THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

INTERNET-BASED REAL TIME COLLABORATION TECHNIQUES FOR UNDERSTANDING 3D ENGINEERING CONCEPTS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

JACKY CHU

Norman, Oklahoma

2006
Internet-based real time collaboration techniques for understanding
3D engineering concepts

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. Kurt Gramoll, Chair

Dr. Jerry E. Crain, Chair

Dr. James Sluss
Acknowledgement

I would like to take this opportunity to express my deepest gratitude to my families and friends who have been supporting me in the past six years in the United States. First of all, I am very thankful that God has blessed me with such a wonderful family, who love and support me no matter what situation is. My mother is a hardworking individual, who works 7 days a week in order to support my education. Because of her, I can be the person I am today. Without her, I would not have the opportunity to study in the States. I thank my father who gives me direction and guidance while I was growing up. He did not have the opportunity to study in his childhood. He understands how important education is to his children. My father worked hard to support what we need in our education. To my younger brother and sister who respect and encourage me in every decision I made. I am very grateful to my family.

Secondly, I would like to thank the committee members for devoting their time and energy in this research. A special thanks to Dr. Gramoll who has given me the opportunity to work in this interesting project. Because of his expertise, he helped me from the beginning to the end of this research. I am very thankful to have Dr. Crain and Dr. Sluss participated in this project. Thank you for all their guidance and supports throughout this research.

Thirdly, there is one person in my life I will never know how to thank. She is always there for me whenever I need encouragement. A deepest appreciation to my
fiancée Jennifer who always gives me encouragements. Thank you for her constant patiently support in my study and career. Thank you for loving me as who I am.

At last, I would like to thank all of my friends I have met in the States. Without them, my life in Norman would be boring and meaningless. They also helped me to become a better person. Thank you to my friends who are being part of my life at University of Oklahoma.
Table of Contents

Acknowledgement ........................................................................................................ IV
List of tables and listings ........................................................................................... IX
List of Illustrations........................................................................................................ X
Abstract ..................................................................................................................... XII

Chapter One: Online Learning .................................................................................. 1

1.1 Server and client online concept ..................................................................... 2
1.2 3D Application Developments ..................................................................... 4
1.3 Internet Collaboration ................................................................................... 5
1.4 Research Objectives ....................................................................................... 7

Chapter Two: Literature Review .............................................................................. 10

2.1 Existing 3D Learning ..................................................................................... 10
2.1.1 Three-Dimensional Visualization ......................................................... 11
2.1.2 Three-Dimensional Simulation Model ................................................. 13
2.2 Authoring Tools ............................................................................................ 14
2.2.1 Macromedia Director and 3D Shockwave ......................................... 15
2.2.2 Macromedia Flash Communication Server MX ................................ 16
2.3 Implementation of Director and Flash Communication Server MX ............ 18

Chapter Three: Background Network Techniques ................................................. 19

3.1 Flash Communication Server ........................................................................ 19
3.1.1 NetConnection ..................................................................................... 19
3.1.2 Shared Object ....................................................................................... 20
3.1.3 NetStream Object ................................................................................ 23
7.2 Recommendations for future researches .................................................62
References ........................................................................................................64
Appendices .......................................................................................................66
Appendix A: Server Side Action Script Code (main.asc) .........................66
Appendix B: Set up NetConnection in Director .......................................70
Appendix C: Setup NetStream in Director ...............................................72
Appendix D: Microphone Pool configuration .......................................74
Appendix E: 3D World Background .......................................................78
List of tables and listings

Table 4.1: Bandwidth demands for number of client connections 44

Listing 4.1: Netconnection setup code 31

Listing 4.2: The onStatus handler 32

Listing 4.3: The context of onSetupSharedObject handler 34

Listing 4.4: The call back handler onSync 35

Listing 4.5: The update of shared object handler BallSO 38

Listing 4.6: The context of handler NetStreamConnection 41

Listing 4.7: The initialization of Netstreams 43
List of Illustrations

Figure 1.1: Client side and server side diagram 2
Figure 1.2: Screen Shots of eCourses.ou.edu 3
Figure 2.1: Screen Shots of 3D Visualization from the 3D Dynamics System 12
Figure 2.2: 3D Plumber Helper using 3D Shockware technology 13
Figure 2.3: Flash communication server and its clients 17
Figure 3.1: NetConnection and data synchronization. 21
Figure 3.2: Flash server and clients sharing information using shared objects 21
Figure 3.3: Synchronization steps using shared objects 22
Figure 3.4: Block Diagram of NetStream objects 24
Figure 3.5: The architecture of collaboration 29
Figure 4.1: Flash Communication Server Application Inspector 34
Figure 4.2: Information about shared objects in Application Inspector 34
Figure 4.3: Two basic 3D objects in a typical 3D world 38
Figure 4.4: Example of 3D objects being collaborated in two Internet browsers 38
Figure 4.5: Property Inspector of a Flash movie object used in Director 43
Figure 4.6: Screen shot of microphone pool system 46
Figure 4.7: Block diagram of microphone pool system 47
Figure 4.8: Logic diagram of model control setting 48
Figure 4.9: The screen shots of control buttons 49
Figure 5.1: Main elements of the 3D collaboration modules 53
Figure 5.2: Picture of 3D Moments on Bracket 56
Figure 5.3: Picture of Electric Generator 57
Figure 6.1: Pictures of evaluation capture by screen capture software MORAE
Abstract

Online learning has become a common medium in education. Numerous companies and universities have adopted distance learning in their institutions. Because of its flexibilities, online learning can provide good quality education without being limited by location and time. Collaboration, an efficient learning technique, has recently increased research interests. Also, 3D problems are common in engineering courses. But traditional learning tools, such as books and lectures have a difficult time teaching 3D concepts. Therefore developing 3D learning tools is needed for engineering students. Also, 3D collaborative environments can enhance distance learning in engineering education. The main objective of this research is to develop a 3D real time collaboration learning environment tool for engineering student.

In this thesis, 3D collaboration learning environments for the students are developed. Various papers and references are reviewed in order to understand the current use of collaboration and in 3D learning environments. Because of the lack of integrated 3D collaboration tools and implementation resources, the implementation of the tools to create 3D collaboration was the most challenging part in this study. Two software development tools, Director and Flash Communication Server (FlashCom), are used to develop prototype modules. Also, this research explores different possible usages of these tools to enhance better learning environments such as better manipulation and communication.
Three, 3D collaboration learning tools related to Mechanical and Electrical Engineering were built using Director and FlashCom. The user interface and the structure of these modules are the same. All modules have an interactive 3D world, which enables students to share and manipulate 3D objects in real time. A customized microphone system is developed in each module to allow a group of students to communicate with one another. This microphone system also allows the instructor conduct an online lecture. Each module has input and output windows so that students can understand different calculations while they are manipulating the 3D objects. Different other technologies and methodologies about constructing the 3D learning modules are also presented in this thesis.

To ensure the user reliability of the 3D modules developed in this research, evaluation of the 3D modules was conducted by a group of engineering students from University of Missouri (UMR). In general, the feedbacks of the modules were positive. The students believed the 3D collaborative environments could help students understand engineering concepts better. Detailed evaluation results are presented in this these as well.
Chapter One: Online Learning

The Internet has not only become an important information source, but has also become a main medium for education over the past decade. One aspect of using the Internet for education is online learning. Internet-based learning has grown tremendously in the past few years [1] in addition to the numerous web sites that have been developed to present research materials. Since online materials are available 24/7 without the constraints of location, researchers and students are able to learn whenever they want. Online learning is not only beneficial in academic oriented environments but also in industry. When the population of an organization increases, the training needs of employees also increase as well. Online training is able to reduce the cost of education without affecting the quality of training [2]. Because of the flexibilities and advantages of online learning, many universities and companies now offer a large variety of online courses.

In order to enhance online learning, researchers are developing different kind of new programs and technologies to improve the learning environment. Collaboration over the Internet has been used in different areas, such as education and commercial [3]. Online conferences and remote teaching are allowing organizations better communication without being constricted by distance. Users can now interact with other people and able to gain new knowledge at the same time. Many technologies such as video, audio, simulation and animation have
been developed to improve online learning. Basic server and client concepts are discussed in the next section.

1.1 Server and client online concept

Most previous online courses and remote learning environments used the HTML approach. Nowadays, many websites have changed to dynamic and interactive pages from plain text website using different multimedia plug-ins such as the Flash player and Shockwave player. If the HTML requires a huge amount of data or information from database, scripting language like ASP, PHP and PERL are needed. The applications like HTML and scripting languages required a server running 24/7 to serve people in different parts of the world. Figure 1.1 is a basic diagram of server side and client side connected with the Internet [4].

Figure 1.1: Client side and server side diagram.
There are many advantages and flexibilities that can benefit students when they are using online learning. Online learning offers different tools and materials that help students learn better in education. For example, Engineering Media Lab (EML) at the University of Oklahoma has been developing different online learning systems which help engineering students learn mechanical engineering concepts [5, 6]. One of the e-learning courses called eCourses was built to provide basic engineering classes such as static, dynamics, mechanics, thermodynamics, fluids, and Math classes. The eCourses websites have different kinds of materials in order to provide a better learning environment to students. Those materials included eBook, homework, quizzes, solutions, and movie lectures. Figure 1.2 shows two screen shots of the eCourses.ou.edu at the University of Oklahoma.

![Figure 1.2: Screen shots of eCourses.ou.edu](image)

The eCourses website has a collaborative drawing tool developed to assist both instructors and students in working together during discussion groups, office hours, and lectures to solve problems from a distance [5]. Online courses with
collaboration programs provide flexibilities to both students and instructors. This allows students to ask questions during office hours without physically being in the office. It saves time for both instructors and students without being restricted to a specific location and or time.

1.2 3D Application Developments

In order to learn engineering theory more efficiently, simulation programs are particularly helpful. On the market today, there are many commercial simulation programs that are designed for learning engineering concepts. Programs like Pro E, Labview, and P-Spice and so on. Those programs are used not only in industry, but also in education as well. Researchers use simulation programs to provide both graphic and data output. Similarly these types of programs can also increase students understanding of concepts more easily than just reading a book. Normally engineering simulation programs incorporate elements like 3D graphics, animations and simulation. Even though some of those programs have 3D graphics, they are not able to imitate the actual model closely. Therefore, many 3D supported software have been developed to fulfill the user’s needs. Adobe, a well known graphic company, has introduced a 3D supported software called Dubbed Acrobat 3D. Companies have long struggled with finding simple ways to share elements of their designs while protecting the intellectual property used in the products and making those documents detailed enough to supply adequate levels of information for testing [7]. Dubbed Acrobat 3D has features like loading 3-D images into PDF documents, viewing and editing
images in any Acrobat reader, rotating and magnifying and cross-section image, and isolating specific components within an images etc. In order to have better communication in term of reading, 3D environments is an important technique. Microsoft, a large software company, has unveiled the first pieces of its “Flash Killer” called Microsoft’s Sparkle on Jan 24th 2006 [8]. In order to compete with Flash, Sparkle has features that support native 3D, C# and Visual Basic. The 3D technologies continue to develop and grow, indicating that they will be even more important in the future.

