

CSE 587

Fall 2000

Project Report: Building an Information Kiosk

Anirudh Modi, Atin Bansal, Gaurav Kumar,
Yashmeet Khopkar and Prital Shah

December 12, 2000

1 Introduction

The aim of the project is to build a working *kiosk* for the Computer Science and Engineering Department. The *kiosk* will be unique in the sense that it will have a 3-D display for the building and the rooms as opposed to the widely used 2D displays for such devices, and will accept input not only via the touch-screen, but also via speech and gesture. The speech and gesture/vision input will be tightly integrated with the display module thus offering a new way to interact with the *kiosk* for gathering information. The display will be rendered in a photorealistic manner making it look very enchanting. The report describes ways in order to achieve this goal by the integration of various modules. The project aims to make the user familiar with the system as well as help him to get some information pertaining to the department and faculty in as little time as possible. The *kiosk* will be extremely user-friendly and will get the user to actually feel as if he or she were walking through the building.

2 Data organization using XML (Yashmeet Khopkar and Prital Shah)

The aim of the project is to come up with an Internet based *kiosk*. Through this *kiosk* the user will be able to navigate through the domain and handle the queries put forward by the user. The domain for this project is the 3rd floor of the Computer Science and Engineering Department. There is a display program, which continuously renders the 3rd floor. But this program requires the information of what to render. This information is given by our routine.

There are two modes in the system. The system can be in a navigation mode or it can be in query mode, but not in both. The user has to give a command to switch between the modes. When the user is in the navigation mode then the display program keeps us

updated with the position of the user. We keep track whether the user is in the corridor or in the room. If the user is in the corridor then we tell the display program to continue. But if the user is in the room then we compute the room in which the user is. Collect all the information that is required to render the room. And write that information in a meaningful format for the display program to generate the room. The information that is to be given is highlighted in the DTD that is attached. The information from the XML file for the room is extracted through the following steps:

1. Parse the XML file with a parser.
2. Get the root node of the XML file.
3. Traverse the entire tree for the node that contains the room information.
4. Once information is found, write it to a text file in a meaningful format.

Now, when the user is in query mode. The keywords that we get from the audio and visual inputs will enable the searching of the XML file. This searching is different because the results will be finer than the previous search results for the room. If the query is vague then the result may or may not exist. We will try to make the query algorithm robust to vague queries.

The following important considerations/studies were made in this work:

1. Study of feasibility of using XML as a data repository.
2. Which XML Parser to use?
3. Efficient Querying.
4. Navigation Mode.
5. Query Mode.

2.1 Study of feasibility of using XML as a data repository

We studied the feasibility of using the XML in our project. We could have used any other databases but there were some points in XML, which were advantageous. With XML we can organize the data in a very efficient way. This leads to a better search results in a Query. Also there is platform independent ness to be considered. With the help of XML we can change the application program smoothly. So if you decide to change your application from C to Java you will not need to do anything with the XML portion of your application. The interoperability property of XML is of great advantageous because of the need for distributed computing. And since the Kiosk has two platforms, one is the Solaris and the other is the Windows. So XML is ideally suited for the Kiosk.

2.2 Which XML Parser to use?

There are many parsers available. They are written in many languages like Java, C, C++, Perl, Python, Tcl and many others. So you can have your application on any platform using any of these parsers. And there need only be one XML file. We started off with C parser from Oracle, but since the Maze program was in C++ we could easily change the Parser. So we are now using the C++ Parser from Oracle. But the XML file is still the same. There are different types of parsers available. They can be categorized as validating and non - validating parsers. Also they can be categorized as SAX and DOM Parsers. A validating parser will use the DTD (Document Type Declaration), which describes the grammar of the XML file to validate the XML file. This is very much important from the application point of view. This ensures that the application gets a valid set of data. Also if the DTD is well defined between two applications then you can use the two XML files interchangeably for the two applications. The SAX is an Event based parser and the DOM is a Document Object Model based parser. We are using the validating parser which is DOM based.

2.3 Efficient Querying

By using XML the querying becomes efficient. This may not be apparent with this project, but in general because of the meaning attached to the name of the nodes in an XML file, the query can be specifically targeted at a particular section of the XML file. So the querying can be done efficiently. So when searching for the Keyword you can, not only search in the text of the XML file, but also in the tags of the XML file. Thus the search results generated are more relevant than the normal text based search results. There are also techniques to query the XML file, which narrows down the search result drastically and effectively. One of them is the X-SQL. But we are not using it in this project because of the small nature of the domain. Instead what we have done is that we have captured the DOM structure of the XML file and then queried this DOM hierarchy for the Key word. If the Key word is found then we get the parent node, which has the information to render the room.

