MCSE Project Report

COMPUTER SUPPORTED COLLABORATIVE WORK (CSCW) ENVIRONMENT
USING VRML AND CGI/PERL

Yanxin Li

Submitted to Committee Members:
Dr. Kai-Hsiung Chang (Chair)
Dr. W. Homer Carlisle
Dr. Alvin S. Lim

Department of Computer Science & Engineering
Auburn University
Auburn, Alabama 36849
December 18, 1997
ABSTRACT

The interactive online 3D technology is gaining more and more popularity on the World Wide Web. Many Web applications are based on this technology, including collaborative working environment. A collaborative working environment allows multiple users to work together through Internet. Virtual Reality Modeling language (VRML) is a cutting-edge 3D technology and has the potential to be the next generation user interface language in which user can interact with colorful, three dimensional objects. VRML allows users to create 3D objects and combine them into a virtual world. Implementation of Computer Supported Collaborative Work (CSCW) environment using VRML becomes appealing for Internet 3D technology research because a conventional CSCW environment normally supports a group of users working as a team in a 2D environment. At this moment VRML doesn’t provide explicit support for multiple users interacting within a single world. By using CGI/Perl to create a VRML world on the fly is one of the solutions. This project focuses on the implementation of a VRML CSCW environment using CGI/Perl programming.
1. INTRODUCTION

2. VRML - THE CUTTING-EDGE 3D TECHNOLOGY
   2.1 VRML - The Script Language
   2.2 VRML Coordinate System and Transformation
   2.3 VRML and World Wide Web
   2.4 VRML Browser
   2.5 VRML Authoring Tool

3. VRML WITH CGI/PERL
   3.1 Introduction to CGI/Perl
   3.2 Creating VRML Worlds Using CGI/Perl

4. PROBLEM STATEMENT AND PROJECT OBJECTIVE

5. IMPLEMENTATION OF VRML CSCW ENVIRONMENT
   5.1 Architecture of VRML CSCW Environment
   5.2 Implementation of VRML CSCW Environment
   5.3 Security

6. FUTURE OF VRML CSCW ENVIRONMENT

7. REFERENCES

8. APPENDICES
LIST OF FIGURES

Figure 1. VRML Coordinate System
Figure 2. Scene Diagram for a Group Hierarchy
Figure 3. Cosmo Player Display Window
Figure 4. Cosmo Player Control Dashboard
Figure 5. Community Place Conductor Edit Screen
Figure 6. General Architecture for CGI/Perl with HTTP Server
Figure 7. Client/Server Architecture of VRML CSCW Environment
Figure 8. VRML CSCW User Interface (Initial Window)
Figure 9. VRML CSCW User Interface (Working Window)
Figure 10. VRML CSCW User Interface (Close View)
Figure 11. VRML CSCW Login Window
LIST OF TABLES

Table 1. List of VRML Browsers

Table 2. Features of VRML CSCW Environment
1. INTRODUCTION

With affordable 3D accelerator chips became available in the market last year, interactive online 3D technology is gaining more and more popularity on the World Wide Web. Web is changing from its current static world to a colorful and interactive world. The demands for 3D technology are growing in the commercial and consumer areas, particularly because of the Internet’s popularity.

VRML, Virtual Reality Modeling Language, is a key interactive online 3D technology. It is a language that describes interactive 3D objects and worlds to be experienced on the World Wide Web. In June, 1994, VRML was just a concept - three dimensional graphics on the Internet. Since then VRML has been growing rapidly and has been the basis for all interactive 3D Web contents.

The most suitable applications of online 3D technology include electronic commerce, collaborative working environment, education and training. The collaborative working environment on the World Wide Web becomes appealing because more and more people want to work together as a team through Internet. For example, a collaborative working environment on World Wide Web could let engineers from different companies or different offices within the same company work together to design a product online.

VRML is especially useful for 3D collaborative work such as a team of architects is working together to design a building which is a 3D object. VRML can provide a three dimensional picture of the building for all architects. To design a building as a team, architects need dynamically interact with 3D objects. This requires VRML to provide explicit support for
multiple users interacting in a single world. So far the latest version VRML 2.0 doesn’t provide this support. By using CGI/Perl script is one of solutions. The purpose of this project is to implement the collaborative working environment to allow multiple users to work in a single VRML world.
2. VRML - THE CUTTING-EDGE 3D TECHNOLOGY

VRML, pronounced “vermel”, stands for Virtual Reality Modeling Language. The goal of VRML is to allow users to describe 3D objects and combine them into scenes and worlds. Since its inception in 1994, it has been growing at a surprising speed. Many people agree that VRML will be the next revolutionary user interface technology. Because “The exploration of information, computers, and people all interacting with each other over the world-wide network requires a new user interface metaphor. The window-based desktops of today were created to allow a single person to interact with a single computer; VRML allows the creation of virtual three-dimensional spaces in which the other people can be included.” [3].

VRML is still changing now. The latest version is VRML 2.0. The future version will include the script language API and multi-user support.

2.1 VRML - The Script Language

VRML, like HTML, is written in plain text, provides descriptions rather than representations, and can be built by hand with an ASCII editor. But VRML is not a “markup language” as HTML. It is a modeling language used to describe 3D scenes. It is more complex than HTML, but less complex than a programming language like C or Java [9].

A VRML file is essentially a collection of objects, arranged in a particular order. As in real life, a VRML object has a particular shape, various surface properties (color, smoothness, shininess, and so on), a position in 3D space. The following program fragment is a simple example of a VRML object. A VRML file starts with the header #VRML V2.0 utf8 as
standalone line. The utf8 specification refers to an ISO standard for text strings known as UTF-8 encoding.

```vrml
#VRML V2.0 utf8
Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0.5 0.5 0.5
    }
  }
  geometry Cylinder {
    radius 3
    height 6
    side TRUE
    top TRUE
    bottom TRUE
  }
}
```

This simple VRML file creates a cylinder object. The four nodes present in this file are the Shape, Appearance, Material, and Cylinder. The information for each node is surrounded by curly braces, “{}”. Nodes contain property fields, whose values define the properties of the object. The Shape node is the basic container node for a geometry object. Each piece of geometry is wrapped in a Shape node, which includes a “slot” for describing an object’s appearance. To create an object, you can choose from among a variety of predefined shapes in VRML, including cubes, spheres and cylinders. Objects can be combined together into VRML worlds by using grouping nodes, such as Group and Transform.

VRML uses the RGB color system. All colors are represented in terms of red, green, and blue in a range of values from 0 to 1. In the example above the diffuseColor is 0.5 red, 0.5 green, and 0.5 blue, which is color gray.
Unlike HTML, VRML is case-sensitive. A VRML file name must have an extension .wrl. For example, you can save the above VRML file as cylinder.wrl. VRML objects can only be viewed by a VRML-enabled browser or a VRML plug-in.