In the past, online learning websites rarely used 3D environments. Now, more companies are using different 3D technologies like VRML, Java 3D, and 3D Director since web-based learning tends to be in 3D mode today. The 3D Director has a strong scripting language in building 3D objects dynamically. It also can be implemented with FlashCom Server MX for collaboration purpose. Therefore it is the first choice for developing 3D learning modules in this study. In the following sections more background information about Internet collaboration will be presented.

1.3 Internet Collaboration

Internet collaboration was commercially introduced about ten years ago. One of the major factors that has helped its rapid use is that 87% of U.S workers have broadband access [9]. Since then many companies have been using collaboration for commercially use. The collaboration horizon did not stop at
companies, but also expanded to include the education section. Many researchers are investigating possible uses for collaboration to enhance the online learning methods [10, 11]. In those researches, different techniques like Shockwave and 2D drawing board were studied. Researchers are developing and investigating the potential use in collaboration in order to improve the environment of online learning.

The collaboration technology today not only applies in audio but also applies both video and visualization of the subject. The use of the Internet collaboration can make educational learning more flexible and better. For example, students and instructors can discuss problems online. An online group discussion can take place anywhere as long as an Internet connection is available. Also, collaboration discussion can be recorded for later use. In order to have better online learning environment, collaboration is one element that should not be neglected.

New techniques like collaboration can improve the learning environment online. Collaboration programs can enhance the interaction between students and instructors. In engineering, there are many topics that involve 3D environments. In today's market, few 3D collaboration tools are found in the education area. Therefore, developing 3D collaboration program is needed to satisfy the needs of instructors and students. In this thesis, literature review, techniques, theory, and testing evaluation of 3D real-time collaboration modules will be presented.

The following section discusses the objectives of this study, different goals and tasks.
1.4 Research Objectives

There are three main objectives in this research to develop different Internet-based collaboration modules in 3D environment. In order to achieve the goals, two popular software Director and Flash Communication Server MX (FlashCom) will be implemented together to develop these modules. The first objective is to test and code the implementation between these programs. Issues addressed include creating shared objects, how to setup network connections between users and how to setup a login system. This latter objective is the most difficult due to the lack of information about the implementation of Director and FlashCom. There is only an unpublished chapter of a book, Director for Multi-user Gaming Environments, by Jesse R. Warden [12] available at this time.

The second objective is to create a better 3D learning environment in real time for students. 3D Director has a powerful script program and can be used to create almost any 3D object for use in a simulation program. However, sharing 3D objects online is not enough. Developers have to consider how to setup the model control setting, microphone system and user interface. These issues are critical for a good learning environment for 3D learning.

The last objective is to test and evaluate the developed modules by an outside party. This project is a co-study with the University of Missouri (UMS). After the learning modules were built, UMS has different student groups test and comment about the modules. This task allowed feedbacks to be gathered which permitted new features and enhancements to be added.
Chapter two discusses the background of authoring tools, which are Director and FlashCom. The concept of server and client in FlashCom, and the brief implementation between Director and FlashCom will be presented.

Chapter three discusses the terminology that is used in this research. In order to let readers have better understanding the term used in the thesis, some basic concepts like Shared object, Netconnection, and Netstream will be explained here. The functionality and the advantage of these tools will be present in this chapter as well.

Chapter four discusses technical concepts on how to setup Netconnection and Netstream, and how to use shared objects. The programming explained in this chapter to build a basic 3D collaboration environment with Director and FlashCom. The completed codes are attached at the end of this thesis. The concept of FlashCom and Director will be discussed step by step in this chapter. The reader should be able to setup their own collaboration program with the contents of this chapter.

Chapter five discusses one of 3D collaboration tools. The user interface of the tool will be explained in this chapter. Some of the potential usage will be suggested in this chapter as well.

Chapter six discusses the evaluation done by University of Missouri. The overall evaluation will be reported in this chapter. This chapter also discusses the methodology that UMR is using. Results and recommendations are also presented in this chapter.
Chapter seven discusses what this research has achieved. It concludes the accomplishment in this study. Recommendations for future researches are also suggested in this chapter.
Chapter Two: Literature Review

This thesis introduces real-time 3D collaboration techniques using 3D Director and Flash Communication Server MX (FlashCom) from Macromedia Inc. 3D collaboration modules are developed to help understand 3D engineering concepts better. In this chapter, existing 3D learning techniques will be discussed. The advantages and disadvantages are explained in each technique. The enhancements due to 3D collaboration will be addressed in this chapter as well.

Due to the lack of studies about the implementation of 3D Director and FlashCom in the market, much of the effort in this study has been spent in the development and implementation of 3D collaboration modules. However, background information about the tools will be introduced in this chapter. The basic concept of implementation between Director and FlashCom will be discussed in following sections.

2.1 Existing 3D Learning

Chapter one discussed the benefits that online learning offers to students and instructors. This section discusses how 3D learning environments can benefit the student in learning engineering concepts and what kind of common 3D tools are used today.

In the engineering field, dealing with three-dimensional models is common. Generally, it is hard to grasp 3D concepts by only reading text materials. In order to understand engineering and technical concepts better, the
best way is to provide students with 3D models [13]. Incorporating three-dimensional visualizations in interactive simulations can be used to assist the user in his or her learning progress by using virtual prototype models that would be available in reality [13]. A number of 3D learning tools will be present in next sections.

2.1.1 Three-Dimensional Visualization

One of the common 3D remote learning techniques is 3D visualization, which includes 3D graphics, animations and simulations in an organized system. Such a system was developed by Nisagornsen and works like a 3D book where students can go through a set of questions related to one specific topic [13]. The questions start from the basic to complex level and different questions are presented until the single topic is covered. If students give the wrong answer, the system will give the student feedback and provide hints. Along with the questions, simulation activities are presented to the students. The 3D simulations can help students have a better understanding of the subject without doing a complicated calculation. Figure 2.1 gives various screen shots of that 3D visualization called the 3D Dynamics System.
One disadvantage of the 3D Dynamic System [13] is the lack of interaction between students and instructor. In this learning environment, the students only communicate with the system. If the students have questions that are not defined or classified in the program, users can be confused in the learning process. Furthermore, there is a limitation of the 3D abilities in this tool. Student cannot manipulate the 3D objects directly, since most of the 3D objects are created by capturing different frames. Even though it provides the 3D learning feeling, it is simply created in 2D pictures. Student can only follow the 3D objects according to the pre-defined questions in the system, and the users have little control of the 3D models.

Figure 2.1: Screen shots of 3D Visualization from the 3D Dynamics System [13]
2.1.2 Three-Dimensional Simulation Model

A second type of a 3D learning tool is to build the specific 3D model with a simulation software tool such as Director 3D Shockwave. This kind of simulation model can be built according to an actual theoretical model. An example is shown in Figure 2.2, which was developed at University of Oklahoma [5].

Figure 2.2: 3D Plumber’s Helper using 3D shockwave technology
In the Plumber’s Helper 3D tool, students are able to manipulate the model freely. Also, users can view and rotate the model at different angles. In this example, the force and moment vector are simulated and calculated when the user manipulates the model. This tool can benefit student in learning 3D questions, since it brings the students closer to the problem when they work on the 3D questions. However it does not present background information efficiently in the modules itself. This application would be normally placed in a website when developers try to simulate a specific model. It works well if it is combined with a detailed background information webpage, such as theories and background information. Like the 3D visualization tool, this tool lacks collaboration interaction between students and the instructors.

The 3D learning tools discussed in the last two sections are useful. However, they are lacking of collaboration. The 3D visualization learning system is not able to provide 3D manipulation to students either. In order to improve current 3D learning environment, a 3D collaboration learning tool is needed. A 3D tool should allow students to manipulate the 3D objects as well as collaborate with others. The next section discusses the tools used in developing 3D collaboration.

### 2.2 Authoring Tools

In order to develop a 3D environment and collaboration application, software tools should be able to perform 3D designs and real time
communications. In this research, two main software tools used are Director and Flash Communication Server from Macromedia Inc. The details about each software tool will be discussed in following sections. Macromedia Inc, one of the biggest web development companies, has been developing different web plug-in programs. Programs like Flash and Director are two of the well known online programs. Designers can use these programs in different applications like web design, online learning, and online game.

2.2.1 Macromedia Director and 3D Shockwave

Director is a well known web development tool for web-based interactive media and is used by developers around the world. Director gives users the ability to create complex and interactive application by using its powerful scripting language, called Lingo. The other advantage of using Director is its cross-platform capabilities and web-based media support. Macromedia has added Shockwave 3D technology into Director 8.5 version. The Shockwave is a 3D animation and interactive learning technology that allows users modify subjects such as 3D objects, cameras and lights in 3D world. Shockwave files can be viewed through a web browser plug-in (the Shockwave player) or multimedia applications that access the player directly. Director 3D allow people combine 3D ability with the real-time interactivity [14]. Director has become popular for developing Internet games and different online applications, because of its powerful 3D functions. This ability not only allows user import different 3D
objects into Director 3D world, but also allow user to create different designs using its scripting language (Lingo). Using Lingo to develop 3D objects enable developers to have better control over 3D objects in applications. The other features of 3D director are discussed in following sectors.

### 2.2.2 Macromedia Flash Communication Server MX

Flash Communication Server is real-time, collaborative software. Macromedia released the first Flash communication Server 1.0 (FlashCom) in September 2002 [15]. FlashCom MX is a new version of FlashCom with new features. FlashCom MX has new abilities that can support Director Shockwave. In this paper, FlashCom means FlashCom MX, which is the latest version.

