UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

A MULTIMEDIA APPROACH TO COLLABORATIVE ENGINEERING DESIGN ON THE WEB

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of requirements for the

degree of

MASTER OF SCIENCE

(Mechanical Engineering)

By

KIAN HUAT TAN
Norman, Oklahoma
1999

A MULTIMEDIA APPROACH TO COLLABORATIVE ENGINEERING DESIGN EDUCATION USING THE WEB

A THESIS APPROVED FOR THE
SCHOOL OF AEROSPACE AND MECHANICAL ENGINEERING

BY

_____
Dr. Kurt Gramoll

_____
Dr. Harold Stafford

_____
Dr. Feng Chyuan Lai

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my deepest appreciation to my advisor Dr. Kurt Gramoll for his guidance, suggestions and encouragement throughout the duration of my graduate studies. Also, I am very grateful to him for providing me with an assistantship and allowing me to further develop on his Truss Solver. I would also like to extend my gratitude to the faculty of Aerospace and Mechanical Engineering Department for imparting their invaluable knowledge to me in my pursuit of quality education. I wish to express my sincere gratitude to Dr. Harold L. Stalford and Dr. Feng C. Lai for serving on my advisory committee.

I am grateful to my colleagues in the Engineering Media Lab and friends for helping me. My sincere thanks to Carl Nelson for his help in developing the Drawing Board. A word of thanks to Julian Yew for providing me with his VRML packing line model as an example in this thesis. Also, thanks to Qiu Li for his advice on LINGO debugging.

Most of all, a very special thank you to my family especially my parents Chik-Seang and How-Keng for giving me this wonderful opportunity to study abroad. In addition, special acknowledgement to my girlfriend Ean-Nee for her love and continuous support when research and thesis writing seems to go on indefinitely.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Over the past decade, there has been an elevation in computer software's role in the educational process. Software developers have started to take advantage of the most significant advances in computer technology most notably, internet-based networks. As resources for education dwindle, many now look towards information technologies to improve productivity by reducing the time, money and teaching resources needed to help students learn. Undeniably, the Internet will be a vital part of education in the near future. Also, the extensive use of computers and the dramatic increase of laptops at many universities, such as the University of Oklahoma, further encourage the need for content development of courses on the Internet.

This thesis explores the use of the World Wide Web (also known as the Internet) as a supplement to conventional teaching methods in engineering education. Since good engineering programs are generally too complex to use, too costly and not widely available to students, a Web-based approach is more favored. In addition, creating engineering applications on the Web that are interactive and collaborative further induces the excitement and creativity in engineering students. Collaboration is defined as the act of working together to accomplish a task.

The applications developed here are meant to be used along with Web pages to promote online collaboration and have students coming back to the Web pages regularly. The ability to collaborate online enables engineering students to cultivate teamwork when they cannot physically meet. The objective of this thesis is to research the use of Director 7 and VRML to collaborate online. There are four tools in this research that illustrate the

ability to collaborate on the Web. The multi-user programs developed here also incorporate text chat ability for online discussion. The first tool is the Multiuser Painting Board that allows students to draw simple sketches for another person over the Web. Next, the Multiuser Drawing Board was developed to provide online collaborative drawing with greater accuracy and also more options. The collaboration drawing is real time and both sides are able to draw at the same time. The Multiuser Drawing Board is used in the Rigid Body Mechanics Online Course at the University of Oklahoma, Norman.

The Multiuser Layout Planner was developed to cater for the needs to collaboratively layout a plant. This tool uses VRML to provide 3-D visualization of the layout design. Besides the Multiuser Layout Planner, a concurrent truss design called Multiuser Truss Solver was also made. The Truss Solver allows the students to concurrently design trusses using the Web platform.

The final part of this thesis concerns the development of Virtual Lathe using VRML and JavaScript. The chapter describes the use of VRML for viewing and JavaScript for interaction. In addition, the chapter also discusses the creation of the VRML models to illustrate manufacturing processes.

CHAPTER I

INTRODUCTION

Until recently, higher education in United States has been viewed as almost trouble-free and the best in the world. Today, however, several problems loom. Many state budgets for higher education are falling, and some are experiencing large reductions even as student populations are increasing and becoming more diverse (McArthur and Lewis, 1998). Policy makers and educators believed that information technology could help higher education reach many of its goals. Their vision is organized around the pervasive use of interactive and high-bandwidth communication networks. The model championed is the Internet, World Wide Web and it's generic tools (e.g. browsers), which are used to run applications. The Internet is an international network of computers connecting together universities, companies, research laboratories, homes and government offices. The World Wide Web is the graphical interface to the Internet. One can access the Internet using a wide variety of applications, the most common of which is the Web browsers.

1.1 THE WORLD WIDE WEB AS A COMMUNICATION MEDIUM

In 1992, Tim Berners-Lee, a Swiss scientist, created the World Wide Web (WWW) at the European Laboratory for Particle Physics (Lemay, 1996). WWW is the overall system consisting of gopherspace, ftp sites, telnet utilities etc. One distinguished feature of the Web is its flexibility. Not only can the sites on the Web post text but also graphics, 3-D models, sound and video. Internet has mimicked almost all kinds of other communication devices and audio-video genre devices. Omitting interactivity, the Internet is like an instructional TV, leaving out the text, it can imitate two way video and

telephone; if multi-casting is not available, the Internet might look like a one-on-one intelligent tutoring system. Not only, can it individually emulate a television, telephone, fax machine, radio, newspaper or textbook but also all at the same time. Text and graphics document can be viewed dynamically using links instead of just forward backward direction like a PowerPoint presentation. Since most universities and colleges provide access to the Internet, most students won't have trouble using the Internet as a mean of communication. For instance, emails have become a daily routine for most higher education students and also high skill workers.

1.2 INTERNET BASED LEARNING – PROS AND CONS

No doubt that there are many technologies other than Internet that can and do play important roles in education. One such technology is educational information delivered on CD-ROM. Still, the Internet is favored because it incorporates functionality of other available tools and it convenient. One major advantage of Internet based learning and collaboration is the cost factor. In the time of dwindling education budget size, the Internet can serve as a cost-effective way of conveying information to a broad base recipient list. The Online courses can offset the lack of faculty members to teach the class. It is also more economical to fund an online course than to create a new faculty position.

Although it is cost effective to use Internet as a tool in education, it is still a concern that sufficient knowledge and suitable equipment is required before one can gain access to the information on the Internet. As the prices of computer and Web-based hardware and software fall from year to year, a significant amount of students will be able to afford it. In the University of Oklahoma, all entering freshmen in the College of

Engineering are now required to have a laptop. With the installation of the wireless network on campus, students can connect to the Internet anytime. Therefore, resources can be shifted to develop more Internet-based course materials. On the other hand, most major universities have well equipped computer labs that can accommodate most of their student's need. Therefore, the idea of Internet-based tools developed in this research for students is quite practical.

In addition, the Internet can meet challenges post by the deliverance of information to geographically isolated areas as well as lifelong learners who must retool their skills. Telephone and television lines can be used to access data from the Web between home users and Internet Service Provides enabling students to obtain information 24 hours a day and 7 days a week. Furthermore, updating the information on the server can be done any time without having to invest funds to reprint newer editions.

Of course, any new technology requires the users to acquire certain amount of new skills before he/she can benefit from the technology. However, with the recent introduction of many high level authoring, animation, modeling and rendering programs for both the Macintosh and Windows based computers significant multimedia courseware programs can be developed by professors and students (Gramoll, 1994). User-friendly and intuitive development software enables the developers to learn within a shorter amount of time.

In reality, a professor cannot always be in the office for the students. Therefore, a concept called 'virtual office hours' is introduced. Virtual office hour is the concept where the professors or teachers can communicate with students through emails and Web-based collaborative tools without having to meet in person. The goal of this thesis is to establish

an additional communication channel through the use of collaboration tools using the Web.

Since it is proposed that the students and professor could communicate through the Web, there exist a need for software that allows it to be done. The emerging Web standards and also easily accessible software on the Web allows this to happen. For example, in the main part of the thesis, Shockwave, a proprietary plug-in is now distributed free in the latest versions of Internet browsers. Although Web authoring software requires money but players or plug-in are freely available to the users. This means that an initial investment on the authoring software allows students to use it for free.

1.3 COLLABORATIVE ENGINEERING DESIGN AND INTERNET

Engineering design courses are an important aspect of engineering education. The theories, principles and materials the students learn in class will eventually be used to do design work in their senior year. In short, the purpose of many designs is to generate a scientific and logical scheme to satisfy the requirements of producing the specified product (Tan, 1997). Teaching an engineering design course is taxing in terms of time and effort. As more and more engineering design materials are emerging on the Web as basic HTML files, there is a need for Web-based simulations that would make design interactive. The use of simulation or design software on the computer would further encourage creativity in an engineering student's mind.

The increasing importance of collaborative work has lead to an important realization – the ability to work interdependently, as part of a team, is a higher level of achievement than working independently. Business managers have discovered that

empowered teams collaborate to achieve impressive results with minimal supervision (Sherman, 1996). Similarly, students can learn and achieve more as a collaborative team than as a group of individuals (Buchal, 1999). According to Michael Schrage (1995), "collaboration is the process of shared creation: two or more individuals with complementary skills interacting to create a shared understanding that none previously possessed or could have come to on their own. Collaboration creates a shared meaning about a process, product or an event".

Continuous improvement in the speed and bandwidth network allows idea of online collaboration to become a reality. A high-bandwidth network connecting digital hypermedia simulation will enable both engineering design operations and information to be exchanged concurrently among users at several remote sites. Research and development of this proposed distributed multimedia environments, should enable students to learn new ways of working with each other using technology, hypermedia and multimedia tools in order to collaborate more effectively.

As the importance of collaboration is described above, the same kind of collaboration can be implemented in the cyber-world. The purpose of this research is to demonstrate one of the ways to do collaboration through the Internet using Director Multiuser Xtra. A program called Multiuser Server runs on the server and client programs (Shockwave movies) can interact with each other by sending data through the server. The ability to send and received data allows interaction. At present, there are few developers on the Web that do online design collaboration using Director Multiuser Xtra. The following chapters will discuss the development of the Multiuser Drawing Board, Multiuser Plant Layout Planner, Multiuser Truss Solver and VRML as an aid to

visualization aid to collaborative learning. In the subsequent chapter, it will be shown how the Multiuser Drawing Board can assist discussion on the Web with drawing capability. Later, the Multiuser Plant Layout Planner and the Multiuser Truss Solver will demonstrate the method that allows engineering students to solve engineering problems through the Web. This research will test the feasibility of implementing online collaboration in a college environment.

There are many benefits of using the Internet as a collaboration platform. One of the advantages is that the user gets to use it for free because the plug-in are free. Since it is Web-based, it is freely accessible or downloadable. This solves the difficulties of mass distribution. The main reason why the Internet platform is favored over a CD-based platform is because that collaboration requires a medium to transmit the data. In this research the medium is through the network of computers and servers. There are more options in Director Multiuser Xtra that will not explored such as a database to handle the connected users. In addition, for the purpose of education, students could also post information like corrections for other classmates. This promotes the willingness of students to work together to accomplish projects and homework.

Any new technological approach doesn't come without a price. There is a steep initial learning curve before one can develop a multi-user application. Also, the users must have access to Internet-enabled computers and the developer either must have access to a web-server to host the Shockwave movie.

Even though Shockwave is good for developing online simulation, it is inadequate when it comes to 3-D graphics visualization. Three-dimensional graphics in Shockwave is possible with the use of a third party plug-in (Xtra) but it is tedious due to the

complexity of showing 3-D perspective. An easier alternative is Virtual Reality Modeling Language (VRML), which is the chosen format in this research. VRML can be written using any text editor. Also, it is the default 3-D standard for the Web and third party players are widely available for download. There are a few commercial VRML authoring software programs that simplify the creation of VRML models. Besides VRML, there is also a 3-D Java format. This format isn't used in this research because of its complex programming procedure and also the lack of authoring software that would simplify the creation of 3-D models.

This research is unique because there currently is no 3-D or 2-D Visual collaboration tools available on the Internet for engineering education. Web developers are more interested in using Director 7 Multiuser Server to develop multi-user games and not for engineering purposes. Also, most Computer-Aided-Design (CAD) software developers are moving towards concurrent design utilizing Intranet or Internet. The research done here shows the practicality of doing it through the Web using Director 7. In the Multiuser Layout Planner, VRML, an established visualization standard is used. Most of the CAD software can export 3-D models into the VRML format and thus the Layout Planner is able to tie both software together to allow design and collaboration at the same time.

CHAPTER II

LITERATURE REVIEW

2.1 INTRODUCTION

The skills and abilities of designers cannot be accumulated without proper guidance and training (Tan, 1997). As such, Gibson (1995) recommends that new information and reference materials for engineering design have to be constantly provided for to understand new ideas and concepts. The Internet serves as a good source of updated information and materials. However, most people think the Internet as consisting of only HTML and JavaScript. Few people realized that the Internet, as an educational tool, could do much more with the addition of newer technology such as VRML, Shockwave, Java, JavaScript, PERL and ASP. All of these technologies have an ever-stronger impact on Internet-based collaboration for education and design.

2.2 EDUCATIONAL SOFTWARE

Many educators have noted the need for development of comprehensive education software. The tools developed for this research are meant to supplement educational software or class teachings and not meant to replace it. The National Science Foundation (NSF) has actively urged the educators to explore the opportunity to use educational software and advanced technologies to enable the students to simulate, visualize, model and experiment with real world problems (Zia and Mulder, 1996). As an example, in chapter 6, the Multimedia Layout Planner captures the essence of simulation, visualization and design all in one application.

2.3 EDUCATION THROUGH THE WEB

Over the last decade, the creation of the Internet has paved a way to another new aspect of communicating information. To help promote and facilitate the concept of distributed computing via the Internet, Tim-Berners-Lee created the World Wide Web in 1992 (Tom, 1998).

Information Technology could reduce teaching costs or increase the speed with which learners acquire knowledge (McArthur, 1998). To create a Web-based classroom requires a number of skills, a fair amount of time and a reasonable level of experiences (McComack, 1998). According to McCormack, the use of Web-based classroom can meet increased participation needs, increased varieties of learning tools, need for increased flexibility, increased expectation and increased competition needs.

2.4 COLLABORATION AND DESIGN ON THE WEB

The ability to collaborate design and work whether in education or industry is an important aspect. For creators and users of CAD data, the Internet is the ideal mechanism for facilitating collaborative design and real-time communication. By its nature, project design is a collaborative process. From concept to final construction or manufacturing, a varying range of disciplines are involved in bringing ideas to reality. The Web, in conjunction with AutoCAD Internet technologies, brings powerful tools for the immediate managing, viewing, accessing, and publishing of CAD data (Configured Systems, 1996). With the integration of Internet access functionality into Release 14, AutoCAD has become "Web-enabled". At the same time, Autodesk's WHIP technology has "CAD-enabled" the Web. The union of these two technologies delivers new powers to the CAD designer. Bids for new projects can be searched, solicited, and submitted,

opening up new opportunities for companies that may not have had the resources to previously participate in such processes. Questions and proposals are e-mailed, and the time saved can be put to better use in refining the bid. A product portfolio of the firm's successful projects can become a compelling Web site. This expands the number of audiences to their services resulting in greater market exposure and potentially higher revenues. In comparison with the collaborative tools developed in this research, it is found that the above system lacked the interactivity for the users to manipulate the end output.

Geraldine Gay from Cornell University has done a study on engineering design students (Gay, 1993). The study aimed to modify and develop communication tools and resources of collaborative design for use in a distributed hypermedia environment. Also, it was intended to analyze communication activities of students in a collaborative design environment in order to promote the teaching of collaboration skills needed for concurrent engineering design. In their study of design, he found that engineering students often work without a context. As a consequence, they may fail to understand the social, historical, economic, and other implications associated with the design and manufacture of products. Also, students often design in an environment that is insular and not imitative of the team development characteristic of engineering in industry.

In addition, Web-based conferencing sites are gaining popularity on the Internet. Among them are WebChat (http://www.irsociety.com/Webchat.html) and WWW ChatServer (htttp://www.bprc.mpss.ohio-state.edu/Chat/). Collaboration through chatting with text has been around for a few years already. For instance, O'Reilly has come out with WebBoard, a tool for fostering communication among people with common interest

on the Web (Peck, 1997). WebBoard's ability to store information by topic, combined with its fast search engine, makes it a good choice for offsite groups collaborating on the same project.

According to Ralph Buchal at the ASEE 1999 Annual Conference, synchronous communication tools are still weak in their ability to record and archive the results of collaborative activity (Buchal, 1999). He showed the use of NetMeeting together with Paint program as a collaboration tool. The combination is good but in terms of flexibility, it is limited. For example, current asynchronous communication tools are primarily text-based. These tools can maintain a record of discussion and collaborative activities, but they are weak in support for multimedia. Thus, this research aimed to fulfill the needs for the development of collaborative tools that can support multimedia.

Founded in 1997, Virtual Ink Corporation is an emerging leader in the development of business collaboration technologies that enable organizations to capture, manage, and share information, improving the productivity and creativity of any business meeting (Virtual Ink, 1999). Their product, Mimio, a completely portable Digital Meeting Assistant that turns any existing whiteboard or flat surface into a powerful medium for capturing, managing and sharing thoughts, ideas, and instruction directly into PC applications. One of the disadvantages is the cost of Mimio that is about USD 499 per set. Also, the information sharing of Mimio is not real time compared to the Multiuser Drawing Board that is developed in this research. It requires the user to manually export pen strokes to other users.

Open Virtual Factory developed by Engineering Animation Inc. (EAI) to be comprehensive manufacturing system. Open Virtual Factory allows colleagues across the

enterprise to design, analyze, visualize, simulate and communicate the production system design. Using an integrated process database (IPD) as a common framework, their solutions and existing software connect and work together (EAI, 1999). EAI claimed that Open Virtual Factory can improve productivity, reduce overall production planning time, save millions of dollars in equipment costs and resources, increase output, and dramatically reduce the time to market. This is the direction, manufacturing in the 21rst Century is heading to as competition gets more intensive, more design and concepts are going digital. The Open Virtual Factory is not designed to incorporate collaboration. The issue of collaboration is addressed in this thesis to include collaboration besides 3-D visualization and layout tools.

Manufacturing collaboration has always been the major focus of engineering. Micheal Bailey at the University of California, San Diego has undertaken the necessary research and development to ensure that it is viable for engineers and scientists to tele-manufacture over long distances (Bailey, 1992). The collaboration of rapid prototypes is useful for companies that have international branches and need engineers from different plants to design a product.

As the Web continues to make collaboration between product development partners easier, the next logical step may be to actually turn over more development work to outside providers. The idea behind ITsquare.com is to collaborate online with software development contractors for outsourcing (PC WEEK, Aug. 1999).

2.5 INTERNET-BASED 3-D GRAPHICS TO FACILITATE COLLABORATION

VRML is an acronym for the Virtual Reality Modeling Language. VRML is chosen in this research to facilitate collaboration because of its 3-D graphics capability.

VRML is still a relatively new concept that evolved due to the need of a common language to specify 3D scene descriptions on the Internet (Marudur, 1998). Data visualization is an important factor in the use of multimedia to improve on the standard learning method of lectures and textbooks (Higuchi, 1993).

The reason development in of Java and VRML has made production and distribution of highly interactive programs. These resources can be developed with inexpensive tools and are distributed through Web technology. Steven Peterson from Vanderbilt University has developed a machine kinetics solver using Java and VRML (Peterson, 1997). VRML has been used at Vanderbilt to help students in ME280 Advanced Dynamics of Mechanical Systems visualize the locations of coordinate systems to define the kinematics analysis of closed loop spatial mechanisms.

At the University of Delaware, VRML model of a person called Jack is use to assist the rapid prototyping of assistive devices (Wieckowski, 1996). The VRML model helps to customize the design of wheel chairs for a person depending on the age.

Manufacturers have dreamed for 30 years of managing their entire operations from design to costing, production and factory planning (Forbes, 1999). Chrysler's new Jeep Grand Cherokee, due to enter production in 2001, will be completely designed in the digital realm. Steadily declining computing costs now let affordable software produce realistic 3-D images on even cheaper PCs.

At Iowa State University, a project was taken to investigate the role of visualization and virtual reality could play in the decision making process when manufacturers are faced with investing in new technology (Wong, 1998). This virtual

factory provides a visual; 3-D space to explore the effect of various products mixes, inspection schedules, the worker experience on productivity.

Recently, designers use VRML, and interactive 3D technology to illustrate complex tasks such as fixing a paper jam on a copier (Emedia, 1999). The animated VRML model was created using Platinum's Cosmo Worlds.

Other VRML projects in progress include the Geo VRML, which maps the whole world as VRML. That project in turn is working with NASA airport project (NetProfessional, 1998).

In this research, VRML is used for collaboration combined with Director Shockwave. The idea of using the combination Shockwave and VRML for collaboration is new. Shockwave handles most calculations and the VRML display the needed 3-D visualization in a collaborative environment. This research is unique because it ties VRML to a multimedia authoring software, Director, for collaboration. Hence, it allows implementation of multimedia into the collaborative environment.

CHAPTER III

MULTI-USER APPLICATION USING DIRECTOR 7

3.1 INTRODUCTION

Since most given tasks or project are usually constrained by time, the ability to do collaboration allows the optimal use of resources. With collaboration in a team, a better project or product can be produced with minimal time possible. Therefore, there exist a need for the development of collaborative tools that allows concurrent work to be done. This is also true for engineering education. In this chapter, Director 7 with Multiuser Xtra is explored as one of the ways to enable collaboration on the Web. The Multiuser Xtra allows Shockwave to communicate data between computers. Given the ability to convey data between computers is the first step towards collaboration.

3.2 DIRECTOR BASICS

Director is an authoring tool for multimedia production for both CD and Web-based delivery (Macromedia, 1998). The current version, 7.0, was released early 1999. Director 7 comes with an extra called Multiuser Server. Multiuser Server is an application that runs on the server and regulates the flow of data between Shockwave movies. The following sections will explain the use of this application to create multi-user application thus enabling collaboration.

Since Director 7 is a multimedia authoring software, it incorporates images, animation, text and sound media. In this thesis, only Shockwave (Web-based) output is discussed. It requires a Shockwave plug-in to play the application. It can be downloaded for free at www.macromedia.com.

When an image is imported into Director, it will become a cast member. The image can then be placed on the score and stage. The score is a chart that shows which member appears on the stage during a specific timeline. The stage shows the particular member position during a particular frame. The member in the score is called a sprite. A Director production is called a movie. Frame is defined as an instance in time in Director. A channel is a numbered position in the Score.

Director can be used to produce multimedia programs and simulations for both CD and Web delivery. For Web delivery, Director saves the program as a Shockwave file. Basically, Shockwave is the technology that enables users to play Director movies in the Web browser. The second type of output is the Projector output. This type of output is intended for CD-based delivery, where the file size is not a major concern. The last output format that Director 7 allows is Java output. The Java output is suitable to be used on the Web and also CD. Also, Java output allows it to be played on any platform regardless Unix or Windows. However in this research, development of collaborative tools are solely for Web-based purposes thereby only the Shockwave output is used.

In order to create a multi-user application using Director 7, a programming language have to be used. Lingo is the programming language inside Director. In general, Lingo can be used to program four scripts in Director that is the behavior script, frame script, movie script and parents script. Behavioral script is a script that controls a sprite. A Frame script is a script that controls a frame in the score. A Movie script controls the entire movie. A parent script is used for creating child instance. This will be further illustrated in chapter 6.

