces. Different force initialization settings affect the amount of information available to the user.

Figure 3: Logitech Wingman Force Feedback Mouse

Proper settings are vital for the correct portrayal of information. For example, forces that are too weak may not be noticeable to the user, whereas strong forces may distract or overwhelm the user from correctly identifying the environment. This paper tests the various strength settings by applying a variety of force settings to environments implemented as the controls. While users each have personal preferences, the usability tests of this paper aim to determine the average force settings that provide a user with the greatest understanding upon exploration of the spatial environment with the Logitech WingMan Force Feedback Mouse.

This paper describes the implementation of the graphical user interface and the capabilities it provides. It then goes on to outline the usability tests implemented and concludes with the analysis, results, and future work.

2. PROGRAMMING TOOLS

The Immersion Corporation develops tactile feedback technology that its licensing partners, including Logitech, can integrate into their products to gain market adoption [3]. Specific to this research, the Logitech Force Feedback Mouse utilizes Immersion TouchSense technology to create a software development kit (SDK) for program developers. The SDK library includes the Immersion Foundation Classes (IFC) that define the effects available for C++ programming, previously listed in section 1.2. Our software implements numerous methods of the IFC to program the mouse to recognize and perform the necessary forces.

3. PRE-TESTING ANALYSIS

Pretests were performed to refine the testing session and to ensure user understanding, comfort, and productivity. Two subjects were tested and asked to provide feedback throughout the pre-testing session.

In comparison with traditional mice, the additional features of the Logitech WingMan Force Feedback Mouse often require a period of orientation for users to become accustomed to using the device. Users traditionally navigate by either sight or sound and thus have not been trained to interpret Graphical User Interface (GUI) navigation with only the sense of touch. The orientation session is especially necessary for blind and visually impaired users who might have little to no experience using a mouse in any situation. Thus both users during pre-testing completed an orientation session followed by nine tests and a set of reaction survey questions.

During the orientation period in which the users were provided time to become comfortable with the mouse and environment, both users said that a thorough explanation of the environment beforehand would increase their understanding of what they felt. They wanted to know the different forces they would encounter, the general layout of the environment, and the possible errors that could occur and how to correct them. We utilized these recommendations to create a set of orientation tasks and instructions, which we present later in this paper.

In order to narrow the field of possible force settings to test, the two users were asked to explore the digital environment and to provide feedback on a set of force settings varying from 100% to 25%. These sessions were timed in order to determine a reasonable number of tests to administer during a testing session. Both users responded by saying that forces must be set to at least 50% in order to receive enough feedback from the mouse to navigate the environment. Based on this recommendation and the average time required to explore the environment, we decided to perform a series of nine tests divided into three force settings of 100%, 75%, and 50%.

Finally, the two users were asked to provide feedback on the initialization settings of one additional force. This force is a grid system of 'raised lines' spaced at equal intervals both vertically and horizontally. The grid is intended to help the user determine distance traveled. Provided with a set of force settings, the two users determined that values between 20% and 30% were the easiest to feel without being obtrusive.

4. METHODS

4.1 ORIENTATION SESSION

The orientation session utilized in this study consists of a series of instructions the user is required to complete. These instructions are a composite of all the skills that could possibly be utilized during the official usability testing sessions. They ensure knowledge of the fundamental forces users may encounter as well as the fundamental actions users may perform. The session has a maximum time constraint of ten minutes, but users will be allowed to proceed at their own pace. Throughout the orientation session there is an experienced user present observing progress and available for questions. The user is informed that they can ask whatever questions they need to understand the device and the environment at anytime during the orientation, but will not be allowed to ask questions during the official testing. Users are alerted to the possible problems they may encounter and provided with advice on what to do should they encounter these errors. Users will also be informed of what they are to expect during the official testing session. The forces throughout the orientation session are programmed at 100% to help beginners become familiar and comfortable using the mouse.

Orientation Session Instructions for a 1x1 Grid:

1) Note the groove between the outside boundary box and the impenetrable square.

2) Trace the square felt through the force feedback device.

3) Note that the mouse can move across the square, yet only when sufficient force is applied.

4) Note occasional shaking of the mouse. This is normal and occurs occasionally because of the placement of competing forces close together.

5) Note that the mouse can move outside of the boundary box. Try to return to the box and re-outline it.

Orientation Session Instructions for a 2x2 Grid:

1) Trace the perimeter of the square felt through the force feedback device.

2) Note the boundary box and the force it takes to jump over it.

3) Note the attraction points at each corner or intersection to alert the user to a possibility for a direction change. These attraction points feel like a divot in the surface of the environment.

