**Chapter 1**

# PREAMBLE

## 1.1 INTRODUCTION

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television. It has the added advantage that, with the computer, we can make pictures not only of concrete real world objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

Using this editor you can draw and paint using the mouse. It can also perform a host of other functions like drawing lines, circles, polygons and so on. Interactive picture construction techniques such as basic positioning methods, rubber-band methods, dragging and drawing are used. Block operations like cut, copy and paste are supported to edit large areas of the workspace simultaneously. It is user friendly and intuitive to use.

OpenGL(open graphics library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex 3D scenes from simple primitives. OpenGL was developed by silicon graphics Inc.(SGI) in 1992 and is widely used in CAD ,virtual reality , scientific visualization , information visualization and flight simulation. It is also used in video games, where it competes with direct 3D on Microsoft Windows platforms.OpenGL is managed by the non-profit technology consortium, the khronos group, Inc

OpenGL serves two main purpose :

* To hide the complexities of interfacing with different 3D accelerators, by presenting programmer with a single, uniform API
* To hide the differing capabilities of hardware platforms, by requiring that all Implementations support the full openGL, feature set.

OpenGL has historically been influential on the development of 3D accelerator, promoting a base level of functionality that is now common in consumer level hardware:

* Rasterized points, lines and polygons are basic primitives.
* A transform and lighting pipeline.
* Z buffering.
* Texture Mapping.
* Alpha
* Blending.

## 1.2 STATEMENT OF PROBLEM

3D Home Architect is a property designing [program.](http://en.wikipedia.org/wiki/Software) Harneet's guide to 3D

Home Architect comes in three designs for specific purposes: Home and Landscape Design Suite, Home Design Deluxe, and Landscape Design Deluxe. Home Design Deluxe simulates home designs, Landscape Design Deluxe simulates landscape designs, and Home and Landscape Design Suite is used for both.

3D Home Architect was introduced by [Broderbund](http://en.wikipedia.org/wiki/Br%C3%B8derbund) in the [1990s](http://en.wikipedia.org/wiki/1990s) and was a scaled down version of a professional home design application called Chief Architect, made by Advanced Relational Technology (ART) Inc. (now renamed to Chief Architect, Inc.). After version 4.0, the agreement between Broderbund and ART Inc. was terminated, and 3D Home Architect 5.0 and later versions are based on a similar professional application called Cad soft Envisioned.

## 1.3 OBJECTIVE OF THE PROBLEM

The narrative mode (also known as the mode of narration) is the set of methods the [author](http://en.wikipedia.org/wiki/Author) of a literary, theatrical, cinematic, or musical [story](http://en.wikipedia.org/wiki/Narrative) uses to convey the plot to the [audience.](http://en.wikipedia.org/wiki/Audience) Narration, the process of presenting the narrative, occurs because of the narrative mode. It encompasses several overlapping areas of concern, most importantly narrative point-of-view, which determines through whose perspective the story is viewed; narrative voice, which determines the manner through which the story is communicated to the author to be the same person. However, the narrator may be a fictive person devised by the author as a stand-alone entity, or even a character. The narrator is considered participant if an actual character in the story, and nonparticipant if only an implied character, or a sort of omniscient or semi-omniscient being who does not take part in the story but only relates it to the audienc

**Chapter 2**

# 2. LITERATURE SURVEY

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube (CRT) screens soon after the introduction of computers.

Computer graphics today largely interactive, the user controls the contents, structure, and appearance of objects and of displayed images by using input devices, such as keyboard, mouse, or touch-sensitive panel on the screen. Graphics based user interfaces allow millions of new users to control simple, low-cost application programs, such as spreadsheets, word processors, and drawing programs.

OpenGL (Open Graphics Library) is a standard specification defining a crosslanguage, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation. It is also used in video games, where it competes with Direct3D on Microsoft Windows platforms (see Direct3D vs. OpenGL). OpenGL is managed by the non-profit technology consortium, the Khronos Group.