3.3 DIRECTOR 7 MULTIUSER SERVER XTRA

The Director Multiuser Server is installed and run on a NT server and it allows the users to collaborate over a network. Essentially Director Shockwave movies can communicate with each other through the Multiuser Xtra. Xtra is a plug-in for Director that gives the authoring software more capabilities. Multiuser Xtra allows Shockwave running on different computer to communicate. Each user runs a copy of the Director movie that is stored on the Web server. The movie allows interaction with other movies by sending data to the server. When the Multiuser Server Xtra receives the message, it will route it to the appropriate user. This is a two-way transmission of data. Each Director movie then has its own the appropriate handler (subroutine) to handle the received message. Not only text can be exchanged but also pictures and sound. The Director Multiuser Server is a server application that controls traffic, groups and access privileges (Macromedia, 1998).

The Multiuser Xtra can establish three different types of connection. The first kind is the Client-to-Director Multiuser Server connection. This kind of connection requires the Multiuser Server application to be running on a host computer (server). Essentially, the server serves as the intermediary between different users connected to the server. The Multiuser application can be kept running 24 hours on the server. As a result, users can log in whenever they want. Also, the server controls which movies are connected, controls the total number of connections and posts information regarding connections and messages in the server's console window (Macromedia, 1999). The configurations are set in the multiuser.cfg file after installation of the Multiuser Server

application on the server. Setting this type of connection requires a host computer that supports the TCP/IP protocol network and a unique IP address on the network.

The second type of connection available in Director Multiuser application is the Peer-to-Peer connection. This connection is a direct connection between the users' computers. However, this requires each user to meet at the same time for collaboration. The users most likely would have to use other means of setting up the collaboration time such as through email, ICQ or IRC. One benefit of this kind of connection is that it avoids sending data through a server and thus decreases the traffic load on the server. Up to 16 other movies can take part in the peer-to-peer connection to the host movie (Macromedia, 1999). Considering that there isn't a server to route the messages, the computer that host the Director movie would act as the server. One deficiency of this connection is that it lacks the server-type functionality such as the groups and database management. In addition, the host movie must be running on a computer with a static IP (Internet Protocol).

The final type of connection is the Client-to-text-based-server. These servers do not run the Director Multiuser Server software. Servers of this type include IRC (Internet Relay Chat) servers, mail servers or HTTP servers. Further configuration and installation instructions for the Multiuser Server software are written in the Macromedia FAQ for Multiuser Xtra (1999). Figure 3.1 summarize the types of connection available for a two-user model.

Generally, using one type of the connections will permit data to be sent back and forth between computers. Consequently, letting collaborative work to be done. The client-to-Director Multiuser Server is preferred over the other two types of connection

because it allows access at anytime. The following section will illustrate the purpose of using Director Multiuser to create collaborative tools.

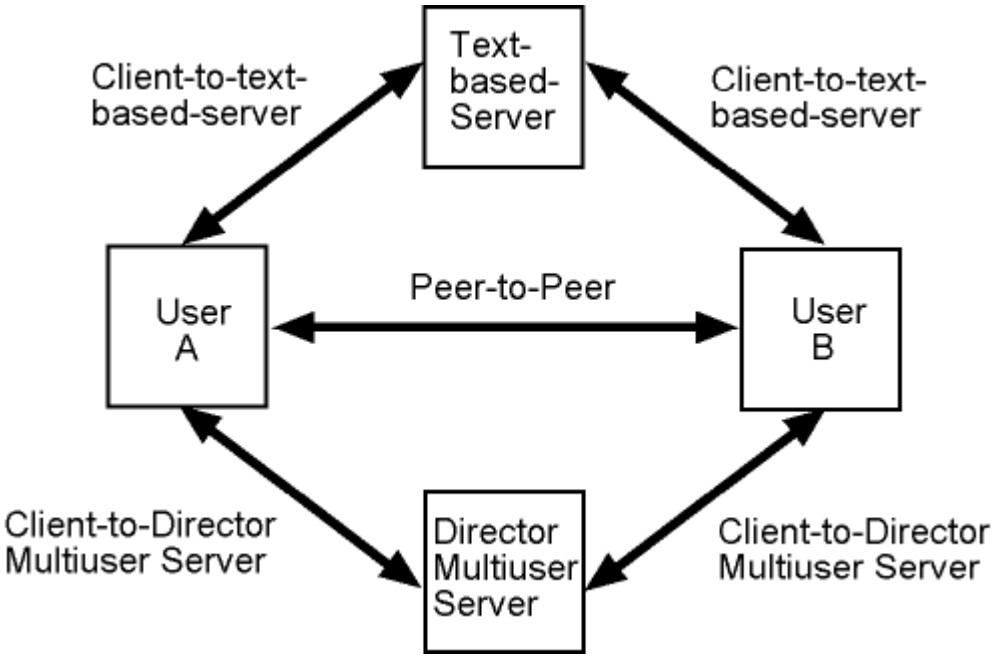


Figure 3.1 Three Different Types of Multiuser Application Connection (two- user model)

3.4 PURPOSE

The task of designing a product is not trivial and can involve a tremendous amount of time, skill, effort and knowledge. The intent of the multi-user application is to create a collaborative and interactive design environment either as a tool for distance learning or a supplement to class-work.

At present, there are few design engineers and Web developers that have used Web-based collaboration technology. Possible reasons include lack of source code and tutorials for Director 7 Multiuser Xtra since it is relatively new. Two good resources on

the Web that provides support for this application currently are the Macromedia Multiuser newsgroup and Macromedia support Web pages. Multiuser applications developed at Engineering Media Lab (EML) can be found at this url, http://www.eml.ou.edu/multiuser/.

The main reason for using Director 7 instead of other alternatives like Java is because of its simplicity. Director 7 has an intuitive programming language that ties back to images and timeline thus making it easier to program. The ease of visual programming allows more time to be devoted to developing the content. Another reason for choosing Director 7 Multiuser is because the player, Shockwave plug-in, is free. Subsequent section will explain the limitations of Director 7 Multiuser Xtra.

3.5 LIMITATIONS

Any kind of technology has strengths and weaknesses, and Director Multiuser application is no exception. Like all other Internet applications, the biggest concern is bandwidth. The more complex the simulation, the bigger the file size; and subsequently, a higher bandwidth is needed. The Multiuser application must minimize the bandwidth need to capture the interest of the user. As the average modem speed for common user increase, there is more flexibility in designing more complex Multiuser application. One way to keep the data traffic low is to send a message to trigger the calculation by the other user's Director movie instead of sending all the calculation results. This method will be illustrated in the Truss Solver application in Chapter 5.

The second limitation to this kind of application is the amount of computer processing unit (CPU) time needed to perform a task. The calculation done by the Director movie on the local computer shouldn't be so complex that it uses all the

memory. However, as the price of computer keeps dropping and computers keep getting more powerful, the power to create more complex application increases. Generally, loops are used sparingly to keep the required resources low on the CPU.

The number of connections allowed on each Multiuser Server is limited to 50. Macromedia does sell a 100, 500 and 1000 users but the basic 50-user version if free with Director 7 package, and is sufficient enough to meet the most collaboration needs. Any higher number of users requires a higher bandwidth in order to avoid congestion to all the connected movies. One way to solve this problem is to install the 50-user version on a number of servers so that more users can access at the same time without having to upgrade the 50-user version to 100-user version. For training and education, the 50-user version is considered sufficient because at a given time usually, not more than 50 users are present.

3.6 STEPS TO ESTABLISH MULTIUSER SERVER CONNECTION

Basically, there are a few steps to follow in order to connect to the Multiuser Server properly. First, the server application has to be installed on the server. This application has to be run whenever the connection is needed. The host movie must be sitting on the server. The second step is to start a client movie that connects to the server or host. Once connected, a movie, will connect to the server, verify that the connection has been made and start the transmission of data back and forth.

3.7 ERROR CODES

One important feature of this Multiuser Xtra is its ability to return error codes when errors are encountered. The getNetErrorString Lingo command is specifically used to get the return error codes. Error codes are always negative integers and a "0" error

code means no errors have been encountered. The following is part of the error code list for the Multiuser server. For the complete list, refer to Macromedia FAQ. The transmission through the server had to be check for errors always to insure proper connection.

Error code	Translated error string
0	No error
-2147216223	Unknown error
-2147216222	Invalid movie ID
-2147216221	Invalid user ID
-2147216220	Invalid password
-2147216219	Incoming data has been lost
-2147216218	Invalid server name
-2147216217	Server or movie is full; no connections available
2147216216	Bad parameter
-2147216214	There is no current connection

Table 3.1 Part of the Error Code List for Multiuser Server

3.8 BASIC MULTIUSER LINGO SYNTAX

The following are some essential commands in a Multiuser script. A complete list of commands can be found in the Director 7 Manual: Lingo Dictionary or the Multiuser FAQ. There are more than 40 commands, functions and properties related to its use and number of commands in the Multiuser Xtra is expected to grow. Note that there are some basic Multiuser Xtra commands in the Library Palette. Also, Macromedia released a correction to some of the behavior scripts in the Library Palette related to Multiuser Xtra.

Command	Function
ConnectToNetServer	Starts a connection to the server and returns an integer indicating the connection's status.
GetNetErrorString	Returns a string explaining the error code that is passed in.
GetNetMessage	Returns the network messages in the Xtra's message queue.
SendNetMessage	Sends message to one or more recipients in the movie on the same server.

Table 3.2 Essential Commands to Create Multiuser Application (Macromedia, 1999)

CHAPTER IV

MULTIUSER APPLICATION: PAINTING AND DRAWING BOARD

4.1 INTRODUCTION

The emerging need of having a collaborative sketching tool that enable digital communication prompt the development of this Painting and Drawing Board. The purpose of the board is to create a collaborative sketching board for online discussion of homework problems. The idea of having a Painting Board is not new. For instance, a computer accessory developed by Virtual Ink of Boston (www.virtual-ink.com) will copy notes written on a conference room whiteboard into the computer (Popular Mechanics, 1999). This is a commercial product that enables managers to teleconference. The sensors on the system track the movement of sleeves placed over the standard dry-ink pen.

The difference between the Painting Board developed using Director Multiuser and that developed by Virtual Ink is the cost. The system created by Virtual Ink is expensive (USD499) because it requires special hardware (sensors) where as the Painting Board using Director Multiuser is fully digital and requires only computers to use and view it. The Multiuser Painting Board is basically free because the required software plug-in is free and Engineering Media Lab does not charge anyone to use it. The "pen" in the Multiuser Painting Board can be the mouse or even trackball. A digital pen is not required to be able to sketch and write on the Painting Board but it would be more convenient. It was designed to be user-friendly and cost effective for students when compared to the system by Virtual Ink that was created for the industry. Moreover, the Virtual Ink system is not real-time and requires the user to export the pen strokes to others for them to view.

4.2 FEATURES AND COMPARISON

The most notable feature of the Multiuser Painting and Drawing Board is it's real-time collaborative digital sketching. It was tested on a Local Area Network with a T-1 line and there appears to be instantaneous with no noticeable lag. When the Painting Board was first developed, it wasn't practical for engineering discussion. Since the Painting Board utilizes a free-form sketching technique, it is not precise. In this case, the Multiuser Drawing Board was developed to enable more sketching control. The Painting Board uses the trail function to draw where as the Drawing Board makes use of the newly introduced vector shapes in Director 7. Besides, the vector shapes allow easier changing of size and color in terms of scripting.

Basically the Painting Board sends packets of X-Y location points through a server when the brush moves and the recipient movie redraws the line with the received data points. The Drawing Board sends packets of vertex points that allow the recreation of the vector shapes. Both boards send packets of data at intervals instead of sending data continuously so that it will reduce the traffic congestion on the Multiuser server. There are a few ways though to check the data flow between the client computer and the server. Two of the ways are the command CheckNetMessages and also time-stamping the data. There are parameters that can be set on the Multiuser Server to allow a determined message size.

Both boards include a chat room to enable text interaction between client computers. The one media missing in both boards is sound media. This tool is specifically design for online engineering education therefore some generic shapes are added. Visual media such as the sketching and text chat is essential for practical

discussion of problems on the Web. Furthermore, the instructor can go online and have a virtual teaching lesson on the Web. Virtual office hours are also possible where the sketch is left on the board so other students can log on and view, for instance, a corrected homework diagram.

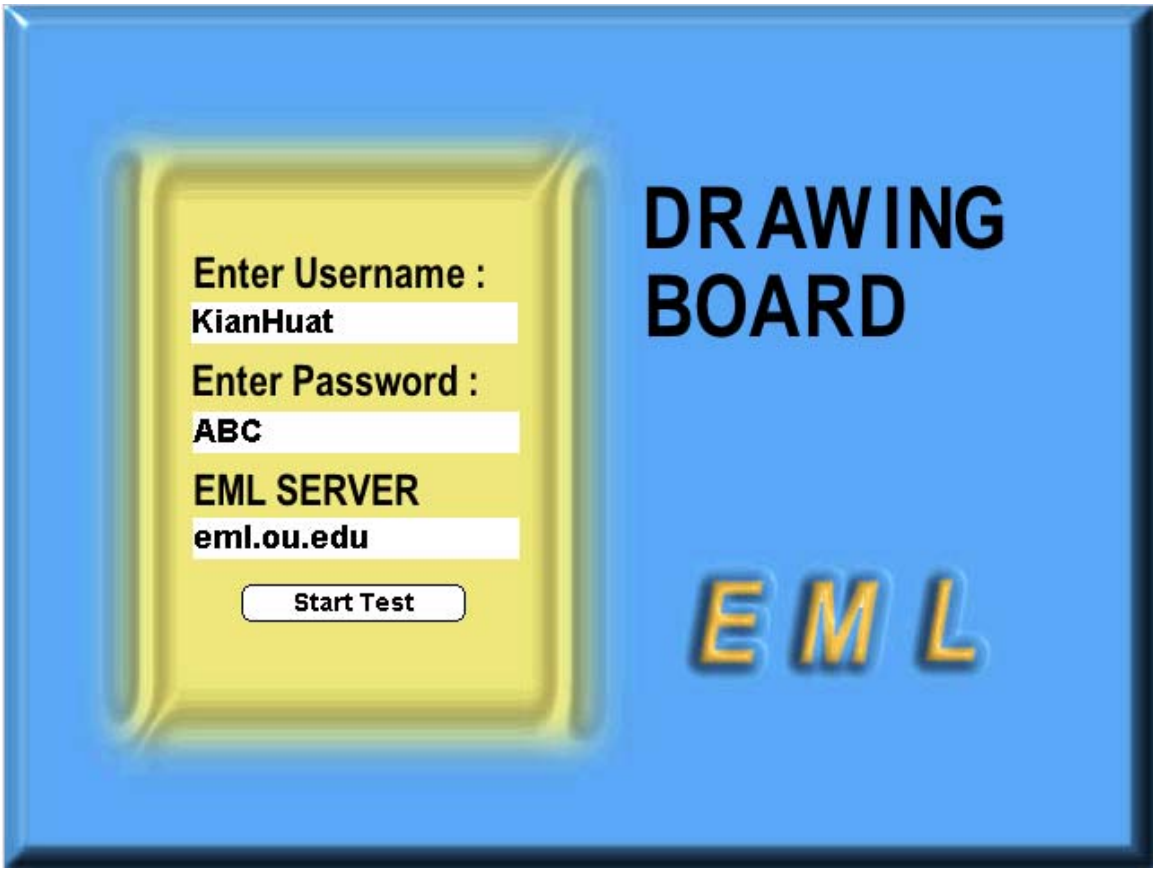


Figure 4.1 Login Page for the Multiuser Painting Board

4.3 MULTIUSER PAINTING BOARD

Figure 4-1 shows the login layout of the Multiuser Painting Board. There are three fields for the input of the username, password and server name. One important note regarding all Multiuser application is that each username that logs on must be unique. For instance, if user ABC logs into Painting Board, subsequent users who use ABC as the

username will be given an error message, “Can’t connect to server, is username correct?" The Multiuser movie can run on other servers that have the Multiuser Server application running and still work fine. When the username has been changed, pressing the “Start Test” button will log the user into the layout of the Painting Board as shown in Figure 4.2.

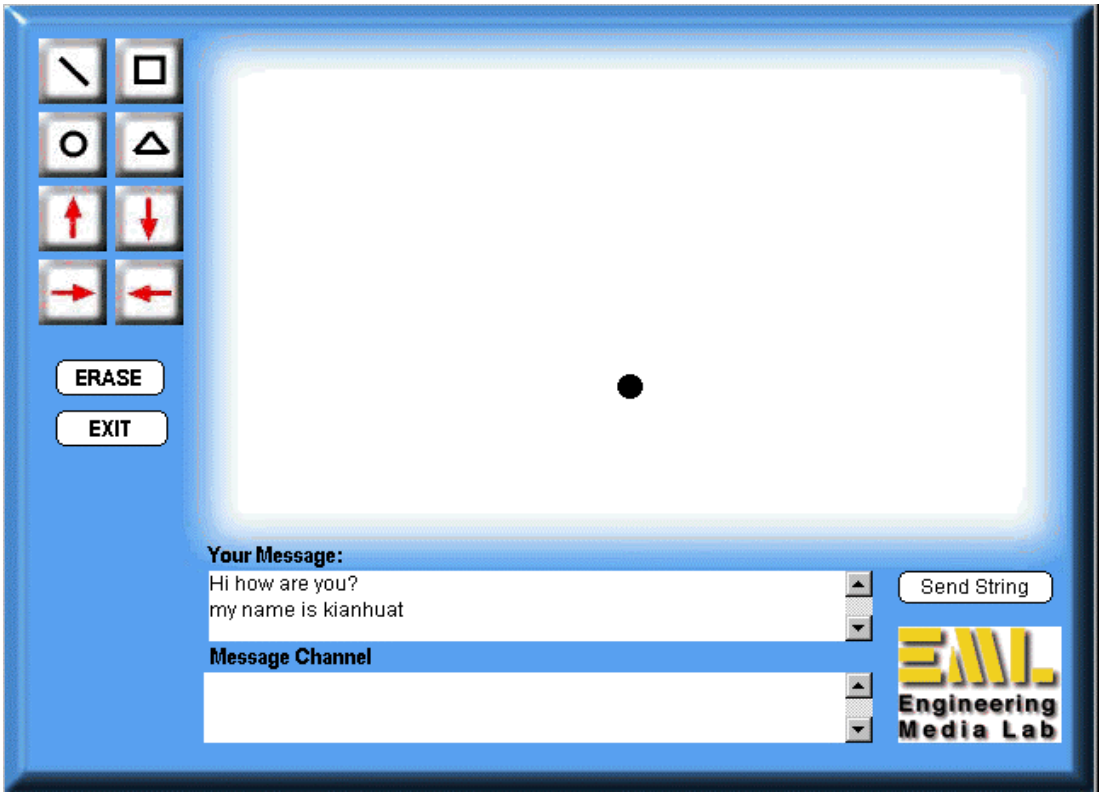


Figure 4.2 Layout of Multiuser Painting Board

The upper left of the Multiuser Painting Board is the set of drawing icons. The drawing icons consist of the line, square, triangle, circle and arrows of 4 directions. The intended use of this Painting Board is to sketch a simple Free-Body diagram for online discussion of problems in Rigid Body Mechanics. The stage, where the actual drawing takes place, is white in color. The brush is shown as a black circle the first time the user

logs in. It is intended to guide the user to the location of the brush head. After the first line is drawn, the brush head is no longer visible. However the mouse cursor within the stage area now acts like a brush head. This is done so that the line drawing is intuitive. The message and channel fields are located below the stage area. The message field provides the user with feedback. Messages that are sent out are printed back out into the channel field. This is to indicate that the message has been successfully sent to the server and to the client. The message that are sent out and received by the message handler, depending on the originating movie, individual movies can decide whether to response to the message. Click the "Send String" button once the message is typed into the message field to send the message out.

The way the brush draws in Multiuser Painting Board is to make a cast member of the brush head appears when the mouse is clicked down. As the mouse is drag from a point to another point, the trail option is turned on to draw the line between both points. Not all points are recorded and send over to the server. It will only take in the points that are significantly different in distance and place it in a list. So when the list reaches four points each time, it will prompt the script to send out the points to the server. The line drawn with this method is not accurate because the receiving computer will run the duplicate brush head following the points using interpolation. If the number of points is huge, it will approximate the original drawing more but as the number of points gets larger, there is a forfeit on the collaboration speed of the program. From Figure 4.3 (a) and (b), it is noticeable that the original drawn line is much smooth and not jagged like the reconstructed line. By comparison, it is found that the distance of the points is not too spaced out in the original source drawing.

This is not the first time the idea of a collaborative drawing or painting board has been implemented. In the American Society for Engineering Education (ASEE) 1999 Annual Conference and Exposition, Ralph Buchal from University of Western Ontario presented a paper on the Internet for team collaboration (Buchal, 1999). He showed the use of NetMeeting together with Paint program as a collaboration tool. The tool is good not only because the users can talk but also chat and sketch with each other. It is professionally done with numerous useful features but in terms of flexibility, it is still limited. Most of the time, instructors need something more than a generic painting tool such as a custom made collaborative tool for a engineering sketching. Also, Microsoft chat server used for NetMeeting is quite congested at times.

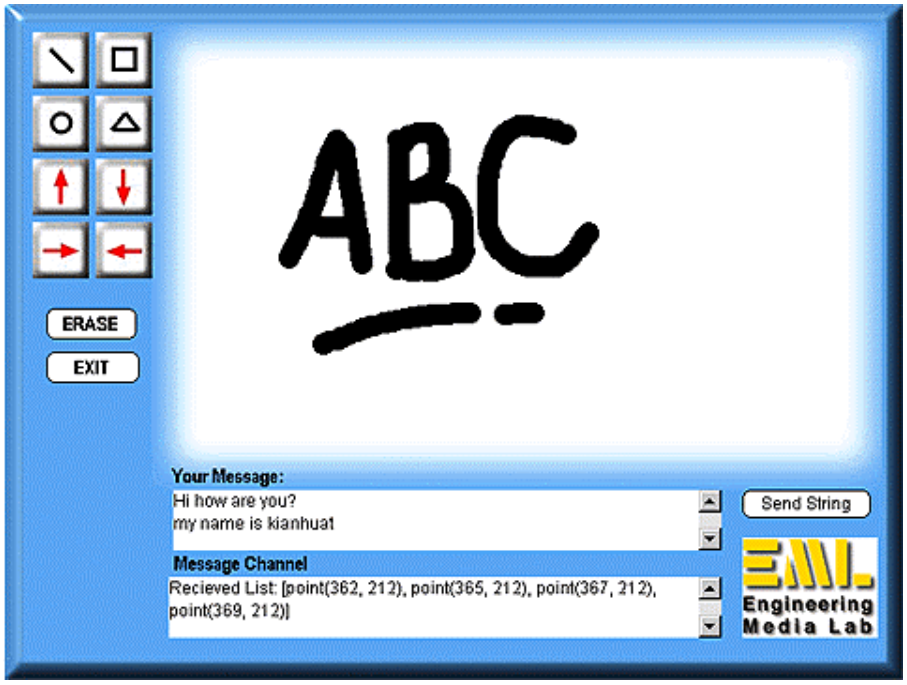


Figure 4.3 (a) The Original Source Drawing

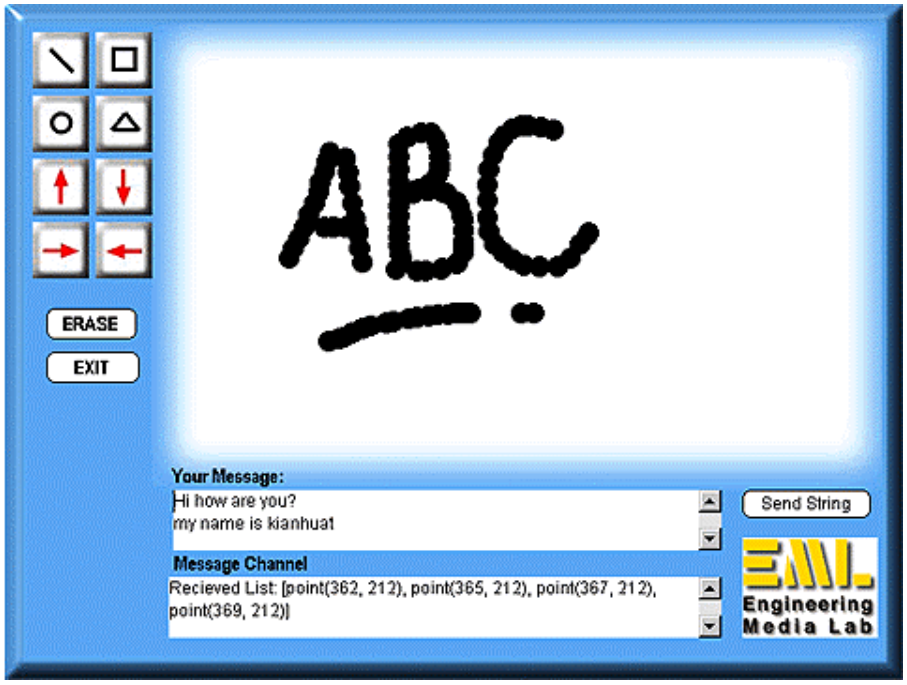


Figure 4.3(b) The Reconstructed Drawing Based on Receiving Points

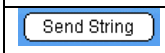
ICONS	DESCRIPTION
	Free-form Line Tool. It is use to draw any line between any number of points.
	Square Shape. Click once to place a square at the desire location.
	Circle Shape. Click once to place a circle at the desire location.
	Triangular Shape. Click once to place a triangular at the desire location.
	Vertical Arrows. Click once to place a vertical arrow at the desire location.
	Horizontal Arrows. Click once to place a horizontal arrow at the desire location.
	Erase Button – Erases all lines and shapes on the stage.
	Exit Button – Exit to the Login Page
	Send Message Button – Send the current message in the message field.