FlashCom is a communication server, which connects different clients and synchronizes data. It provides an extensible and customizable platform for developing web-based communication applications. FlashCom is easy to use since it takes care of the network connection requirements to create networked applications [15]. FlashCom server runs on Windows or Linux. Flash clients can run on different platforms including Windows, Linux, and Macintosh. FlashCom can offer real-time video, audio, text, and data exchange. It can upload video and audio from the user’s camera and microphone. In order to make it easy to use, Macromedia has provided high-level communication components that can be used with a minimum source code to create a variety of basic applications. The components like Simpleconnection, Videochat, and Whiteboard are easy to
implement in the design under the Flash environment. These features allow users to communicate over the Internet browsers without involving any other complexity programming setting.

The concept of FlashCom is similar to a traditional server and client relationship. FlashCom server associates with all the clients in the application. The only difference is that FlashCom allows clients in the same application to interact with one another. All users share the same information and data. If any variable or object changes, the server will synchronize the changes to all clients. Figure 2.3 is a diagram of two clients and FlashCom server in the same application [16]. This is the server system used in FlashCom applications.

![Figure 2.3: Flash communication server and its clients](image-url)
2.3 Implementation of Director and Flash Communication Server MX

The implementation of Director and Flash Communication server (FlashCom) is the main task in this study. Both of Director and FlashCom are mature and powerful multimedia software, and they can be implemented together [12]. FlashCom is designed to support a heavier user load than an earlier program from Macromedia, called Shockwave Multi-user Server. It has been tested successfully with tens of thousands of simultaneous connection [17]. However, they are not commonly to be used together. This is the difficult issue for application developers since there are no published resources. Macromedia have mentioned FlashCom and Director can be implemented together; however little information is found online. Therefore programmers most likely have to spend time experimenting with programming. However, there are some references that provide steps implementing Flash and FlashCom server [18]. Director and Flash use different languages, but the steps in constructing FlashCom connection is similar.

Another issue is FlashCom has many useful components defined in Flash. Moreover, those components may not be supported in Director. There are rarely any references that discuss using FlashCom components in Director. Developers have to develop their own components, like login and microphone system to suit their module designs. The combination of Director and FlashCom enable developer to develop real-time learning modules involving 3D concepts. In the next chapter, basic concepts and theories of these techniques will be introduced.
Chapter Three: Background Network Techniques

In order for readers to understand better in further discussion, this chapter will explain the terms and concepts of FlashCom and Director. The concept likes how to collaborate in real time, how to construct a connection between client and server, and how to share audio and video. The functionalities and the advantages of these tools will be present in this chapter as well.

3.1 Flash Communication Server

3.1.1 NetConnection

NetConnection is an important object in the FlashCom application. It is a built-in object with predefined functions that sets up the communication link between the client Flash applications and the central server. NetConnection object also maintains a two-way connection between the Flash server and the Flash clients. Once a user connects to the application, NetConnection will be created if there are no technical errors such as a bad Internet connection. NetConnection is also used when Shared Objects and NetStream Objects (will be discussed later) exchange data and streams to the server. Communication between client and server is achieved using the Real Time Messaging protocol (RTMP) which is built by Macromedia [15]. The RTMP is specifically designed for FlashCom applications by Macromedia. Once the NetConnection object has been successfully established, data and shared objects will be synchronized. The values
in the shared objects will then be copied to the local clients from the server. This is how clients share information with the FlashCom server. Figure 3.1 shows the flow of NetConnection.

![NetConnection diagram](image)

Figure 3.1: NetConnection and data synchronization

### 3.1.2 Shared Object

How can data exchange in FlashCom application? The answer to this question is the “shared object”. Shared object enables data exchange between the FlashCom server and its clients. Figure 3.2 shows the shared object between FlashCom server and clients.

![Shared Object diagram](image)

Figure 3.2: Flash server and clients sharing information using shared objects
Different types of data can be stored in a Shared object such as numbers, strings and arrays. The server holds the value of the shared objects. If any of its clients change the local value of the shared object, the server will update the data automatically. After the server updates the value of shared object from one of the client, it will distribute the value of shared object to all clients. At this point, the server and all the clients will hold identical values of the data. Figure 3.3 explains the steps of synchronization with Shared Objects.

Figure 3.3: Synchronization steps using shared objects
By applying this concept to the modules, FlashCom can synchronize different types of data such as input value, output result, text message and etc. All users will share the same information in the application. 3D environments usually deal with rotation and position of 3D objects in the 3D world. In order to share 3D models in Director, the developer has to share the rotation and position values of the 3D objects by using Shared objects. When one of the clients moves or rotates the 3D objects, FlashCom will synchronize the value of the shared object. The rest of client will move or rotate the 3D objects according to the shared object from the server. This is the basic idea how Shockwave performs simply 3D collaboration. Because of this ability, designers can construct collaboration applications in a 3D environment. This ability is convenient to programmers, since they do not need to update the data themselves. The shared object will perform the synchronization itself as long as the clients make any changes to the shared objects.

How is a shared object created? The first person who logs in to the module will start the application in the FlashCom server. The application will create the shared objects for specific objects, which are predefined by the programmer using Lingo inside Director. At this stage, the shared objects are just created for the application. Shared object will be deleted, if the application is stopped. After someone starts the application, anyone who is logged in to the module will automatically synchronize to the local values from the server.
3.1.3 NetStream Object

The previous paragraph discussed the NetStream object and how it allows users to send and receive live streaming video and audio. Users can also send synchronized messages within the stream to others with the NetStream objects. Most 3D modules developed modules in this research use only the audio ability of NetStream object due to bandwidth limitations. If the application has 40 to 50 people, it will not be possible to publish video stream to all users from a single server. On the other hand, if the application is a 2 or 3 people online conference, the bandwidth issue will not be a problem at all. In this study, video stream is not considered for use in the 3D collaboration modules, since video is not important in this particular study. Students are still able to collaborate with one another with audio and modules functions. The only advantage taken from NetStream object in this research will be audio ability.

The communication tools between two different NetStream objects are OutStream and InStream. OutStream is used to publish audio, and InStream is used to subscribe it. The basic diagram of NetStream Communication is shown in Figure 3.4.

Figure 3.4: Block Diagram of NetStream objects
Each user has only one OutStream, but it can have different InStreams. This means a user can listen to severed different people at once. For example, if one of the users has two InStreams, they can listen to two different users at the same time.

Shared object and NetStream objects give the developers the capability to communicate in terms of audio, video and data exchange. Everything can be shared by using shared objects and NetStream objects. Because of its simplicity and its supports to Shockwave in many ways [14], Flash Communication Server is a good tool to use in this research as a 3D collaboration development tool.

3.1.4 Server-Side Action Script (SSAS)

Server-Side Action Script (SSAS) is used to define all the behaviors of Flash Communication Server (FlashCom) applications. The behaviors are such as starting up FlashCom application, connecting clients to server, and disconnecting clients from server. The SSAS is written in an .asc file located in the server. SSAS can be used to manage and obtain different information from the users in the applications such as the connection time, user’s IP and online period. Because of these powerful features, the designer can program the login system or security system to enhance the application. Login System is not easy to define in Director, since the modules will be exported in HTML format. The only way to build a secure login system is using SSAS. The SSAS is similar to Macromedia Flash
Action Script except it exists on the server. It cannot directly control the elements of 3D shockwave such as buttons, light, and 3D objects etc. However, it can directly create, modify shared objects to store data, and handle communication events. The Main.asc file is the heart of the application. The SSAS is placed in Main.asc file. When the first user starts the application, the server will define a user list by SSAS. The user list is shared object in array format. Once the shared object of user list has been updated, Director will displace the content of the list from shared object into the stage. If someone logs in after the first user, the list will be updated by SSAS. The SSAS will synchronize the user list to all the clients once the change is made at the server. If any user logs out of the system, the name of that person will be searched and removed from the user list.

### 3.1.5 Bandwidth Limitation

The bandwidth limitation of Flash Communication Server depends on what kind of license is. The developer edition is a 1Mbit/10 user license ($499). The professional license is 10Mbit/500 users. According to these numbers, it is obvious that using FlashCom for audio and video will occupy most of bandwidth. However if the shared data is pure text, the bandwidth issue will not be a problem at all. The 3D modules developed in the research have been tested with 20 people, there is no any degradation because of the bandwidth issue. Using audio stream normally has around 0.2 second to 1 second delay.
3.2 Director

Director is a well-known multimedia development software tool and has numerous features and capabilities. However, its 3D capabilities will be the focus of this section. Director provides “real-time” 3D (RT3D) developmental environment [14]. There is a significant difference between RT3D and what developers usually call 2.5D. The 2.5D uses pre-rendered images to depict 3D, but there has no geometry control over the model because the environment is comprised of drawn bitmaps. Director’s 3D environment is not made of pre-rendered images. Real-Time 3D environments are rendered at runtime by the client machine [14]. The next two sub-sections will discuss the tools used in this research.

3.2.1 Shockwave 3D Castmember

Shockwave 3D Castmember is a central role in the development of any 3D environment in Director. Every 3D model that is created is shown in Shockwave 3D Castmember. The 3D Castmember contains many internal elements that form the basis of the 3D environments. These internal elements include Lights, Cameras, and Models. These are the essential elements of the 3D world [14]. Director’s scripting language, Lingo, allows direct control of all 3D objects in 3D Castmember. 3D models in Director can be created by different ways such as import from other 3D software or create basic shape (e.g. sphere, box, and cylinder) in Director. In this project, all the 3D models are created by using the last method, since it allows more control over the objects. For example, if the
length or the shapes of objects are to be modified, those changes can be achieved by using Lingo. Understand the internal elements of the 3D Castmember and learn how to control 3D object with Lingo are the main skills to develop a good 3D environment in Director.

### 3.2.2 Model Behaviors

In order to control the 3D models, model behaviors are needed to be applied in the 3D world to perform specific functions. The behaviors can be either defined by programmers themselves or found from Library panel in Director. Creating customized behaviors gives designers more control over 3D models in the 3D world, however it requires complicated coding and design. It is achievable, but it will consume certain amount of time in coding development. Using built-in behaviors can be easier and save time in the design. On the other hand, the main disadvantage is the limitation of the behaviors. Sometimes the behaviors are not able to provide what the designers want in the 3D controls, and then they have to design their own behavior. The behaviors used in this study are mostly found in the Director Library, because these behaviors can provide enough model controls in this research.

### 3.3 Collaboration in Director with Flash Communication Server

The architecture of collaboration with Director and Flash Communication Server (FlashCom) is shown in figure 3.5. FlashCom will be installed at the
server. The developed files such as the server side action script (SSAS), HTML file, and 3D Shockwave file are placed at FlashCom folder. In order to view Director file through the Internet, Director file need to be published in HTML format and Director plug-in Shockwave player have to be download in local client machine. Afterward the clients are able to view Director file. See figure 3.5 for the details.