### 2.2 VRML Coordinate System and Transformation

VRML uses a Cartesian, right-handed, 3-dimensional coordinate system where the right is positive X, up positive Y, and toward the viewer is positive Z (see Figure 1). By default, objects are rendered onto a 2-dimensional device by projecting them in the direction of the positive Z axis, with the positive X axis to the right and the positive Y axis up. A camera or modeling transformation may be used to alter this default projection. The standard unit for lengths and distances specified is meter. The standard unit for angles is radian. A VRML world may contain an arbitrary number of local coordinate systems, defined by modeling transformations using Translate, Rotate, Scale, Transform, and MatrixTransform nodes. The various local coordinate transformations map objects into the world coordinate system.

![Figure 1. VRML Coordinate System](image)
The Transform node contains a couple of advanced fields with which you can do more complicated transformations, combining a simple translation, a simple rotation, and a simple scale together. An example is given in the following and Figure 2 is the scene diagram for this program.

```
Transform {
    translation  2  5  7.8
    rotation      0  1  0  1.57
    scale          1.5  1   1
    children [
        Shape {
            appearance Appearance {
                material Material { } } 
            geometry Cylinder { } }
        } 
    Transform {
        translation 0 1.5 0 
        children [
            Shape {
                geometry Box {
                    size 4.5 6 8
                }
            ]}
    ]}
}
```

Figure 2. Scene Diagram for a Group Hierarchy
2.3 VRML and World Wide Web Server

The World Wide Web server is a computer program that handles requests for documents located at a particular host computer. The Web server must be properly configured to establish the VRML file suffix to MIME type mapping. This allows the HTTP protocol to use the MIME types to identify files coming across the Internet as VRML files.

Different HTTP servers have different ways of adding entry into the configuration file. Even the same sever, under different versions, may require a different file. But one thing is certain, i.e., you must have administrative privileges on your server to modify the configuration file. The MIME type for VRML is

\texttt{x-world/x-vrml}.

After the World Wide Web server is properly configured, the VRML files are delivered by the Web server in much the same way as HTML files. The server processes a given request and detects that the file is with a .wrl extension, and returns a MIME content type \texttt{x-world/x-vrml} inserted to the front of the contents of the requested file itself. The Web browser, in turn, parse the content type, and if it supports a VRML plug-in or other VRML viewers, the VRML 3D world is displayed.

2.4 VRML Browser

In order to view a VRML world a VRML browser is needed. Many browsers are available from different companies for different platforms. This project uses a VRML plug-in for Netscape Navigator 4.03 on Windows 95 platform because of the unavailability of VRML plug-
in for the UNIX environment in the Engineering Network at Auburn University. The primary VRML browsers for Windows 95 are listed and compared in Table 1.

<table>
<thead>
<tr>
<th>Browser</th>
<th>Platform</th>
<th>Current Version</th>
<th>Script Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community Place</td>
<td>Windows95</td>
<td>2.0</td>
<td>Java</td>
</tr>
<tr>
<td>Cosmo Player</td>
<td>Windows95</td>
<td>2.0</td>
<td>Java and JavaScript</td>
</tr>
<tr>
<td>GL View/D3D</td>
<td>Windows95</td>
<td>3.03</td>
<td>VRML Script</td>
</tr>
<tr>
<td>Liquid Reality</td>
<td>Windows95</td>
<td>1.0b</td>
<td>Java</td>
</tr>
</tbody>
</table>

Table 1. List of VRML Browsers

This project uses a VRML plug-in, Cosmo Player 2.0. Cosmo Player 2.0 is a free software from Silicon Graphics which can be found at http://cosmo.sgi.com. It is a 3-dimensional viewer for World Wide Web. With Cosmo Player, users can easily navigate and manipulate 3D scenes created in VRML. It includes many features such as VRML 1.0 to VRML 2.0 converter and GZIP. Cosmo Player will automatically convert VRML 1.0 files to 2.0 on the fly. If you prefer to permanently convert your VRML 1.0 files to 2.0, a standalone converter, vrml1tovrml2.exe, is also provided. All data formats can also be gzip compressed. This could provide significant improvement in download time for larger files [1].

Figure 3 shows a sample VRML world using Cosmo Player plug-in for Netscape Navigator 4.03. Figure 4 illustrates the control dashboard of Cosmo Player. There are some buttons on the control dashboard. Using these buttons you can change the your point, move
Figure 3. Cosmo Player Display Window

Figure 4. Cosmo Player Control Tool Bar
through the VRML world, slide backward/forward/left/right/diagonally, tilt your view of a world by dragging in the direction you want to look.

2.5 VRML Authoring Tool

There are two ways to create a VRML world: using VRML authoring tool and by hand. Sometimes it is necessary to create and modify the code for efficiency on the Internet. But for a complicate VRML world it is tedious to build VRML worlds by hand. The fastest and easiest way is to use a VRML authoring tool.

A VRML world authoring tool is a 3D drawing application used to create VRML worlds. A VRML authoring tool normally presents user an interactive user interface from which the user can add, delete, and modify scene contents.

Like VRML browsers, the list of VRML authoring tools is expanding rapidly as VRML’s popularity continues to grow. Community Place Conductor 1.0 from Sony is used in this project to create VRML worlds. This VRML authoring tool allows users to create a virtual world on Internet. It conforms to VRML 2.0. The friendly user interface provides simple operations by pressing buttons and dragging-and-dropping to create a 3D world. User can edit the world by specifying the values of node attributes while seeing the result on the screen. User can also specify the textures for animated pictures, sound, and objects. Figure 5 is the edit screen of Community Place Conductor.

As shown in the figure, there are several windows which allows users to create and modify the VRML worlds. The Conductor window manages the overall procedure for using the Conductor. The 3D Perspective View window displays objects in the perspective projection mode. The Scene Graph window displays the tree structure of a world. The Attribute window
Figure 5. Community Place Conductor Edit Screen
specifies the attribute and name of a node. The Appearance window specifies the color and texture of a world. The Resources Library window displays the resources that come with Community Place Conductor. The Script Editor window is the text editor to edit Java files. The Script Expert window allows you to create a script model for your VRML world. Finally the Route window specifies the routing for events [9].
3. VRML WITH CGI/PERL

3.1 Introduction to CGI/Perl

CGI stands for Common Gateway Interface. *Common* means that CGI can be used by many languages and interact with many types of different systems. *Gateway* suggests that CGI’s strength lies not on what it does by itself, but on the potential access it offers to other systems such as databases and graphic generators. *Interface* implies that CGI provides a well-defined way to call up its features [5].

CGI is a mechanism for communicating between a browser and a server for processing user input through a script and generating output. The output of the script is then sent back to the user’s browser and displayed.

The CGI script is usually written in a simple programming language such as Perl. Perl (Practical Extraction and Report Language) is not a CGI-specific programming language. In fact it is a powerful language with many applications far beyond the needs of CGI.