Table 4.1 List of Icons and Descriptions for the Multiuser Painting Board

4.4 MULTIUSER DRAWING BOARD

The Multiuser Drawing board was developed specifically for engineering problem discussion. Although, the Multiuser Painting Board can do simple sketching, it is not precise and not flexible enough to implement in an online class. When it comes to flexibility, Multiuser Drawing board differs from the Multiuser Painting Board. Since each drawing shape consist of a set of vertex points, editing the list of vertex points is made possible. A drawing member can be bent and curved. In a close loop drawing member, there is an option to fill it with colors.

The Multiuser Drawing board can be used for the Online Rigid Body Mechanics class and also the Physic Web site at the Engineering Media Lab. The login page as shown in Figure 4.4, is essentially the same as the Multiuser Painting Board. To ease the creation of a Multiuser application, a set of log in scripts and layout is always used. The server connection scripts are listed in Appendix A. There is always a "connecting server" window to let the user know that the application is connecting to the server.

One very important feature of any drawing-based drawing is the ability to scale to any size and still maintain shape and clarity (Rosenzweig, 1999). In Figure 4.5, the layout consists of the drawing area, tool palette and chat area. Using the mouse as a guide, the drawing shapes can be drawn on the stage. The list of vertex will create new drawing members on the recipients' Shockwave movie. The vector shapes are accurately reconstructed on the receiving computers. The reason for this is because the same number of vertex points is send from the original drawing. The vector shapes use vertex points to regenerate the shape and therefore the drawings are the same for both ends.

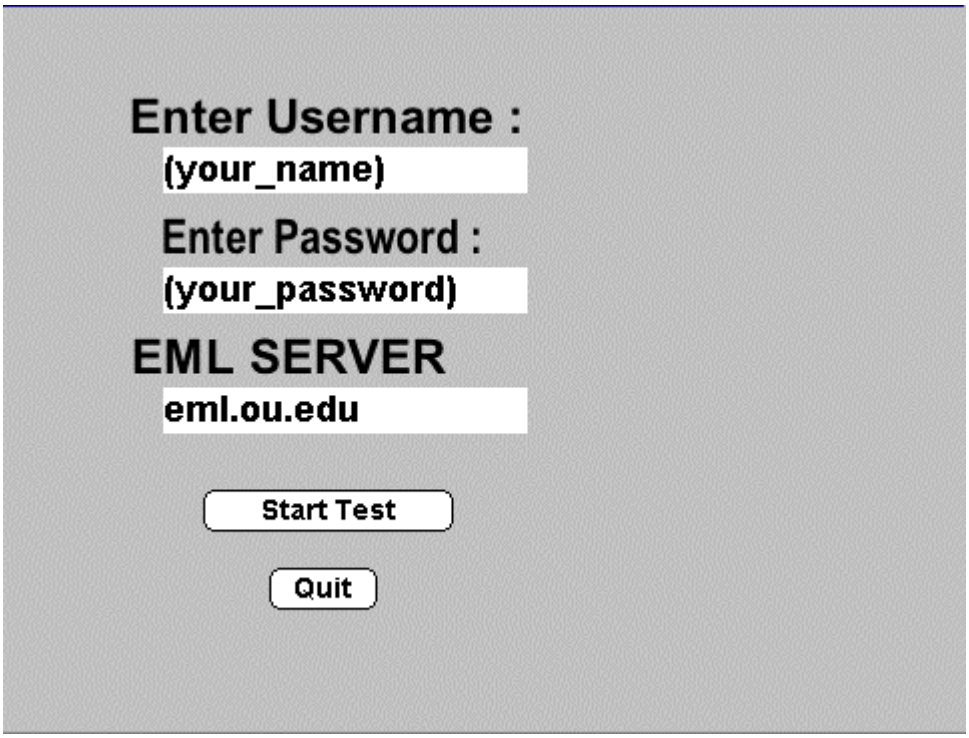


Figure 4.4 Login Page to the Multiuser Drawing Board

Table 4.2 shows the list of icons found in the Multiuser Drawing Board. There are three editing buttons (move, delete and change) in the Drawing Board. Of the three buttons, the most unique is the change button. When the properties of the vector shape is selected, an option menu as shown in Figure 4.6 appears. The text icon allows the user to insert text at any location of the Multiuser Drawing Board by double clicking the mouse. Remaining icons consist of different vector shapes such as line, circle, square, free-form line and line with an arrow. The most unique of these vector shapes is the free-form line. Spline lines can be drawn using the free-from line icon forming complex shapes.

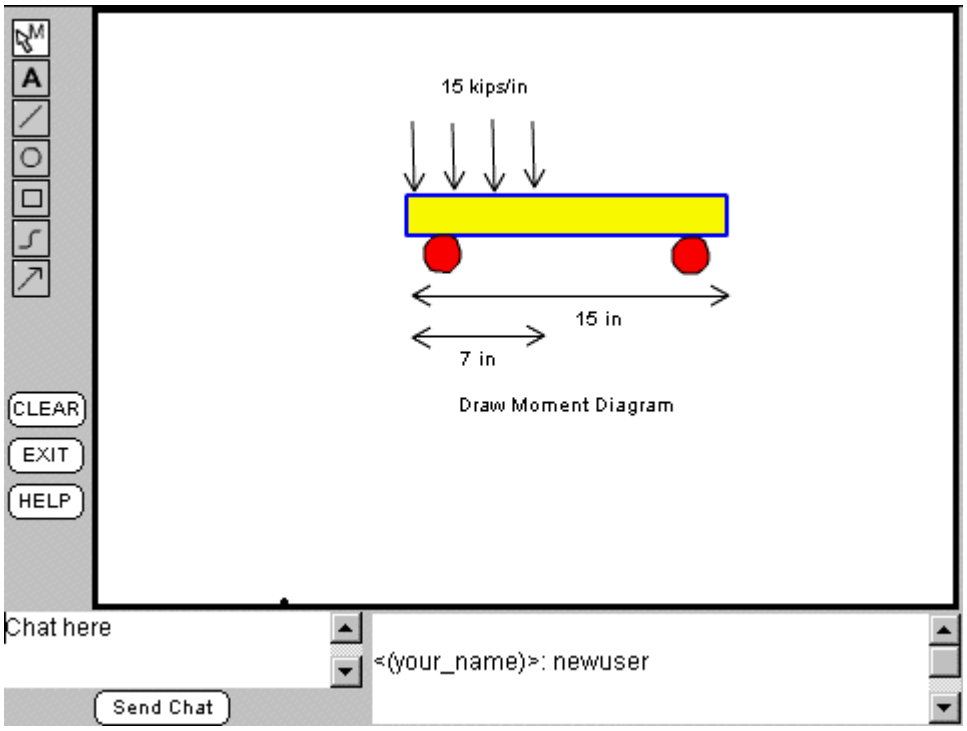


Figure 4.5 Layout of the Multiuser Drawing Board

ICONS	DESCRIPTION
	This icon lets the user choose the move, delete and change icon.
MOVE	This icon allows the mouse pointer to move the drawing shape by its vertex points.
DELETE	This icon allows the mouse pointer to delete the drawing shape on the stage
CHANGE	This icon allows the user to change the line color and fill color.
A	Text icon
	Line icon – allows the user to draw a drawing line between two points.
	Circle icon – to draw a circle
	Square icon – to draw a square
	Freeform line icon – to draw a Spline line
	Arrow line icon – to draw an arrow line

Table 4.2 List of Icon Set for the Multiuser Drawing Board

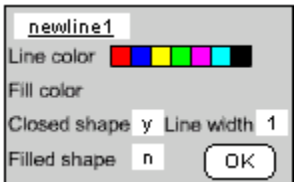


Figure 4.6 Option Menu to Change Line Color and Fill Color

CHAPTER V

MULTIUSER APPLICATION: 2-D TRUSS SOLVER

5.1 INTRODUCTION

The use of multimedia in engineering classes is fast gaining popularity, since it is both educational and fun. This chapter of the research attempts to introduce the use of the Multiuser technology to further enhance the concept of simple trusses. The multi-user program presented here is the Multiuser Truss Solver.

The reason for developing the Multiuser Truss Solver is to heighten the concept of virtual collaboration and interaction between students in the Rigid Body Mechanics class. With this tool, student can grasp engineering concepts better, for instance, learn to identify zero force members by just looking at the structure and applied forces. The students can also double-check their homework problems by constructing similar trusses in the Truss Solver. Furthermore students can meet each other on the Internet while using the Truss Solver so they can collaborate and discuss their homework.

5.2 SIMPLE TRUSSES

The definition of a truss is a structure composed of slender members joined together at their end points (Hibbeler, 1995). In this Multiuser Truss Solver only a simple planar truss will be discussed. Essentially, planar trusses are two 2-D problems because it lies in a single plane. To prevent collapse, the arrangement of the truss must be rigid. Two important assumptions are made in the design of trusses. First, the forces are assumed to be placed at the joints. Secondly, trusses are assumed to be joined by smooth pins. As a result, each truss member acts as a two-force member. If a member is

elongated, the force acting on it is tensile and similarly if, a member shortens, the force is a compressive force.

There are two simple methods of solving static truss problem by hand but there are other methods when a computer is used. The method of joints assumes that the joints are in equilibrium. The method of joints examines each joint as an independent static structure. The summation of all forces acting on the joint must equate to zero. The 2-D equations are,

$$\Sigma F_x = \Sigma F_y = 0$$

Since each joint must be in equilibrium these two equations can be applied to each joint. Generally, the support reactions should be determined first, and then the reaction joints should be solved. The joint sequence for solving can be important for multiple joints.

The method of sections splits the truss up into two or more parts and then applies both force and moment equilibrium equations. The biggest benefit of this method is the ability to solve for member forces in the middle of the truss without solving for all the other members. The basic equations for each section are

$\Sigma F_x = \Sigma F_y = 0$ (two dimensional scalar form)

$\Sigma M_z = 0$ (two dimensional scalar form)

It is important that the student learn to identify zero-force member in a complex truss system. The Multiuser Truss Solver allows instant confirmation of a zero force member thus helping the students to grasp the concept better.

5.3 FEATURES OF MULTIUSER TRUSS SOLVER

The Multiuser Truss Solver is an intelligent simulation that allows students to collaborate interactively when designing their own truss structures. This simulation is designed to supplement the learning of simple trusses. The Truss Solver is intelligent enough not only to solve but also to detect errors. One added aspect to the Truss Solver is the ability to chat and collaborate with other users through the Web. The solver includes sets of truss, force and support. Although the solver here is not generic, it can solve a wide variety of simple truss problems that is either angled at multiples of 45 or 90˚. In addition, the solver incorporates the ability to change the parameters of the Force and Truss.

A good simulation solver should be able to provide feedback to the user. In the Multiuser Truss Solver, errors are checked before the problem is solved and if any errors are encountered they are reported on the message field. The program checks for a minimum of one load and a minimum of two supports. The second error check is to identify any force or support that is not applied at any joints. If there are any force or support that are outside the structure, they will be deleted. Also, it is important to check for a singular matrix. An important property associated with the determinant of a matrix is that if the determinant of a matrix is zero, $|a|=0$, then it is said to be singular (Logan, 1993). The solving engine will be discussed in the next section of this chapter. In addition, the solver checks for large displacement because large displacement is an indication that the structure is not stable and in equilibrium. Lastly, error checking is also done to eliminate the possibility of dividing by 0.

A required feature of a multi-user program is the ability to communicate between Web users. The Multiuser Truss Solver allows the users to communicate either by text chat or by constructing truss structures on the grids. When an action is performed on one of the Multiuser Truss Solver, changes will be reflected immediately on the other computers. The collaboration is made possible using the Multiuser Server and Director Multiuser Xtra. Commands are send from one client movie to another client movie. The recipient of the command initiates a subroutine (called a handler in Director) that executes the command. The commands are automated so that the user does not need to execute any command manually. Below is a partial list of the commands in the Multiuser Truss Solver.

COMMANDS	**DESCRIPTION**
Calculate	Execute the CalcNodeDisp handler
Cleared	Clear all trusses and results
Creating Horizontal Force	Create a horizontal force on the stage with a default 1 N
Creating Pin Support	Place a pin support on the stage

Table 5.1 Partial List of Commands in Multiuser Truss Solver

Since there isn't a need to manually enter the commands into the message field, the commands are only used to force an action. In fact all the actions can be executed by clicking on the buttons instead of typing in the command. Figure 5.2 depicts the flowchart of the program.

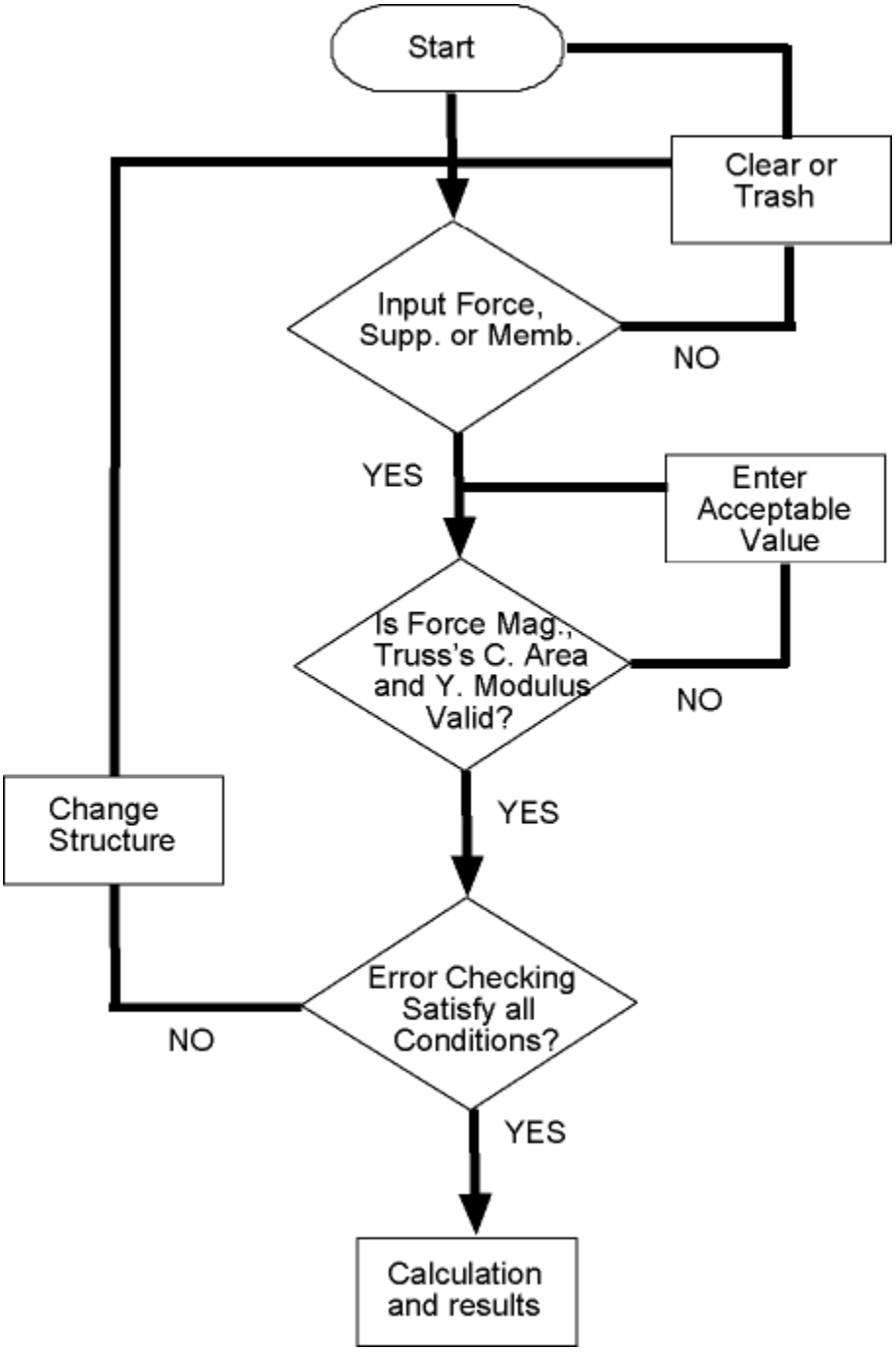


Figure5.1 Flowchart of Multiuser Truss Solver

5.4 NUMERICAL SOLUTION METHOD

The matrix stiffness method is used to calculate the deflection of the trusses in the program. It is essentially a subset of the finite element model method in which small discrete elements are modeled by a displacement function (Gramoll, 1994). The total global stiffness matrix for a structures is,

$$\underline{F}=\underline{K}\,\underline{d}$$

The total stiffness matrix $\underline{K}$ is defined as,

$$\underline{K} = \sum_{e=1}^{N} \underline{k}^{(e)}$$

where N is the total number of elements. The local stiffness matrix $\underline{k}$ for each member in explicit form is given by,

$$\underline{k} = \frac{AE}{L}\begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix}$$

where C is the cos and S is the sin of the angle of each truss member. A is the cross sectional area of the truss, E is the Young's Modulus and L is the length of the truss. The program sets L to be 1 for the horizontal and vertical bars.

Displacement $\underline{d}$ can be defined as,

$$\underline{\hat{d}} = \underline{T}\ \ \underline{d}$$

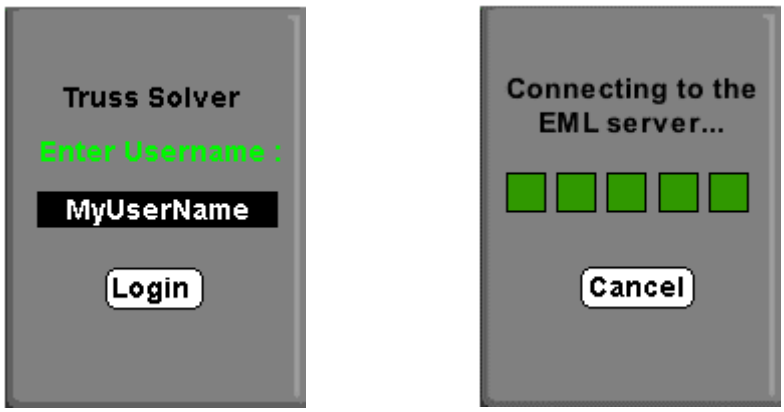
where,

$$\underline{T} = \begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & C & S \\ 0 & -S & C & \end{bmatrix}$$

After the global stiffness matrix had been assembled, the program solves for displacements. A numerical method called Cholesky Decomposition is used here to solve for the displacements. This algorithm is based on the fact that a symmetric matrix can be decomposed (Chapra, 1988). The approach is particularly well suited for matrix analysis of structures because it provides the efficiency of the well-known Gaussian elimination process within a matrix format. The computer algorithm and routines for solving using Cholesky method can be found in the book "A Matrix Analysis of Framed Structures" (Weaver, 1980).

5.5 INTERFACE

Much like any other software, the ease of use is an important aspect in a software interface. Figure 5.2 shows the login and connecting layout. The layout is much simpler than the layout in the Multiuser Painting board. It is simpler because it isn't required for the user to enter the password and server name.



(a) *(b)*

Figure 5.2 Login Frame(a) and Connecting to Server Frame(b)

Since a database of users is not maintained on Engineering Media Lab (EML) server, anybody could access to the application. Therefore, a password field is not

needed. Unlike Multiuser Painting Board, the Truss Solver connects only to the EML server because there is no field to change the Multiuser Server. The server field is the text box that allows the user to specify which Multiuser server to connect to. If the default settings file, multiuser.cfg, is kept the same, there isn't a difference between a Multiuser server located in EML or outside EML. The reason for requiring users of Multiuser Truss Solver to log on only to the Multiuser server in EML is to allow future modifications. Future modifications to the server settings can allow better control of the connected movies.

While connecting to the server, it is important to give the user feedback. Feedback is important so that the user is aware that the program is still running. The progress bar moves from left to right to provide this kind of feedback. The cancel button on Figure 5.2 (b) allows the user to return to the login page.

Figure 5.3 shows the entire layout of the truss solver. Unlike the Multiuser Painting Board, the icon set is on the right side of the stage. Objects are made to snap to the grid to for the ease of the users and also the ease of computing the results. Right below the stage is the color gradient bar to indicate the load on the truss. There is a line of message field at the lower left of the Shockwave movie. The purpose of the message field is to inform the user of any errors and also give instructions. The chat area is in yellow. One notable difference between the chat area in Multiuser Painting Board and in the Truss Solver is that foes not need to press a button to send the message string to the server. After typing the message in the white colored field, pressing Enter will send the message.