![Figure 3.5: The architecture of collaboration](image)

### 3.1 Share 3D object using Director and Flash Communication Server

From the previous sections, different features of Flash Communication Server (FlashCom) and Director have been discussed. If those features are combined together, collaboration in 3D environment can be achieved with FlashCom and Director. Collaboration with 3D objects is the main goal in this project. This project aims to share 3D objects with many users. In order to achieve
this task, two parameters of any 3D object are required. These parameters are rotation vectors and position vectors. Rotation and position vectors allow users to share the same location and rotation of the object in the 3D world. The shared object is used to share the position vectors and rotation vectors. The sharing process can be summarized in four steps,

Step 1: Move or rotate the 3D object in any clients.

Step 2: Update the changes of the shared object value from the client to the server.

Step 3: Synchronize the changed shared object to all other clients.

Step 4: Move or rotate the 3D objects locally in each client.

After these steps, users are able to collaborate and manipulate 3D objects in real-time to others. Further details and codes will be explained in Chapter 4.
Chapter Four: Collaboration Tool Development

The previous chapter discussed the concepts which are commonly used in this project. In this chapter, technical details will be presented about how to setup real time 3D collaboration using Director and FlashCom. Since this chapter involves many different processes, four different sections will be used to help guide the user. The four sections are FlashCom in Director, Collaboration in Director, Microphone pool system and Model Control Setting.

Details for creating and modifying basic 3D objects in Director, such as World, Light, Camera, Shader, Resource and Model, are not given in this thesis. However, understanding these elements can help user develop better 3D models and 3D environments. A good reference named Director’s Third Dimension by Paul Catanese [14] provides detailed information of these elements.

4.1 Flash Communication Server MX in Director

To implement Flash Communication Server (FlashCom) in Director, there are three basic steps. The sequence is, first create a Netconnection, second, connect to the FlashCom server, and third, synchronize the shared objects with the server. There are two callback functions used when either Netconnection is connected to FlashCom server or the shared objects are synchronized from the server. The listing 4.1 is the first step of the sequence.
global gNC

on createConnection
gNC = newObject("NetConnection")
setCallback(gNC, "onStatus", #onStatus)
user_name = member("usernameinput").text
gNC.connect("rtmp:/Bracket/", user_name)
end

Listing 4.1 Netconnection setup code

The first line of listing 4.1 gives the reference to the Flash Netconnection object. The Netconnection object is used to connect to the FlashCom server. The second line sets the onStatus callback to a Director handler. The first parameter of the setcallback function gives a reference to the FlashCom object that it is calling to. The third parameter is the name of Director handler. When the Netconnection onStatus event is triggered, the Director handler will be executed. The Director handler is shown in listing 4.2, following this paragraph. After creating the Net connexion and handler, the connection to the FlashCom server is requested by executing the last line of the function. The first parameter ("rtmp:/Bracket/") is the name of the application in the FlashCom server. In this case, the name of the application is “Bracket”. The second parameter of this code is the name of user.

The only information needed to create the Netconnection from the server is the name of the application. If the developer wants to add other features to the system such as login system for security purposes, additional information user name and user ID from clients will be needed. In those cases, different information can be obtained from the client in these parameters. Different parameters can be defined as long as the parameters are consistent with Server Side Action Script (SSAS) at the server.

on onStatus void, ncObject, info_obj
```
put "*** gNC.onStatus ***"
if info_obj.code = "NetConnection.Connect.Success" then
    onSetupSharedObject
    put "Netconnection is connected successfully."
else if info_obj.code = "NetConnection.Connect.Closed"
    put "Netconnection is failed to connect."
end if
end
```

Listing 4.2: The onStatus handler

The handler of “onStatus” is shown in listing 4.2 has two parameters. The first parameter is ncObject which is a reference to the Netconnection object that is calling this handler. The second parameter is the information object. The main purpose of this handler is to report the connection success of Netconnection. The code for the information object will display the connection status. If the Netconnection is successfully connected, the shared objects can be setup at this point. Once the NetConnection is successfully connected, the Communication Application Inspector will run the application in server. Figure 4.1 is the image of Application Inspector.

Figure 4.1: Flash Communication Server Application Inspector
In the Application Inspector, all information about the shared objects, Netstreams, and connection activities can be monitored in this application. As an example, Figure 4.2 shows the shared objects for this application.

![Image: Application Inspector](image)

Figure 4.2: Information of shared objects in Application Inspector

If the Netconnection is not able to connect for certain reasons, the application can let the users know that the Netconnection has failed to connect. The function `onSetupSharedObject` is called if the Netconnection is connected to the server successfully. The listing 4.3 is the context of the function `onSetupSharedObject`.

```javascript
global gNC

on onSetupSharedObject
    so = newObject("SharedObject")

    -- setup shared object for box
    gSO = so.getRemote("gSO", gNC.uri, false)
    setCallback(gSO, "onSync", #onSync)
    gSO.connect(gNC)
```

33
-- setup shared object for ball
BallSO = so.getRemote("BallSO", gNC.uri, false)
setCallback(BallSO, "onSync", onSyncBall)
BallSO.connect(gNC)
end

Listing 4.3: The context of onSetupSharedObject handler

In the previous code listing, the reference to the shared object has been created at 3rd line. In this case, “so” refers to the shared object class. In this function, two shared objects (gSO and BallSO) are set up. The function “getRemote” is used to pass the name of the Remote Shared object in the uri address where it is the address of NetConnection. The last parameter of getRemote is true or false. If the value is true, then the server can modify the shared object; otherwise it cannot be modified by server. Every shared object must be connected to the Netconnection by using the handler “connect()”. The connect parameter is the reference to the Netconnection. The setCallback is used to call back which handler to run when the value of shared object has changed. The setCallback function will call “onSync” handler in this example every time the context of gSO has changed. The structure of onSync is shown in the listing 4.4.

on onSync void, soObject, change_array
set i = 0
repeat while i < change_array.length
  if (change_array[i].code = "change") then
    BoxVector = sprite(1).member.model("obj").transform.position
    if (change_array[i].name = "xvalue") then
      BoxVector[1] = gSO.data.xvalue
      sprite(1).member.model("obj").transform.position = BoxVector
    else if (change_array[i].name = "yvalue") then
      BoxVector[2] = gSO.data.yvalue
      sprite(1).member.model("obj").transform.position = BoxVector
    else if (change_array[i].name = "zvalue") then
      BoxVector[3] = gSO.data.zvalue
  end
end
The handler “onSync” is called whenever the context of shared object has changed. The first parameter of onSync is the name of the shared object that will be modified. Second parameter is the array of operation, which contains the name of shared object variables and the operation code. When “onSync” has been called, there are five possible operations which may be executed [18].

**Code**

- **Change** – This operation is called whenever any Flash client receives changes from server.

- **Success** – This operation is called when the Flash client that make successfully changes to the shared object.

- **Reject** – This operation is called when any Flash client that make unsuccessfully changes to the Shared object. This is the case when a collision happened. For example, if two clients try to write to the same property at the same time, it will cause an error.

- **Clear** – This operation is called when Flash client connects to a shared object at the first time.

- **Delete** – This operation is called when the shared object has been deleted from server.
The next section explains the collaboration in Director and Flash Communication Server.

4.2: Collaboration in Director and Flash Communication Server MX

After setting up the Netconnection and the shared object in the application, the collaboration in real time is closed. Shared objects share the rotation and position value of 3D objects in this study. However, it is also possible to share other data if needed. In this chapter, a simple example will be presented. Figure 4.3 shows two 3D objects created in Shockware Director using Lingo.

![Figure 4.3: Two basic 3D objects in a typical 3D world](image)

In order to rotate or move a model, either predefined behaviors can be used or they can be programmed. Either method will give the same result. It depends on what the application is. After defining the behaviors, simply apply the
behaviors to the 3D world. Then user can move and rotate the model around in the 3D world. Figure 4.4 shows the collaboration between two browsers.

![Figure 4.4: Example of 3D objects being collaborated in two browsers](image)

If the user manipulates the 3D object locally, the changes in any 3D objects will be shared. Recall, there are four general steps in sharing objects in Chapter 3.3. The first step has been discussed in the last two paragraphs, which are the manipulation from client. Second step is to update the changes to shared object on the server.

In order to perform this, the changes of the shared objects have to be updated to server whenever the 3D model has been moved. The listing 4.5 shows the handler updating the shared object values to the server.

```javascript
global BallSO
on updateBallSO
  positionboxVector = sprite(1).member.model("obj").transform.position
  BallSO.data.xposition = positionboxVector[1]
  BallSO.data.yposition = positionboxVector[2]
  BallSO.data.zposition = positionboxVector[3]
```
rotationVector = sprite(1).member.model("obj").transform.rotation
BallSO.data.xrotation = rotationVector[1]
BallSO.data.yrotation = rotationVector[2]
BallSO.data.zrotation = rotationVector[3]
end

Listing 4.5: The update of shared object handler BallSO

In the listing 4.5, the shared object BallSO stores both position and rotation values. In the function upDateBallSO, the local position values of 3D model name “Obj” are copied into temporary vectors called positionboxVector. The following code gives the temporary vector three different parameters which are x, y, and z coordinates. Programmers copy each parameter into x, y and z position of the shared object BallSO at server.

positionboxVector
= sprite(1).member.model("obj").transform.position

Afterward, the position of the shared object will be updated to the server. Those are the steps to update shared object value to server.

Once the server receives the changes from one of the clients, the server will broadcast to each client in the application by calling “onSync” handler defined in last section. This handler simplifies the programming, since this step is done by FlashCom itself automatically. As long as “onSync” handler is defined, Director will perform the change as what the handler is defined. The following code is quoted from “onSync” handler in previous code example.
if (change_array[i].code = "change") then
BoxVector = sprite(1).member.model("obj").transform.position
    if (change_array[i].name = "xvalue") then
BoxVector[1] = gSO.data.xvalue
sprite(1).member.model("obj").transform.position = boxVector
else if (change_array[i].name = "yvalue") then
boxVector[2] = gSO.data.yvalue
sprite(1).member.model("obj").transform.position = boxVector
end if
end if

Once "onSync" has been called, the value of the shared objects will be

copied to the server and then to the client. After the values have been copied,

Director will move and rotate the 3D models using the following code.

sprite(1).member.model("obj").transform.position = boxVector

At this point, the rest of clients in the application receive the update and
change the movement of 3D object named "obj" locally. It should be noted, the
shared objects can store different type of variables such as arrays, strings, and
integers. By using shared object, 3D objects can be manipulated in different ways.
For instance, the length of the 3D objects or the color of the object can be
changed in real time. All the changes will be collaborated to all the users in the
application.