Perl is an interpreted language optimized for scanning arbitrary text files, extracting information from those text files, and printing reports based on that information. The most significant characters of Perl are portability, modification, and ease of use. Perl is portable to any kind of platforms as long as a Perl interpreter is installed. Perl is also easily modifiable. Users don’t have to compile them in order to run them on the machine, such that any superficial changes in the code can be made by editing the one file that is the script. Comparing with other programming languages, Perl is not a very hard language to learn [6].
The Perl script/HTML interaction is that the CGI engine moderates the feed between the HTML client and the Perl script. The CGI engine receives the data from the client process, sets up the running environment for the scripts, and then returns any output from the script to the HTML client.

Most CGI applications involve manipulating data in some fashion and accessing external programs and applications. Perl provides easy ways for those purposes.

### 3.2 Creating VRML Worlds Using CGI/Perl

By using the CGI/Perl program the designers can implement interactive 3D applications that are created on the fly. Usually a CGI/Perl program resides in Web server. The Web server must be configured to allow users to execute the CGI/Perl script. When the program is called, it outputs the VRML file to the users’ browsers. A simple CGI/Perl script for generating a VRML world can be illustrated by the following program fragment.

```perl
#! /usr/local/bin/perl
print "Content-type: x-world/x-vrml\n\n";
print "<\"END\";="#VRML V2.0 utf8
Transform {
  translation  2     5    7.8
  rotation      0      1    0   1.57
  scale           1.5   1    1
  children [
    Shape {
      appearance Appearance {
        material Material { }
      }
      geometry Cylinder { }
    }
  ]
}
```
First this script identifies the content type it is sending back to the browser. In Perl, the syntax to identify VRML content is

```
print “Content-type: x-world/x-vrml\n\n”;
```

Then this script outputs the VRML header. All VRML contents begin with the standard VRML header, followed by the VRML file. Because most time a VRML file is very long and not possible be placed in a CGI/Perl script, the script also can open an exiting VRML file and output to the browser. The following script fragment shows this idea.

```
#!/usr/local/bin/perl

print “Content-type: x-world/x-vrml\n\n”;
open (GATE, ‘gate.wrl’)
while (<GATE>)
    print;
}
close(GATE);
```

The script above opens a VRML file called “gate.wrl” and then simply outputs the VRML file. The VRML header is not included in the script because the header does exist in the file “gate.wrl”.

4. PROBLEM STATEMENT AND PROJECT OBJECTIVE

The computer software geared for supporting collaborative work is expected to flourish because of the exploration of Internet and the fast advances in computer network technology [2]. The computer supported collaborative work (CSCW) environment allows multiple users to contribute to the same piece of work any time and any where. Combining the interactive, colorful 3D VRML worlds with computer supported collaborative work (CSCW) environment is interesting and appealing. Because a conventional CSCW environment normally supports a group of users working as a team in a two dimensional environment.

In order to support collaborative work environment using VRML, VRML worlds must provide multiple users access capability. But the problem is that “VRML 2.0 doesn’t provide explicit support for multiple users interacting in a single world; handling multi-user environments is scheduled to be part of the VRML 3.0 specification.” [3].

CGI/Perl script can create a VRML world interactively on the fly. The objective of this project is to implement CSCW environment using CGI/Perl script and simulate the multiple users VRML world to support collaborative work through World Wide Web.
5. IMPLEMENTATION OF VRML CSCW ENVIRONMENT

5.1 Architecture of VRML CSCW Environment

The general architecture for using CGI/Perl through HTTP server is illustrated in Figure 6.

Figure 6. General Architecture for CGI/Perl with Web Server

CGI serves as a connection between the browser and HTTP server. Processing requests flow from the HTML documents through the CGI, where Perl programs receive the requests and process the requests. The usual response provided by a CGI/Perl script takes the form of another HTML page, which is frequently constructed “on the fly” to meet the specific needs of the client [8].

The HTML documents and the CGI programs usually are located on the same server. CGI/Perl program also resides in a separate directory on the server such as /cgi-bin/. Because CGI programs sometimes must access server files and other resources, security measures with
rules and procedures for uploading and using CGI programming is a highly priority for Web site system administrators.

Based on the general architecture, the architecture of VRML CSCW environment is displayed in Figure 7.

![Client/Server Architecture of the VRML CSCW Environment](image)

A user at the client side needs a VRML authoring tool to create a VRML object and save it to a VRML file. The VRML file can be delivered to the HTTP server and processed by a CGI/Perl program residing on the server. After HTTP server completes the process, the VRML file is sent back to the client side for displaying. The user also needs a HTTP browser with a VRML plug-in to view the updated VRML world.
5.2 Implementation of VRML CSCW Environment

Suppose we have an architecture design team of four architects to design a temple with a garden. The garden includes a gate, a pavilion, and some trees. According to the agreement, one architect is responsible for designing the temple, one for the garden gate, one for the pavilion, and one for landscape using trees. They all use the same VRML authoring tool to create the VRML objects and upload their VRML files to the server. The CGI/Perl script on the server assembles all parts together to give each architect a complete picture of the project.

In this VRML CSCW environment, each client needs to send the VRML file to the server. This goal can be achieved by implementing an upload mechanism using CGI/Perl. The following program fragment uses an HTML Form to upload a text file.

```html
<FORM ENCTYPE="multipart/form-data"
ACTION="http://roadrunner.eng.auburn.edu/cgi-bin/cse532/liyanxi/file-upload"
METHOD="POST">
Filename to Be Saved:
<INPUT TYPE="TEXT" NAME="save-as-filename" SIZE=27>
File to Be Uploaded:
<INPUT TYPE="FILE" NAME="upload-file" SIZE=27>
<INPUT TYPE="SUBMIT" VALUE="Upload VRML File!">
</FORM>
```

The Form prompts user to input file name to be saved in the server and allows user to browse the file to be uploaded. When the user clicks button labeled “Upload VRML File”, a CGI/Perl script on the server called “file-upload” processes the user’s request and saves the file to a predetermined directory. After all VRML files are available on the server, a CGI/Perl script
called “assembler” combines all parts together to generate a whole picture. The skeleton of “assembler” is as follows:

```perl
#!/usr/local/bin/perl

# print the VRML MIME type.
sub MIMEHeader { return "Content-type: x-world/x-vrml\n\n";
}
sub VRMLHeader { return "#VRML V2.0 utf8\n\n";
}
sub TransTop { return "Transform \n\n";
}
sub ChildTop { return "children \n\n";
}
sub TransBottom { return "\] \n\n";
}

# Change all VRML files' permissions and
# make sure they are readable by CGI/Perl program.
chmod (0644, "../vrml/gate.wrl");
chmod (0644, "../vrml/temple.wrl");
chmod (0644, "../vrml/pavilion.wrl");
chmod (0644, "../vrml/tree.wrl");

# print MIME type and VRML header
print (&MIMEHeader);
print (&VRMLHeader);

# Transform the temple object to its proper position
print (&TransTop);
print ("translation 0 0 0\n");
print ("rotation 0 0 0 0\n");
print ("scale 1.3 1.3 1.3\n");
print (&ChildTop);
# Open and output VRML object temple
open(TEMPLE, '../vrml/temple.wrl');
while(<TEMPLE>) {
    # Looking for a special string pattern with VRML header
    # and the line with this pattern is deleted because of only
    # one header needed.
    if (/\d+\.\d+\.\d+\.\d+/) { }
    else{ print; }
}
close(TEMPLE);
print(&TransBottom);

# Transform the gate object to its proper position
print (&TransTop);
```
This script first defines some useful subroutines to output the VRML MIME type, VRML header, and other symbols for Transform node. Then it changes the permissions of VRML files to make sure they are readable by the script. After it outputs the VRML header, the script opens and outputs a VRML file and transforms the VRML world to its proper position. This script can open unlimited number of VRML files and combine them together into a single VRML world. For example, this script opens a VRML file, gate.wrl and combines it with temple.wrl.