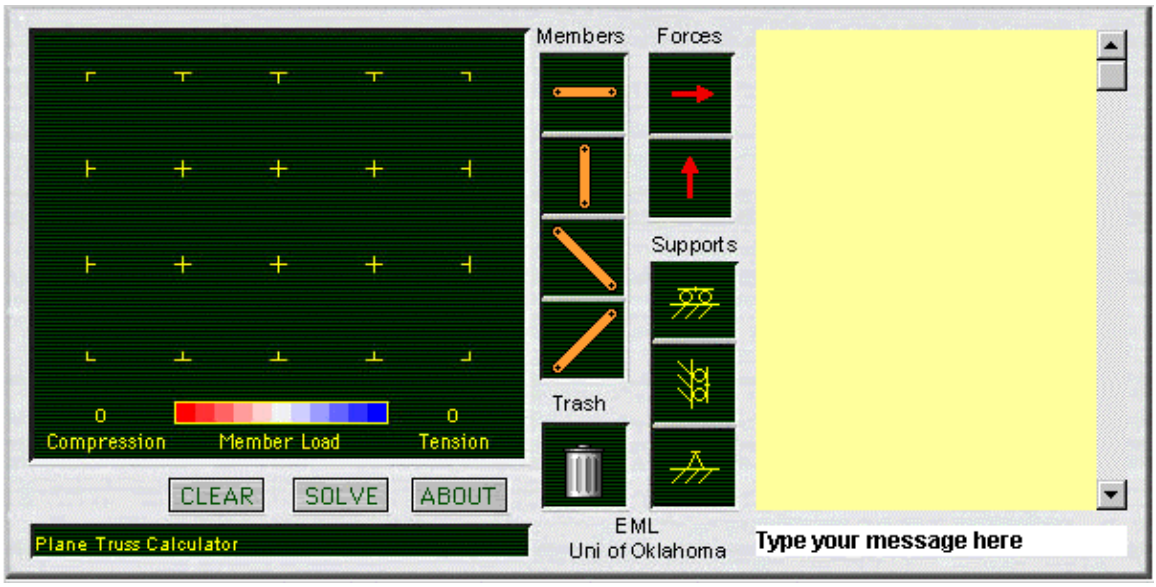


Figure 5.3 Layout of the Multiuser Truss Solver

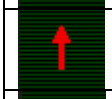
ICON	DESCRIPTION
	Horizontal Truss Member – double click to edit Young's Modulus and Cross Sectional Area
	Vertical Truss Member – double click to edit Young's Modulus and Cross Sectional Area
	Left Diagonal Truss Member – double click to edit Young's Modulus and Cross Sectional Area
	Right Diagonal Truss Member – double click to edit Young's Modulus and Cross Sectional Area
	Trash Bin – to delete objects, drag over the trash bin and drop
	Horizontal Force – double click to edit the magnitude and direction of force
	Vertical Force – double click to edit the magnitude and direction of force
	Horizontal Roller
	Vertical Roller
	Pin Joint
	Clear Button – deletes all object on the stage
	Solve – compute results
	About - a brief explanation of the truss solver

Table 5.2 List of Icons in the Multiuser Truss Solver

5.6 POSSIBLE FUTURE DEVELOPMENT

The Multiuser Truss Solver can be developed further to include 3-D trusses that also incorporate VRML models of the trusses. Three dimensional truss solver would be more practical to use since it is considerably more difficult for students to visualize. Furthermore, a generic truss solver that allows the user to place trusses at any angle could be developed. Other than a truss solver, other rigid body mechanics solvers can be developed, for instance the Beam Analysis tool developed by Gramoll at Georgia Tech (Gramoll, 1994).

CHAPTER VI

MULTIUSER APPLICATION: 2-D FACTORY LAYOUT PLANNER

6.1 INTRODUCTION

Plant layout is a branch of the broader category of facilities layout (Cedarleaf, 1994). Generally, facilities planning can be subdivided into the design of three components of a facility: the structure, layout and handling system. Layout planning is complex, broad, and cuts across specialized disciplines (Thompkins, 1984). For example within the engineering profession, civil, electrical, industrial and mechanical engineers are all involved. This chapter introduces the use of Director Multiuser and multimedia technology to visualize a plant layout. Visualization of a plant in 3-D is important to make better layout decisions. From a wide variety of engineering layouts planning, only the plant layout will be the focus here as shown in Figure 1.2.

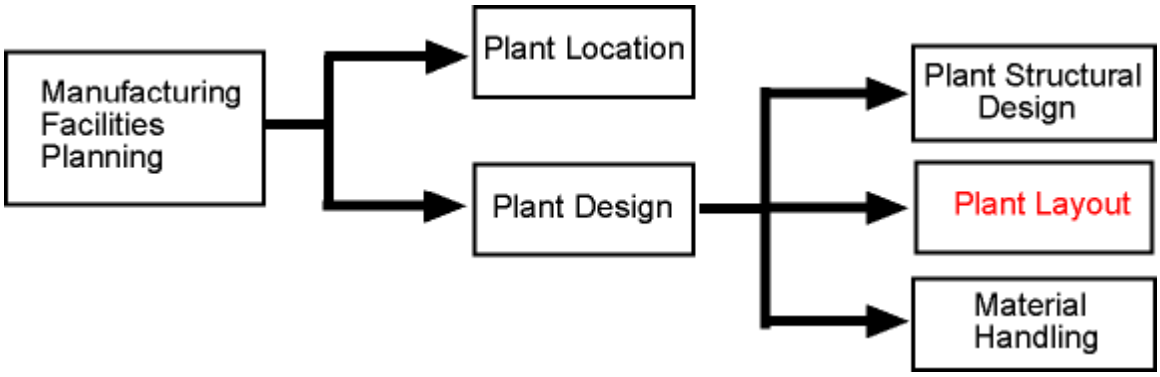


Figure 6.1 Plant Layout, a Sub Part of the Manufacturing Facilities Planning

The Multiuser Plant Layout developed here is a demonstration of the potentials of the Internet for manufacturing layout design. The technology used here is still and under utilized by people in the industry and educational institution.

The role of plant layout plays a major role in the future operations of the plant. It is important so that process flow is smooth, floor area is optimized, and presentation and decision making can be made.

At present, one of the major aids in selling a factory or plant layout to management or a client is the iconic model of the layout. Admittedly, the iconic model is useful but it is also costly and requires time to make. It is proposed that a digital layout will save time because an actual model does not had to be constructed and the planning can be done in digital realm. It is also economical since all the data is kept digital and management can easily view it through the Internet.

A three-dimensional representation of a proposed layout is helpful for selling design ideas to the manager (Francis, 1992). It is now possible to visualize 3-D information by modeling it using the Virtual Reality Modeling Language (VRML). It saves time compared to creating a physical model because VRML can be exported from the Computer-Aided-Design (CAD) software. Even if the VRML model needed to be drawn, there are many third parties VRML authoring software that allow quick and easy VRML model creation. Besides, time saving, 3-D digital representation by VRML saves the cost of raw materials needed to build a physical model. Changes to the model can also be done conveniently in digital format.

As manufacturing gets more complex and digital manufacturing more popular, the layout planner can meet the need for a collaborative tool on the Web. A Web-based collaborative tool is beneficial not only for marketing but also to promote concurrent engineering in new product development and manufacturing. Pushing for digital

manufacturing could save millions of dollars and also introduce a better product faster to market.

6.2 MULTIUSER PLANT LAYOUT PLANNER

The purpose of developing this Multiuser 2-D Plant Layout Planner is to implement the idea of virtual plant design layout for industrial engineers and management. As mentioned in the previous section, there are many notable benefits of doing the initial design in digital form. Figure 6.2 shows the entire layout of the Multiuser Plant Layout Planner.

Figure 6.2 Multiuser Plant Layout Planner.

The size of the browser window is 610 by 515 pixel. It is specifically designed to fit even a 800 by 600 pixel monitor. Generally, the browser window is intended to be unscrollable so that the space is optimized for a standard 15" monitor. This is done using a simple JavaScript to specify the size of the window. The browser window is divided into two frames, the top and the bottom. The top frame contains the Director Shockwave movie and the bottom frame contains the 3-D VRML model. The login and connecting frame is similar to the Multiuser Truss Solver.

6.3 FEATURES AND LIMITATIONS

The Shockwave movie allows the user to design the plant layout with a 2-D representation of the production floor. The layout of the tool bar, stage, chat channel and message field is quite similar to the Truss Solver. On the left, the stage allows the designer to place objects on the stage. Below the stage is the message bar. The message bar will display details and also errors. At the middle top of the Director Multiuser Plant Layout Planner is the location indicator. The location indicator relates the X and Y position of the object with respect to the coordinate system. The coordinate system is shown in Figure 6.3. Different from the standard Cartesian coordinate system, the coordinate system used here follows the coordinate system in Director 7. The coordinate system can be changed though by writing another handler and inverts the Y location.

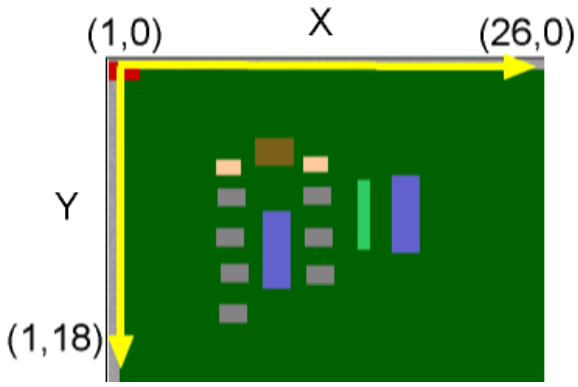
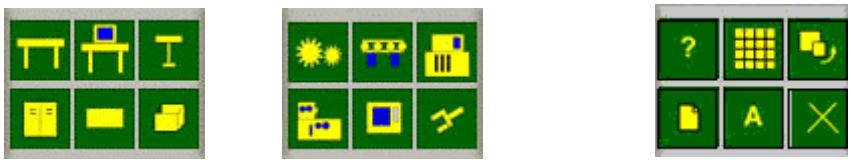


Figure 6.3 Coordinate System of the Multiuser Layout Planner

The palette of icons represents the various models that can be placed in the VRML world. There are two sets of icons as shown in Figure 6.4(a). Figure 6.4(b) shows the command and properties icon. Description of each icon is listed in Table 6.1.



(a) *(b)*

Figure 6.4 Icon Sets in Multiuser Layout Planner

The importance of saving the latest setting of the layout is not overlooked in this planner. The layout file is updated on the server and is automatically saved. A text file is written when the VRML model is updated. The text file contains parameters of the X-Y location. Once the user logs in, the location, number and type of objects are automatically restored regardless when and where. One limitation is that only one version of the settings can be saved. In the future, an additional file handler can be written to allow different file settings to be saved and displayed whenever needed. Also, the main purpose

of saving the settings is to make sure that users who come in later have the coherent arrangement settings as the other users. This is important to prevent any false error messages. For instance, user A created 3 tables before user B enters the planner. If user B starts to create another 6 tables, user A will receive alert messages saying that only 6 tables are allowed at any instance of time because user B adds another 6 more to the existing 3 tables.

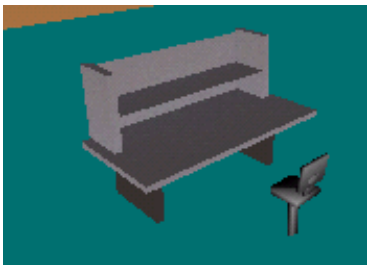
ICON	VRML	DESCRIPTION
	Fig 6.5(a)	Table Icon – Place a Assembly table plus a stool
	Fig 6.5(b)	3-Seater Technician Table – inclusive of Analyzers & Monitors
	Fig 6.5(c)	Single Stool
	Fig 6.5(d)	Storage Cabinet
	Fig 6.5(e)	Partition
	Fig 6.5(f)	Storage Box
	Fig 6.5(g)	Packing Line
	Fig 6.5(h)	Conveyor Belt
	Fig 6.5(i)	Semiconductor Mounting Technology (SMT) Machine
	Fig 6.5(j)	Lathe Machine
	Fig 6.5(k)	Industrial Oven
	Fig 6.5(l)	Robotic Arms
	-	Description – briefly describe what the layout planner can do
	-	Grid Properties – not available yet
	-	Delete Objects – rotation and scaling not available yet
	-	File Upload – not available yet
	-	Credits and Icon to refresh page after time out
	-	Quit to Login Page
	-	Update VRML button – to refresh the VRML models

Table 6.1 Icon Description for the Multiuser Layout Planner

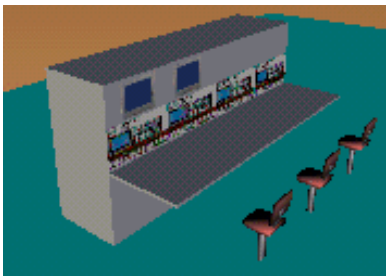
Currently, there are 12 models available in the layout designer. They are as listed in Table 6.1. The VRML models are custom made for this research. In order to change or add VRML models, a user had to have access to the server. The concern with adding new VRML models is foreseen and in the future, an option is allowed so that the user can upload new VRML files to the server. This can be done using JavaScript henceforth will enlarge the database to include more VRML models. A snap shot of all available VRML models are shown in Figure 6.5. More details on VRML modeling are discussed in Chapter 7.

The number of objects is in Director limited so that the file size and memory usage will be minimized which will make the system more responsive. Each child instance has a parent script that creates a child object whenever the icon is clicked. The following section will describe in more detail the programming of this planner. The limits are set at 6 assembly tables, 3 technician tables, 3 stools, 3 cabinets, 6 partitions, 5 boxes, 5 conveyor belts, 5 SMT machine, 5 lathes, 2 ovens, 2 set of robotic arms and 2 packing line.

The floor layout has been kept simple without any walls in order to have easier navigation and visualization. Properties like rotation and scaling have not been developed yet. One important limitation to this layout prototype is that it doesn't allow the user to delete individual objects and change the grid properties. The intent of this thesis is to demonstrate the potentials of using Director Multiuser technology with the Internet to enable collaborative work in the manufacturing field and not as a multipurpose layout tool. The methodology of this application will be discussed in the next section.

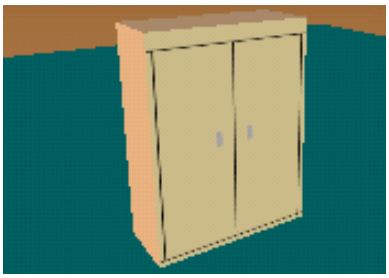


(a) Assembly Table



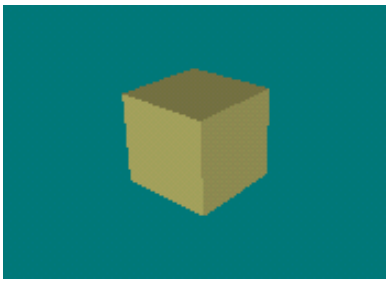
(b) Technician Table

(c) Stool

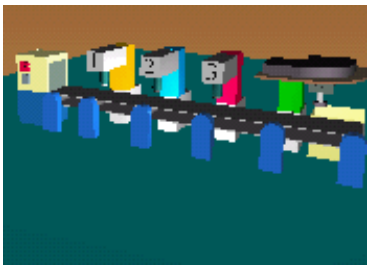


(d) Storage Cabinet

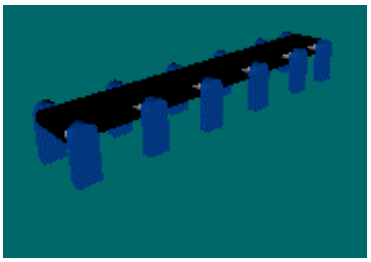
(e) Partition



(f) Packing Box

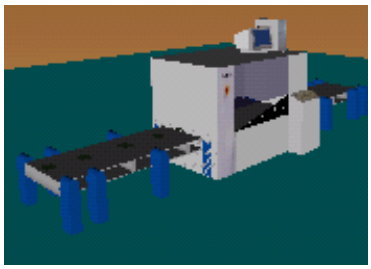


(g) Packing Line

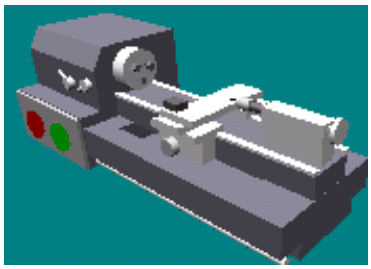


(h) Conveyor Belt

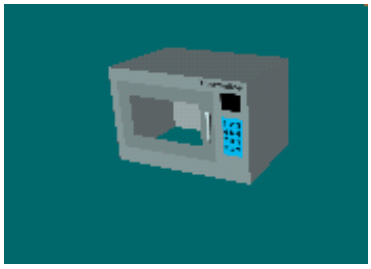
Continue Next Page



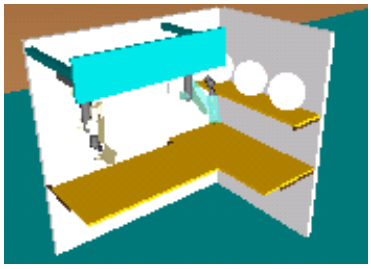
(i) SMT Machine



(j) Lathe



(k) Industrial Oven



(l) Robotic Arms

Figure 6.5 VRML Models available in Multiuser Plant Layout Planner

6.4 PROGRAMMING METHODOLOGY

The Multiuser Layout Planner uses three different scripting languages, LINGO(Director), VRML and PERL. Most scripting was done in the Director Shockwave using LINGO. The Shockwave does most of the location calculation and displays a two-dimensional layout. The frame that contains the VRML model is also refreshed by a command (getNetPage) in Shockwave. VRML fits into the role of providing moderate quality 3-D visualization. PERL executed on the server, is used to write the setting files and the VRML file.

6.4.1 LINGO PROGRAMMING IN LAYOUT PLANNER

Like the Multiuser Truss Solver, the scripting in this program requires object-oriented programming (OOP) within Director. OOP is important to generate objects that

behave and respond similarly yet still behave independently of each other (Macromedia, 1998). This is important because it is not efficient to create, for example a table each time it is needed. Each instance of a parent script is called a child object. By using parent scripts in Director, a specified number of tables (child objects) can be created but with individual properties such as X-Y position. In order to place a child object on the stage, a temporary sprite has to be first in the sprite channel. The number of child objects is limited by the amount of RAM memory of the computer.

A behavior script is a script that is attached to sprites or frames in the score (Bacon, 1999). When behaviors are attached to a score, it is called a score script. On the other hand, behaviors attached to sprites are called a sprite script. A behavioral script is attached to each individual icon's sprite. This script allows the check for the active menu of icons so that the appropriate commands can be executed. The command then will execute a handler that creates the child object by calling the parent script. Also, it will send out a command string to the server to prompt other connected movies to reflect the changes. The following script is the behavioral script attached to the sprite member of each icon.

```
On MouseUp
  global gconnection,menuflag
  if (menuflag=1) then
    CreateTable
    regenTab="Regenerate Table Child Object"
    sendNetMessage( gConnection, "@TypeTestGroup", "testStr", regenTab, 0 )
  end if
  if (menuflag=2) then
    CreateMach
    regenTab="Regenerate Machine Child Object"
    sendNetMessage( gConnection, "@TypeTestGroup", "testStr", regenTab, 0 )
  end if
end
```

The "MouseUp" is an event handler that contains statements that are activated when the mouse button is released. The global variables here are the "gconnection" and "menuflag". The condition of "MenuFlag=1" indicates that the icon set one is active. "CreateTable" is the handler that calls the parent script. The string "Regenerate Table Child Object" is sent by the sendNetMessage command to notify the server and other client movies of the changes. "Gconnection" is the global variable to when TRUE indicates an active connection. When the connection is not active a send message will result in an alert message.

If an object is moved on the stage, it is reflected immediately on other connected Layout Planner when the mouse is released. This is made possible by another behavioral script that sends the X-Y points to the server using the SendNetMessage command. In addition, a location script allows the position to be displayed when the mouse rolls over an object. A channel has to be turn into a puppet channel for LINGO to take control over the sprite.

The following is a sample of the script that establishes the puppet channel, constraints it and also creates table child objects. Note that a counter is placed to count the number of child instances created. This is important so that a child instance is not placed in a location where it is not needed or where it doesn't have a temporary sprite.

```
------------------
-- For PuppetSprite
------------------
global tableList
tableList=[]
repeat with spriteChannel = 41 to 46
  puppetSprite spriteChannel, TRUE
  set the constraint of sprite SpriteChannel=gridconstr
end repeat
```

```
  ------------------
  -- Create new table
  ------------------
on CreateTable
  global tableList,gConnection, tableName
  if count(tableList)>=6 then
    Alert "Maximum number of table allowed!"
  else
    add(TableList,new(script"CreateTable Parent",count(tableList)+1))
  end if
end Createtable
```

To create that a new instance of the child object the command new ("name of parent script") is executed. Below is the parent script to create the child instances. Note that all other child objects share the properties given here. After the child object is created, it can have its own individual properties like X-Y location but the height and width parameter cannot be changed. In this case the horizontal and vertical positions are set at (50,50). This means that the child instance will be created and place at (50 pixels, 50 pixels) from the default coordinate system in Director.

```
property Mysprite
On new me, tableListPos
  global tableSpr1 ,tableName, gConnection,TableChd
  tableSpr1=32
  set myListPos to tableListPos
  set mysprite to myListPos +40
  set the memberNum of sprite mysprite to tableSpr1
  set the width of sprite mysprite to 18
  set the height of sprite mysprite to 12
  sprite(mysprite).locH=50
  sprite(mysprite).locV=50
  sprite(mysprite).moveableSprite=TRUE
  tableNum=string(myListPos)
  tableName=string("tablechild")&&tableNum
  TableChd=string(tableName)
  return me
end
```

It is also important to note how the received message is handled. The setup message handler (subroutine) will route the data received to appropriate handlers that are assigned to handle that type of data. The senderID is checked so that the same command is not execute twice by the originating Shockwave. If the sender and receiver is the not the same, it will proceed to execute the command. Below are the setup handler script and also the message handler that executes the command.

```
on SetupNetMessageHandlers me
  set errCode = SetNetMessageHandler( conn, #TableChildHandler, me,
"TableChild" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #TableChildHandler"
    return
  end if
end SetupNetMessageHandlers

on StringMsgHandler me
  global UserPathNameWRL,gConnection
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    sender = newMsg.senderID
    myname = member("UserNameField").text
    TempText=member("MsgChannel").text
    NewTempText=getAprop(newMsg,#content)
    member("MsgChannel").text=string(TempText & return&& NewTempText)
     if (getAProp(newMsg, #content)="Reloading VRML") then
      gotoNetPage (UserPathNameWRL, "VRML")
    end if

  if myname <> sender then
      if(getAprop(newMsg,#content)="Regenerate Table Child Object") then
        CreateTable
      end if
    end if
      else
    alert "*** Error in StringMsgHandler: " & newMsg
  end if
end StringMsgHandler
```

6.4.2 PERL PROGRAMMING FOR LAYOUT PLANNER

The basic outline of the scripting used in Director was shown in the previous pages. Since the file writing capability in Director is limited, another language must be used to write the setting files and edit the VRML file. For this research, PERL was chosen because of its simplicity and also its functional capabilities. PERL is an acronym that stands for Practical Extraction and Report Language (Zhang, 1997). PERL was originally designed to parse different inputs and generate output based on incoming data. PERL is an interpreted language; there are no compilers and it is platform independent. It provides some well-defined networking capabilities and limits the scope of intentional or unintentional security problems.

In the Multiuser Layout Planner, calculations are made in Director Shockwave movie before outputting the processed data to the PERL program. This allows the PERL program to write the VRML file. The Shockwave and PERL program are linked together using the postNetText command. The following is the sample command to post data from Shockwave to PERL program.

netID = postNetText ("http://eml.ou.edu/multiuser/cgi-bin/table.pl", infoList, "Win")

The information is sent in the same way a browser posts an HTML form with METHOD=POST. An alternate way is to use postText command in Director but it is not recommended because of posting incompatibilities with HTML forms. The infoList variable in the above command line contains the list of data to be sent over to the PERL program. The serverOS parameter "Win" or "Mac" translate any Return characters in the postText argument into those used on the server to avoid confusion. Most of the time this is unnecessary.

The PERL program uses a standard module to write files (CGI qw/:standard/:). As shown below, the input data is split into the variable name and variable values. Then the input variables are inserted into the program that writes the VRML file. The PERL script only modifies one main VRML files and if a new object is created, it will have the VRML file placed as an inline to the main VRML file. By doing this, the PERL file will be simpler and easier to edit. PERL writes out a file called tableset.txt to store the object settings. Below is a part of the PERL file to generate the VRML table model and place it at the correct X-Y position.