4.3 Microphone Pool System

This section discusses the procedures of setting up the Netstream and the
concept of microphone pool system used in this research. The detailed codes of
microphone pool system are attached at the appendix. The procedures of setup Netstream are:

1. Create the Netconnection for just Netstream
2. Import a movie object from Flash
3. Add the reference to microphone and camera
4. Publish and subscribe using Outstream and Instream, respectively

Creating the Netconnection for Netstream is similar to creating Netconnection. Example code for setting up the NetStream is shown in listing 4.6.

```plaintext
on NetStreamConnection
    -- initialize a flash movie object reference
    pSprite = sprite(55)

    -- create a new NetConnection object
   pNetConn = pSprite.newObject("NetConnection")

    -- declare an onStatus callback handler
    pSprite.setCallback(pNetConn, "onStatus", #myOnStatus2)

    -- set the connecting URL
    tURL = "rtmp:/NetStream"

    -- connect the object to the server
   pNetConn.connect(tURL)
end
```

Listing 4.6: The context of handler NetStreamConnection

The above code is slightly different than NetConnection from the last section. Since the setup of Netstream is involved with Flash movie object, Flash object reference has to be attached to the Netstream. In this example, the sprite(55) is the flash movie object, which is used to for DTS (Data Transmission System). The rest of the code identifies and connects to the url address of the application.
The second step is to export a Flash movie object from Flash into Director. Simply publish a Flash movie file from Flash, and import it into Director. Then, in the property inspector of the Flash movie object in Director, click on the box “DTS” and “Audio”. Make sure the boxes are checked, otherwise the Netstreams will not able to be created. Figure 4.5 is an image of the property inspector of the Flash movie object.

![Property Inspector of Flash movie object used in Director](image)

After importing the Flash movie object in Director, microphone and camera references have to be added to the Flash movie. The code example is shown in listing 4.7 below adds the reference and creates the Instream and Outstream. The code also initiates the Netstreams. The initiateStream handler is called whenever the connection of Netstream is successfully created.

```plaintext
global pInStream, pOutStream
on initiateStreams
  pSprite = sprite(55)
```

41
-- gain references to the camera and microphone
tCamObj = pSprite.newObject("Camera")
pCamera = tCamObj.get()
tMicObj = pSprite.newObject("Microphone")
pMicrophone = tMicObj.get()

-- create a NetStream object
pOutStream = pSprite.newObject("NetStream",pNetConn)

-- attach the camera and microphone to the outgoing stream
pOutStream.attachAudio(pMicrophone)
pOutStream.attachVideo(pCamera)

-- create In NetStream object
pInStream = pSprite.newObject("NetStream",pNetConn)
end

Listing 4.7: The initialization of Netstreams

The concept of Netstream can be recalled from the last chapter in
Netstream section at the Figure 3.4. The last step is to publish and subscribe the
Netstream accordingly. The simply handler publishStream and subscribeStream
are shown below:

on publishStream
    pOutStream.publish(member("usernameinput").text,"live")
end

on subscribeStream
    pInStream.play(MicPoolSO.data.Mic1)
end

In the handler publishStream, the Outstream can be named anything. In
this case it uses the name of the person who publish the Outstream is more
suitable, since other users are able to know who is publishing in the application.

In the handler subscribeStream, the InStream will be automatically
subscribed according to the name from the shared object. Since the only way user
knows who is publishing Netstream is to check the Shared object that contains the
name of the Outstream. In this case, Mic1 is the variable of Outstream, and it stores a name (in string format) in the shared object named MicPoolSO. The first part of this section discussed the steps of setup the Netsteam, the rest of this section will discuss the concept of MicPool system used in this project.

The MicPool system used in this project allows 4 people to communicate in the module. All users who login the system are able to listen those who are talking in the MicPool. Because of bandwidth limitation, it may not be possible to allow everyone to talk at the same time. The bandwidth will increased exponentially if more people publish in the system. Table 4.1 compares the differences between the bandwidth of client and server streams.

<table>
<thead>
<tr>
<th>No. of Clients</th>
<th>Client Upstream</th>
<th>Client Downstream</th>
<th>Server Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8 KB/s</td>
<td>8 KB/s</td>
<td>32 KB/s</td>
</tr>
<tr>
<td>3</td>
<td>8 KB/s</td>
<td>16 KB/s</td>
<td>72 KB/s</td>
</tr>
<tr>
<td>4</td>
<td>8 KB/s</td>
<td>24 KB/s</td>
<td>128 KB/s</td>
</tr>
<tr>
<td>5</td>
<td>8 KB/s</td>
<td>32 KB/s</td>
<td>200 KB/s</td>
</tr>
<tr>
<td>6</td>
<td>8 KB/s</td>
<td>72 KB/s</td>
<td>800 KB/s</td>
</tr>
<tr>
<td>7</td>
<td>8 KB/s</td>
<td>152 KB/s</td>
<td>3,200 KB/s</td>
</tr>
<tr>
<td>8</td>
<td>8 KB/s</td>
<td>392 KB/s</td>
<td>20,000 KB/s</td>
</tr>
</tbody>
</table>

Table 4.1: Bandwidth demands for number of client connections

As the table shows [15], if more users publish or subscribe to the NetStream, the server will not able to deal with such large bandwidth. Therefore
the modules limit simultaneous talking to four people. Another reason using the microphone pool system can prevent all the students from talking in the discussion at once. Imagine if all students talk in the classroom, it certainly disturbs the learning environment.

In order to create a four people microphone system, a shared object will be needed to store the name of publishers. It will hold four different strings (the name of the publishers). If a user presses the microphone button, the name of the user will be updated into one of the string of the shared object. Once the shared object has been update the name of the string, the local client will subscribe according to name in the shared object. For example, if there is only one publisher then one string value of the shared object will hold the name of publisher and the rest will be “NAN”. The term “NAN” does not have any specific meanings, it just means not occupied. If no one publishes in the application, the strings of the shared object will be “NAN”. Figure 4.6 shows example of “no one” publishing and “one person” publishing.

![Image of microphone pool system](image)

(a): No one publish in the system  (b): One person publish in the system

Figure 4.6: Screen shots of microphone pool system
The system will not subscribe any Netstream from server as long as the string of the shared object is “NAN”. All the users will subscribe to the NetStream according to the string of the shared object. Figure 4.7 shows the block diagram of microphone Pool system.

![Figure 4.7: Block diagram of microphone pool system](image)

### 4.4 Model Control Setting

If all users in the application are able to control, move, and rotate 3D models, users can be confused with the changes of the model. In order to prevent this from happening, the system should have a model control setting which allows only one person to manipulate 3D model one time. In the other words, two people cannot move the model at the same time. If the model control is gained by someone, the rest of the user should not able to control the model until the person release the control. Also the instructor should be able to take control of the model.
even if a student currently has control. For instance, if a student has the model control, instructor should able to get back the control from the student without waiting for him to release the model. There should be different ways to construct the logic by using shared object and if-end statement. Figure 4.8 shows the logic diagrams for model control settings used in this study.

(a) Gain Model Control by a client

(b) Release Model Control by a client

Figure 4.8: Logic Diagram of model control settings

According to the above diagram, there is only one shared object. The shared object contains two pieces of information, which are the name of the person who controls the model, and the control status (1/0). The control status is a Boolean logic. If the logic is 1, which means the model is occupied. On the other hand, if the control status is 0, which means the model is available to anyone. If one of the clients gains or releases the model control, the shared object will be updated to server. If no one controls the model, the shared object value will be
“NAN”, which is mean not occupied. In this case, the control is available to anyone in the system.

In order to prevent others from controlling the model, there are two things that have to be dealt with, model control buttons and 3D world. Once the shared object synchronizes a client who controls the model, control buttons will be available and 3D world will be uncontrollable to the rest of clients. On the other hand, if someone releases the control, control buttons will be available to anyone. Once the other client gains the model control, the 3D world will be controllable. Figure 4.9 shows the control buttons.

![Control Buttons](image)

Figure 4.9: The screen shots of control buttons, (a) without model control, (b) with model control
To prevent 3D world from being controlling by other user, one of the simplest ways is to put a transparent layer on top of all the 3D objects. Since this layer is not controllable, the user will not able to manipulate any 3D objects in the 3D world. After someone gains the control of the model, the rest of the users will be blocked by this transparent layer. This step is done once the server synchronizes the control person to all of its clients. The follow codes shows the BlockControl and AllowControl functions.

```plaintext
on BlockControl
    sprite(1).member.modelResource("my_big_sphere_resource").radius = 200
    sprite(1).member.model("MyBigBall").shader.blend = 0
end

on AllowControl
    sprite(1).member.modelResource("my_big_sphere_resource").radius = 1
    sprite(1).member.model("MyBigBall").shader.blend = 0
end
```

If the model control is gained by someone, the function BlockControl will be called to block the local 3D world. If no one controls the model, the function AllowControl will be called to unblock the 3D world. The first function is to block any 3D control. What it does is to change the radius of the transparent ball becomes large enough to cover the whole 3D world. The second statement of the function is to make the object transparent. The function AllowControl is to allow the 3D control. It changes the radius of the transparent ball to become almost zero. Afterward the user will able to move of rotate the rest of 3D objects in the 3D world.
After the user clicks on the control button “on”, Director will jump to the control frame (Figure 4.9 b), which holds all the control buttons. If the user releases the control, it will jump back to the original frame (Figure 4.9 a). Those settings can ensure no more than one user controlling the 3D model at the same time.

Different implementation techniques of Director and FlashCom were discussed in this chapter. The techniques are, implementing FlashCom in Director, performing collaboration in Director, setting up microphone pool system and model control setting. Each of the technique is important. The 3D collaboration modules cannot be constructed without any of these techniques.
Chapter Five: Online 3D Collaboration Module

Chapter 4 discussed the implementation between Director and Flash Communication Server MX (FlashCom). Three 3D-collaboration modules were developed based on the concepts discussed from last chapter. These modules are 3D Moments on Bracket, 3D loads on Truss, and Electric Generator.

These 3D modules were developed using same methodologies. They each have same login system, microphone system and model control system. In next section, one of the modules will be presented as example to explain how the user interface constructed. Sections 5.1.2 and 5.1.3 will briefly introduce the other two modules, which are 3D Loads on Truss and Electric Generator.