Because each VRML file has one header and we just need one header for the final VRML world, all headers in all VRML files need to be deleted. For this purpose a special string 555.555.555.555 is put in the line with the VRML header as follows:

```
#VRML V2.0 utf 8    #555.555.555.555
```

When the script reads a VRML file, it tries to find the line with this special string. If found, the line with the special string is skipped. The following program fragment is to designed for this purpose.

```python
open(TEMPLE, '../vrml/temple.wrl');
while(<TEMPLE>) {
    if (/\d+\.\d+\.\d+\.\d+/) { }
    else{ print;}  
}
```
The HTML program to call “assembler” script is as follows:

```html
<FORM METHOD = "GET" ACTION = "display.html" TARGET = "cscw2">
<input TYPE = "submit" VALUE = "Update">
</FORM>
```

A program fragment in display.html is as follows:

```html
<EMBED SRC = "http://roadrunner.eng.auburn.edu/cgi-bin/cse532/liyanxi/assembler"
WIDTH = 680, HEIGHT = 275 BORDER = 0>
```

When the user clicks a button labeled “Update”, the HTML file, display.html, is executed. Then the request is passed to a CGI/Perl script, assembler, on the server, roadruuner. The returned VRML world is displayed on a VRML plug-in (Cosmo Player in this project), and it is embedded in a HTML browser (Netscape Navigator 4.0 in this project).

Of course the collaborative working environment needs some kind of communication mechanism to allow all the architects to communicate with each other. This project also implements a “Talk Room” to increase the understanding among the team members. All architects in the project can talk to each other in the “Talk Room” and also can find out who are in the room at any moment.

Figure 8 shows the initial window of the user interface of VRML CSCW environment. There are totally four windows inside: the top left window is the Control window which allows user to control other windows; the top right window is the Display window which displays a VRML world; the bottom left window is the Upload window which allows user to upload VRML files; the bottom right window is the “Talk Room” to enable the communication among the architects.
Figure 8. VRML CSCW User Interface (Initial Window)
After the user clicks on the buttons in the Control window, the contents can be displayed as shown in Figure 9. Because VRML allows user to walk through the world, Figure 10 is a close view of the temple achieved by walking through the gate to the temple.

The main features of the VRML CSCW environment is listed in Table 2.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upload Files</td>
<td>User can upload text files by browsing the file directory and specifying the file name to be saved.</td>
<td>CGI/Perl</td>
</tr>
<tr>
<td>Talk Room</td>
<td>Users can talk to each other and monitor who is in the room.</td>
<td>Java Applet</td>
</tr>
<tr>
<td>Password</td>
<td>Users must have a username/password to gain access. users can view VRML world using Cosmo Player 2.0 plug-in.</td>
<td>CGI/Perl</td>
</tr>
<tr>
<td>VRML Browser</td>
<td>Cosmo Player is written in Java. 3D worlds are written in VRML.</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Features of VRML CSCW Environment

5.3 Security

The security issues should be carefully considered for all CSCW environment. The basic security policy requires that only authorized users can access the system. This project also implements a password mechanism to protect the VRML CSCW environment. Users must have a username/password to gain access as shown in Figure 11. The form is as follows:

```html
<FORM ACTION = "cscw.html">
User Name : <INPUT TYPE = "TEXT" NAME = "user" VALUE = ">
Password : <INPUT TYPE = "PASSWORD" NAME = "usrpwd" VALUE = ">
< INPUT TYPE ="SUBMIT" VALUE="Login">
< INPUT TYPE ="RESET" VALUE="Reset">
```
Figure 9. VRML CSCW User Interface (Working Window)
Figure 10. VRML CSCW user Interface
Figure 11. VRML CSCW Log-in Window
The SUBMIT will send the form INPUT to the ACTION script. The RESET will return any input on the form to their preset values, or clear the form if no presets were given.