2.4 Navigation mode

The system can work in two modes. One is the Navigation mode and the other is the Query mode. Both modes for now are mutually exclusive of each other. For the navigation mode the Maze program constantly calls one of our routine giving the present (x,y) coordinates. The routine checks whether the user is in the room or not. If the user is not in the room then it simply returns. But if the user is in the room, then another routine is called which will get the information about the room in which the user is. This information is then populated into a structure, which is returned to the Maze program. The Maze program picks up these values and renders the room with the help of this information.

2.5 Querying mode

The querying mode has been implemented. Essentially the way it is supposed work is that if a query from speech comes then the Keyword is searched against the XML file and if the Keyword exists then the relevant node that contained the information is returned as a structure. Right now this is fully functional.

This is the DTD that we have come up for the Department for now.

```
<!ELEMENT Department (Room+)>
<!ELEMENT Room (Room_Number,Professor,Introduction,Research,Texture,
Image,NumberWalls,RoomX1,RoomX2,RoomX3,RoomX4,RoomX5,RoomX6,
RoomX7,RoomX8,RoomX9,RoomX10,RoomY1,RoomY2,RoomY3,RoomY4,RoomY5,RoomY6,
RoomY7,RoomY8,RoomY9,RoomY10,CenterX,CenterY,TableX,TableY,NormalTable,
LightX,LightY)>

<!ELEMENT Room_Number (#PCDATA)>
<!ELEMENT Professor (#PCDATA)>
<!ELEMENT Introduction (#PCDATA)>
<!ELEMENT Research (#PCDATA)>
<!ELEMENT Texture (#PCDATA)>
<!ELEMENT Image (#PCDATA)>
<!ELEMENT NumberWalls (#PCDATA)>
<!ELEMENT RoomX1 (#PCDATA)>
<!ELEMENT RoomX2 (#PCDATA)>
<!ELEMENT RoomX3 (#PCDATA)>
<!ELEMENT RoomX4 (#PCDATA)>
<!ELEMENT RoomX5 (#PCDATA)>
<!ELEMENT RoomX6 (#PCDATA)>
<!ELEMENT RoomX7 (#PCDATA)>
<!ELEMENT RoomX8 (#PCDATA)>
<!ELEMENT RoomX9 (#PCDATA)>
<!ELEMENT RoomX10 (#PCDATA)>
<!ELEMENT RoomY1 (#PCDATA)>
<!ELEMENT RoomY2 (#PCDATA)>
<!ELEMENT RoomY3 (#PCDATA)>
<!ELEMENT RoomY4 (#PCDATA)>
<!ELEMENT RoomY5 (#PCDATA)>
<!ELEMENT RoomY6 (#PCDATA)>
<!ELEMENT RoomY7 (#PCDATA)>
<!ELEMENT RoomY8 (#PCDATA)>
<!ELEMENT RoomY9 (#PCDATA)>
<!ELEMENT RoomY10 (#PCDATA)>
<!ELEMENT CenterX (#PCDATA)>
<!ELEMENT CenterY (#PCDATA)>
<!ELEMENT TableX (#PCDATA)>
<!ELEMENT TableY (#PCDATA)>
<!ELEMENT NormalTable (#PCDATA)>
<!ELEMENT LightX (#PCDATA)>
<!ELEMENT LightY (#PCDATA)>
```

This is a sample XML file that conforms to the above DTD:

```
<?xml version="1.0"?>
<Department>
  <Room>
    <Room_Number>317</Room_Number>
    <Professor>Dr. Sharma</Professor>
    <Introduction_Text>This is Dr. Sharma's Room</Introduction_Text>
    <Research>Computer Vision</Research>
    <Publications>http://www.cse.psu.edu/gradbroc/gbFsharma.html</Publications>
```

```

<Texture>tile</Texture>
<Image>rsharma.jpg</Image>
    <NumberWalls>3</NumberWalls>
    <RoomX1>16.7</RoomX1>
    <RoomX2>16.7</RoomX2>
    <RoomX3>16.7</RoomX3>
    <RoomX4>18.5</RoomX4>
    <RoomX5>18.5</RoomX5>
    <RoomX6>18.5</RoomX6>
    <RoomY1>6.1</RoomY1>
    <RoomY2>3.1</RoomY2>
    <RoomY3>6.1</RoomY3>
    <RoomY4>6.1</RoomY4>
    <RoomY5>6.1</RoomY5>
    <RoomY6>3.1</RoomY6>
    <CenterX>3.0</CenterX>
    <CenterY>3.0</CenterY>
    <TableX>-1</TableX>
    <TableY>-1</TableY>
    <NormalTable>North</NormalTable>
    <LightX>1.0</LightX>
    <LightY>1.0</LightY>
</Room>
</Department>