5.1 3D Loads on Truss

This section discusses the structure of the 3D modules developed in this research. One of the modules, 3D Loads on Truss, is used to demonstrate the structure. This module simulates a single force applied to a 3-member truss frame. By applying different magnitudes and directions to the force, the reaction force on the each member will change accordingly. When the user manipulates the model, the module will calculate the forces of each member automatically. The tool can help students understand better about 3D trusses. It has a real-time 3D world that all students and the instructor work together to manipulate 3D truss model. The single load of the model can be rotated and moved by user. Figure 5.1 shows the
main elements of the 3D modules. The brief functions of each element will be explained in following paragraphs.

Shockwave 3D world is used to place 3D objects. All the 3D objects were created using Lingo. All users in the application will share the same vision in this window. Both 3D manipulation and collaboration are performed in this 3D world.

The function of the people list is to show how many people are in the application. The People list is monitored by Server Side Action Script (SSAS).
 Whoever logs in the system, SSAS will add the name of user into array which is
the shared object. The context of the array will be displaced in this people list.

 Connection light indicates the connection status. Green color means the
client is connected to server. On the other hand, red color means disconnected. If
the server is disconnected in the middle of a discussion, it will tell the users the
server is not connected. Afterward, Director will jump to login window and let
user re-login in the system again.

 The purpose of the microphone pool is to show who is presenting in the
application. Without the microphone pool it is hard to tell who is speaking in the
discussion. This microphone system is used as walkie-talkie style. This style can
prevent some of the users from occupying the microphone. Using this design,
users have to press the microphone buttons whenever they want to present in the
discussion. However this does not apply to instructor, who can use the
microphone at any time.

 The model control button shows different manipulated functions in the
module. The functions are such as reset the 3D world, rotate the model, modify
the input and etc. In this module the user can rotate two different objects, which
are the whole 3D model and the force vector. The force joint can be moved with
the “control” key. The model control will be disappeared when other user
manipulating 3D model. It prevents the 3D world from being controlled by more
than one person.

 The output area is used to show the calculation results that are related to
the 3D model while the manipulation has been taking place. The calculation
formulas are predefined and are coded with Director scripting. In this 3D truss module, it calculates the reaction force on three different members when different angle and magnitude of the force applied at the top joint. The input area is used to input data that can affect the output calculation.

Flash player provides a private window to ensure user’s privacy is protected. This happens when any Flash movie tries to access devices like the microphone or camera [18]. This design is to protect the privacy and the security of the Flash application user. Without allowing this prompt, FlashCom will not to access any camera or microphone. One the anther hand, the allow button has to be clicked or user will not able to participate in audio discussion.

The chat board is used to communicate in text form between users. Beside the audio communication, instructor and students can question and answer problems. The chat board can give a place to students for group discussions. For instance, the instructor can put math equations and solving steps in text chat area to answer student’s questions.
5.2 3D Moments on Bracket

The module in Figure 5.2 is called 3D Moments on Bracket and it simulates a single force applied on a bracket. The force can be placed at four different joints on the bracket. An interactive 3D world allows users to manipulate the force and model. By moving the force, the moment will be automatically updated. Also, the magnitude of the moment and the force will be calculated while users are manipulating the model. This tool can help the students understand the relationship between force and 3D moment, since the module gives the students the ability to view and manipulate the model at different angles.

Figure 5.2: Picture of 3D Moments on Bracket
5.3 Electric Generator

The module shown in Figure 5.3 is called Electric Generator. The tool demonstrates the simple design of an electric generator. Different components of generator have been simulated in the modules. It shows how electrical energy is converted from mechanical energy. The module provides the calculation of the voltage according to the Faraday’s Law. There are four different input parameters can affect the voltage outputs. When different input parameters are applied to the module, the users will see the changes dynamically in the 3D world. For example, in the simulation students can change the speed of the rotor, the number of the wire turns and the area of the rotor. The users are able to discuss the design of generator in real-time.

Figure 5.3: Picture of Electric Generator
5.4 Usage

There are different possible uses for online 3D collaboration modules. The first application can be used for online lecture. Instructor can use different 3D collaboration module in topic he wants and conduct online lecture using this tool. Second usage can be used for online conferences. Commercial companies always need the online conference to share idea and design for people who are far away from each other. A 3D collaboration module can be used as a conference tool. It not only allows users to share audio and video, but also allows users to share 3D objects such as designs and products. All the participants of the application can manipulate the model real-time online. The other use can be developed as online games. Game developers can create simple online 3D game using the features of this tool. Different possible usage can be found in manipulation and collaboration in Director and FlashCom Server MX.
Chapter Six: Testing and Performance

After successfully developing different 3D collaboration tools using Director and Flash Communication Server MX (FlashCom), two of the 3D modules which are 3D Moments on Bracket and 3D Loads on truss were tested and evaluated. The modules were placed in eCourses server at University of Oklahoma at http://eml11.ou.edu/3d/. The evaluation was conducted by University of Missouri (UMR). This project is a co-project with UMR. The evaluation is called Laboratory for Information Technology Evaluation, which is done by two professors and students from UMR. The professors who conducted this evaluation are Dr. Timothy A. Philpot, Dr. Richard H. Hall from Department of Basic Engineering. This chapter discusses the methodology, results and recommendations from this evaluation.

6.1 Methodology

This evaluation was done by ten undergraduate students from UMR. Three of the modules were tested in this evaluation. These modules are 3D Moments on Bracket, 3D Loads on Truss, and Bracket Deflections with Load. The last module was developed by a PhD student whose name is Hengzhond Wen [19]. These ten students were enrolled in Mechanics of Materials class in UMR. Since all of the testing modules are related to Mechanics topics, the mechanics students were able to evaluate the modules better than students from other departments. The students
were paired into 5 teams. Each team collaborated in separated room. During the evaluation, web cam and screen capture software (MORAE) were used. The function of MORAE is to record real-world actions, such as user speech and facial expressions, along with detailed application and computer system data to give usability professionals a unique view into the way that software and web sites are seen and experienced. [20]. Figure 6.1 shows the picture captured by MORAE software.

Figure 6.1: Pictures of evaluation captured by screen software MORAE
Each student had a post experiment interview after using the modules. The interview collected their opinion, output, and suggestion. The summary results of the evaluation are presented in next section.

### 6.2 Result Overview

In general, the students had positive feedback about the 3D collaboration modules. They believe the environment of the modules can benefit them in learning engineering concepts. On the other hand, they addressed a couple of issues and made suggestions in the user interface. Those issues are related to the control, and the communication.

The students commented the general model control was easily accomplished. They felt the model is easy to be manipulated in 3D world. There was one inconvenient issue about Model control was not obvious when it was designed. When someone in the application has the model control, the model control panel of other user’s window will be gone. Students found that they do not understand what the module does in this case. To solve this issue, a dimmed out panel can be used instead of taking off the panel. In this way the students can realize who control. The other control suggestion was to have a bookmark, since students always want to compare different inputs (like magnitude, rotation, etc) of the model to see how they affect the outputs. Having a bookmark can help users for comparison and improve the discussion environment.

There were two recommendations about the microphone pool system. The recommendations are the activation of the microphone and the microphone
setting. Students appreciated the current design of the microphone, but since it is walkie-talkie style, students found that they cannot talk and manipulate the 3D models at same time. Student found walkie-talkie system is not convenient during the discussion with others. One way to solve this issue is to use keyboard activation instead of using the mouse. Second concern is that the microphone setting is not convenient for small group of students. The reason for using four people microphone pool system is to prevent bandwidth overflow. However if there are only couple of students using the application, bandwidth will not be an issue. One solution to this problem is to set the microphone on while the number of user is less than or equal to four people. This solution can prevent the issue of bandwidth overflow while many users in the system. On the other hand it makes the small group of discussion more convenient while the bandwidth is not an issue.

This evaluation is considered successful. The suggestions discussed from last paragraph were well-concerned. Since this was the first time the 3D collaboration modules was tested, some of the minor issue like labels of the model may not be well considered at the design stage. This is the reason to have this evaluation. There are no major crashes or errors during the testing. All the concerns about the 3D modules are minor issues, and those issues will be modified at the end of this research.
Chapter Seven: Conclusion

7.1 Summary

The primary objective of this research was to use Shockwave Director and Flash Communication Server (FlashCom) to create the 3D real time collaboration learning environment. Since there are few studies related to 3D collaboration in academic researches, this thesis is a pioneer study in the 3D collaboration area.

The major of the effort in this research was spent in researching the implementation between Director and FlashCom. The other focus was developing different 3D engineering learning modules. After three 3D modules were built, two of them were tested by, the University of Missouri (UMR). This task allowed feedback to be gathered, which suggested new features to improve the 3D modules.

Various objectives were accomplished through the research. Director and FlashCom were successfully implemented together to build 3D real time collaboration tools. Three different 3D modules were developed using the same methodology. These 3D modules are related to different engineering areas, which are Statics, Dynamics, and Electrical sciences. Since all these areas are general requirements for engineering students, the tools can benefit all engineering students. Two of the 3D collaboration tools were evaluated by UMR. The students involved had positive feedback about the 3D modules, and they believed the collaboration environment and visualization abilities of the modules can benefit
them in learning engineering concepts. There were no major problems during the testing. A few recommendations related to the interface were suggested.

The first 3D module is related to 3D Moments on Bracket. It allows the instructor and students to move the direction of a point load at the end of a simple three member bracket. Users can manipulate the magnitude and direction of the load. The second module is 3D Loads on Truss. It has a 3D world that allows users to manipulate a basic 3D truss. A single load is applied at the top of the truss. By rotating and moving the load, the magnitude of all the trusses will be calculated. The last module is an electric generator. It is a simple simulation of a basic generator. By using this module, users can understand how electrical energy is converted from mechanical energy. The electric generator simulation performs according to different inputs such as the number of coils, magnetic field and other factors. All three 3D collaboration modules are constructed using Director and FlashCom, the modules provide not only theoretical calculations but also offer interactive environments. They also allow the instructors and the students to have an online lecture that provides 3D collaboration as well as voice and text communication. These 3D modules can also be integrated into online courses.

7.2 Recommendations for future researches

In this research, the primary goal was to implement Director and FlashCom. The 3D real time collaboration was achieved. Instructor and students can manipulate 3D objects online in the 3D modules. However, the functionalities
of these 3D modules were not well constructed. Instructors may not be able to conduct an efficient online lecture using current settings, therefore, a better module control setting functionality is recommended in future research.