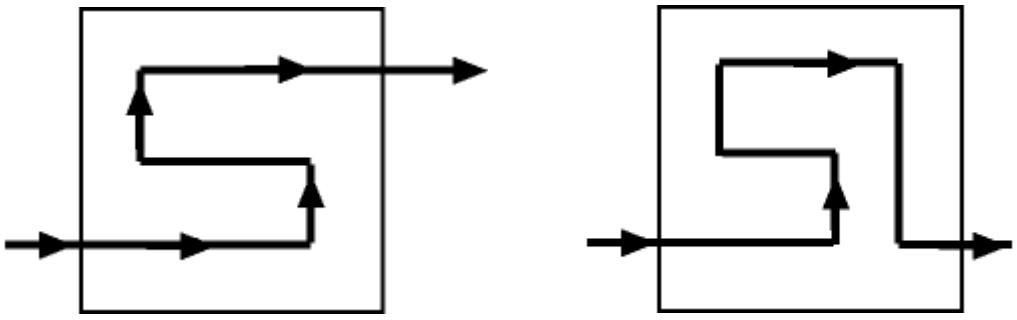
```
use CGI qw/:standard/;
# split name and value into pairs
@nvpairs = split(/&/, $buffer);
#table 1 x pos
$pair4 = @nvpairs[3];
($tab1xname,$tab1x) = split(/=/,$pair4);
#table 1 y pos
$pair5 = @nvpairs[4];
($tab1yname,$tab1y) = split(/=/,$pair5);


print INFO <<endfile;
#VRML V2.0 utf8
#Cosmo Worlds V2.0
$abc=1
if ($tabNum>=$abc1){
print INFO <<table1seg;
DEF table1 Transform {
  children        Inline {
    url    "table.wrl"
  }
  translation $tab1x 0 $tab1y
 }
table1seg
}
close(INFO);
```

One last part of scripting in Multiuser Layout Planner is the frame script in Director to read in the setting files that the PERL wrote. The frame script reads the text file word-by-word and assigns a variable to the value. Multiuser Layout Planner uses these variables to restore the latest updates the in arrangement.

6.5 USE FOR MANUFACTURING

Other than providing visualization, the Layout Planner can be utilized as a teaching tool in a manufacturing class especially if it is a distant learning class or a laptop class. Initial ideas or concepts of a layout can be quickly assembled and presented. Also, concepts like flow patterns can be illustrated more easily as shown in Figure 6.6. A virtual plant visit is also made possible in a class equipped with computers and projector.



(a) (b)

Figure 6.6 Vertical Flow Pattern

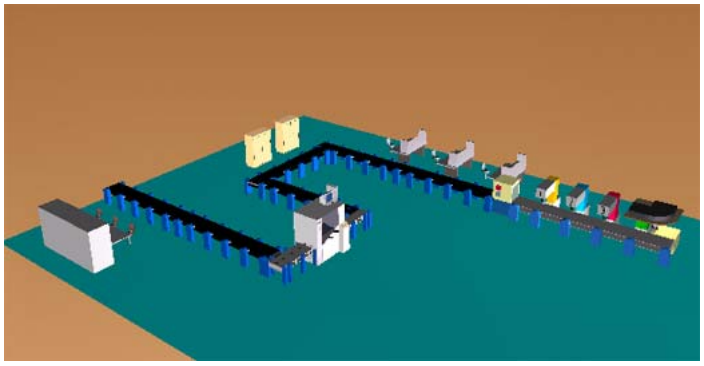


Figure 6.7 Three-dimensional VRML representation of vertical flow pattern in Figure 6.6 (a)

The Multiuser Layout Planner is generally not specific enough to support industrial use nevertheless it demonstrate what can be done with Internet technology. At present, the Plant Layout Planner is only good for visualization and demonstration purposes. More analysis tools can be developed to make it useful for actual manufacturing purposes.

There are a few weaknesses in a Virtual Factory concept. First the models might be too poor in detail to provide accurate answers or solutions (Will, 1995). Subtle changes in a factory layout can produce large fluctuations in factory operations. More accurate representation like dimension of machines can be added. Wrong representations of process might lead to wrong answers. Moreover, simulation might run too slow to provide timely answers and also in accurate process representation.

There is few commercially available software for layout planning. Among them is ARENA from Systems Modeling, Virtual Factory by EAI and AWESIM. Commercial software incorporates better simulation and calculations. One notable weakness in the commercial simulation software is the lack of collaboration tools to unleash the power of the Internet for concurrent work.

CHAPTER VII

VRML FOR VISUALIZATION

7.1 INTRODUCTION

Software developers have started to take advantage of the most significant advances in computer technology most notably, internet-based networks. VRML or Virtual Reality Modeling Language is one of the many potential educational tools on the Internet. It is the acknowledged three dimensional Web standard for visualization. It allows the viewer to examine the model at different angles and at different distances, all within a Web browser. Animations, sounds and interactivity are also possible with VRML.

This chapter describes the application of VRML and the imparting of manufacturing education to engineering students using the Internet. Good engineering programs are generally too complex to use, too costly and not widely available for engineering education. Engineering Design Graphics and Manufacturing Processes are generally compulsory courses in both the Aerospace and Mechanical engineering syllabus. The extensive use of computer with the implementation of the laptop program in University of Oklahoma further encourages the need for virtual machines and a factory to supplement engineering coursework. Also students can access the virtual models anytime and anywhere with a computer and a VRML player. The use of visualization tools also promotes interest and curiosity towards a manufacturing course.

Frequently, mechanical and aerospace engineering students don't have enough exposure to the use of basic industrial and manufacturing equipment. It is not possible to bring real machinery into the classroom and on the other hand it is not easy to teach

students outside the classroom particularly in the workshop. Therefore there is a need to have some medium of teaching that is easily available and allow students to visualize interactively. VRML models supplemented with class notes on the Web page can solve this problem by bringing virtual machine models into the classroom. The instructors will need a laptop, a VRML capable browser and a projector to present the VRML models.

The Internet or the World Wide Web is a good source of vast amounts of information for research and education. Over the pass few years, the Internet has seen a tremendous growth in the amount of users and information. There has been research by many universities and government agencies around the world in developing VRML models on the Web (VRML Works, 1999). VRML developers have published their field of work ranging from virtual places, artificial intelligence, chemistry, engineering and many more. VRML models are emerging as fast as homepages appeared a few years ago. The purpose is to explain the potential use of VRML models for engineering education. This chapter introduces visualization of manufacturing process and equipment, an area of engineering education which VRML can be used to supplement conventional teaching methods.

7.2 COMPUTING POWER AND VRML

The amount of data that can be transmitted increases as the bandwidth of the Internet grows. This allows for more 3D-model visualization over the Internet that can be CPU intensive in addition to accommodating large file sizes. For example, even in VRML, a common 3D file format for Web use, the files are generally designed to be small but these files can quickly become complex when trying to realistically model an

actual object or system. Thus, it is expected that the increase in bandwidth will be offset by more complex models.

In addition to the file size considerations, the ever increasing computer processing capability is also an advantage to the VRML world because more details can be added to a model without a large penalty on modeling time. It can be frustrating to wait for model details to load up and render as one moves about in a VRML world. Therefore the ever-increasing CPU power will enable more complicated VRML models to be created and used in engineering education.

7.3 VRML

One of the methods to present visualization models for engineering education on the Internet is VRML (Virtual Reality Modeling Language). VRML is a platform-independent file format for sharing 3D worlds on the Web (Ames, 1997). VRML worlds are extremely versatile in that they support interactivity, animation, user modification, and embedded hyperlinks to other Web documents. Furthermore, VRML file formats are independent, therefore all computers (PC, Mac, Unix, etc.) that use a Web browser can use and view VRML models with a VRML player. In addition to these advantages, VRML is inexpensive and flexible, hence this file format was chosen for this research project. VRML format is less complex when compared to other 3-D CAD programs and can easily be published on any Web site.

The actual VRML file is a textual description of the model that can be created using any text editor or word processor. VRML file names end with the extension .wrl and contains seven main types of components including a VRML header, prototypes, shapes, interpolators, sensors and scripts and routes.

VRML building instructions must include precise sizes and distances to control the size and placement of shapes built within VRML's three-dimensional space. Currently, VRML only uses polygons to build the 3-D objects, however future releases are expected to include spline and NURB based objects. Geometric transformation such as scaling, rotation and translation is also possible in VRML. To create interactivity in the VRML world, a wiring route or path between two nodes is required. Once a route is built between two nodes, the first node can send messages to the second node. These messages are known as events and contain a value, similar to field values within nodes.

7.3.1 VRML BROWSER/PLAYER

Similar to other software that needs a container program to play it, VRML has its own player to view the VRML models. This is similar to other media types, such as sounds and movies, where the Web browsers passes that file information to helper applications called plug-ins. Plug-ins are programs that enable you to view non-HTML information within the Web window. To view VRML documents, the user needs a VRML helper application or plug-in, called a VRML player, that works inside the browser. When the player reads a VRML file, it builds the world described in the file and places the model inside an interface within the browser window.

There are a number of VRML players that are available for both main browsers (Internet Explorer and Netscape Navigator). Some of the most common VRML browsers available are Silicon Graphics' CosmoPlayer (www.cosmosoftware.com), Intervista's WorldView (www.intervista.com), and Dimension X's Liquid Reality (www.blaxxun.com).

7.3.2 VRML AUTHORING SOFTWARE

Due to the expected need for creating 3-D models quickly and efficiently, software developers have come out with authoring tools for VRML. Even though VRML files can be constructed and edited with any text editor, the files quickly become large and unmanageable using just a text editor. To assist in the designing of VRML worlds, there are a number of VRML authoring software available that enables the creation of VRML worlds to be much faster and simpler than using just a text editor. Among these are Cosmo Worlds, Homespace Builder, VR Creator and Site Pad. Most of these software programs provide advanced graphical tools for creating, editing, optimizing and packaging VRML content.

In addition to VRML editors, most CAD software programs support VRML as an export option because of the growing importance of having a user's 3D work published on the Web (ASME, 1997). However, one drawback of using the export option in CAD programs is the lack of VRML code optimization. Thus CAD generated VRML files generally have too much detail and takes more polygons to describe the geometry then what is needed. The end result is a large file that causes long download times. It should also be noted that most CAD programs do not read or import VRML files.

In the future, Interactive 3D graphics will be embedded in word processors, spreadsheets, and presentation software. Currently Microsoft is working towards improving data visualization for PowerPoint, Word and Excel. Users have realized that there is a need for a 3D format that can serve the ever expanding Internet community. VRML format appears to be able to serve this purpose well.

7.4 VRML LIMITATIONS AND OPTIMIZATION

VRML is designed to work well when used with both very powerful computers and very inexpensive machines, allowing the VRML to trade off image or simulation quality for improved performance and increased hardware performance (Carey, 1997). VRML worlds should also scale with network performances, from 14.4K modems that are common today to multi-gigabit connections that might become common in the future. Similar to other file formats, VRML has it own share of limitations. One of the limitations is that the amount of polygons in the VRML dictates the rendering time. The more polygons present, the more CPU (Computer Processing Unit) time are needed to render the 3-D object. Simplicity is a design constraint, compromising details for faster rendering time. On the other hand, if the file size gets too large, a faster Internet connection speed will be needed.

As mentioned previously, there are some tradeoffs in performance versus contents of a VRML file. Therefore, optimization is an essential part of creating VRML worlds. Given existing limitation on bandwidths and rendering speed, any world, which isn't carefully optimized, may be too slow to capture the audience's interest (Hartman, 1998). Content must be balanced with the performance requirements to keep everything running at interactive speeds. Level of Detail (LOD) node can enclose different versions of a shape in varying levels of details. The LOD node is used to reduce the number of polygons visible at one time. For instance, if a level of detail node is used on a box object, when the user is far away from the object, the VRML browser would load up the model with less detail so that it does not need to render the extra details. This directly decreases the rendering time of the CPU, which then translates to faster movement in the

VRML world. Other methods of optimization include, replacing complex surfaces with textures, inline code, collision nodes, and reducing the precision of floating point numbers to reduce file size and use of compression (gzip).

7.5 APPLICATIONS OF VRML

Due to the need for 3-D visualization, Web-based VRML can be used in almost all fields of study including art, business, engineering and sciences. The main benefit for the student or learner is the ability to view and explore a virtual environment that they can visit or enter. Examples range from working in a factory to exploring the inside of bacteria. In short, if a picture is worth a thousand words then a VRML model is worth a thousand pictures. Not only is VRML a picture, but it is also dynamic 3–D environment that interacts with the user. However, VRML is not the only way to publish 3-D but it is one of the cheapest and easiest methods. Three-dimensional models can also be published on the Web using Java 3-D.

Most Computer-Aided-Design (CAD) developers realized the importance for engineers to publish their work on the Web. For instance, the newly released Pro/ENGINEER 2000i from Parametric Technology Corp provides capabilities to publish designs on the Web, attach hyperlinks to components in an assembly, and view and access 3D assemblies from any Web browser. Though Java programs in commercial software are highly capable of publishing the actual model, it is not widely available to most engineers. The high cost of getting a commercial CAD program that can publish models to the Web prevent most engineers from sharing their ideas on the Web. The importance of publishing one's engineering work on the Web, in addition to journals is to gain recognition and also share ideas with other engineers.

7.5.1 APPLICATIONS IN MANUFACTURING

A good example of VRML usage in engineering is the 3-D visualization of processes, machinery and robotics in the manufacturing field. Engineers can use VRML to display three-dimensional manufacturing data or models to managers so they could make better decision together. For instance, a group of students taking the capstone class could present to their sponsor the 3-D VRML model using any Internet capable computer in a conference room instead of having to meet in the CAD lab to demonstrate their model.

The model can then be examined and determined as to whether the design expectations have been met. It is easier for mechanical and aerospace students to explain their design with a 3-D model rather than presenting 2-D drawings. Note that 2-D drawings are still important when the design is to be submitted to the machine shop.

7.5.2 ENGINEERING EDUCATION

VRML worlds that are broadcast over the Internet have many potential uses in instructional and educational settings. VRML can be helpful in an engineering course because it can provide better flexibility in viewing data. Views can be customized for each application and animated. This makes learning easier and fun. For example, a 3-D VRML model could show the 3–D stresses of a beam under a loading. The student could examine, rotate and view it at different angles which makes learning the concept of beam theory easier especially when it comes to showing the location of maximum or minimum stresses. Another good example that can illustrate the use of VRML in engineering education is the VRML mechanical catalog. An example of this can be found at the site http://www.evl.uic.edu/fred/cat_dev/vscript2/cat1_main.wrl. The catalog allows the

viewer to see a mechanical part, explode the assembly, and link to the respective URLs that contains the parts specification. Another capability of VRML is its ability to view 3-D objects by altering the transparency. This can be effectively used to show different level of parts or how they fit together. For example, the user could set the transparency of the outside parts to 100 percent so that the inner components are viewable and how it works. Figure 7.1 shows a VRML model of a hard drive where the user can specify the height and width of case, number of disk, radius of disk and most importantly adjust the transparency with the slider.

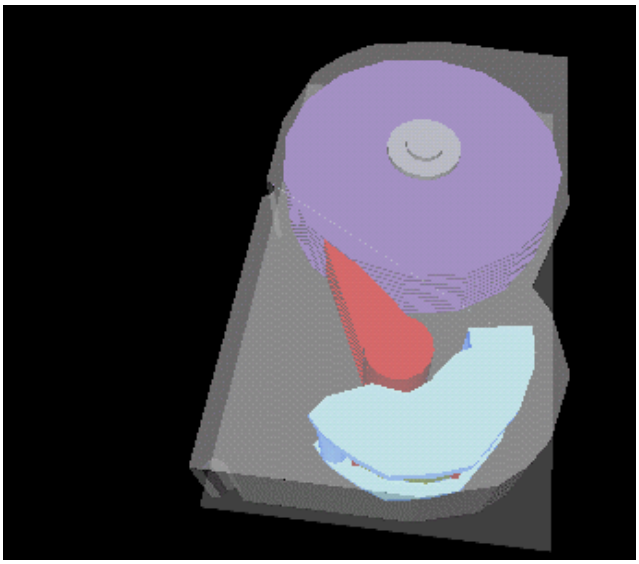


Figure 7.1 Hard Drive Design and Visualization

One of the focuses of this thesis is to introduce animated manufacturing processes. For example, a manufacturing process of packing potato chips allows the viewer to understand how each process works and how the end product is packaged. Generally students learn better when they are able to visualize it. Students can learn even better and have fun with the interactive machining process. Figure 7.2 depicts the fully

operational VRML model when viewed in a browser. When the process button is on, a tin would appear and it would go through the first process of chip dispensing into the tin subsequently to the sealing of the tin, lid placement and moving the cans to the packing box.

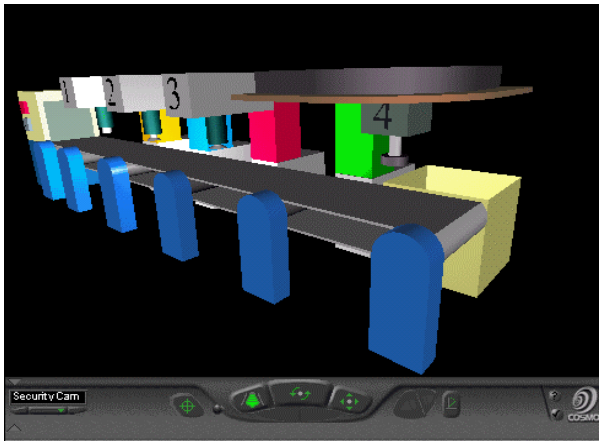


Figure 7.2 Packing Line

A virtual machine is a software emulation of a real machine. A virtual machine has an instruction set, just like a physical machine. Frequently mechanical and aerospace students don't have enough exposure to the use of manufacturing equipment in their manufacturing class. By creating three dimensional machine and components; not only can the students rotate and examine the part from different angles but also operate or run the virtual model. By imparting the concept of each part of the machine, the students can easily pick up skills needed to enable them to design something that can be practically machined.

Within the VRML code, scripts and sensors can be used to detect the actions of the user. Also, interpolator nodes can be used to animate any object within the world,

such as mechanical components or parts. The interactivity that can be added to a VRML world also includes the ability to change parameters while the user works with the object. Another focus of this paper is to introduce the virtual lathe made in VRML, which illustrates the ability for a user to operate virtual equipment over the Internet. This lathe

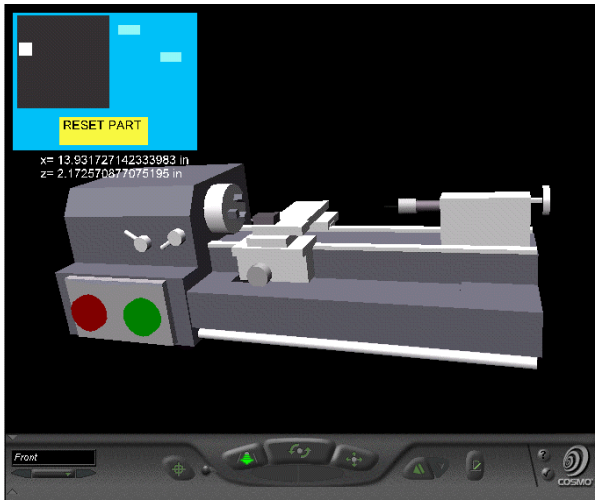


Figure 7.3 Virtual Lathe

has interactivity built in it by using JavaScript and routing nodes and sensors and is shown in Figure 7.3. Like a real physical machine, the virtual lathe can be instructed to perform certain tasks by pushing or touching buttons on the control panel.

The virtual lathe is useful to illustrate to students how products are machined and what are the processes involved. For instance, it is easier to illustrate in the class, material removal sequence efficiently. This lathe is really a simplified version of the actual lathe but it is sufficient enough to show how the lathe works. The large red and green buttons on the side of the lathe are used to start and stop the lathe chuck from spinning. The lathe has a control panel that pops up when the green button is clicked which has additional capabilities as seen in Figure 7.3. The control panel consists of the indicator, two sliders

and one reset button. The indicator on the monitor panel and the sliders are used to translate the movement of the table and tool. The numbers below the control panel shows the tool location.

7.5.3 CONSTRUCTION OF THE VIRTUAL LATHE IN VRML

The virtual lathe in Figure 7.3, was modeled in Cosmo Worlds and includes JavaScript to facilitate user interaction. The final VRML model contained 1072 polygons and had a file size of 29 Kb. To further reduce the file size, the file was compressed into a 'gzip' file type that reduced the file to only 5 Kb. It should be noted that VRML players recognize 'gzip' files and automatically uncompressed the file when loaded into the player. Gzip can be done using any compression software that supports this file format. For this example, gzip is done automatically when the VRML world is published in Cosmo Worlds. This is certainly acceptable for downloading through the Internet, even with a home modem. The majority of polygons were due to the cylinders used to model the gear settings, drill and the chuck. It is for cosmetic purposes and can be taken off to reduce the file size even further. There are two types of objects in VRML. One consists of a customized shape node such as cone, box and cylinder. Basic parameters can be changed on this shape object such as the radius of the cone. The second object consists of points, lines and polygons. In other words, the object is created with the use of line, surfaces and points. Since points, lines and surfaces create it, it is allowable to change each single parameter of the entity, thereby giving much more flexibility to draw complex geometry rather than using only the basic shapes. Cosmo Worlds refers to this type of object as PEP (Points, Lines and Polygons). The lathe motor is broken into a PEP

object and some faces of the box are extruded out. All other parts of the lathe uses the basic shapes available in VRML.

To allow the user to interact with the lathe, touch sensors were placed on the sliders, indicators and lathe buttons. When the start button is touched, the sensor will activate the script and the time sensor. The script will take in the SFTime field value and activate the control panel that pops up. SFTime is a floating point value that gives the absolute time measured in seconds since 12:00 midnight, GMT, January 1,1970. In other words, it takes in the time when you activate the sensors and perform a job that the script defines. This time field value is typically used to start or stop animation. In addition, the input to the time sensor would activate the Orientation Interpolator, which in turn makes the chuck rotate. The Orientation Interpolator in VRML computes a rotation based on a list of key rotations specified in radians. Both touch and plane sensors are used to control the slider and the indicator for controlling the tool and table position in the X and Z-axis. The touch sensor's value is true when an object is touched it when the pointer is activated. The translation of the sliders and the indicator is routed to the script. The script takes in the field SFVec3F values and output it to the table and tool. The SFVec3F field is a 3-D floating- point drawing that specifies the 3D position. The output values however are multiplied with a factor so that a small translation on the indicator results in a bigger translation change in the tool and table.

Subsequently, the translation values of the tool will be sent back to the script, which takes in the value and output it to a string in a Text node. Since the model is drawn in meters, the script converts the units to meter. The Text node then displays the position change as the tool position is moved. Constraints have been set on the plane sensor and

the position of the tool so that it would not overlap with the other objects. Using the if and then statements in the script allows a more realistic model for instance restrict the maximum movement of the machine table so that it does not go into the machine casing. Setting the maximum distance to a specified maximum realistic distance does this. Sliders' translation and indicator's position is routed together through the script in order for the other to move accordingly when one of them is repositioned. The touch sensor on the reset button sends the value of the touchTime to the script. The script in turn set the X and Z position back to zero. Thereby setting all the parameters back to 0.