Due to security issues associated with system encrypted passwords, the password can’t be decrypted without the knowledge of the original transformation keys. Only the system administrator can control this feature. To use VRML CSCW practically the system administrator should add this feature, so a CGI/Perl script can use username and password to identify users. Here is fragment of the CGI/Perl script:

```perl
if ($user == "") {
   printf (“Username can’t be null. Try again!”);  
   exit(1);
}
elsif ($user = "tiger") {
   printf (“Welcome to VRML CSCW Project.”);  
}
...
6. FUTURE OF VRML CSCW ENVIRONMENT

VRML 3.0 specification is under discussion and the VRML networking API in the new version will allow clients to interact with the VRML world. This project implements a framework for computer supported collaborative work environment using CGI/Perl.

The Script node in VRML provides a way for designer to include more complex forms of interaction. A Script node receives incoming events, performs processing, and generates outgoing events. The part of the node that performs the processing is called a “script” and can be written in any programming language the VRML browser supports. A browser can not execute a script in a language it doesn’t support. The languages most likely to be supported is Java [3].

Java is an object-oriented, interpreted, portable, high-performance, and dynamic language. The java.net package provides a powerful and flexible infrastructure for networking. The combination of VRML and Java will enable a very powerful multi-user environment with strong networking capability. For example the Socket class in Java allows you to connect to a specified port on a specified Internet host and read and write data using the InputStream and OutputStream classes of the java.io package. If you want to implement a server to accept connections from clients, you can use the related ServerSocket class. Both functions will enhance the collaborative working environment [11].
7. REFERENCES

[1] Cosmo Player (http://vrml.sgi.com)

CASE and CAD Applications”, Final Report to NIST, November 23, 1996.


Inc. (1996).


(1996).

Inc. (1996).

[9] VRML Repository (http://www.sdsc.edu/vrml/)

[10] VRML Consortium (http://www.vrml.org)

8. APPENDICES

Appendix A. CGI/Perl Program - assembler

Appendix B. HTML Program

(1) control.html
(2) display.html
(3) upload.html
(4) cscw.html

Appendix C. VRML Program

(1) temple.wrl
(2) gate.wrl
Appendix A

Assembler

#!/usr/local/bin/perl

############################################################
##                              Assembler
## The CGI/Perl program to translate all VRML objects to their proper
## positions and assemble these VRML objects into one VRML world.
##
## Writen by Yanxin Li, November 1997
############################################################

# unbuffer output stream
$| = 1;

# print the VRML MIME type.
sub MIMEHeader {
    return "Content-type: x-world/x-vrml\n\n";
}

# print VRML header
sub VRMLHeader {
    return "#VRML V2.0 utf8\n\n";
}

sub TransTop {
    return "Transform \n";
}

sub ChildTop {
    return "children \n";
}

sub TransBottom {
    return "\n\n";
}

# Change all VRML files' permissions and
# make sure they are readable by CGI/Perl program.
chmod (0644, "../vrml/gate.wrl");
chmod (0644, "../vrml/temple.wrl");
chmod (0644, "../vrml/pavilion.wrl");
chmod (0644, "../vrml/tree.wrl");

# print MIME type and VRML header
print (&MIMEHeader);
print (&VRMLHeader);

# Transform the temple object to its proper position
print (&TransTop);
print ("translation 0 0 0\n");
print ("rotation 0 0 0\n");
print ("scale 1.3 1.3 1\n");
print (&ChildTop);

# Open and output VRML object, temple
open(TEMPLE, '../vrml/temple.wrl');
while(<TEMPLE>) {
    # Looking for a special string pattern with VRML header
    # and the line with this pattern is omitted because of only
    # one header needed.
    if (\d+\.\d+\.\d+\.\d+/) { }
    else{
        print;
    }
}
close(TEMPLE);
print(&TransBottom);

# Transform the gate object to its proper position
print (&TransTop);
print ("translation -1.8 -1 2\n");
print ("rotation 0 0 0\n");
print ("scale 0.8 0.8 0.8\n");
print (&ChildTop);

# Open and output VRML object gate
open(GATE,'../vrml/gate.wrl');
while(<GATE>) {
    # Looking for a special string pattern with VRML header
    # and the line with this pattern is omitted because of only
#one header needed.
if (/\d+/\d+/\d+/\d+/ { })
else{
    print;
}
}
close(GATE);
print(&TransBottom);

#Transform the pavilion object to its proper position
print (&TransTop);
print ("translation 40 0 -40\n");
print ("rotation 0 0 0 0\n");
print ("scale 1 1 1\n");
print (&ChildTop);

#Open and output VRML object pavilion
open(PAVILION, '../vrml/pavilion.wrl');
while(<PAVILION>) {
    #Looking for a special string pattern with VRML header
    #and the line with this pattern is omitted because of only
    #one header needed.
    if (/\d+/\d+/\d+/\d+/ { })
        else{
            print;
        }
}
close(PAVILION);
print(&TransBottom);

#Transform the tree object to its proper position
print (&TransTop);
print ("translation -10 0 -10\n");
print ("rotation 0 0 0 0\n");
print ("scale 1 0.7 1\n");
print (&ChildTop);

#Open and output VRML object tree
open(TREE, '../vrml/tree.wrl');
while(<TREE>) {
    #Looking for a special string pattern with VRML header
    #and the line with this pattern is omitted because of only
    #one header needed.
if (\d+\d+\d+\d+) { }
    else{
        print;
    }
} close(TREE);
prient(&TransBottom);

#Transform the tree object to its proper position
print (&TransTop);
prient ("translation -10 0 -30\n");
prient ("rotation 0 0 0 0\n");
prient ("scale 1 0.7 1\n");
prient (&ChildTop);

#Open and output VRML object tree
open(TREE, '../vrml/tree.wrl');
while(<TREE>) {
    #Looking for a special string pattern with VRML header
    #and the line with this pattern is omitted because of only
    #one header needed.
    if (\d+\d+\d+\d+) { }
    else{
        print;
    }
} close(TREE);
prient(&TransBottom);

#Transform the tree object to its proper position
print (&TransTop);
prient ("translation 10 0 -10\n");
prient ("rotation 0 0 0 0\n");
prient ("scale 1 0.7 1\n");
prient (&ChildTop);

#Open and output VRML object tree
open(TREE, '../vrml/tree.wrl');
while(<TREE>) {
    #Looking for a special string pattern with VRML header
    #and the line with this pattern is omitted because of only
    #one header needed.
    if (\d+\d+\d+\d+) { }
    else{
print;
}
}
close(TREE);
print(&TransBottom);

#Transform the tree object to its proper position
print (&TransTop);
print ("translation 10 0 -30\n");
print ("rotation 0 0 0 0\n");