```

3 Input, Display and Integration (Anirudh Modi)

The program will receive the input from the *kiosk*. There will be several input modalities provided using the touch-screen, voice commands or gesture (only keyboard-based and touch-screen based input modalities were implemented in this project). In the most basic form, input will be taken using the touch-screen which will have four arrows (front, back, left and right) showing on the screen. Pressing on each arrow will generate a corresponding movement in the position of the viewer in the 3-D world which depicts the interior of the building. This information is reflected in a sub-window that is present in the top 1/10th of the screen, and is updated in real-time. For the speech input, voice commands such as “*Move Front*”, “*Move Back*”, “*Move Left*”, “*Move Right*” get the same response. Additional voice commands such as “*Open the door*”, “*Go to Room 324*”, *etc.* will have the corresponding effect of opening the door or navigating the way automatically to the room requested respectively.

Owing to the unavailability of the *kiosk* for the most part of the project, only a keyboard based input using arrow keys was integrated, which is a simpler version of the touch-screen based input. Had things gone as planned, there was to be a marker on the touch screen that would have shown the directions in which the user could move. Touching one of the directions would move the user by a fixed quantity Δx or Δy in the direction of motion which was to be constantly written to an input file. The display program could then poll the input file several times a second to continuously update the display based on the users current position. Input was expected to be taken roughly 50 times a second so that there was no lag between the users entry and the subsequent output by the program. My job was prominently to write the basic display routine and to take each person’s work and integrate it in into the main project order to get a complete working *kiosk*. An impor-

tant part of my work was thus to co-ordinate and help the other people in the group get their work done.

The display program is written in C++ using the OpenGL API for graphics. Due to the use of open-source platform independent APIs like GLUT and OpenGL, the program can be compiled and run on any operating system. Currently, the program has been tested successfully under Linux, Solaris and Microsoft Windows 9x/NT/2000. This program takes the plan of the building as an input file which has a fixed format decided upon by us. The input is taken either by keyboard, mouse or another modality like touch, speech or gesture. Since the unit in question is a *kiosk*, the latter modalities based on the touch-screen, voice commands and hand gestures will be more appropriate. To simplify things, if another tracking/input program is used, the interpreted input data is constantly written to a fixed file. This file is then constantly read by the display program using polling. This works fairly well provided the number of times we poll the file every second is reasonable (50 in our case). Eventually, this process can be made more efficient by establishing a direct socket connection with the tracking program. From our standpoint, adding the latter input modalities is a simple extension to the keyboard input routine. Our work, for the most part, was independent of the input modalities, since it mainly involved the creation, display and update of the 3-D content.

The exact coordinates of the building is stored in a file beforehand in a specific file format. The corridor is stored as a continuation of line segments. A break in the coordinates of the line segments signifies an opening in the wall which can be a door or a staircase. The specifications of the corridor is followed by the specifications of the room. These will correspond to the three (or more) walls of the room, the room number, the name of the person occupying the room, an image file name of the person who occupies the room, a coordinate center where the image will be placed and another file name in which further information about the occupants of the room will be stored.

This information will include the introduction text, the research area, publications and a URL to the person's homepage. It will also have options for a texture image used to render the three walls of the room. There will also be a URL attached to every room which when selected will open a browser window that will then open the given URL.

The display program will be sending the (x,y) coordinates at all times to a tracking routine. The tracking routine will obtain the information about every room from an XML input file. With the help of these coordinates, the separate tracking routine will keep track on the user, i.e., it will be able to know the precise position of the user in the building. So if the user is in the corridor, the tracking routine will consult the plan database to verify this and will convey this information to the display routine which will display it in the live sub-window. If the user is found to be in a room, then it will determine the particulars of the room and provide detailed information (location of tables and chair, location of light sources, texture information, etc.) to the display routine to render this room. So when a user enters a room, the room will be displayed along with the correct wall-texture, various objects like tables and chairs in the correct places, etc. Also the user can directly ask for a particular room (using the speech interface), in which case this information would be passed on to the display routine which will display an automated walk-through of the user

from the current location in the building to the particular room he just requested.