4.2 USABILITY TESTS: PART I

These tests determine the force initialization settings programmed into the force feedback device that provide the user with the highest percentage of proper environment identification and the highest comfort level. Proper environment identification is based on the user's ability to define the grid dimensions of a set of squares, all equal in size. Comfort levels are determined by user ratings upon completion of each test. During testing the length of time and screenshots are recorded from the start time and position until users successfully determine the grid dimensions. The screen shots are made possible with the Supreme Spy computer monitoring software [8].

The subjects tested comprised a group of six college students with at least minimal computer experience. This software is designed to be very simple to use and subjects were able to gain additional skills needed during the orientation session. While the software is intended for visually impaired users, sighted users frequently have more experience using mice. Additionally, previous studies have shown success utilizing both blind and sighted users to fully test software. [2] Thus the group consisted of and equal number of both visually impaired and sighted students in order to fully examine the force settings felt with the mouse.

The subjects were each asked to complete a series of nine tests. The order of the tests was statistically randomized for each user to ensure that users could not predict the dimension settings. Also, the monitor was not visible by the user during testing to mimic the feeling of being blind. Upon completion of each test, subjects were told whether or not they correctly identified the dimensions and were asked to rate their reaction to the preset force setting. After all testing was completed, subjects were asked to complete a survey to gauge their reaction to the use of the Logitech Force Feedback Mouse in the spatial environment.

Test	Dimensions	Strength Percentage
1	3 x 1	50%
2	3 x 1	75%
3	3 x 1	100%
4	2 x 2	50%
5	2 x 2	75%
6	2 x 2	100%
7	2 x 3	50%
8	2 x 3	75%
9	2 x 3	100%

Table 1: Individual Test Information

4.2.1 Tests1-9

The user tests assess user ability to use the force feedback mouse to determine the grid dimensions in the 3D digital environment (see Section 1.1). The forces were programmed at 50%, 75%, or 100% strength. Nothing below 50% strength is utilized because pretests revealed that such a setting is too low for navigation. Table 1 reveals the setting values of the nine tests.

4.2.2 Grid System Testing

Upon completion of the nine tests, users were asked to explore one additional environment and provide feedback of their opinion of the additional force programmed into the mouse. This force is an underlying grid system of 'raised lines' intended to help alert the user to distance traveled. These grid lines are placed at intervals of 50 pixels and are programmed to two strengths: 20% and 30%, which were determined during pre-testing (see Section 3). The underlying grid was applied to a 2x2 grid of squares initialized to 75% strength.

5. ANALYSIS AND RESULTS

5.1 Test Results

Our analysis was comprised of both quantified and qualified tests relating to the ability of each subject to successfully identify the environment, the time needed to complete the test, user ratings of the various force settings, and a set of user reaction survey questions.

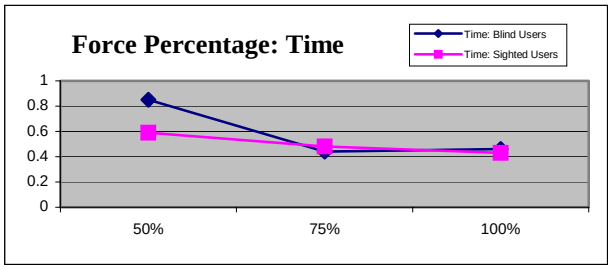


Figure 4: Force Percentage Results - Time

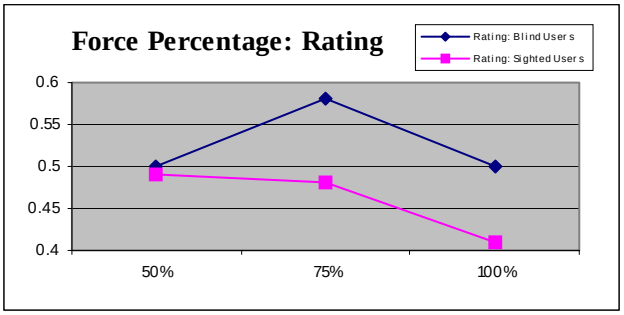


Figure 5: Force Percentage Results – Rating

The tests were analyzed in two ways. First, the results were grouped together according to the strength of the forces to determine which setting (100%, 75%, or 50%) the users rate highest.

According to the calculated results, both blind and sighted users on average required the longest amount of time to respond when the forces were set at 50%. When the forces were set at 75% and 100%, users on average took the same amount of time (see Figures 4 and 5).

The user ratings were based on a scale of 0.1 to 0.7, 0.1 meaning they did not like the settings and 0.7 meaning they really liked the settings. The results for the user ratings are close, however blind users foremost preferred the forces at 75%, then 50%, and finally at 100%. Sighted users liked both 50% and 75% about the same, and again ranked 100% as the least favorite. As the majority of the users verbally expressed, they preferred the settings at 100% when they first began, but as they became comfortable with the mouse and software environment, they actually preferred lighter settings. After testing, users felt they were at an intermediate level and thus preferred the forces at 75%. The majority then expressed that with a little more time and practice they would prefer the settings closer to 50%. They reasoned that forces which were too strong impeded their ability to move out of the attractions boxes.