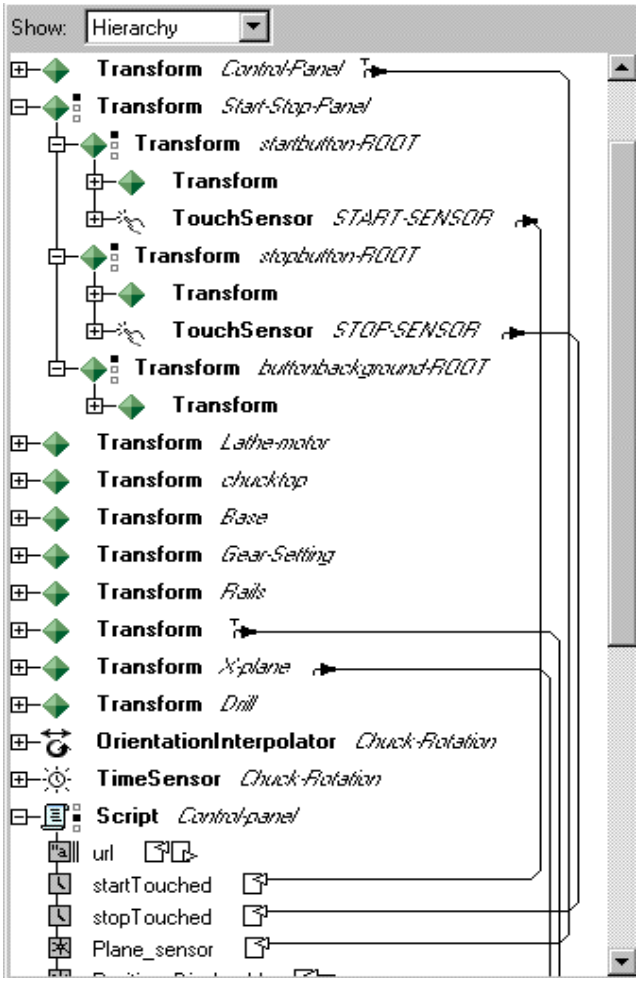


Figure 7.4 Values Routing in Cosmo Worlds 2.0

Once the control panel of the virtual lathe is activated, it stays in that position regardless of whether the user rotates the lathe or zooms out. The control panel transform is actually placed inside a collision node together with the proximity sensor. The field values of position_changed and orientation_changed from the proximity sensor are routed to the translation and rotation of the control panel's transform. The purpose of doing that is to allow the position_changed and orientation_changed fields to recalculate the drawing each time the user moves around so that the control panel always stays on the upper left hand corner. The objective is to recalculate the fixed position of the control panel from the viewer whenever the viewer moves. The collision node is set so that the control panel transform does not collide with the lathe's transform. There is a switch node in the overall collision node. This is for turning the control panel on and off. When the touchTime on the start button is sensed, the script would send a 0 value into the field whichChoice and it would display the control panel. The touchTime field is to obtain the time when the touch is detected and the whichChoice field activates the script. On the default however, the field value would be –1 so that it wouldn't display the control panel. Vice versa, the stop button when touched will send in a –1 value into the whichChoice field. The routing of the values can be shown in Figure 7.4.

At present, when the lathe is turned on, an audio file is played to simulate the sound of an actual lathe. In VRML, sound is accessed through the use of the 'AudioClip' node that links to a sound source. Both WAV files and MIDI files can be used with VRML. WAV files are easy to use but can cause file size problems since they are not compressed. On the other hand MIDI files are not true digitized sound waves like WAVs, but equations used to reconstruct the sound. This allows MIDI files to be small, but does

not accurately reproduce the actual sounds. In this virtual lathe a small WAV file size of 3K was used. To minimize the file size, a short sound was looped indefinitely until the virtual lathe is turn off. One advantage of using sound files in VRML is that it supports sound spatialization. Sound spatialization is a digital sound that appears to emit from a specific point in 3-D space. For instance when the user gets nearer to the sound source the sound appears to be louder or when the sound source is towards the right, the right speaker would sound louder than the left speaker would.

As the user monitors the lathe, information about the system is written to the VRML world. This was done with a simple JavaScript that truncates the floating-point numbers and then writes them to a text field in the VRML world. The truncation is done with a JavaScript that takes in the floating-point numbers and converts it to text string. From the text string, only those digits that are needed are processed and output back to the text node.

Attempts have been made to create a Proto node in which the inner radius and other radius of a hollow cylinder can be changed. Proto node is a customize node that can be reusable. However the Proto node has not been integrated into the virtual lathe. It is possible to write a script that gives a warning whenever the tool cuts too much of the workpiece material. Further work can be done to change the speed of the chuck and display some animated sparks when the tool touches the workpiece.

7.5.4 INDUSTRIAL APPLICATIONS

The rise of Intranets – private network based on Internet, is promoting concurrent engineering, communication, and cross platform sharing of information especially between manufacturers, suppliers and customers. As a result, customers or clients can

view the VRML product while it is in the designing stage and give feedback before it is manufactured.

Customers or clients too don't need to have the same CAD software to be able to view the VRML model. What they need is just a VRML browser. Furthermore, VRML parts are cheap and easy to develop compared to standardizing a CAD software for all the suppliers and manufacturers so that they can all view it. VRML is also a good presentation tool for meetings or when proposing a prototype. It would make the proposal much more attractive.

Combined with JavaScript or Java, operators could be trained using a virtual machine before using the real machine. Therefore saving training cost, machine cost, freeing up the real machine for the real work and ensuring the operators' fitness to operate the real machine.

Furthermore better visualization tools such as VRML can enable engineers to develop better manufacturing processes. Usually in a factory, the CAD model is not easily accessible to the manufacturing engineers; therefore they need some visualization tools to view how the end product should look like to help them in their job. Furthermore, license of the CAD software is usually limited in one company. Therefore preference would go to the R&D engineers first instead of the manufacturing engineers.

Instead of hiding all the information in many levels of queries, it is easier to publish the part and the VRML model in the Intranet so that all levels of workers could access it or password the site to allow only certain group of people to view it. Sometimes when the engineer can't remember the part number and can only remember how it looks like, he/she has to go to the trouble of pulling up the Bill of Material (BOM). On the

other hand if he knows the part number but can't remember how it looks like, he/she can download the design drawings and determine which part it actually is. Therefore if the manufactured parts are link by some organized Web pages, it would provide easier retrieval of information and accessibility through any computer in the factory.

7.6 VRML SUMMARY

In an effort to bring more visualization into design education, VRML has been shown to be a useful 3-D file format that can used on the Internet. The use of VRML as a tool in manufacturing and engineering education is beneficial to the engineering community since virtual environment makes learning fun for engineers and education. Most importantly, there is no constraint on the time and place for the users to view and interact with the virtual models. For instance, students can view and try out the VRML models outside their class-time. If engineers share exported VRML design in the Web, other engineers can study and comment about the design. This in turn promotes concurrent engineering. Manufacturers of mechanical parts can maintain a VRML Web site that shows 3-D VRML models and save cost from printing the part catalogs. On the other hand, engineers can examine the 3-D model of the parts before buying. The engineer's judgement is generally better with a 3-D model compared to a 2-D drawing in the catalog. In conclusion, VRML is not at it's full potential and could well develop to make engineering education a much more interesting experience.

CHAPTER VIII

SUMMARY AND CONCLUSION

8.1 SUMMARY

The main objective of this research is to facilitate the development of a collaborative design through the Web using Director Multiuser Server. To accomplish the objective, four multi-user applications have been developed. At present, there is a need for collaborative and design through the Web. This research concentrates more on utilizing a combination of Internet technology (Director 7 Multiuser Xtra, JavaScript, PERL and VRML) to demonstrate the potential of collaborative design work for enhancing education and industry work. Development of this kind of multi-user tools allows new methods to communicate design ideas and permit concurrent design without physically being together. However, like most new technology, creating a multi-user tool has an initial learning curve and also requires practice before it can be done efficiently.

This research also investigated some of the potential benefits to collaborative learning and a concurrent design environment when the powerful communication tools of high-bandwidth networking and digital hypermedia are joined. The environment created in this research enable student to collaborate with drawing, text chat and also construction of 2-D layout and truss design. VRML was used to present the data in a 3-D format on the Internet.

Most efforts in this research were devoted to the development of Web-based collaboration and concurrent design than on content development for the multi-user tools. The Multiuser Painting Board was a multi-user program prototype that allows simple digital painting with collaboration feature. Following the prototype Multiuser Painting

Board, the Multiuser Drawing Board was developed. The Drawing board was a more polished and practical program because of its ability to change the size of vector shapes and other parameters. Next was the Multiuser Truss Solver that performs simple collaborative truss design. The last application, Multiuser Plant Layout Planner, was specifically targeted for industrial engineers or students learning to do layout and plan. All multi-user programs written in this research include the ability to text chat with other users.

The Virtual Lathe model with JavaScript demonstrated a 3-D model of a machine with interaction. VRML is platform independent format and hence it is suitable as a 3-D visualization tool on the Internet. This method of collaborative design is still new and there is more room for improvement.

The development of multi-user applications in this research demonstrated the feasibility of doing collaborative work on the Web. It accomplished the objective of establishing a new method to do online design collaboration and incorporates multimedia into it. The multi-user tools are useful because it allows people to work together electronically. VRML was used in this research to permit visualization of collaborative design. The integration of multi-user and engineering concepts on the Web, fulfills the need for online collaborative tools.

8.2 CONCLUSIONS

Several conclusions can be drawn from this detailed study on multi-user and VRML applications. It is shown that a collaborative design environment on the Internet is feasible and cost efficient. It encourages the use of Internet as a learning tool. More

resources should be invested into developing digital courses and multi-user tools that are more economical in meeting demands.

Furthermore, one good use of the collaborative multi-user tools is to implement them into engineering courses through the Internet. It is recommended that implementation of multi-user tools should coexist with related HTML, pictures, sound and videos of that subject. For example, a multi-user truss solver can serve as a simulation inside HTML Web page containing theories, solution and explanation of trusses. This way, students would be able learn the concepts as they browse through the theories and collaborate with the multi-user tools.

It also can be concluded that for a multi-user application to be flexible and effective, a combination of Internet technologies have to be used to realize the advantages of each one of them. PERL is good for server-based programming which allows writing text files on a server. Director Multiuser is well suited for authoring a multimedia collaborative simulation. VRML is the common format to show 3-D objects on the Web. JavaScript makes it easy to produce interaction on the Web.

8.3 RECOMMENDATIONS FOR FUTURE RESEARCH

The contents of the multi-user tools can be further developed. In this thesis, the development the collaborative environment is more emphasized compared to the content development of the tools. More content development allows the tools developed here to be more useful and generic. The capability of this Web collaboration technique can be further improved with the use of database. The Multiuser Server can link to database in dBASE IV format with FoxPro index files, and each movie can access several databases.

The ability to keep database with Director Multiuser means more complex multi-user application can be programmed and make tracking records easier.

Currently, the multi-user application is available for any person on the Internet to use but only those that have Director 7 can develop the application. In the future, more control might be needed to allow only certain user access. Director Multiuser Server allows the designer to set which movie can connect to the server, user access level, and database levels, create groups and etc.

The Multiuser Truss Solver would be more useful for undergraduate student if a 3-D version of it could be developed. The Multiuser Layout Planner can be made to do more than a fixed plant simulation. Different typical floor plants could be introduced to improve the variety of the layouts.

As for the Virtual Lathe, more details can be added to make the visualization more realistic. For Instance, the Virtual Lathe can show a stock being cut by customizing the VRML nodes.

REFERENCES

Ames, A., David, R., John, L., (1997) VRML 2.0 Sourcebook, 2nd Ed., John Wiley, New York

ASME Magazine, (1997) Vol. 119 March Issue, BPA International

Bacon, J. et al, (1999), "Director 7 and Lingo Bible", IDG Books Worldwide Inc., New York, pp. 417

Bailey, M.J., (1992), "Tele-Manufacturing: Rapid Prototyping on the Internet", http://www.sdsc.edu/tmf/

Buchal, R., (1999), "Engineering Education in the 21rst Century", ASEE Annual Conference Proceedings

Carey, R., Bell, G., (April 1997), "The Annotated VRML 2.0 Reference Manual", 1st Ed., Addison Wesley

Cedarleaf, J., (1994),"Plant Layout and Flow Improvement", McGraw-Hill, pp.1-12

Chapra, S., Canale, R.P., (1988) "Numerical Methods for Engineers", 2nd Ed., McGraw-Hill, pp. 288-290

Configured Systems, (1996), http://www.config-sys.com/software/air.htm

Emedia Magazine, (1999), "Web 3D heats up", January Issue

Engineering Animation Inc., (1999), http://www.eai.com/

Francis, L.R., (1992), "Facility Layout and Location: An Analytical Approach", 2nd Ed., Prentice Hall International Series in Industrial and System Engineering, pp.19

Forbes Magazine, (1999), "Chrysler's new Jeep Grand Cherokee", October Issue

Gay, G., (1993), http://www.ehr.nsf.gov/EHR/RED/AAT/Award9253085.html

Gibson, I. S., (1995), "Professional Development and Education in Engineering Design",

International Journal of Engineering Education, Vol. 85, No. 1, pp. 61-67.

Gramoll, K., (1994), “Use of Multimedia Development Software for Engineering Courseware”, ASEE Annual Conference Proceedings, pp. 1217-1222

Gramoll, K., (1994), "Interactive Beam Analysis Program for Engineering Education" ASEE Annual Conference Proceedings, pp. 469-475

Hartman, J., Wendy, V., (1998), Cosmo Worlds: User's Guide, First Edition, Silicon Graphics

Hibbeler, H. C., (1995), Engineering Mechanics Statics, 7th Ed, Prentice Hall, pp.241-302

Higuchi, H. and E.F. Spina, (1993), "Improving Undergraduate Engineering Education Through Scientific Visualization", 1993 ASEE Conf. Proc., pp. 1581-1583

Lemay, L., (1996), Teach Yourself Web Publishing with HTML 3.0 in a Week, Sams. Net Publishing.

Logan, D.,(1993), A First Course in the Finite Element Method, 2nd Ed., PWS Publishing Company, pp. 595-596

Macromedia, (1998), "Macromedia Director 7: Using Director", Macromedia Inc.

Macromedia, (1998), "Macromedia Director 7: Lingo Dictionary", Macromedia Inc.

Macromedia Multiuser Newsgroup (1999), news://forums.macromedia.com/macromedia.director.multiuser

Macromedia FAQ for Multiuser Xtra (1999) http://www.macromedia.com/support/director/internet/multiuser/

Marudur, K., (1998), "Concurrent-Interactive Design and Analysis using the Internet and VRML", Thesis, School of Aerospace and Mechanical Engineering at Uni. of Oklahoma, Norman

Mayfield, J., Ali, K.S., (1996),"The Internet as an Educational Tool", Computers and Industrial Engineering, Vol. 31, No.1, pp. 21-24

McArthur, D. J. and Lewis, M. W., (1998) "Untangling the Web: Applications of the Internet and Other Information Technologies to Higher Learning", pp. 1-20

McCormack, C., Jones, D., (1998),"Web-Based Education System", John Wiley & Sons Inc., pp. 1-28

NetProfessional Magazine, (1998), "Virtual Daylight", September/October Issue

PC Week, (August 1999), "Using Online Collaboration to Speed Software Development, Reduce Cost"

Peck, S., Scherf, B.M.,(1997), " Building Your Own Web Conferences", O'Reilly & Associates Inc., pp. 8

Peterson, S., (1997)," Using Java and VRML in Teaching Machine Kinematics", 9th Annual Conference Technology-Based Engineering Education Consortium Proceedings

Popular Mechanics, (1999), May Edition, pp. 22

Rosenzweig, Gary, (1999)" Special Edition Using Macromedia Director 7",Que Publishing, pp. 18

Selby, M., "Algorithm and Flowcharts Menu" "http://www.eng.iastate.edu/efmd/16algor.htm"

Sherman, Stratford, (1996), “Secrets of HP’s ‘Muddled’ Team”, Fortune, Vol. 133, No. 5, March 18 Issue

Shcrage, Michael, (1995), “No More Teams: Mastering the Dynamics of Creative Collaboration, Currency Doubleday”, pp. 31

Tan, C. K., (1997), “Use of World Wide Web as a Resource for Mechanical Design Education”, Thesis, School of Industrial Engineering, Uni. of Oklahoma, Norman, pp.1-16

Tom, S. et al., (1998), " Using HTML", 2nd Ed., Que Publishing, pp. 26Thompkins, J.A., White, J.A., (1984), "Facilities Planning", John Wiley & Sons, pp. 1-14

Virtual Ink, (1999), http://www.virtual-ink.com/company_overview.htm

VRML Works,(1999), http://hiwaay.net/~crispen/vrml/

Weaver, W., Gere, J.M.,(1980), "Matrix Analysis of Framed Structures",2nd Ed., D. Van Nostrand Company, pp.459-472

Wieckowski, M., (1996), “Rapid Prototyping of Assistive Devices”, http://www.asel.udel.edu/rapid/

Will, P. et al., (1995), "Information Technology for Manufacturing", National Academy Press, pp. 110-118

Wong, M. et al., (1998), “Flexible Accounting Systems in Dynamic Manufacturing Environments”, http://www.eng.iastate.edu/abstracts/abstracts/ra1998-0483.html

Zhang, T., Till, D., (1997),"PERL 5 for Windows NT", Sams Publishing, 1st Ed., pp. 1-117

Zia, L.L. and Mulder, M.C., (1996), "NSF Workshop on Information Technology and Undergraduate Science, Mathematics, Engineering, and Technology Education: Challenges and Opportunities", IEEE Transactions on Education, Vol. 39, No. 3, pp. 452-454

APPENDIX A

MULTIUSER SCRIPTS TO ESTABLISH

CONNECTION WITH DIRECTOR MULTIUSER SERVER

```
-------------------------------------------------------------------------------------------------------------------
-- Script for "login" button at start
-------------------------------------------------------------------------------------------------------------------
on mouseUp
  global gLocalUserID, gLocalPassword, gConnection, gConnSysID

  set gConnSysID = the text of field "ServerName"
  set tempName = the text of field "UserNameField"
  set tempPass = the text of field "UserPasswordField"

  if ( gConnection = 0 ) then
    set gConnection = new( script member "Connection" )
  end if

  -- If we have a valid user name, go start the connection
  -- otherwise go back to the start
  set newName = CheckUserName( gConnection, tempName )
  if ( length( newName ) > 0 ) then
    set gLocalUserID = newName
    set gLocalPassword = tempPass
    go to "Connecting"
  else
    go to "login"
  end if
end
-------------------------------------------------------------------------------------------------------------------
-- Movie Script
----- ---------------------------------------------------------------------------------------------------------------
on prepareMovie
  InitMovie
end
-----------------------------------------------------------------
-- Shutdown tasks
-----------------------------------------------------------------
on stopMovie
  BreakConnection
  updatestage
end
-----------------------------------------------------------------
-- Initialization
-----------------------------------------------------------------
on InitMovie

  global gConnection     -- script object for connection
  global gAppID          -- Application name we connect under
  global gLocalUserID    -- our User ID
  global gLocalPassword  -- our password
  global gConnSysID      -- system we want to connect with
  global gConnUserID     -- remote connecting user ID we let connect to us
  global gConnPassword   -- remote connecting user's password
  global RcontrolNum

  set gAppID = "MultiUserGroup"
  set gLocalUserID = "MyUsername"
  set gLocalPassword = "ABC"
```

```
  -- Put in the server name or the server IP address for gConnSysID  --  set gConnSysID =
"129.15.98.23"
  set gConnSysID = "eml.ou.edu"
  set gRefreshUserList = 0

  ---------------------------------------------------------------
  -- User Configuration
  ---------------------------------------------------------------
   set gConnection = 0
  set the text of field "UserNameField" = "Type user name here"
  set the text of field "UserPasswordField" = "Type password here"
  set the text of field "ServerName" to gConnSysID
  set the text of field "UserNameField" = "MyUserName"
  set the text of field "UserPasswordField" = "ABC"
  set the text of field "ServerName" to gConnSysID
end

---------------------------------------------------------------
-- BreakConnection
---------------------------------------------------------------
on BreakConnection
  global gConnection

  if ( gConnection <> 0 ) then
    delete gConnection
    set the text of member "MsgChannel" to ""
  end if
  set gConnection = 0
end BreakConnection

---------------------------------------------------------------
-- InitTypeTest - setup the test
---------------------------------------------------------------
on InitTypeTest
--  set the text of member "RecvMsg" to ""
end InitTypeTest
---------------------------------------------------------------
-- SayHowLong
---------------------------------------------------------------
```