print ("scale 1 0.7 1\n");
print (&ChildTop);

#Open and output VRML object tree
open(TREE,'../vrml/tree.wrl');
while(<TREE>) {
    #Looking for a special string pattern with VRML header
    #and the line with this pattern is omitted because of only
    #one header needed.
    if (/\d+\.\d+\.\d+\.\d+/) { }
    else{
        print;
    }
}
close(TREE);
print(&TransBottom);
Appendix B

cscw.html

<html>
<head>
<title>VRML CSCW Project</title>
</head>
<base href="http://www.eng.auburn.edu/~liyanxi/">
<FRAMESET rows="65%, 35%">
  <FRAMESET cols="10%, 90%">
    <FRAME SRC="cscw/control.html" name="cscw1" border=1 noresize>
    <FRAME SRC="cscw/window.html" name="cscw2" border=1 noresize>
  </FRAMESET>
  <FRAMESET cols="50%, 50%">
    <FRAME SRC="cscw/window.html" name="cscw3" border=1 noresize>
    <FRAME SRC="cscw/window.html" name="cscw4" border=1 noresize>
  </FRAMESET>
</FRAMESET>
</html>
<HTML>
<HEAD>
<title>VRML CSCW Project</title>
</HEAD>
<body background="../picture/marble.gif">
<p><br></p>
<form action="display.html" target="cscw2" method="GET">
<input type="submit" value="Update">
</form>
<p></p>
<form action="upload.html" target="cscw3" method="GET">
<input type="submit" value="Upload">
</form>
<p></p>
<form action="talk.html" target="cscw4" method="GET">
<input type="submit" value="Talk">
</form>
<p></p>
<form action="talk.html" target="cscw4" method="GET">
<input type="submit" value="Help">
</form>
</body>
</HTML>
<HTML>
<HEAD>
<TITLE>VRML CSCW Project</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFF" background="../picture/marble.gif">
<FORM ENCTYPE="multipart/form-data" ACTION="http://roadrunner.eng.auburn.edu/cgi-bin/cse532/liyanxi/file-upload" METHOD="POST">
<bold><font color="FF0000">Filename to Be Saved:</font></bold><br>
<INPUT TYPE="TEXT" NAME="save-as-filename" SIZE=27><br>
<bold><font color="FF0000">File to Be Uploaded:</font></bold><br>
<INPUT TYPE="FILE" NAME="upload-file" SIZE=27><br><br>
<center>
<INPUT TYPE="SUBMIT" VALUE="Upload VRML File!">
</center>
</FORM>
</BODY>
</HTML>
#VRML V2.0 utf8  #555.555.555.555

#Specifying Ground Color
Background {
 groundColor [ 
    0 0.6 0,
    0 0.8 0,
    0 1 0
  ]
groundAngle [ 
    1.05, 1.65
  ]
}

#Specifying Sky Color
skyColor [ 
  0 0 0.2,
  0.1 0.1 0.8,
  0.2 0.2 1
  ]
skyAngle [ 
  1.05, 1.57
  ]
}

#specifying the road from gate to temple
Transform { 
  translation 0 -3 -20
  children [ 
    Shape { 
      appearance Appearance { 
        material Material { 
          diffuseColor 0.9 0.9 0.9 
          shininess 1
        }
      }
    }
  ]
}
geometry Box {
    size 2 0.05 80
}

# specifying the temple.
Transform {
    translation 0 0 -60
    scale 3 3 3
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 1 0.5 0
                    shininess 1
                }
            }
        }
    ]
}

geometry IndexedFaceSet {
    coord Coordinate {
        point [
            -1.21126 -0.933773 -1.21126,
            -0.980099 -0.801326 -0.980099,
            -1.21126 -0.933773 -1.21126,
            -1.21126 -0.933773 1.21126,
            -0.980099 -0.801326 -0.980099,
            -0.980099 -0.801326 -0.980099,
            -1.21126 -0.933773 1.21126,
            -0.980099 -0.801326 0.980099,
            -1.21126 -0.933773 1.21126,
            1.21126 -0.933773 1.21126,
            -0.980099 -0.801326 0.980099,
            -0.980099 -0.801326 0.980099,
            1.21126 -0.933773 1.21126,
            0.980099 -0.801326 0.980099,
            0.980099 -0.801326 0.980099,
            0.980099 -0.801326 -0.980099,
            0.944536 -0.668872 0.944536,
            0.944536 -0.668872 0.944536,
            0.980099 -0.801326 -0.980099,
            -0.980099 -0.801326 -0.980099,
            0.944536 -0.668872 -0.944536,
            0.944536 -0.668872 -0.944536,
        ]
    }
}
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.660033</td>
<td>0.192052</td>
<td>0.660033</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.37553</td>
<td>0.443707</td>
<td>0.37553</td>
</tr>
<tr>
<td>0.37553</td>
<td>0.443707</td>
<td>0.37553</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.37553</td>
<td>0.443707</td>
<td>0.37553</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.37553</td>
<td>0.443707</td>
<td>0.37553</td>
</tr>
<tr>
<td>0.37553</td>
<td>0.443707</td>
<td>0.37553</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.178807</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>-0.553344</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>-0.553344</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>-0.553344</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>0.553344</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>0.553344</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.553344</td>
<td>0.549667</td>
<td>0.553344</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.553344</td>
<td>0.549667</td>
<td>0.553344</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>0.411093</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>0.357749</td>
<td>0.721852</td>
<td>0.357749</td>
</tr>
<tr>
<td>0.357749</td>
<td>0.721852</td>
<td>0.357749</td>
</tr>
<tr>
<td>0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>0.357749</td>
<td>0.721852</td>
<td>-0.357749</td>
</tr>
<tr>
<td>0.357749</td>
<td>0.721852</td>
<td>-0.357749</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
<tr>
<td>-0.357749</td>
<td>0.721852</td>
<td>-0.357749</td>
</tr>
<tr>
<td>-0.411093</td>
<td>0.642382</td>
<td>-0.411093</td>
</tr>
</tbody>
</table>
0.108809 0.708607 0.108809,
0.108809 0.708607 -0.108809,
0.0199017 1 0.0199017,
0.0199017 1 0.0199017,
0.108809 0.708607 -0.108809,
0.0199017 1 -0.0199017,
0.108809 0.708607 -0.108809,
-0.108809 0.708607 -0.108809,
0.0199017 1 -0.0199017,
0.0199017 1 -0.0199017,
-0.108809 0.708607 -0.108809,
-0.0199017 1 -0.0199017,
-0.108809 0.708607 -0.108809,
-0.108809 0.708607 0.108809,
-0.0199017 1 -0.0199017,
-0.0199017 1 -0.0199017,
-0.108809 0.708607 0.108809,
-0.0199017 1 0.0199017,
-0.108809 0.708607 0.108809,
0.108809 0.708607 0.108809,
-0.0199017 1 0.0199017,
-0.0199017 1 0.0199017,
0.108809 0.708607 0.108809,
0.0199017 1 0.0199017
}
}
}
coordIndex [ 
    39, 40, 41, -1,
    42, 43, 44, -1,
    45, 46, 47, -1,
    48, 49, 50, -1,
    51, 52, 53, -1,
    54, 55, 56, -1,
    57, 58, 59, -1,
    60, 61, 62, -1,
    63, 64, 65, -1,
    66, 67, 68, -1,
    69, 70, 71, -1,
    72, 73, 74, -1,
    75, 76, 77, -1,
    78, 79, 80, -1,
    81, 82, 83, -1,
    84, 85, 86, -1,
    87, 88, 89, -1,
    90, 91, 92, -1,
normalPerVertex FALSE
solid FALSE
creaseAngle 0.523599

Shape {
appearance Appearance {
material Material {
diffuseColor 0 1 0
shininess 1
}
}
}