The results were then grouped according to the number of squares programmed for each test. This grouping is important when analyzing the time taken for users to respond since a larger number of squares implies a larger surface area to navigate.

According to the calculated results for the size of the grid, the number of squares correlates to the amount of time users, both blind and sighted, needed to determine the correct dimensions. Users, again both blind and sighted rated each grid size at approximately the same value since all three grid sizes were implemented at all three force percentage settings (see Figures 6 and 7).

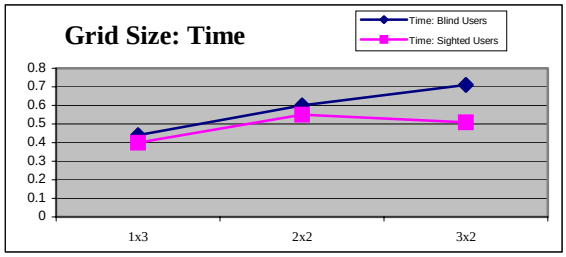


Figure 6: Grid Size Results – Time

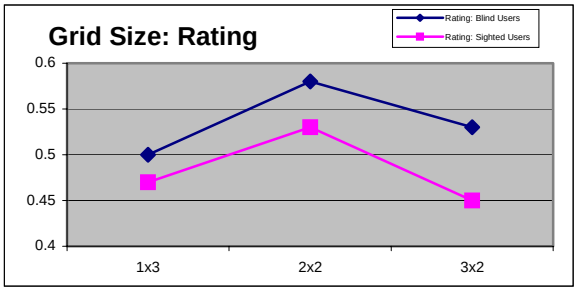


Figure 7: Grid Size Results – Rating

User reactions to the addition of the grid system of 'raised lines', or tick marks, equally spaced to indicate distance traveled was very informative. All users agreed that they could feel the tick marks, but struggled to count how many they crossed with the mouse. Additionally, users noted that they were less able to distinguish between the tick marks and attraction points since the two forces felt very similar.

For this reason, we proposed a new solution to the users. Since our studies show that attraction points are easy to count, we proposed that instead of the tick marks, we would place attraction points at equal intervals along the paths. Then in order to alert the user of an intersection, we would implement a new force such as a small vibration in the mouse. All users agreed that they would prefer the use of attraction points over tick marks.

6. USABILITY TESTS: PART II

Upon completion of Part I testing and analysis, we decided that the results would improve if we adjusted our testing strategy. Instead of randomizing the force settings for the nine tests, we chose to organize the tests in groups based on their initialization percentages. This second round of testing employed the same orientation session, however during testing the nine tests were given in order, beginning with 100% and working down to 50% strength. Since there are three grid dimensions at each percentage, the order of these three tests was randomized. Users were alerted when the force setting changed.

Since the results for both the blind and sighted correlate very closely, we decided that further testing on sighted users is sufficient to determine the proper settings for user by blind users. The users tested include two college students and two middle age adults. Again, all users had at least minimal computer experience.

6.1 Analysis and Results

As users became acquainted with the mouse and their sense of touch increased, they preferred to use weaker force settings. Our new testing approach implements this pattern. Three of the four testers rated the 50% setting as their preferred, then 75%, and lastly 100%, which they stated was too strong to manipulate the mouse with (see Figure 9). User 3 preferred the stronger forces during testing, however clearly stated that as they became more comfortable with the mouse, they too would prefer a force setting lighter than 100%.

No conclusions can be determined concerning the correlation between grid size and time since the results show no pattern (see Figure 10).

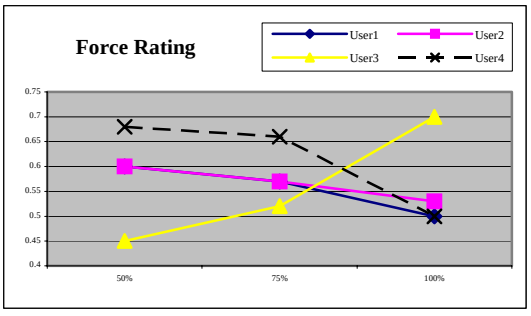


Figure 9: Force Percentage Results – Rating

As in the first usability tests, users informed us that they did not find the tick marks very useful and felt they made the environment more confusing to navigate. However, all four users again agreed that the implementation of attraction points at equal intervals along the paths would help them determine distance traveled. Additionally they agreed that a new force should be used at the intersections to alert the user.