In order to design a better control setting, the microphone system and model control system should allow instructors to have control over the modules. Instead of letting the students take the control of the model themselves, the instructors should have the power to assign the control to the students they choose. In the same way, instructors should also be able to assign the microphone to specific students. However, this is not an easy task to accomplish in Director. Director has to deal with Server Side Action Script (SSAS), since SSAS has greater control over the users in the system. By combining Director with SSAS, a better control environment can be created.

The 3D collaboration protocol developed in this research can also be used in industry to develop online design conference systems. This protocol enables engineers to share their designs and ideas with their clients and co-workers in three dimensions. Although this was an academic research with limited resources, it demonstrated that 3D applications can be developed for high performance. The 3D modules were constructed by using basic techniques from Director and Flash Communication Server MX (FlashCom). Different other functionalities of the software were not used in this study. Companies should explore the other functions and implementations of Director and FlashCom that can create well-designed collaborative environments.
References

[9] A safer place to meet (2003), FORBES, April 28, 2003, Nicole Ridgway


Appendices

Appendix A: Server Side Action Script Code (main.asc)

//-------------       Server Side Action Script      --------------//

function User(client, userName) {
    this.client = client;
    this.userName = userName;
    this.connectTime = new Date().getTime();
};

User.prototype.getTimeConnected = function() {
    var now = new Date().getTime();
    return (now - this.connectTime)/1000;  // seconds
};

User.prototype.getPingTime = function() {
    var client = this.client;
    if (client.ping()) {
        return client.getStats().ping_rtt/2;
    }
    else {
        return "Client is not connected."
    }
};

User.prototype.getConnectionInfo = function() {
    return "  IP: " + this.client.ip + ", user name: " + this.userName + ", connection time: " + this.getTimeConnected() + ", ping time: " + this.getPingTime();
};

function listCurrentUsers() {
    trace("----------------- Current Users -------------------");
    var i, user;
    var total = application.clients.length;
    trace("There are " + total + " current users.");
    for (i = 0 ; i < total; i ++) {
        user = application.clients[i].user;
        trace(user.getConnectionInfo());
    }
    trace("-----------------------------------------------");
}
localNameList = []; 
localNameList = userlist_so.getProperty("peopleList"); 
trace("localNameList: " + localNameList); 
trace("peopleList size: " + localNameList.length);

application.onAppStart = function(){
    trace("onAppStart> " + application.name + " is starting at " + new Date());
    userlist_so = SharedObject.get("peopleListSO");
    userlist_so.setProperty("userNo",0);
    miclist_so = SharedObject.get("MicPoolSO");
    buttom_message_so = SharedObject.get("messageSO");
    buttom_message_so.setProperty("buttomstatus",0);
    buttom_message_so.setProperty("modelcontrolperson", "NAN");
    miclist_so.setProperty("MicUserNo",0);
    //try mic1, mic2, mic3
    miclist_so.setProperty("Mic1", "NAN");
    miclist_so.setProperty("Mic2", "NAN");
    miclist_so.setProperty("Mic3", "NAN");
    miclist_so.setProperty("Mic4", "NAN");
    miclist_so.setProperty("buttomstatus",0);

    setInterval(listCurrentUsers, 10000);

    testingList = ["firstuser", "seconduser"]; 
    userlist_so.setProperty("peopleList", testingList);
};

application.onStatus = function(info){
    trace("onStatus> info.level: " + info.level + ", info.code:" + info.code);
    trace("onStatus> info.description: " + info.description);
    trace("onStatus> info.details: " + info.details);
};

application.onConnect = function(client, userName){
    //application.onConnect = function(client, userName, password){         //
    //password login

    trace("Connection attempt from IP: " + client.ip + " userName: " + userName);
    var user = users[userName];

};
// undate sharedobject -- peopleistSO
// Property: userNo
// property: peopleList
userlist_so = SharedObject.get("peopleListSO");
userlist_so.setProperty("userNo", userlist_so.getProperty("userNo") + 1);
client.user = new User(client, userName);
application.acceptConnection(client)

// push a user in the list when user login
testingList =[];
testingList = userlist_so.getProperty("peopleList");
testingList.push(userName);
testingList.sort();
userlist_so.setProperty("peopleList", testingList);

trace("Connection accepted at " + new Date());

application.onDisconnect = function(client){
  userlist_so = SharedObject.get("peopleListSO");
  userlist_so.setProperty("userNo",userlist_so.getProperty("userNo") - 1);
  buttom_message_so = SharedObject.get("messageSO");
  modelcontrol = buttom_message_so.getProperty("modelcontrolperson");

  if (modelcontrol == client.user.userName)
  {
    buttom_message_so.setProperty("buttomstatus",0);
    buttom_message_so.setProperty("modelcontrolperson", "NAN");
  }

  micpoolSO = SharedObject.get("MicPoolSO");
  mic1 = micpoolSO.getProperty("Mic1");
  if (mic1 ==client.user.userName) // catch instructic logout without turn
    off the mic
  {
    micpoolSO.setProperty("Mic1","NAN")
  }

  localNameList = [];
  localNameList = userlist_so.getProperty("peopleList");
  var i, user, listlength;
  listlength = localNameList.length;
  for( i = 0; i<localNameList.length ; i++){
user = localNameList[i];
if (client.user.userName == user) {
    trace("localNameList: before delete: " + localNameList);
    delete localNameList[i];
    localNameList.sort();
    localNameList.length = listlength - 1;
    listlength - 1;
}
}
trace("localNameList: after delete: " + localNameList);
userlist_so.setProperty("peopleList", localNameList);

trace(client.user.userName + " is disconnection at: " + new Date());
trace(client.user.userName + " was connected for" +
client.user.getTimeConnected() + " seconds.");

};

application.onAppStop = function(info){
    trace(application.name + " is stopping at " + new Date());
    application.clearSharedObjects("/");

};
Appendix B: Set up NetConnection in Director

```plaintext
on createConnection
    -- connection
gNC = newObject("NetConnection")
setCallback(gNC, "onStatus", #onStatus)
user_name = member("usernameinput").text
    gNC.connect("rtmp://Truss/", user_name)
    put "after..... setcallback"
    -- initialize the message and chat field
    member("messagefield").text = ""
    member("chatfield").text = ""
end

on onStatus void, ncObject, info_obj
    put "*** gNC.onStatus ***"
    put "nc object:" & ncObject
    put "info_obj.code:" & info_obj.code
    put "info_obj: " & info_obj
    if info_obj.code = "NetConnection.Connect.Success"
        onSetupSharedObject
        go to frame 5
        initiateConnection2
        createCheckConnection
    else if info_obj.code = "NetConnection.Connect.closed"
        put " lost connection...."
    end if
end

on onSetupSharedObject
    so = newObject("SharedObject")
    -- setup shared object for box
    gSO = so.getRemote("gSO", gNC.uri, false)
    setCallback(gSO, "onSync", #onSync)
    gSO.connect(gNC)
    -- setup shared object for ball
    BallSO = so.getRemote("BallSO", gNC.uri, false)
    setCallback(BallSO, "onSync", #onSyncBall)
    BallSO.connect(gNC)
    -- setup shared object for peopleList
```

70
peopleListSO = so.getRemote("peopleListSO", gNC.uri, false)
  setCallback(peopleListSO, "onSync", #onSyncpeopleList)
  peopleListSO.connect(gNC)
  -- peopleListSO.setFps(10)

-- setup shared object for message
messageSO = so.getRemote("messageSO", gNC.uri, false)
setCallback(messageSO, "onSync", #onSyncmessage)
messageSO.connect(gNC)

MicPoolSO = so.getRemote("MicPoolSO", gNC.uri, false)
setCallback(MicPoolSO, "onSync", #onSyncMicPool)
MicPoolSO.connect(gNC)

-- setup shared object for calculation
calSO = so.getRemote("calSO", gNC.uri, false)
setCallback(calSO, "onSync", #onSyncCal)
calSO.connect(gNC)

-- create a private flag property named synchronized
gSO.synchronized = false
BallSO.synchronized = false
peopleListSO.synchronized = false
messageSO.synchronized = false

end
Appendix C: Setup NetStream in Director

```plaintext
on initiateConnection2
    -- initialize a sprite reference
    pSprite = sprite(55)

    -- create a new NetConnection object
   pNetConn = pSprite.newObject("NetConnection")

    -- declare an onStatus callback handler
    pSprite.setCallback(pNetConn,"onStatus",#myOnStatus2)

    -- set the connecting URL
    tURL = "rtmp://NetStream3"

    -- connect the object to the server
   pNetConn.connect(tURL)
end

on myOnStatus2 void,aArg1, aArg2
    put "myOnStatus2"
    put "aArg2.comd : " & aArg2.code
    if (aArg2.code = "NetConnection.Connect.Success")
        then
            put "NetStream connection is successful"
            initiateStreams2
        else if (aArg2.code = "NetConnection.Connect.closed")
            then
                put "NetStream connection is close"
        end if
    end if
end

on initiateStreams2
    pSprite = sprite(55)
    put(pSprite)
    -- gain references to the camera and microphone
    tCamObj = pSprite.newObject("Camera")
    pCamera = tCamObj.get()
    tMicObj = pSprite.newObject("Microphone")
    pMicrophone = tMicObj.get()
```
pMicrophone.setGain(65)
pMicrophone.setuseechosuppression(true)

-- create a NetStream object
pOutStream = pSprite.newObject("NetStream",pNetConn)

-- attach the camera and microphone to the outgoing stream
pOutStream.attachAudio(pMicrophone)
-- pOutStream.attachVideo(pCamera)