On SayHowLong taskStr

```
  set connTime = float( the timer )
  --  set the text of member "StatMsg" to taskStr & " required " & string(integer(connTime)) & "
ticks, or " & string( connTime / 60.0 ) & " seconds"
end SayHowLong

-------------------------------------------------------------------------------------------------------------------------
-- Connection Scripts
-------------------------------------------------------------------------------------------------------------------------
property conn
property connState
property connIsText
property connUserName
```

```
-----------------------------------------------------------------
-- Create the script - get the xtra object loaded
-----------------------------------------------------------------
on new me
  set conn = new( xtra "Multiuser" )
  set connState = #disconnected
  set connIsText = false
  set connUserName = ""
  return me
end new

-----------------------------------------------------------------
-- Cleanup.  Delete the connection Xtra so the connection is closed
-----------------------------------------------------------------
on delete me
  if ( conn <> 0 ) then
    set conn = empty
  end if
  set connState = #disconnected
end delete

-----------------------------------------------------------------
-- Connect to another computer
-----------------------------------------------------------------
on ConnectToHost me, hostName, hostPort, myUserName, myPassWord, appName, isText
  set connState = #connecting
  if ( conn = empty ) then
    set conn = new(xtra "Multiuser")
  end if
  set connIsText = isText
  set connUserName = myUserName

  -- Setup the connection message handler
  set errCode = SetNetMessageHandler( conn, #InitializeNetMessage, me )

  if ( errCode = 0 ) then
    set errCode = ConnectToNetServer( conn, myUserName, myPassWord,hostName, hostPort,
appName, connIsText )
  end if

  -- Debugging message
  if ( errCode <> 0 ) then
    set errString = GetNetErrorString( conn, errCode )
    alert "*** Error connecting to host: " & errString
  end if

  return errCode
end

-----------------------------------------------------------------
-- Wait for connection from other system
-----------------------------------------------------------------
on WaitForConnection me, myUserName, myPortNum, connUserName, connPassWord,
appName
  set connState = #waiting
```

```
  -- Create the connection Xtra object
  if ( conn = empty ) then
    set conn = new( xtra "Multiuser")
  end if

  -- Setup the connection message handler
  set errCode = SetNetMessageHandler( conn, #InitializeNetMessage, me )
  if ( errCode = 0 ) then
    -- Start waiting for a message.  It replies with
    -- a message handled by InitializeNetMessage
    set errCode = WaitForNetConnection( conn, myUserName, myPortNum, connUserName,
connPassWord, appName )
  end if

  -- Debugging message
  if ( errCode <> 0 ) then
    set errString = GetNetErrorString( conn, errCode )
    alert "*** Error waiting for connection: " & errString
  end if

end WaitForConnection
-----------------------------------------------------------------
--  Check that the user name is entered OK.
-- to do - remove spaces, not-alphanumerics
-----------------------------------------------------------------
on CheckUserName me, userName
  set newName = userName
  set charNum = the length of userName

  if charNum > 26 then
    alert "Please limit your user name to 26 characters or less." & RETURN¬
&& "You currently have" && charNum && "characters."
    set newName = ""
  else if charNum < 1 then
    alert "Please enter your desired user name in the field provided"
    set newName = ""
  else
    repeat with index = 1 to charNum
      set c = char index of userName
      if ( c = ENTER or c = RETURN or c = SPACE ) then
        delete char index of newName
      end if
    end repeat
  end if
  return newName
end
-----------------------------------------------------------------
-- Handler to recieve initial net message from server
-----------------------------------------------------------------
on InitializeNetMessage me
  if ( conn <> 0 ) then
    set numMessages = GetNumberWaitingNetMessages( conn )
    if ( numMessages > 0 ) then
      set msgList = GetNetMessage( conn )

      -- Check that we got acknowledgement from the server
```

```
      if ( getAProp( msgList, #errorCode ) = 0 ) then
        if ( getAProp( msgList, #senderID) = "System" ) then
          if ( getAProp( msgList, #subject ) = "ConnectToNetServer" ) then
            SetupNetMessageHandlers( me )
            set connState = #connected
          end if
        end if
      else  -- Have an error
        put "Error in initial connection: " & msgList
        set connState = #disconnected
      end if
    end if -- numMessages > 0
  end if   -- conn <> 0
end InitializeNetMessage
-----------------------------------------------------------------
-- SetupNetMessageHandlers - configure our handlers
-----------------------------------------------------------------
on SetupNetMessageHandlers me

  set errCode = SetNetMessageHandler( conn, #DefaultMsgHandler, me )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #DefaultMsgHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #JoinGroupMsgHandler, me, "JoinGroup" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #JoinGroupMsgHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #StringMsgHandler, me, "testStr" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #StringMsgHandler"
    return
  end if
      return
  end if


  set errCode = SetNetMessageHandler( conn, #ListMsgHandler, me, "testList" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #ListMsgHandler"
    return
  end if

   set errCode = SetNetMessageHandler( conn, #DateMsgHandler, me, "testDate" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #DateMsgHandler"
    return
  end if

end SetupNetMessageHandlers
-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
```

```
on DefaultMsgHandler me
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    alert "*** Unexpected message in DefaultMsgHandler: " & newMsg
  else
    alert "*** Error in DefaultMsgHandler: " & errCode
  end if
end DefaultMsgHandler
-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on JoinGroupMsgHandler me
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode <> 0 ) then
    alert "*** Error in JoinGroupMsgHandler: " & newMsg
  end if
end JoinGroupMsgHandler
-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on StringMsgHandler me
  global gConnection

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string" )

    sender = newMsg.senderID
    myname = member("UserNameField").text

    TempText=member("MsgChannel").text
    NewTempText=getAprop(newMsg,#content)
    member("MsgChannel").text=string(TempText && NewTempText& return)

    if (getAProp(newMsg, #content)="Cleared") then
      put "your message have got thru!!!!"
      InitLists
    end if

  else
    alert "*** Error in StringMsgHandler: " & newMsg
  end if
end StringMsgHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on DateMsgHandler me
  set newMsg = GetNetMessage( conn )
  put newMsg
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a date" )
```

```
    set the text of member "RecvMsg" to "Recieved Date: " & string( getAProp(newMsg, #content)
)
  else
    alert "*** Error in DateMsgHandler: " & newMsg
  end if
end DateMsgHandler
----------------------------------------------------------------
-- Send a message to the server
----------------------------------------------------------------
on SendNetMessage me, recipient, subject, message, errCodeToSend
  set errCode = 5
  if ( conn <> 0 and connState = #connected ) then
    set errCode = SendNetMessage( conn, recipient, subject, message, errCodeToSend )
    if ( errCode <> 0 ) then
      alert "Error sending message: " & errCode
    end if
  end if
  return errCode
end SendNetMessage

----------------------------------------------------------------
-- Start the connection
----------------------------------------------------------------

global gConnection, gRefreshUserList
global gAppID, gLocalUserID, gLocalPassword, gConnSysID

on exitFrame
  if ( gConnection = 0 ) then
    go "Login"
  end if

  --  startTimer
  set err = ConnectToHost( gConnection, gConnSysID, 1626, gLocalUserID, gLocalPassword,
gAppID, false )

end


----------------------------------------------------------------
-- Script to join test group
----------------------------------------------------------------
on exitFrame
  global initMembChan,Maxchannels

  repeat with i = initMembChan to MaxChannels   -- Members, Forces and Supports
    puppetSprite i, TRUE
  end repeat


  SayHowLong( "Connecting" )
  set curDate = the date
  set the itemDelimiter = "/"
  set mon = item 1 of curDate
  if ( the number of chars of mon < 2 ) then
    set mon = "0" & mon
```

```
  end if
  set day = item 2 of curDate
  if ( the number of chars of day < 2 ) then
    set day = "0" & day
  end if
  set year = "19" & item 3 of curDate
  set the text of member "DateField" to year & mon & day

  global gConnection
  -- Join the test group.  We're not checking
  -- for the proper replies, which get sent back.
  sendNetMessage( gConnection, "System", "JoinGroup", "@MultiUserGroup", 0 )
end

---------------------------------------------------------------------
-- Script to wait for server connection to be acknowledged
---------------------------------------------------------------------
on exitFrame
  global gConnection
  set curState = the connState of gConnection
  if ( curState = #connecting ) then
    -- We have a timeout of 15 seconds here
    if ( the timer > (60 * 1005) ) then
      alert "Unable to connect to the server - is the name and password correct ?"
      BreakConnection
      go "SignIn"
    else
      go to the frame
    end if
  else if ( curState = #connected ) then
    InitTypeTest
    go to "Start"
  else
    alert "Unable to connect to the server - is the name correct ?"
    BreakConnection
    go to "login"
  end if
end
```

APPENDIX B

PROGRAM LISTING FOR DIRECTOR

MULITUSER PLANT LAYOUT PLANNER

```
-------------------------------------------------------------------------------------------------------------------------------
-- Movie Script
-------------------------------------------------------------------------------------------------------------------------------
On startmovie
  --------------------------
  -- For PuppetSprite
  --------------------------
  global tableList
  tableList=[]
  techList=[]
  stoolList=[]
  cabinetList=[]
  partList=[]
  boxList=[]
  convList=[]
  smtList=[]
  latList=[]
  ovenList=[]
  robList=[]
  machList=[]
   repeat with spriteChannel = 41 to 89
    puppetSprite spriteChannel, TRUE
    set the constraint of sprite SpriteChannel=gridconstr
  end repeat
  ---------------------------------------------------------------
  -- Create new table
  ---------------------------------------------------------------
on CreateTable
  global tableList,gConnection, tableName
  if count(tableList)>=6 then
    Alert "Maximum number of table allowed!"
  else
    add(TableList,new(script"CreateTable Parent",count(tableList)+1))
  end if
end Createtable
-------------------------------------------------------------------------------------------------------------------------------
-- All Other Objects are similarly created using a handler like the CreateTable handler above
-------------------------------------------------------------------------------------------------------------------------------
end startMovie

---------------------------------------------------------------
-- SetupNetMessageHandlers - configure our handlers
---------------------------------------------------------------
on SetupNetMessageHandlers me

  set errCode = SetNetMessageHandler( conn, #DefaultMsgHandler, me )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #DefaultMsgHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #JoinGroupMsgHandler, me, "JoinGroup" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #JoinGroupMsgHandler"
    return
  end if
```

```
set errCode = SetNetMessageHandler( conn, #StringMsgHandler, me, "testStr" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #StringMsgHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #TableChildHandler, me, "TableChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #TableChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #TechChildHandler, me, "TechChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #TechChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #StoolChildHandler, me, "StoolChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #StoolChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #CabinetChildHandler, me, "CabinetChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #CabinetChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #PartChildHandler, me, "partChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #PartChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #boxChildHandler, me, "boxChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #boxChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #convChildHandler, me, "convChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #convChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #smtChildHandler, me, "SMTChild" )
if ( errCode <> 0 ) then
  alert "SetNetMessageHandler error with #smtChildHandler"
  return
end if

set errCode = SetNetMessageHandler( conn, #latheChildHandler, me, "latheChild" )
```

```
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #latheChildHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #ovenChildHandler, me, "ovenChild" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #ovenChildHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #robChildHandler, me, "robChild" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #robChildHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #machChildHandler, me, "machChild" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #machChildHandler"
    return
  end if

  set errCode = SetNetMessageHandler( conn, #PointMsgHandler, me, "testPoint" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #PointMsgHandler"
    return
  end if
  set errCode = SetNetMessageHandler( conn, #ListMsgHandler, me, "testList" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #ListMsgHandler"
    return
  end if
  set errCode = SetNetMessageHandler( conn, #DateMsgHandler, me, "testDate" )
  if ( errCode <> 0 ) then
    alert "SetNetMessageHandler error with #DateMsgHandler"
    return
  end if
end SetupNetMessageHandlers


-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on DefaultMsgHandler me
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    alert "*** Unexpected message in DefaultMsgHandler: " & newMsg
  else
    alert "*** Error in DefaultMsgHandler: " & errCode
  end if
end DefaultMsgHandler


-----------------------------------------------------------------
```

```
-- Handler to recieve net messages from server
-----------------------------------------------------------------------

on TableChildHandler me

  tableChild1=41
  tableChild2=42
  tableChild3=43
  tableChild4=44
  tableChild5=45
  tableChild6=46

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    tabChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(tableChild1).loc= getAProp(tabChildList,#loc1)
      sprite(tableChild2).loc= getAProp(tabChildList,#loc2)
      sprite(tableChild3).loc= getAProp(tabChildList,#loc3)
      sprite(tableChild4).loc= getAProp(tabChildList,#loc4)
      sprite(tableChild5).loc= getAProp(tabChildList,#loc5)
      sprite(tableChild6).loc= getAProp(tabChildList,#loc6)
    end if


  else
    alert "*** Error in ChildMsgHandler: " & newMsg
  end if
end TableChildHandler

-----------------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------------

on TechChildHandler me

  techChild1=49
  techChild2=50
  techChild3=51

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    techChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
```

```
    sprite(techChild1).loc= getAProp(techChildList,#loc1)
    sprite(techChild2).loc= getAProp(techChildList,#loc2)
    sprite(techChild3).loc= getAProp(techChildList,#loc3)
   end if


  else
   alert "*** Error in TechChildHandler: " & newMsg
  end if
end TechChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on StoolChildHandler me

  stoolChild1=52
  stoolChild2=53
  stoolChild3=54

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
   SayHowLong( "Sending a string")
   stoolChildList= getAProp(newMsg, #content)

   sender = newMsg.senderID
   myname = member("UserNameField").text

   if myname <> sender then
    sprite(stoolChild1).loc= getAProp(stoolChildList,#loc1)
    sprite(stoolChild2).loc= getAProp(stoolChildList,#loc2)
    sprite(stoolChild3).loc= getAProp(stoolChildList,#loc3)
   end if


  else
   alert "*** Error in StoolChildHandler: " & newMsg
  end if
end StoolChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on CabinetChildHandler me

  cabinetChild1=55
  cabinetChild2=56
  cabinetChild3=57

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
   SayHowLong( "Sending a string")
```

```
    cabinetChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(cabinetChild1).loc= getAProp(cabinetChildList,#loc1)
      sprite(cabinetChild2).loc= getAProp(cabinetChildList,#loc2)
      sprite(cabinetChild3).loc= getAProp(cabinetChildList,#loc3)
    end if


  else
    alert "*** Error in CabinetChildHandler: " & newMsg
  end if
end CabinetChildHandler

----------------------------------------------------------------------
-- Handler to recieve net messages from server
----------------------------------------------------------------------

on PartChildHandler me

  partChild1=58
  partChild2=59
  partChild3=60
  partChild4=61
  partChild5=62
  partChild6=63

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    partChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(partChild1).loc= getAProp(partChildList,#loc1)
      sprite(partChild2).loc= getAProp(partChildList,#loc2)
      sprite(partChild3).loc= getAProp(partChildList,#loc3)
      sprite(partChild4).loc= getAProp(partChildList,#loc4)
      sprite(partChild5).loc= getAProp(partChildList,#loc5)
      sprite(partChild6).loc= getAProp(partChildList,#loc6)
    end if


  else
    alert "*** Error in PartChildHandler: " & newMsg
  end if
end PartChildHandler

----------------------------------------------------------------------
-- Handler to recieve net messages from server
```

```
-----------------------------------------------------------------

on BoxChildHandler me

  boxChild1=64
  boxChild2=65
  boxChild3=66
  boxChild4=67
  boxChild5=68

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    boxChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(boxChild1).loc= getAProp(boxChildList,#loc1)
      sprite(boxChild2).loc= getAProp(boxChildList,#loc2)
      sprite(boxChild3).loc= getAProp(boxChildList,#loc3)
      sprite(boxChild4).loc= getAProp(boxChildList,#loc4)
      sprite(boxChild5).loc= getAProp(boxChildList,#loc5)
    end if


  else
    alert "*** Error in boxChildHandler: " & newMsg
  end if
end boxChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on convChildHandler me

  convChild1=69
  convChild2=70
  convChild3=71
  convChild4=72
  convChild5=73

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    convChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(convChild1).loc= getAProp(convChildList,#loc1)
```

```
    sprite(convChild2).loc= getAProp(convChildList,#loc2)
    sprite(convChild3).loc= getAProp(convChildList,#loc3)
    sprite(convChild4).loc= getAProp(convChildList,#loc4)
    sprite(convChild5).loc= getAProp(convChildList,#loc5)
   end if


  else
   alert "*** Error in convChildHandler: " & newMsg
  end if
end convChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on smtChildHandler me
  smtChild1=74
  smtChild2=75
  smtChild3=76
  smtChild4=77
  smtChild5=78

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
   SayHowLong( "Sending a string")
   smtChildList= getAProp(newMsg, #content)

   sender = newMsg.senderID
   myname = member("UserNameField").text

   if myname <> sender then
    sprite(smtChild1).loc= getAProp(smtChildList,#loc1)
    sprite(smtChild2).loc= getAProp(smtChildList,#loc2)
    sprite(smtChild3).loc= getAProp(smtChildList,#loc3)
    sprite(smtChild4).loc= getAProp(smtChildList,#loc4)
    sprite(smtChild5).loc= getAProp(smtChildList,#loc5)
   end if


  else
   alert "*** Error in smtChildHandler: " & newMsg
  end if
end smtChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on latheChildHandler me
  latheChild1=79
  latheChild2=80
  latheChild3=81
  latheChild4=82
  latheChild5=83
```

```
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    latheChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(latheChild1).loc= getAProp(latheChildList,#loc1)
      sprite(latheChild2).loc= getAProp(latheChildList,#loc2)
      sprite(latheChild3).loc= getAProp(latheChildList,#loc3)
      sprite(latheChild4).loc= getAProp(latheChildList,#loc4)
      sprite(latheChild5).loc= getAProp(latheChildList,#loc5)
    end if


  else
    alert "*** Error in latheChildHandler: " & newMsg
  end if
end latheChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on ovenChildHandler me
  ovenChild1=84
  ovenChild2=85

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    ovenChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(ovenChild1).loc= getAProp(ovenChildList,#loc1)
      sprite(ovenChild2).loc= getAProp(ovenChildList,#loc2)
    end if


  else
    alert "*** Error in ovenChildHandler: " & newMsg
  end if
end ovenChildHandler


-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
```

```
on robChildHandler me
  robChild1=86
  robChild2=87

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    robChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(robChild1).loc= getAProp(robChildList,#loc1)
      sprite(robChild2).loc= getAProp(robChildList,#loc2)
    end if


  else
    alert "*** Error in robChildHandler: " & newMsg
  end if
end robChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------

on machChildHandler me
  machChild1=88
  machChild2=89

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string")
    machChildList= getAProp(newMsg, #content)

    sender = newMsg.senderID
    myname = member("UserNameField").text

    if myname <> sender then
      sprite(machChild1).loc= getAProp(machChildList,#loc1)
      sprite(machChild2).loc= getAProp(machChildList,#loc2)
    end if


  else
    alert "*** Error in machChildHandler: " & newMsg
  end if
end machChildHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
```

```
on StringMsgHandler me
  global UserPathNameWRL,gConnection

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a string" )

    sender = newMsg.senderID
    myname = member("UserNameField").text
    -- "VRML" is the name of the frame
    -- load VRML file with new data

    TempText=member("MsgChannel").text
    NewTempText=getAprop(newMsg,#content)
    member("MsgChannel").text=string(TempText & return&& NewTempText)

    if (getAProp(newMsg, #content)="Reloading VRML") then
      gotoNetPage (UserPathNameWRL, "VRML")
    end if

    if myname <> sender then
      if(getAprop(newMsg,#content)="Regenerate Table Child Object") then
        CreateTable
      else
        if(getAprop(newMsg,#content)="Regenerate Tech Table Child Object") then
          CreateTech
        else
          if(getAprop(newMsg,#content)="Regenerate Stool Child Object") then
            CreateStool
          else
            if(getAprop(newMsg,#content)="Regenerate Cabinet Child Object") then
              CreateCabinet
            else
              if(getAprop(newMsg,#content)="Regenerate Partition Child Object") then
                CreatePart
              else
                if(getAprop(newMsg,#content)="Regenerate Box Child Object") then
                  Createbox
                else
                  if(getAprop(newMsg,#content)="Regenerate Conveyor Child Object") then
                    CreateConveyor
                  else
                    if(getAprop(newMsg,#content)="Regenerate SMT Child Object") then
                      CreateSMT
                    else
                      if(getAprop(newMsg,#content)="Regenerate Lathe Child Object") then
                        CreateLathe
                      else
                        if(getAprop(newMsg,#content)="Regenerate Oven Child Object") then
                          CreateOven
                        else
                          if(getAprop(newMsg,#content)="Regenerate Robot Child Object") then
                            CreateRob
                          else
                            if(getAprop(newMsg,#content)="Regenerate Machine Child Object") then
```

```
                    CreateMach
                  else
                    if (getAprop(newMsg,#content)="Deleting All Objects") then
                      global tableList,TechList,StoolList, CabinetList,PartList,boxList,convList
                      global smtList,latList,ovenList,robList,machList
                      tableList=[]
                      TechList=[]
                      StoolList=[]
                      CabinetList=[]
                      PartList=[]
                      boxList=[]
                      convList=[]
                      smtList=[]
                      latList=[]
                      ovenList=[]
                      robList=[]
                      machList=[]
                      the memberNum of sprite 12 to 57
                      repeat with spriteChannel = 41 to 46
                        the memberNum of sprite spritechannel to 57
                        sprite(spritechannel).loc = point(-50,-50)
                      end repeat

                      repeat with spriteChannel = 49 to 89
                        the memberNum of sprite spritechannel to 57
                        sprite(spritechannel).loc=point(-50,-50)
                      end repeat
                    end if
                  end if
                end if
              end if
            end if
          end if
        end if
      end if
    end if
  end if

  else
    alert "*** Error in StringMsgHandler: " & newMsg
  end if
end StringMsgHandler
-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on PointMsgHandler me

  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a point" )
```

```
    set the text of member "RecvMsg" to "Recieved point: " & getAProp(newMsg, #content)
    set the loc of sprite 12 = getAProp(newMsg, #content)
  else
    alert "*** Error in PointMsgHandler: " & newMsg
  end if
end PointMsgHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on ListMsgHandler me
  global rpoints

  rpoints=[]
  set newMsg = GetNetMessage( conn )
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a list" )
    set the text of member "RecvMsg" to "Recieved List: " & string( getAProp(newMsg,#content) )
  else
    alert "*** Error in ListMsgHandler: " & newMsg
  end if
end ListMsgHandler

-----------------------------------------------------------------
-- Handler to recieve net messages from server
-----------------------------------------------------------------
on DateMsgHandler me
  set newMsg = GetNetMessage( conn )
  put newMsg
  set errCode = getAProp( newMsg, #errorCode )
  if ( errCode = 0 ) then
    SayHowLong( "Sending a date" )
    set the text of member "RecvMsg" to "Recieved Date: " & string( getAProp(newMsg, #content)
)
  else
    alert "*** Error in DateMsgHandler: " & newMsg
  end if
end DateMsgHandler

-----------------------------------------------------------------
-- Send a message to the server
-----------------------------------------------------------------
on SendNetMessage me, recipient, subject, message, errCodeToSend
  set errCode = 5
  if ( conn <> 0 and connState = #connected ) then
    set errCode = SendNetMessage( conn, recipient, subject, message, errCodeToSend )
    if ( errCode <> 0 ) then
      alert "Error sending message: " & errCode
    end if
  end if
  return errCode
end SendNetMessage

-----------------------------------------------------------------
-- Script to grab settings file
```

```
----------------------------------------------------------------------
on exitFrame
  --  --- Get the Initial Settings text file
  global wd
  wd=[]

  theNet=getNetText("http://eml.ou.edu/multiuser/factory/temp/tableset.txt")
  if netDone(theNet) then
    set the text of member ("initSet")=netTextResult(theNet)
  else
    go to "File"
  end if
  -- Use repeat with in next version so that we can keep track at the moment
  repeat with i = 1 to 108
    wd[i]=integer(word i of the text of member("initSet"))
  end repeat
  -- 1-15 Table
  -- 16-22 Tech Table
  -- 23-29 Stool
  -- 30-36 Cabinet
  -- 37-49 Partition
  -- 50-60 Box
  -- 61-71 Conveyor
  -- 72-82 Smt
  -- 83-93 Lathe
  -- 94-98 Oven
  -- 99-103 Robot
  -- 104 108 Machine
end
----------------------------------------------------------------------
-- Script restore saved settings
----------------------------------------------------------------------
on exitFrame
  global wd


  --Table
  repeat with i=1 to wd[1]
    createTable
  end repeat

  sprite(12).loc=point(wd[2],wd[3])
  sprite(41).loc=point(wd[4],wd[5])
  sprite(42).loc=point(wd[6],wd[7])
  sprite(43).loc=point(wd[8],wd[9])
  sprite(44).loc=point(wd[10],wd[11])
  sprite(45).loc=point(wd[12],wd[13])
  sprite(46).loc=point(wd[14],wd[15])

  -- Tech Table
  repeat with i=1 to wd[16]
    createTech
  end repeat

  sprite(49).loc=point(wd[17],wd[18])
  sprite(50).loc=point(wd[19],wd[20])
```

```
sprite(51).loc=point(wd[21],wd[22])

-- Stool
repeat with i=1 to wd[23]
  createStool
end repeat

sprite(52).loc=point(wd[24],wd[25])
sprite(53).loc=point(wd[26],wd[27])
sprite(54).loc=point(wd[28],wd[29])

--Cabinet
repeat with i=1 to wd[30]
  createCabinet
end repeat

sprite(55).loc=point(wd[31],wd[32])
sprite(56).loc=point(wd[33],wd[34])
sprite(57).loc=point(wd[35],wd[36])

--Partition
repeat with i=1 to wd[37]
  createPart
end repeat

sprite(58).loc=point(wd[38],wd[39])
sprite(69).loc=point(wd[40],wd[41])
sprite(60).loc=point(wd[42],wd[43])
sprite(61).loc=point(wd[44],wd[45])
sprite(62).loc=point(wd[46],wd[47])
sprite(63).loc=point(wd[48],wd[49])

--Box
repeat with i=1 to wd[50]
  createBox
end repeat

sprite(64).loc=point(wd[51],wd[52])
sprite(65).loc=point(wd[53],wd[54])
sprite(66).loc=point(wd[55],wd[56])
sprite(67).loc=point(wd[57],wd[58])
sprite(68).loc=point(wd[59],wd[60])

--Conveyor
repeat with i=1 to wd[61]
  createConveyor
end repeat

sprite(69).loc=point(wd[62],wd[63])
sprite(70).loc=point(wd[64],wd[65])
sprite(71).loc=point(wd[66],wd[67])
sprite(72).loc=point(wd[68],wd[69])
sprite(73).loc=point(wd[70],wd[71])

--Smt
repeat with i=1 to wd[72]
```

```
    createSmt
  end repeat

  sprite(74).loc=point(wd[73],wd[74])
  sprite(75).loc=point(wd[75],wd[76])
  sprite(76).loc=point(wd[77],wd[78])
  sprite(77).loc=point(wd[79],wd[80])
  sprite(78).loc=point(wd[81],wd[82])

  --Lathe
  repeat with i=1 to wd[83]
    createLathe
  end repeat

  sprite(79).loc=point(wd[84],wd[85])
  sprite(80).loc=point(wd[86],wd[87])
  sprite(81).loc=point(wd[88],wd[89])
  sprite(82).loc=point(wd[90],wd[91])
  sprite(83).loc=point(wd[92],wd[93])

  --Oven
  repeat with i=1 to wd[94]
    createOven
  end repeat

  sprite(84).loc=point(wd[95],wd[96])
  sprite(85).loc=point(wd[97],wd[98])

  --Robot
  repeat with i=1 to wd[99]
    createRob
  end repeat

  sprite(86).loc=point(wd[100],wd[101])
  sprite(87).loc=point(wd[102],wd[103])

  --Mach
  repeat with i=1 to wd[104]
    createMach
  end repeat

  sprite(88).loc=point(wd[105],wd[106])
  sprite(89).loc=point(wd[107],wd[108])
------------------------------------------------------
-- Monitor text input and send a message
------------------------------------------------------
on keyUp me
  global gConnection
  myMember = sprite( the spriteNum of me ).member
  if ( the key = RETURN ) or ( the key = ENTER ) then
    if ( gConnection <> 0 ) then
      msgText = myMember.text
      myMember.text = ""
      if ( msgText <> 0 ) then
        txtLength = the length of msgText
        if ( char txtLength of msgText = RETURN or char txtLength of msgText = ENTER ) then
```

```
        msgText = char 1 to (txtLength - 1) of msgText
        msgText=the text of cast "UserNameField"&&">"&&msgText
      end if

      if ( gConnection <> 0 and the length of msgText > 0 ) then
        sendNetMessage( gConnection, "@TypeTestGroup", "testStr", msgText, 0 )
      end if

    end if
    updatestage
  end if
 end if
end

-----------------------------------------------------------------------------------------------------------------------
-- Location Script
-----------------------------------------------------------------------------------------------------------------------
on mouseWithin me
member("LeftRight").text = string((sprite(the spriteNum of me).right - sprite(11).locH)/10)
 member("FrontBack").text = string((sprite(the spriteNum of me).bottom - sprite(11).locV)/10)
end mouseWithin
-----------------------------------------------------------------------------------------------------------------------
-- Parent Script to create Child Instances
-----------------------------------------------------------------------------------------------------------------------

property tableLoc,Mysprite, table

On new me, tableListPos
 global tableSpr1 ,tableName, gConnection,TableChd
 tableSpr1=32
 set myListPos to tableListPos
 set mysprite to myListPos +40
 set the memberNum of sprite mysprite to tableSpr1
 set the width of sprite mysprite to 18
 set the height of sprite mysprite to 12
 sprite(mysprite).locH=50
 sprite(mysprite).locV=50
 sprite(mysprite).moveableSprite=TRUE
 tableNum=string(myListPos)
 tableName=string("tablechild")&&tableNum
 TableChd=string(tableName)
 return me
end

On MouseUp
 global gconnection,menuflag

 if (menuflag=1) then
   CreateTable
   regenTab="Regenerate Table Child Object"
   sendNetMessage( gConnection, "@TypeTestGroup", "testStr", regenTab, 0 )
 end if

 if (menuflag=2) then
   CreateMach
   regenTab="Regenerate Machine Child Object"
```

```
  sendNetMessage( gConnection, "@TypeTestGroup", "testStr", regenTab, 0 )
 end if
 end
-----------------------------------------------------------------
-- Script to send an point value
-----------------------------------------------------------------
on mouseUp me
 global gConnection,tableChildProp
 global tableList
 tabNum= count(tableList)

 if tabNum>=1 then
  set x1 = sprite(41).locH
  set y1 = sprite(41).locV
  set pt1 = point (x1,y1)
 else
  pt1=point(-50,-50)
  member("ErrorMsg").text = "Error In Table Child Numbering Script"
 end if

 if tabNum>=2 then
  set x2 = sprite(42).locH
  set y2= sprite(42).locV
  set pt2= point (x2,y2)
 else
  set pt2=point(-50,-50)
 end if

 if tabNum>=3 then
  set x3 = sprite(43).locH
  set y3 = sprite(43).locV
  set pt3 = point (x3,y3)
 else
  set pt3=point(-50,-50)
 end if

 if tabNum>=4 then
  set x4 = sprite(44).locH
  set y4 = sprite(44).locV
  set pt4 = point (x4,y4)
 else
  set pt4=point(-50,-50)
 end if

 if tabNum>=5 then
  set x5 = sprite(45).locH
  set y5 = sprite(45).locV
  set pt5 = point (x5,y5)
 else
  set pt5=point(-50,-50)
 end if

 if tabNum=6 then
  set x6 = sprite(46).locH
  set y6 = sprite(46).locV
  set pt6 = point (x6,y6)
```

```
  else
    set pt6=point(-50,-50)
  end if

  set tableChildProp = [#name:"TableChild", #tableNum:tabNum ,#loc1:pt1, #loc2:pt2,
¬#loc3:pt3,#loc4:pt4,#loc5:pt5 ,#loc6:pt6]
  sendNetMessage( gConnection, "@TypeTestGroup", "TableChild", tableChildProp, 0 )
end
-----------------------------------------------------------------------------------------------------------------------
-- All Other Objects have the similar the above three scripts
-----------------------------------------------------------------------------------------------------------------------


-----------------------------------------------------------------------------------------------------------------------
-- Script for Message Bar
-----------------------------------------------------------------------------------------------------------------------
on exitFrame
    -- On Off Menu Selections Arrows
  --Menu Flag =1 Furniture Menu
  --Menu Flag =2 Machines Menu
  global MenuFlag , FwdArrowSpr, BckArrowSpr, MsgFlag1, MsgFlag2, Msgflag3

  if MenuFlag=1 then                        --Off back arrow
    puppetSprite 20, TRUE                    --PuppetSprite Menu Channel
    sprite(BckArrowSpr).visible = FALSE
    sprite(FwdArrowSpr).visible = TRUE
    set the memberNum of sprite 20 to 22
    -- Change Menu sprite when a button an icon is rolled-over
    MsgFlag1=0
    case the rollover of

      27: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 23
        member("ErrorMsg").text=string("Assembly Table")
        updateStage
      28: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 24
        member("ErrorMsg").text=string("Technician Table")
        updateStage
      29: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 25
        member("ErrorMsg").text=string("Stool")
        updateStage
      30: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 26
        member("ErrorMsg").text=string("Cabinet")
        updateStage
      31: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 27
        member("ErrorMsg").text=string("Partition")
        updateStage
      32: puppetsprite 20, TRUE
        set the memberNum of sprite 20 to 28
        member("ErrorMsg").text=string("Box")
        updateStage
      otherwise:  MsgFlag1=1
    end case
```

```
end if

if MenuFlag=2 then                    -- Off fwd arrow
  puppetSprite 20, TRUE
  sprite(BckArrowSpr).visible = TRUE
  sprite(FwdArrowSpr).visible=FALSE
  set the memberNum of sprite 20 to 29
  MsgFlag2=0
  case the rollover of

    27: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 30
      member("ErrorMsg").text=string("Machine")
      updateStage
    28: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 31
      member("ErrorMsg").text=string("Conveyor Line")
      updateStage
    29: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 32
      member("ErrorMsg").text=string("SMT Chip Mounter")
      updateStage
    30: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 33
      member("ErrorMsg").text=string("Lathe")
      updateStage
    31: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 34
      member("ErrorMsg").text=string("Testing Oven")
      updateStage
    32: puppetsprite 20, TRUE
      set the memberNum of sprite 20 to 35
      member("ErrorMsg").text=string("Robotic Arm")
      updateStage
    otherwise:  MsgFlag2=1
  end case
end if


MsgFlag3=0
case the rollover of

  33: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 37
    member("ErrorMsg").text=string("Help")
    updateStage
  34: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 38
    member("ErrorMsg").text=string("Change Dimension of Factory Floor")
    updateStage
  35: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 39
    member("ErrorMsg").text=string("Change Object Properties")
    updateStage
  36: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 40
```

```
    member("ErrorMsg").text=string("Add VRML Object")
    updateStage
  37: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 41
    member("ErrorMsg").text=string("Credits and Refresh HTML")
    updateStage
  38: puppetsprite 20, TRUE
    set the memberNum of sprite 21 to 36
    member("ErrorMsg").text=string("QUIT")
    updateStage
  otherwise : set the memberNum of sprite 21 to 36
    MsgFlag3=1
end case

-- Error Messages
global tableSpr,msgfieldSpr,submitSpr

tableSpr=12
msgfieldSpr=4
submitBtnSpr= 14
addBtnSpr=28
avdBtnSpr=19

case the rollover of

  tableSpr : member("ErrorMsg").text=string("Table")
  41: member("ErrorMsg").text=string("Table1")
  42: member("ErrorMsg").text=string("Table2")
  43: member("ErrorMsg").text=string("Table3")
  44: member("ErrorMsg").text=string("Table4")
  45: member("ErrorMsg").text=string("Table5")
  46: member("ErrorMsg").text=string("Table6")
  49: member("ErrorMsg").text=string("Tech Table 1")
  50: member("ErrorMsg").text=string("Tech Table 2")
  51: member("ErrorMsg").text=string("Tech Table 3")
  52: member("ErrorMsg").text=string("Stool1")
  53: member("ErrorMsg").text=string("Stool2")
  54: member("ErrorMsg").text=string("Stool3")
  55: member("ErrorMsg").text=string("Cabinet1")
  56: member("ErrorMsg").text=string("Cabinet2")
  57: member("ErrorMsg").text=string("Cabinet3")
  58: member("ErrorMsg").text=string("Partition1")
  59: member("ErrorMsg").text=string("Partition2")
  60: member("ErrorMsg").text=string("Partition3")
  61: member("ErrorMsg").text=string("Partition4")
  62: member("ErrorMsg").text=string("Partition5")
  63: member("ErrorMsg").text=string("Partition6")
  64: member("ErrorMsg").text=string("Box1")
  65: member("ErrorMsg").text=string("Box2")
  66: member("ErrorMsg").text=string("Box3")
  67: member("ErrorMsg").text=string("Box4")
  68: member("ErrorMsg").text=string("Box5")
  69: member("ErrorMsg").text=string("Conveyor1")
  70: member("ErrorMsg").text=string("Conveyor2")
  71: member("ErrorMsg").text=string("Conveyor3")
  72: member("ErrorMsg").text=string("Conveyor4")
```

```
   73: member("ErrorMsg").text=string("Conveyor5")
   74: member("ErrorMsg").text=string("Smt & Conveyor1")
   75: member("ErrorMsg").text=string("Smt & Conveyor2")
   76: member("ErrorMsg").text=string("Smt & Conveyor3")
   77: member("ErrorMsg").text=string("Smt & Conveyor4")
   78: member("ErrorMsg").text=string("Smt & Conveyor5")
   79: member("ErrorMsg").text=string("Lathe1")
   80: member("ErrorMsg").text=string("Lathe2")
   81: member("ErrorMsg").text=string("Lathe3")
   82: member("ErrorMsg").text=string("Lathe4")
   83: member("ErrorMsg").text=string("Lathe5")
   84: member("ErrorMsg").text=string("Oven1")
   85: member("ErrorMsg").text=string("Oven2")
   86: member("ErrorMsg").text=string("Robot1")
   87: member("ErrorMsg").text=string("Robot2")
   88: member("ErrorMsg").text=string("Machine1")
   89: member("ErrorMsg").text=string("Machine2")
   submitBtnSpr : member("ErrorMsg").text=string("Submit / Refresh VRML Button")
   msgfieldSpr :member("ErrorMsg").text=string("Type in your message here")
   1:member("ErrorMsg").text=string("Factory Layout Planner")
   2:member("ErrorMsg").text=string("Factory Layout Planner")
   3:member("ErrorMsg").text=string("Factory Layout Planner")
   5:member("ErrorMsg").text=string("Factory Layout Planner")
   6:member("ErrorMsg").text=string("Factory Layout Planner")
   10:member("ErrorMsg").text=string("Factory Layout Planner")
   11:member("ErrorMsg").text=string("Factory Layout Planner")
   13:member("ErrorMsg").text=string("Factory Layout Planner")
   otherwise :member("LeftRight").text = string("--")
     member("FrontBack").text = string("--")
     dummyflag
  end case

  updateStage
  go to the Frame
end exitframe

-- Bluff :if only one msgFlag=1 the dont print else print string
on dummyflag
  global MsgFlag1, Msgflag2, Msgflag3

  if (MsgFlag1+MsgFlag2+MsgFlag3=3)then
    member("ErrorMsg").text=string("Factory Layout Planner")
    MsgFlag=0
  end if
 end

-------------------------------------------------------------------------------------------------------------------------------
-- Script to Post Data to PERL Script and Also Refresh VRML page
-------------------------------------------------------------------------------------------------------------------------------
on mouseUp
  global gConnection,tableList,TechList,StoolList, CabinetList,PartList
  global boxList,convList,smtList,latList,ovenList,robList,machList

  DName = member("DirName").text
  BldgLR =((sprite(12).locH - sprite(11).locH)/10.0)+1.0
  BldgFB =((sprite(12).locV - sprite(11).locV)/10.0)+1.0
```

```
  x1=((sprite(41).locH- sprite(11).locH)/10.0)+1.0
  y1=((sprite(41).locV- sprite(11).locV)/10.0)+1.0
  x2=((sprite(42).locH- sprite(11).locH)/10.0)+1.0
  y2=((sprite(42).locV- sprite(11).locV)/10.0)+1.0
  x3=((sprite(43).locH- sprite(11).locH)/10.0)+1.0
  y3=((sprite(43).locV- sprite(11).locV)/10.0)+1.0
  x4=((sprite(44).locH- sprite(11).locH)/10.0)+1.0
  y4=((sprite(44).locV- sprite(11).locV)/10.0)+1.0
  x5=((sprite(45).locH- sprite(11).locH)/10.0)+1.0
  y5=((sprite(45).locV- sprite(11).locV)/10.0)+1.0
  x6=((sprite(46).locH- sprite(11).locH)/10.0)+1.0
  y6=((sprite(46).locV- sprite(11).locV)/10.0)+1.0
          .
          .
          .
   -- Robot
  x44=((sprite(86).locH- sprite(11).locH)/10.0)+1.0
  y44=((sprite(86).locV- sprite(11).locV)/10.0)+1.0
  x45=((sprite(87).locH- sprite(11).locH)/10.0)+1.0
  y45=((sprite(87).locV- sprite(11).locV)/10.0)+1.0

  -- Mach
  x46=((sprite(88).locH- sprite(11).locH)/10.0)+1.0
  y46=((sprite(88).locV- sprite(11).locV)/10.0)+1.0
  x47=((sprite(89).locH- sprite(11).locH)/10.0)+1.0
  y47=((sprite(89).locV- sprite(11).locV)/10.0)+1.0

  tabNum= count(tableList)
  techNum=count(TechList)
  stoolNum=count(StoolList)
  cabNum=count(CabinetList)
  partNum=count(partList)
  boxNum=count(boxList)
  convNum=count(convList)
  smtNum=count(smtList)
  latNum=count(latList)
  ovenNum=count(ovenList)
  robotNum=count(robList)
  machNum=count(machList)

  infoList = ["DirName":DName,"BldgLR":BldgLR, "BldgFB":BldgFB¬
    ,"tab1x":x1,"tab1y":y1,"tab2x":x2,"tab2y":y2,"tab3x":x3,"tab3y":y3¬
    ,"tab4x":x4,"tab4y":y4,"tab5x":x5,"tab5y":y5,"tab6x":x6,"tab6y":y6,"tabNum":tabNum¬
    ,"tech1x":x7,"tech1x":y7,"tech2x":x8,"tech2y":y8,"tech3x":x9,"tech3y":y9, "techNum":techNum¬

,"stool1x":x10,"stool1y":y10,"stool2x":x11,"stool2y":y11,"stool3x":x12,"stool3y":y12,"StoolNum":st
oolNum¬

,"cab1x":x13,"cab1y":y13,"cab2x":x14,"cab2y":y14,"cab3x":x15,"cab3y":y15,"cabNum":cabNum¬

,"part1x":x16,"part1y":y16,"part2x":x17,"part2y":y17,"part3x":x18,"part3y":y18,"part4x":x19,"part4y
":y19¬
    ,"part5x":x20,"part5y":y20,"part6x":x21,"part6y":y21,"partNum":partNum¬
    ,"box1x":x22,"box1y":y22,"box2x":x23,"box2y":y23,"box3x":x24,"box3y":y24¬
    ,"box4x":x25,"box4y":y25,"box5x":x26,"box5y":y26,"boxNum":boxNum¬
    ,"conv1x":x27,"conv1y":y27,"conv2x":x28,"conv2y":y28,"conv3x":x29,"conv3y":y29¬
```

```
  ,"conv4x":x30,"conv4y":y30,"conv5x":x31,"conv5y":y31,"convNum":convNum¬
  ,"smt1x":x32,"smt1y":y32,"smt2x":x33,"smt2y":y33,"smt3x":x34,"smt3y":y34¬
  ,"smt4x":x35,"smt4y":y35,"smt5x":x36,"smt5y":y36,"smtNum":smtNum¬
  ,"lat1x":x37,"lat1y":y37,"lat2x":x38,"lat2y":y38,"lat3x":x39,"lat3y":y39¬
  ,"lat4x":x40,"lat4y":y40,"lat5x":x41,"lat5y":y41,"latNum":latNum¬
  ,"oven1x":x42,"oven1y":y42,"oven2x":x43,"oven2y":y43,"ovenNum":ovenNum¬
  ,"rob1x":x44,"rob1y":y44,"rob2x":x45,"rob2y":y45,"robNum":robotNum¬
  ,"mach1x":x46,"mach1y":y46,"mach2x":x47,"mach2y":y47,"machNum":machNum]

 -- Save changed data to file on server, then load
 netID = postNetText ("http://eml.ou.edu/multiuser/factory/cgi-bin/table.pl", infoList, "Win")
 set tempStr = "Reloading VRML"
 startTimer
 sendNetMessage( gConnection, "@TypeTestGroup", "testStr", tempStr, 0 )

end
```