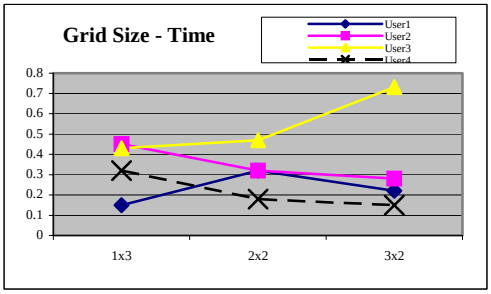


Figure 10: Grid Size Results – Time

7. USER SURVEY RESULTS

Based on the reactions of the users, this Logitech Force Feedback Mouse and the accompanying software have a very small learning curve (see Figure 11). Testing on average took only 30 minutes. Within this time period, every user went from no experience with a force feedback mouse to an extremely high comfort level. In discussions with each user upon completion of the tests, all users confirmed the tested fact that they preferred the force settings at 75%, but with more practice, would probably not need the forces to remain as strong. When the blind users were asked if they would use the Logitech Force Feedback Mouse again, all three confirmed that they would if the software they were using would be enhanced by the use of the mouse.

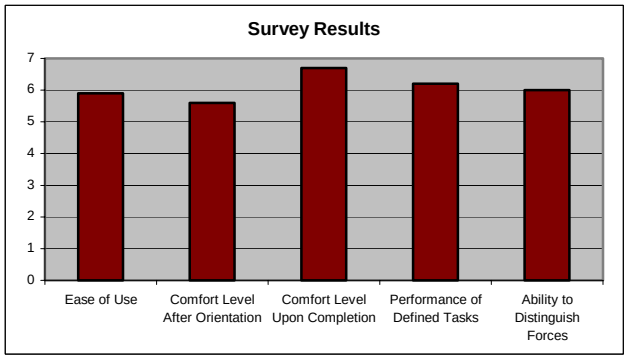


Figure 11: Survey Results

8. CONCLUSION

Usability testing affirmed that the Logitech Force Feedback Mouse can successfully provide a blind user with information about a spatial environment. As expected, users each had personal preferences for the initialization values of the forces, yet as a whole, these preferences have a strong correlation.

Beginning users prefer to start with the forces set at 100% and work towards lighter force settings as they become more comfortable with the device and software. We recommend that future developers do not initialize force levels to values less than 50% of their full strength in order to provide enough feedback for users to navigate the environment. Further, we recommend that program developers implement the ability for users to decrease the strength levels of the forces as they become more comfortable with the force feedback mouse.

In order to implement a guide for the distance traveled along paths, we recommend that developers utilize attraction points that are equally spaced since such forces are easy for users to feel and count. Developers must then implement a new force such as a vibration at intersections to alert the user.

Survey results confirm a small learning curve for users and a strong ability to distinguish between the programmed forces. Those tested with visual impairments confirmed that they would use the Logitech Force Feedback Mouse if they were provided with software that implements this force feedback device.

9. FUTURE WORK

The next stage in this project is to implement vector forces that can pull and guide a user between two defined paths. Further usability testing should determine initialization values of the vector forces that provide the user with the greatest amount of guidance.

Long-term goals are to create a program that can layer these forces on top of a digital map in order to make the map interactive with a force feedback device.

10. ACKNOWLEDGMENTS

Thanks to Robert Manduchi at the University of Santa Cruz, California for his imagination to get this project underway during the summer of 2004. To all my friends, family and classmates that willingly took the time to be subjects in my studies. Finally to my roommates who graciously put up with my conversion of our living room into a testing center.

11. REFERENCES

[1] Gardner, J. and Bulatov, V., Smart Figures, SVG, And Accessible Web Graphics, In Center On Disabilities Technology And Persons With Disabilities Conference. 2001.

[2] Hwang, F., Keates, S., Langdon, P., and Clarkson J. Mouse Movements of Motion-Impaired Users: A Submovement Analysis. In Assets 2004: The Sixth International ACM SIGACCESS Conference on Computers and Accessibility. October 18-20, 2004. 102-109.

[3] Immersion Corporation. Technology: TouchSense Devices. http://www.immersion.com/developer/technology/devices/index.php 2003. 22 January 2005

[4] Kurze, M., Polxfub, J. and Krauss, M. TGuide: A Guidance System for Tactile Image Exploration, In Proceedings of the Third International ACM Conference on Assistive Technologies, Marina del Rey, California, United States, 1998, 85 – 91.

[5] Royal National College for the Blind. http://www.talktab.org/ 26 March 2005

[7] Sallnas, E., Rassmus-Grohn, K. and Sjostrom, C. Supporting presence in collaborative environments by haptic force feedback. In ACM Transactions on Computer-Human Interaction (TOCHI). December 2000. 461 – 476

[8] Supreme Spy Software. Freeware demo download. http://www.downloadjunction.com/product/software/65977/ 2 March 2005