-- create In NetStream object
pInStream = pSprite.newObject("NetStream",pNetConn)
pInStream2 = pSprite.newObject("NetStream",pNetConn)
pInStream3 = pSprite.newObject("NetStream",pNetConn)
pInStream4 = pSprite.newObject("NetStream",pNetConn)
-- publishStream

end
Appendix D: Microphone Pool configuration

```plaintext
-- setup shared object for MicPoolSO
MicPoolSO = so.getRemote("MicPoolSO", gNC.uri, false)
setCallback(MicPoolSO, "onSync", #onSyncMicPool)
MicPoolSO.connect(gNC)

-----------------------------------
on onSyncMicPool void, soObject, change_array
  set i = 0
  repeat while i < change_array.length
    if (change_array[i].code = "change") then
      put "The changed property of onSyncMicPool: " & change_array[i].name
      if (change_array[i].name = "Mic1") then
        member("Mic1").text = MicPoolSO.data.Mic1
        subscribeStream
      else if (change_array[i].name = "Mic2") then
        member("Mic2").text = MicPoolSO.data.Mic2
        subscribeStream
      else if (change_array[i].name = "Mic3") then
        member("Mic3").text = MicPoolSO.data.Mic3
        subscribeStream
      else if (change_array[i].name = "Mic4") then
        member("Mic4").text = MicPoolSO.data.Mic4
        subscribeStream
      end if
    else if (change_array[i].code = "success") then
      put "Handle a successful change of server data from this player"
    else if (change_array[i].code = "reject") then
      put "Handle a rejected SharedObject write operation"
    else if (change_array[i].code = "clear") then
      put "clear operation...... "
    else if (change_array[i].code = "delete") then
      put "Handle deleted Attributes"
    end if
    i = i + 1
  end repeat
end
```
on publishStream
  if (Micisfull = false) then
    pOutStream.publish(member("usernameinput").text,"live")
  end if
end

on subscribeStream
  user_name = member("usernameinput").text
  if (MicPoolSO.data.Mic1<>"NAN") then
    if (MicPoolSO.data.mic1<>user_name) then
      pInStream.play(MicPoolSO.data.Mic1)
    else
      pInstream.close()
    end if
  else
    pInStream.close()
  end if

  if (MicPoolSO.data.Mic2<>"NAN") then
    if (MicPoolSO.data.mic2<>user_name) then
      pInStream2.play(MicPoolSO.data.Mic2)
    else
      pInstream2.close()
    end if
  else
    pInStream2.close()
  end if

  if (MicPoolSO.data.Mic3<>"NAN") then
    if (MicPoolSO.data.mic3<>user_name) then
      pInStream3.play(MicPoolSO.data.Mic3)
    else
      pInstream3.close()
    end if
  else
    pInStream3.close()
  end if

  if (MicPoolSO.data.Mic4<>"NAN") then
    if (MicPoolSO.data.mic4<>user_name) then
      pInStream4.play(MicPoolSO.data.Mic4)
    else
      pInstream4.close()
    end if
  else
    else
pInStream4.close()
end if
end

on updateMic
if (member("usernameinput").text = profname) then
    member("Mic1").text = member("usernameinput").text
    MicPoolSO.data.Mic1 = member("usernameinput").text
    Micisfull = false
else if (MicPoolSO.data.Mic2 = "NAN") then
    member("Mic2").text = member("usernameinput").text
    MicPoolSO.data.Mic2 = member("usernameinput").text
    Micisfull = false
else if (MicPoolSO.data.Mic3 = "NAN") then
    member("Mic3").text = member("usernameinput").text
    MicPoolSO.data.Mic3 = member("usernameinput").text
    Micisfull = false
else if (MicPoolSO.data.Mic4 = "NAN") then
    member("Mic4").text = member("usernameinput").text
    MicPoolSO.data.Mic4 = member("usernameinput").text
    Micisfull = false
else
    Micisfull = true
end if
end

on removeMic
if (member("usernameinput").text =
    member("Mic1").text) then
    member("Mic1").text = "NAN"
    MicPoolSO.data.Mic1 = "NAN"
pInStream.close()
else if (member("usernameinput").text =
    member("Mic2").text) then
    member("Mic2").text = "NAN"
    MicPoolSO.data.Mic2 = "NAN"
pInStream2.close()
else if (member("usernameinput").text =
    member("Mic3").text) then
    member("Mic3").text = "NAN"
    MicPoolSO.data.Mic3 = "NAN"
pInStream3.close()
else if (member("usernameinput").text =
    member("Mic4").text) then
    member("Mic4").text = "NAN"
    MicPoolSO.data.Mic4 = "NAN"
end
pInStream4.close()

end if

end
on background
  scene = member("3DMember")

  -- cylinder resource
  mymodelresourceCylinder = sprite(1).member.newModelResource("My cylinder
resource", #cylinder)
  mymodelresourceCylinder.topRadius = 0
  mymodelresourceCylinder.bottomRadius = 4
  mymodelresourceCylinder.height = 10

  mymodelresCylinder =
  sprite(1).member.newModelResource("My cylinder Res", #cylinder)
  mymodelresCylinder.topRadius = 1
  mymodelresCylinder.bottomRadius = 1
  mymodelresCylinder.height = 100

  -- create axiz cone
  cone = sprite(1).member.newModel("my cylinder",
    mymodelresourceCylinder)

  Ycone = scene.model("my cylinder").clone("my ycone")
  Xcone = scene.model("my cylinder").clone("my xcone")
  Zcone = scene.model("my cylinder").clone("my zcone")
  negXcone = scene.model("my cylinder").clone("my
    negXcone")
  negZcone = scene.model("my cylinder").clone("my
    negZcone")

  Ycone.translate(0,100,0)
  Xcone.translate(100,0,0)
  Xcone.rotate(0,0,-90)
  Zcone.translate(0,0,100)
  Zcone.rotate(90,0,0)

  negXcone.translate(-100, 0,0)
  negXcone.rotate(180,0,-90)
  -- scene.deleteModel("my cylinder")
  -- cone = negZcone
  cone.translate(0,0,-100)
  cone.rotate(-90,0,0)

  -- create axiz
myYaxiz = sprite(1).member.newModel("my y", mymodelresCylinder)
myXaxiz = scene.model("my y").clone("my x")
myZaxiz = scene.model("my y").clone("my z")

mynegXaxiz = scene.model("my y").clone("my negx")
mynegZaxiz = scene.model("my y").clone("my negz")

mynegXaxiz.translate(-50,0,0)
mynegXaxiz.rotate(0,0,90)
mynegZaxiz.translate(0,0,-50)
mynegZaxiz.rotate(-90,0,0)

myYaxiz.translate(0,50,0)
myXaxiz.translate(50,0,0)
myXaxiz.rotate(0,0,90)
myZaxiz.translate(0,0,50)
myZaxiz.rotate(-90,0,0)

-- add cone to axiz chrild
scene.model("my y").addchild(scene.model("my ycone"))
scene.model("my ycone").addchild(scene.model("my y"))
scene.model("my x").addchild(scene.model("my xcone"))
scene.model("my z").addchild(scene.model("my zcone"))
scene.model("my negx").addchild(scene.model("my negXcone"))
-- my cylinder to negZ cone
scene.model("my neg2").addchild(scene.model("my cylinder"))

-- add color
Shd = scene.newshader("axizshd", #standard)
Shd.diffuse = rgb(10,10,20)
Shd.texture = void
redshd = scene.newshader("x-shader", #standard)
redshd.diffuse = rgb(215, 43, 108)
redshd.specular = rgb("CC0000")
redshd.texture = void
blueshd = scene.newshader("z-shader", #standard)
blueshd.diffuse = rgb(67, 67, 247)
blueshd.specular = rgb("003399")
blueshd.texture = void
greenshd = scene.newshader("y-shader", #standard)
greenshd.diffuse = rgb(47, 196, 43)
greenshd.specular = rgb("336633")
greenshd.texture = void

Xcone.shaderlist = redshd
Ycone.shaderlist = greenshd
Zcone.shaderlist = blueshd
negXcone.shaderlist = shd
cone.shaderlist = shd
-- negZcone.shaderlist = shd
myXaxiz.shaderlist = redshd
myYaxiz.shaderlist = greenshd
myZaxiz.shaderlist = blueshd
mynegXaxiz.shaderlist = shd
mynegZaxiz.shaderlist = shd

-- create center point
centerresource = sprite(1).member.newModelResource("centerResource", #sphere)
centerresource.radius = 1
CenterModel = sprite(1).member.newModel("My Center", centerresource)
scene.model("My Center").addchild(scene.model("my y"))
scene.model("My Center").addchild(scene.model("my x"))
scene.model("My Center").addchild(scene.model("my z"))
scene.model("My Center").addchild(scene.model("my negx"))
scene.model("My Center").addchild(scene.model("my negz"))

-- transparent the center model
centershader = scene.newshader("centershader", #standard)
centershader.diffuse = rgb(30, 30, 200)
centershader.texture = void
centershader.specular = rgb("0000ff")
CenterModel.shaderlist = centershader
scene.model("My Center").shader.blend = 10

resetbase = sprite(1).member.model("My Center").getworldtransform()
create3dText
create3dtextforXYZ
createlight
end

on create3dtext
scene = member("3DMember")
resA = member("A").extrude3d(scene)
obj = scene.newmodel("mytextA", resA)
obj.shaderlist.texture = void
obj.translate(-5,0,-55)
resA.tunneldepth = 3

resB = member("B").extrude3d(scene)
objB = scene.newmodel("mytextB", resB)
objB.shaderlist.texture = void
objB.translate(60,0,60)
resB.tunneldepth = 3

resC = member("C").extrude3d(scene)
objC = scene.newmodel("mytextC", resC)
objC.shaderlist.texture = void
objC.translate(-35,0,45)
resC.tunneldepth = 3

resD = member("D").extrude3d(scene)
objD = scene.newmodel("mytextD", resD)
objD.shaderlist.texture = void
-- objD.translate(-25,55,0)
objD.translate(-20,55,0)
resD.tunneldepth = 3
end

on create3dtextforXYZ
scene = member("3Dmember")
resX = member("X").extrude3d(scene)
obj = scene.newmodel("mytextX", resX)
obj.shaderlist.texture = void
obj.translate(40,0,0)
resX.tunneldepth = 3

resY = member("Y").extrude3d(scene)
objY = scene.newmodel("mytextY", resY)
objY.shaderlist.texture = void
objY.translate(-65,100,0)
resY.tunneldepth = 3

resZ = member("Z").extrude3d(scene)
objZ = scene.newmodel("mytextZ", resZ)
objZ.shaderlist.texture = void
objZ.translate(-65,0,100)
resZ.tunneldepth = 3
redshd = scene.newshader("xshader", #standard)
redshd.diffuse = rgb(215, 43, 108)
redshd.specular = rgb("CC0000")
redshd.texture = void
blueshd = scene.newshader("zshader", #standard)
blueshd.diffuse = rgb(67, 67, 247)
blueshd.specular = rgb("003399")
blueshd.texture = void
greenshd = scene.newshader("yshader", #standard)
greenshd.diffuse = rgb(47, 196, 43)
greenshd.specular = rgb("336633")
greenshd.texture = void

Obj.shaderlist = redshd
objy.shaderlist = greenshd
objz.shaderlist = blueshd

end

on createlight
    scene = member("3DMember")
    light1 = scene.newLight("Lightone", #directional)
    light2 = scene.newLight("Lighttwo", #directional)

    light1.transform.rotation = vector(45, -45, 0)
    light2.transform.rotation = vector(0, 0, 0)
    light1.color = rgb("BFBFBF")
    light2.color = rgb("330066")
end