APPENDIX C

PERL PROGRAM LISTING FOR

MULTIUSER PLANT LAYOUT PLANNER

```
#################################################################################
## PERL Script to  Generate VRML File and Settings File
#################################################################################
# table.pl
use CGI qw/:standard/;
#################################################
## Use loop to read in variables instead later
#################################################
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});

# split name and value into pairs
@nvpairs = split(/&/, $buffer);

# single > writes file (even if existing)  >> appends file
$pair1 = @nvpairs[0];
($Dname, $DirVolName) = split(/=/, $pair1);

# place default left and right
$pair2 = @nvpairs[1];
($tab0xname, $tabLR) = split(/=/, $pair2);
# place default table up and back
$pair3 = @nvpairs[2];
($tab0yname, $tabFB) = split(/=/, $pair3);
#table 1 x pos
$pair4 = @nvpairs[3];
($tab1xname,$tab1x) = split(/=/,$pair4);
#table 1 y pos
          .
          .
          .
#TabNum
$pair16 = @nvpairs[15];
($tabNumName,$tabNum) = split(/=/,$pair16);
          .
          .
          .
#machNum pos
$pair109 = @nvpairs[108];
($machNumname,$machNum) = split(/=/,$pair109);

$FullPathWrl = ">multiuser/factory/temp/facfloor.wrl";
open(INFO, $FullPathWrl);

print INFO <<endfile;
#VRML V2.0 utf8

#Cosmo Worlds V2.0

DEF navicontrols NavigationInfo {
  avatarSize      [ 0.25, 1.6, 0.75 ]
  headlight       TRUE
```

```
  speed 1
  type    [ "FLY",
              "WALK" ]
  visibilityLimit    0
}
Background {
  groundAngle    1.57079
  groundColor    [ 1 0.8 0.6,
              0.6 0.4 0.2 ]
  skyAngle       [ 0.2, 1.57079 ]
  skyColor       [ 1 1 0,
              1 1 1,
              0.2 0.2 1 ]
}
DEF Perspective Viewpoint {
  position       33.054 10.831 28.985
  orientation    -0.502 0.836 0.224  0.889
  fieldOfView    0.785
  description    "Perspective View"
}
DEF top Viewpoint {
  position       14.301 29.154 9.021
  orientation    -1 0 0  1.571
  fieldOfView    0.786
  description    "Topview"
}
DEF side Viewpoint {
  position       -4.502 1.893 7.271
  orientation    0.057 0.997 0.051  4.641
  fieldOfView    0.785
  jump   TRUE
  description    "Sideview"
}

DEF factory Transform {
  children       DEF factfloor Transform {
    children     Shape {
      appearance Appearance {
            materialMaterial {
              ambientIntensity          0
              diffuseColor     0 0.8 0.8
              specularColor  0.5 0.5 0.5
              emissiveColor 0 0.15 0.15
              shininess        0.2
              transparency   0
            }

      }

      geometry     DEF _0 IndexedFaceSet {
            coord    Coordinate {
              point  [ -1 -1 0,
                       -1 1 0,
                       1 1 0,
                       1 -1 0 ]
            }
```

```
          coordIndex       [ 0, 1, 2, 3, -1 ]
          solid     FALSE
          normalIndex      [ ]
          texCoordIndex  [ ]
     }

    }

    translation      13 -1.46488e-005 8.99998
    rotation         1 1.76536e-008 2.44939e-008  1.57079
    scale 15 11 1.14843
    scaleOrientation         0 0 1  0
  }

}
DEF table0 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation       $tabLR 0 $tabFB
}

endfile

$abc1=1;
$abc2=2;
$abc3=3;
$abc4=4;
$abc5=5;
$abc6=6;

if ($tabNum>=$abc1){
print INFO <<table1seg;
DEF table1 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab1x 0 $tab1y
 }
table1seg
}

if ($tabNum>=$abc2){
print INFO <<table2seg;
DEF table2 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab2x 0 $tab2y
 }
table2seg
}
```

```
if ($tabNum>=$abc3){
print INFO <<table3seg;
DEF table3 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab3x 0 $tab3y
 }
table3seg
}

if ($tabNum>=$abc4){
print INFO <<table4seg;
DEF table4 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab4x 0 $tab4y
 }
table4seg
}

if ($tabNum>=$abc5){
print INFO <<table5seg;
DEF table5 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab5x 0 $tab5y
 }
table5seg
}

if ($tabNum>=$abc6){
print INFO <<table6seg;
DEF table6 Transform {
  children          Inline {
    url     "table.wrl"
  }

  translation $tab6x 0 $tab6y
 }
table6seg
}
            .
            .
            .

if ($machNum>=$abc1){
print INFO <<mach;
DEF mach1 Transform {
  children          Inline {
```

```
    url     "packing.wrl"
  }

  translation $mach1x 0.2 $mach1y
 }
mach
}

if ($machNum>=$abc2){
print INFO <<mach;
DEF mach2 Transform {
  children          Inline {
    url     "packing.wrl"
  }
  translation $mach2x 0.2 $mach2y
 }
mach
}
close(INFO);
$FilePath = ">multiuser/factory/temp/tableset.txt";
open(DATA, $FilePath);
print DATA <<endfile;
$tabNum
$ptab0x $ptab0y
$ptab1x $ptab1y
$ptab2x $ptab2y
$ptab3x $ptab3y
$ptab4x $ptab4y
$ptab5x $ptab5y
$ptab6x $ptab6y
            .
            .
            .
$machNum
$pmach1x $pmach1y
$pmach2x $pmach2y
endfile
close(DATA);
```