The next step was to integrate the display of our objects in the maze. It involved a lot of work, as we tested our final display on three Operating Systems, windows NT, Linux and Sun Solaris. A lot of hard work was put up to choose the correct position of chairs and tables in the rooms. I had to write OpenGL routines for collision detection, so that a person coming inside a room is not able to go over a table or a chair (i.e., his motion should be blocked by the chair or the table as in a real world). For this, simple routines in C++ were written that calculated the bounding box of the table and chairs, and then ensured that the user cannot enter the bounding box. We also included an internet browser functionality in the plan. We can automatically launch an Netscape or Internet Explorer window as soon as the user enters a specific room. The browser window then shows relevant information about the person in the room or the purpose of the room if it is a lab. The sub-window on the upper portion of the screen displayed the location of the user in the plan. For example, if you are in the corridor, the sub-window displays “You are in the corridor”, and if you are in a particular room, it displays some basic information about the room. It was ensured that the text part was displayed in the middle of the display window, symmetrically, whatever may be the length of the text. Simple routines in OpenGL were written which provided for certain actions to be performed by the press of a button, like by pressing “W”, you can go through the walls. For this, we disabled the collision detection while one goes through the walls. Beside the four arrow keys, which are the most popular way to navigate through a display, a functionality of mouse button was also added, so that you could traverse through the plan, by press of the left button of the mouse.

The original plan for the third floor of Pond Lab (CSE department) is shown in Figure 1. A screenshot of the pop-up 3-D display of the same information in digital format is also shown in Figure 2.

4 Photorealism and content creation (Gaurav Kumar and Atin Bansal)

One of the most important tasks of our group is the rendering of information on the screen, i.e, the display aspect of the project. This involves creating surroundings such that the user gets a feeling of actually being in the building and moving around in it. The basic code has already been written by *Anirudh* in C++ using the *OpenGL API* for graphics, so an extension to the same will be written by us and integrated with the display program. The tasks require placing of objects such as tables and chairs in the room. Properties of objects such as colour, texture, reflectance, shadowing, etc. will be studied and applied so as to make the display photorealistic. A brick or wall paper pattern will be developed for each room and texturing applied so as to give a good look and differentiate between the rooms. Also, portrait snapshots of the person inhabiting the room will be put on the wall next to the door in order to give additional information to the user before he decides to enter the room.

Our sub-group (along with help from Anirudh) had the following tasks:

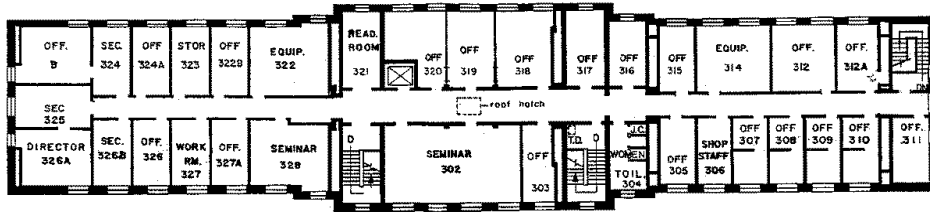


Figure 1: Original plan for the third floor of Pond Lab

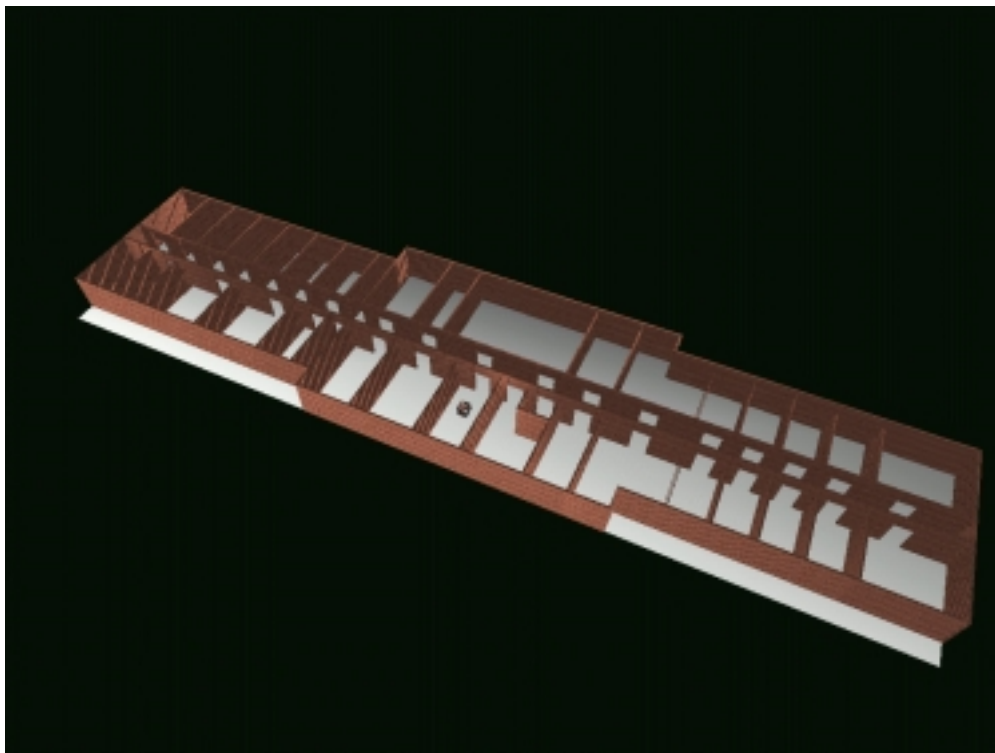


Figure 2: Screenshot showing a pop-up view of the 3-D display of the third floor

1. Making the plan of the 3rd Floor Pond Lab.
2. Photorealism.
3. Display of Objects in the Rooms.
4. Content creation.
5. Help Menu.

The first task that our group did was to procure the architectural plan of the third floor of Pond laboratory. We procured the plan, and mapped it into a 3-D model. We worked upon several problems encountered regarding the specification of corners of a room, specification of the boundary of the room, etc. We worked with the XML team to convey our format for the plan so that they could easily translate it into their own. We have tried to stick to the original architectural plan as much as possible. We measured all the information in the plan and made a data file specifying all the corners of all the rooms on the third floor and the co-ordinates of the door in the room.

The second task in front of us was to get familiar with the OpenGL library in order to contribute to the project, which was dominantly graphical in nature. The task assigned to us was to make all the objects in the room, chair an a table and the door of each room, set proper lightning and then apply proper textures on the objects drawn and also on the walls of the room. This was achieved by going through a few OpenGL book extensively and by following some easy-to-understand web-based tutorials on OpenGL.

Our task involved drawing simple 3-D objects and making sure they were proportionate in all their three dimensions. A person who enters a room can go in various directions, and he should be able to see the object clearly from any perspective. We also had to make sure that the objects in the room look good aesthetically. To complete the task, we had to present a realistic view of the objects for which we set about testing with different lightning models for the room. First, a simple model was drawn on the paper which was translated into the display coordinates. Then using simple OpenGL commands the objects were drawn on the screen. Using `gluLookat` command which defines the views of an object from various directions, it was checked whether the view was correct and wholesome. First, a wire frame model was created which was analyzed from an aesthetic point of view. Then came the task of defining proper lightning for the objects. To define lightning, we first had to decide the position of the light source, then what type of light would be most suitable for a room, diffusive, specular or ambient (light in different rooms may be positioned differently and may have different characteristics). We had to define normals for each surface of an object, whether it was a table or a chair, so that lightning could be properly rendered. Our next job was to define proper textures for the chair, table, rooms of the wall, and the door. We searched through various images of the textures on the web, took some photographs of textures around us with the help of Anirudh, and after great deliberation, decided on the different textures for different objects. While deciding on the texture, we had to take some special precautions to make sure that the texture is

mapped correctly onto every surface, otherwise the surfaces may look weird. A sample wireframe model of a table and chair without and with texture is depicted in Figure 3.

Another part of the project assigned to our group was to take care of the “Help” portion of the project. We have designed an interactive help system for the viewer. It provides the function of an interactive interface with the user of the kiosk.

The moment you press F1, a pop-up screen which lists all the help commands is displayed on the screen (Figure 4). We designed a *Help Window* which can be accessed on the press of a button while you are traversing through the plan. The help window superimposes itself on the window where the user was viewing the room and can be switched on or off by the press of a key. So the user does not have to take the effort to memorize various commands. We also assigned the various keys on the keyboard to the various functionalities in the viewer, like going in the various directions, seeing the top view of the plan, etc.

5 Future work

While a lot of work has been completed in this project, a significant amount still needs to be done. A complete integration of the various input modalities (speech and gesture) with the content display system is highly desirable for adding value to the *kiosk*. Textured doors can also be added to all the rooms, so as to give them a more realistic view. We deliberately left the preliminary model of the doors that we had designed as it did not seem upto the task. Some more objects can be added ion the rooms such as bookshelves, computers, etc. The plan can be expanded to diplay all the three floors of the Pond Lab.