geometry IndexedFaceSet {
    coord Coordinate {
        point [
            -1.47798 -1 1.47798,
            1.47798 -1 -1.47798,
            1.47798 -1 1.47798,
            -1.47798 -1 -1.47798
        ]
    }
    coordIndex [
        0, 1, 2, -1,
        0, 3, 1, -1
    ]
    normalPerVertex FALSE
    solid       FALSE
    creaseAngle 0.523599
}
}
Shape {
    appearance Appearance {
        material Material {
            diffuseColor 1 0 0
            shininess 1
        }
    }
    geometry IndexedFaceSet {
        coord Coordinate {
            point [
                0.0199017 1 0.0199017,
                0.0199017 1 -0.0199017,
                -0.0199017 1 0.0199017,
                -0.0199017 1 -0.0199017
            ]
        }
        coordIndex [
            0, 1, 2, -1,
            1, 3, 2, -1
        ]
        normalPerVertex FALSE
        solid       FALSE
        creaseAngle 0.523599
    }
}
]
DEF GatePole Transform {
    translation -1 -0.6 -10
    scale 2 2 2
    children [
        Shape {
            appearance Appearance {
                material Material {
                    ambientIntensity 0.249
                    diffuseColor     1 0 0
                }
            }
            geometry Cylinder {
                radius 0.4
                height 4
                side TRUE
                top TRUE
                bottom TRUE
            }
        }
        Transform {
            translation 0 2.3 0
            children [
                Shape {
                    appearance Appearance {
                        material Material {
                            ambientIntensity 0.249
                            diffuseColor     1 1 1
                        }
                    }
                    geometry Sphere {
                        radius 0.4
                    }
                }
            ]
        }
    ]
}
Transform {
  translation 20 0 0
  children USE Pole
}

#Fencing Horizontal Bars
DEF Bar Transform {
  translation 17.5 0.5 -10
  scale 2 4 2
  rotation 0 0 1 1.57
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 1 1 1
          shininess 1
        }
      }
      geometry Cylinder {
        radius 0.2
        height 6
        side TRUE
        top TRUE
        bottom TRUE
      }
    }
  ]
}

Transform {
  translation 0 -3 0
  children USE Bar
}

Transform {
  translation -5 -5 -5
  scale 0.1 0.1 0.1
  children [
    Shape {
      appearance Appearance {
        material Material {
          ambientIntensity 0.249
          diffuseColor 0.996 0.996 0.996
        }
      }
    }
  ]
}
geometry IndexedFaceSet {
  coord Coordinate {
    point [
      #all the points here are omitted for simplicity
    ]
  }
}

coordIndex [
  #all the points here are omitted for simplicity
]
normalPerVertex FALSE
ccw FALSE
creaseAngle 0.5
}
Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.249
      diffuseColor 0.996 0.996 0.996
    }
  }
}
geometry IndexedFaceSet {
  coord Coordinate {
    point [
      -36.01 24 -77.945, 
      -38.546 24 -72.507, 
      10.393 24 -49.686, 
      12.929 24 -55.124, 
      12.929 18 -55.124, 
      -36.01 18 -77.945, 
      10.393 18 -49.686, 
      -38.546 18 -72.507
    ]
  }
}

coordIndex [
  0, 1, 2, 3, -1, 
  4, 5, 0, 3, -1, 
  6, 4, 3, 2, -1, 
  7, 6, 2, 1, -1, 
  5, 7, 1, 0, -1, 
  4, 6, 7, 5, -1
]
Anchor {
  description "VRML CSCW"
  url "http://www.eng.auburn.edu/~liyanxi"
  bboxSize 0 0 0
  children [
    Shape {
      appearance Appearance {
        material Material {
          ambientIntensity 0.0993333
          diffuseColor 0.796 0.398
        }
      }
      geometry IndexedFaceSet {
        coord Coordinate {
          point [
            #all the points here are omitted for simplicity
          ]
        }
        coordIndex [
          #all the points here are omitted for simplicity
        ]
        normalPerVertex FALSE
        ccw FALSE
        creaseAngle 0.5
      }...
    }
  ]
}

Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.0993333
      diffuseColor 0.796 0.398
    }
  }
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
        #all the points here are omitted for simplicity
      ]
    }
    coordIndex [
      #all the points here are omitted for simplicity
    ]
    normalPerVertex FALSE
    ccw FALSE
    creaseAngle 0.5
  }
}
Shape {
    appearance Appearance {
        material Material {
            ambientIntensity 0.0413333
            diffuseColor 0.496 0
        }
    }
}

geometry IndexedFaceSet {
    coord Coordinate {
        point [
            #all the points here are omitted for simplicity
        ]
        normalPerVertex FALSE
        ccw FALSE
        ccw FALSE
        creaseAngle 0.5
    }
}

Shape {
    appearance Appearance {
        material Material {
            ambientIntensity 0.0413333
            diffuseColor 0.496 0
        }
    }
}

geometry IndexedFaceSet {
    coord Coordinate {
        point [
            #all the points here are omitted for simplicity
        ]
        normalPerVertex FALSE
        ccw FALSE
        ccw FALSE
        creaseAngle 0.5
    }
}

Shape {
    appearance Appearance {
        material Material {
            ambientIntensity 0.0413333
            diffuseColor 0.496 0
        }
    }
}
ambientIntensity 0.0413333
diffuseColor 0 0.496 0
}
g
geometry IndexedFaceSet {
  coord Coordinate {
    point [
      -22.44 29.964 -69.41,
      -22.722 30.999 -71.17,
      -19.234 37.306 -66.812,
      -23.326 29.982 -67.55,
      -24.863 31.043 -66.679,
      -26.15 32.525 -67.306,
      -26.433 33.56 -69.066,
      -25.546 33.542 -70.926,
      -24.009 32.481 -71.797
    ]
  }
  coordIndex [
    0, 1, 2, -1,
    3, 0, 2, -1,
    4, 3, 2, -1,
    5, 4, 2, -1,
    6, 5, 2, -1,
    7, 6, 2, -1,
    8, 7, 2, -1,
    1, 8, 2, -1,
    0, 3, 4, 5, 6, 7, 8, 1, -1
  ]
  normalPerVertex FALSE
  ccw FALSE
  creaseAngle 0.5
}
}

Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.249
      diffuseColor 0.996 0.996 0.996
    }
  }
  geometry IndexedFaceSet {
    coord Coordinate {

point [
#all the points here are omitted for simplicity
]
}
coordIndex [
#all the points here are omitted for simplicity
]
normalPerVertex FALSE
cw FALSE
creaseAngle 0.5
}
}
Shape {
appearance Appearance {
    material Material {
        ambientIntensity 0.249
diffuseColor 0.996 0.996 0.996
    }
}
}
geometry IndexedFaceSet {
    coord Coordinate {
        point [
            -52.025 88.157 -92.033,
            -54.561 88.157 -86.595,
            -49.451 90.209 -84.213,
            -46.916 90.209 -89.65,
            -17.161 0 -75.777,
            -22.271 -2.051 -78.159,
            -19.697 0 -70.339,
            -24.807 -2.051 -72.722
        ]
}
coordIndex [