In the 1980s, developing software that could function with a wide range of graphics hardware was a real challenge. By the early 1990s, Silicon Graphics (SGI) was a leader in 3D graphics for workstations. SGI's competitors (including Sun Microsystems, Hewlett-Packard and IBM) were also able. In addition, SGI had a large number of software customers; by changing to the OpenGL API they planned to keep their customers locked onto SGI (and IBM) hardware for a few years while market support for OpenGL matured to bring to market 3D hardware, supported by extensions made to the PHIGS standard. In 1992, SGI led the creation of the OpenGL architectural review board (OpenGL ARB), the group of companies that would maintain and expand the

OpenGL specification took for years to come. On 17 December 1997, Microsoft and

SGI initiated the Fahrenheit project, which was a joint effort with the goal of unifying the

OpenGL and Direct3D interfaces (and adding a scene-graph API too). In 1998 HewlettPackard joined the project.[4] It initially showed some promise of bringing order to the world of interactive 3D computer graphics APIs, but on account of financial constraints at SGI, strategic reasons at Microsoft, and general lack of industry support, it was abandoned in 1999[8].

Many opengl functions are used for rendering and transformation purposes.

Transformations functions like glRotate (), glTranslate (), glScaled () can be used.

OpenGL provides a powerful but primitive set of rendering command, and all higherlevel drawing must be done in terms of these commands. There are several libraries that allow you to simplify your programming tasks, including the following:

OpenGL Utility Library (GLU) contains several routines that use lower-level OpenGL commands to perform such tasks as setting up matrices for specific viewing orientations and projections and rendering surfaces.

OpenGL Utility Toolkit (GLUT) is a window-system-independent toolkit, written by Mark Kill guard, to hide the complexities of differing window APIs.

To achieve the objective of the project, information related to the light sources is required with OpenGL we can manipulate the lighting and objects in a scene to create many different kinds of effects. It explains how to control the lighting in a scene, discusses the OpenGL conceptual model of lighting, and describes in detail how to set the numerous illumination parameters to achieve certain effects. This concept is being obtained from .

To demonstrate the transformation and lightening, effects, different polygons have to be used. Polygons are typically drawn by filling in all the pixels enclosed within the boundary, but we can also draw them as outlined polygons or simply as points at the vertices.

This concept is obtained from.

The properties of a light source like its material, diffuse, emissive, has to mention in the project. So to design the light source and the objects, programming guide of an OpenGL is used.

**Chapter 3**

# SYSTEM REQUIREMENTS SPECIFICATION

## 3.1 HARDWARE REQUIREMENTS

Minimum hardware specification

* Microprocessor: **1.0 GHz** and above CPU based on either AMD or INTEL Microprocessor Architecture
* Main memory : **512 MB RAM**
* Hard Disk : **40 GB**
* Hard disk speed in RPM:**5400 RPM**
* Keyboard: **QWERTY** Keyboard
* Mouse :**2 or 3** Button mouse
* Monitor : **1024 x 768** display resolution

## 3.2 SOFTWARE REQUIREMENTS

Minimum software specification

* Operating system : UBUNTU 10.10
* Tool Used : Eclipse
* OPENGL Library
* X86
* X64(WOW)
* Mouse Driver
* Graphics Driver
* C Language