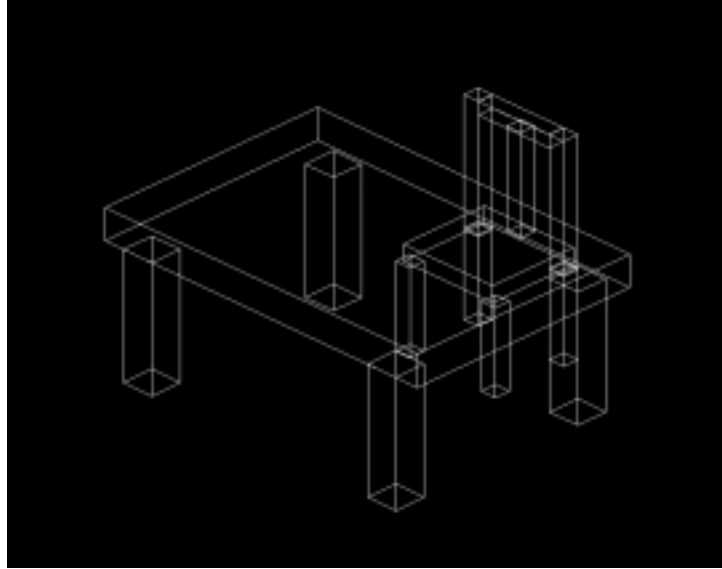


Figure 3: Wireframe model of a table and chair without and with texture

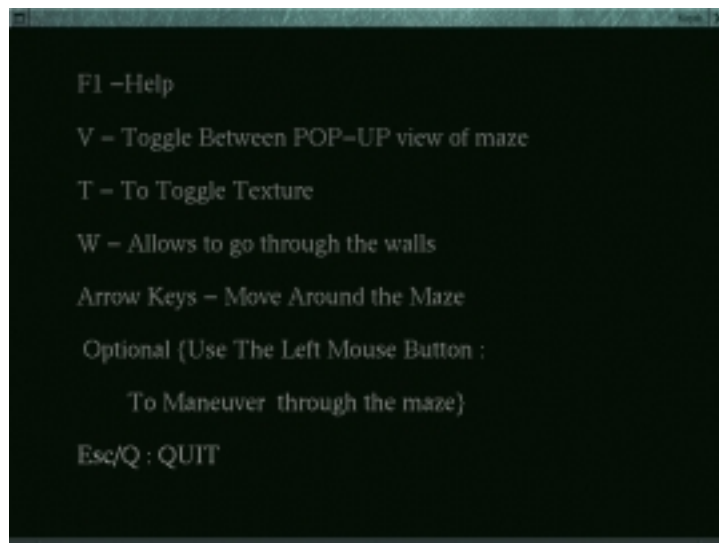


Figure 4: Screenshot of the *Help* screen for the program

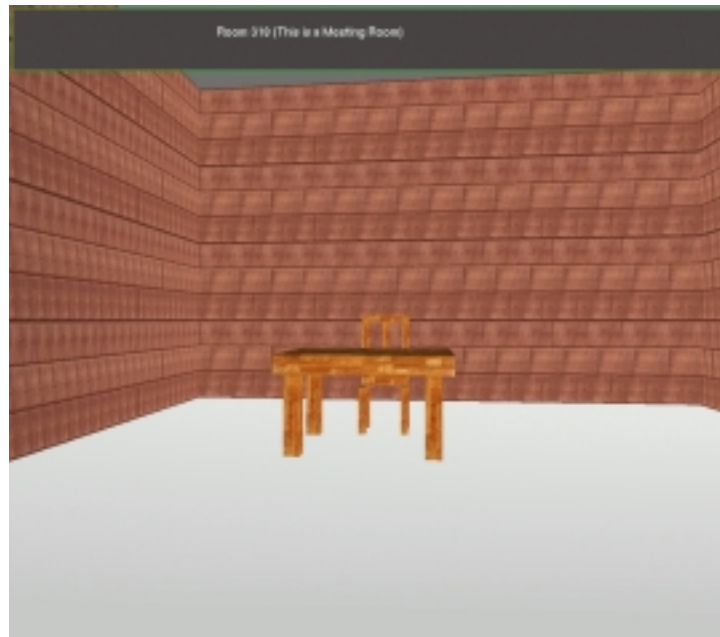


Figure 5: Screenshots showing a view of the 3-D display of the third floor