0, 1, 2, 3, -1,
4, 5, 0, 3, -1,
6, 4, 3, 2, -1,
7, 6, 2, 1, -1,
5, 7, 1, 0, -1,
4, 6, 7, 5, -1
]
Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.0413333
      diffuseColor 0 0.496 0
    }
  }
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
        #all the points here are omitted for simplicity
      ]
    }
    coordIndex [
      #all the points here are omitted for simplicity
    ]
  }
}

Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.0413333
      diffuseColor 0 0.496 0
    }
  }
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
        #all the points here are omitted for simplicity
      ]
    }
    coordIndex [
      #all the points here are omitted for simplicity
    ]
  }
  normalPerVertex FALSE
  ccw FALSE
  creaseAngle 0.5
}
}
coordIndex [
    #all the points here are omitted for simplicity
]
normalPerVertex FALSE
ccw FALSE
creaseAngle 0.5
}
}

Shape {
appearance Appearance {
    material Material {
        ambientIntensity 0.0413333
diffuseColor 0 0.496 0
    }
}
}
geometry IndexedFaceSet {
    coord Coordinate {
        point [
            -14.154 49.358 -59.478,
            -12.641 50.411 -60.4,
            -9.055 47.094 -55.997,
            -15.009 49.376 -57.604,
            -14.706 50.455 -55.874,
            -13.422 51.962 -55.303,
            -11.909 53.015 -56.225,
            -11.054 52.996 -58.1,
            -11.357 51.918 -59.829
        ]
    }
    coordIndex [
        0, 1, 2, -1,
        3, 0, 2, -1,
        4, 3, 2, -1,
        5, 4, 2, -1,
        6, 5, 2, -1,
        7, 6, 2, -1,
        8, 7, 2, -1,
        1, 8, 2, -1,
        0, 3, 4, 5, 6, 7, 8, 1, -1
    ]
}
normalPerVertex FALSE
ccw       FALSE
craseAngle 0.5
}
}
Shape {
appearance Appearance {
material Material {
    ambientIntensity 0.0413333
    diffuseColor     0 0.496 0
}
}
geometry IndexedFaceSet {
    coord Coordinate {
        point [
            -17.383 27.027 -60.984,
            -15.691 27.852 -61.823,
            -12.831 23.546 -57.758,
            -18.235 27.041 -59.108,
            -17.749 27.887 -57.293,
            -16.208 29.069 -56.602,
            -14.516 29.895 -57.441,
            -13.664 29.88 -59.317,
            -14.151 29.034 -61.132
        ]
    }
    coordIndex [
        0, 1, 2, -1,
        3, 0, 2, -1,
        4, 3, 2, -1,
        5, 4, 2, -1,
        6, 5, 2, -1,
        7, 6, 2, -1,
        8, 7, 2, -1,
        1, 8, 2, -1,
        0, 3, 4, 5, 6, 7, 8, 1, -1
    ]
    normalPerVertex FALSE
ccw       FALSE
ccw       FALSE
craseAngle 0.5
}
}
Shape {
    appearance Appearance {
        material Material {
            ambientIntensity 0.0413333
            diffuseColor     0.496 0
        }
    }
}

geometry IndexedFaceSet {
    coord Coordinate {
        point [
            6.253 54.587 -53.548,
            7.025 54.481 -53.901,
            7.332 54.492 -54.7,
            6.993 54.615 -55.476,
            6.208 54.778 -55.775,
            5.436 54.885 -55.422,
            5.13 54.873 -54.623,
            5.468 54.75 -53.847,
            -0.665 19.903 -56.707,
            0.119 19.741 -56.408,
            -1.004 20.027 -57.483,
            -0.697 20.038 -58.282,
            0.074 19.931 -58.636,
            0.859 19.769 -58.336,
            1.197 19.645 -57.56,
            0.891 19.634 -56.761
        ]
    }
    coordIndex [
        0, 1, 2, 3, 4, 5, 6, 7, -1,
        8, 9, 0, 7, -1,
        10, 8, 7, 6, -1,
        11, 10, 6, 5, -1,
        12, 11, 5, 4, -1,
        13, 12, 4, 3, -1,
        14, 13, 3, 2, -1,
        15, 14, 2, 1, -1,
        9, 15, 1, 0, -1,
        8, 10, 11, 12, 13, 14, 15, 9, -1
    ]
    normalPerVertex FALSE
    ccw      FALSE
    creaseAngle 0.5
Shape {
  appearance Appearance {
    material Material {
      ambientIntensity 0.0413333
      diffuseColor 0 0.496 0
    }
  }
  geometry IndexedFaceSet {
    coord Coordinate {
      point [
        -33.825 47.067 -69.754,
        -33.561 47.015 -70.492,
        -33.718 46.729 -71.084,
        -34.205 46.378 -71.183,
        -34.737 46.166 -70.731,
        -35.002 46.219 -69.994,
        -34.844 46.504 -69.402,
        -34.357 46.856 -69.303,
        -19.874 24.836 -62.55,
        -19.343 25.048 -63.002,
        -20.362 24.485 -62.649,
        -20.519 24.199 -63.241,
        -20.255 24.147 -63.979,
        -19.723 24.359 -64.43,
        -19.236 24.71 -64.331,
        -19.078 24.996 -63.739
      ]
      coordIndex [
        0, 1, 2, 3, 4, 5, 6, 7, -1,
        8, 9, 0, 7, -1,
        10, 8, 7, 6, -1,
        11, 10, 6, 5, -1,
        12, 11, 5, 4, -1,
        13, 12, 4, 3, -1,
        14, 13, 3, 2, -1,
        15, 14, 2, 1, -1,
        9, 15, 1, 0, -1,
        8, 10, 11, 12, 13, 14, 15, 9, -1
      ]
      normalPerVertex FALSE
    }
  }
}
Anchor {
    description "VRML CSCW"
    url "http://www.eng.auburn.edu/~liyanxi"
    bboxSize 0 0 0
    children [
        Shape {
            appearance Appearance {
                material Material {
                    ambientIntensity 0.249
                    diffuseColor 0.996 0.996 0.996
                }
            }
            geometry IndexedFaceSet {
                coord Coordinate {
                    point [
                        -96.925 38.461 -89.649, 
                        -101.523 33.846 -89.247, 
                        -96.925 29.23 -89.649, 
                        -73.936 29.23 -91.66, 
                        -73.936 38.461 -91.66, 
                        -74.138 38.461 -93.959, 
                        -97.126 38.461 -91.948, 
                        -74.138 29.23 -93.959, 
                        -97.126 29.23 -91.948, 
                        -101.724 33.846 -91.546
                    ]
                }
                coordIndex [
                    0, 1, 2, 3, 4, -1,
                    5, 6, 0, 4, -1,
                    7, 5, 4, 3, -1,
                    8, 7, 3, 2, -1,
                    9, 8, 2, 1, -1,
                    6, 9, 1, 0, -1,
                    5, 7, 8, 9, 6, -1
                ]
                normalPerVertex FALSE
            } ccw FALSE
        }
    ]
}
creaseAngle 0.5

Shape {
appearance Appearance {
  material Material {
    ambientIntensity 0.149
    diffuseColor 0.796 0.597 0.398
  }
}

graph IndexedFaceSet {
  coord Coordinate {
    point [
      -71.638 40.769 -91.862,
      -71.839 40.769 -94.16,
      -74.138 40.769 -93.959,
      -73.936 40.769 -91.66,
      -73.936 -0.769 -91.66,
      -71.638 -0.769 -91.862,
      -74.138 -0.769 -93.959,
      -71.839 -0.769 -94.16
    ]
  }
  coordIndex [
    0, 1, 2, 3, -1,
    4, 5, 0, 3, -1,
    6, 4, 3, 2, -1,
    7, 6, 2, 1, -1,
    5, 7, 1, 0, -1,
    4, 6, 7, 5, -1
  ]
  normalPerVertex FALSE
  ccw FALSE
  creaseAngle 0.5
}
}