**CHAPTER 4**

## DESIGN

### 4.1 EXISTING SYSTEM

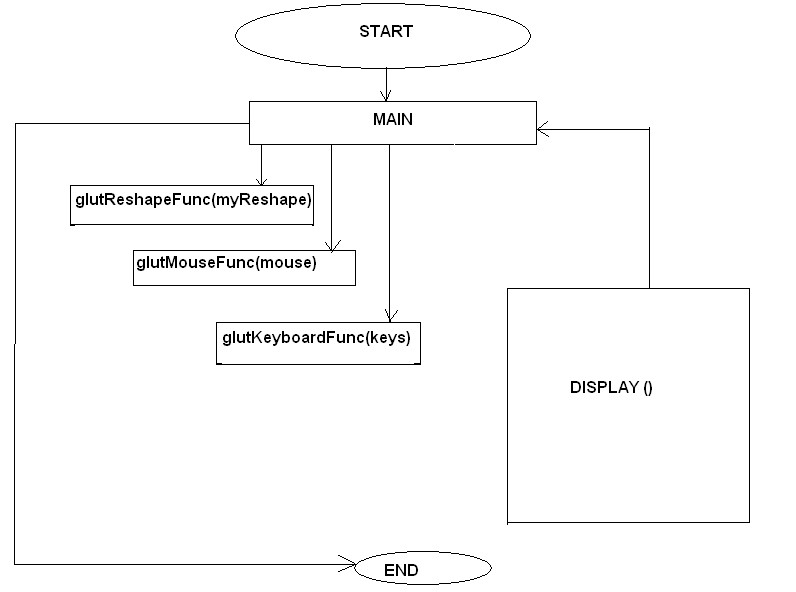
Existing system for a graphics is the TC++. This system will support only the 2D graphics. 2D graphics package being designed should be easy to use and understand. It should provide various options such as free hand drawing, line drawing, polygon drawing, filled polygons, flood fill, translation, rotation, scaling, clipping etc. Even though these properties were supported, it was difficult to render 2D graphics cannot be very difficult to get a 3 Dimensional object. Even the effects like lighting, shading cannot be provided. So we go for Microsoft Visual Studio software.

### 4.2 PROPOSED SYSTEM

To achieve three dimensional effects, open GL software is proposed. It is software which provides a graphical interface. It is a interface between application program and graphics hardware. The advantages are:

1. Open GL is designed as a streamlined.
2. It’s a hardware independent interface i.e it can be implemented on many different hardware platforms.
3. With openGL we can draw a small set of geometric primitives such as points, lines and polygons etc.
4. It provides double buffering which is vital in providing transformations.
5. It is event driven software.
6. It provides call back function.

### 4.3 Low level design



**Chapter 5**

## IMPLEMENTATION

**5.1**

### Functions

The **glColor3f (float, float, float) :-** This function will set the current drawing color

**gluOrtho2D (GLdouble left, GLdouble right, GLdouble bottom,** **GLdouble top):-** which defines a two dimensional viewing rectangle in the plane z=0.

**glClear( ):-**Takes a single argument that is the bitwise OR of several values indicating which buffer is to be cleared.

**glClearColor ():-**Specifies the red, green, blue, and alpha values used by **glClear** to clear the color buffers.

**GlLoadIdentity( ):-**the current matrix with the identity matrix.

**glMatrixMode(mode):-**Sets the current matrix mode, mode can be GL\_MODELVIEW, GL\_PROJECTION or GL\_TEXTURE.

**Void glutInit (int \*argc, char\*\*argv):-**Initializes GLUT, the arguments from main are passed in and can be used by the application.

**Void glutInitDisplayMode (unsigned int mode):-**Requests a display with the properties in mode. The value of mode is determined by the logical OR of options including the color model and buffering.

**Void glutInitWindowSize (int width, int height):-** Specifies the initial position of the topleft corner of the window in pixels

**Int glutCreateWindow (char \*title):-**A window on the display. The string title can be used to label the window. The return value provides references to the window that can be used when there are multiple windows.

**Void glutMouseFunc(void \*f(int button, int state, int x, int y):-**Register the mouse callback function f. The callback function returns the button, the state of button after the event and the position of the mouse relative to the top-left corner of the window.

**Void glutKeyboardFunc(void(\*func) (void)):-**This function is called every time when you press enter key to resume the game or when you press ‘b’ or ‘B’ key to go back to the initial screen or when you press esc key to exit from the application.

**Void glutDisplayFunc (void (\*func) (void)):-**Register the display function func that is executed when the window needs to be redrawn.

**Void glutSpecialFunc(void(\*func)( void)):-**This function is called when you press the special keys in the keyboard like arrow keys, function keys etc. In our program, the func is invoked when the up arrow or down arrow key is pressed for selecting the options in the main menu and when the left or right arrow key is pressed for moving the object(car) accordingly.

**glut PostReDisplay ( )** :-which requests that the display callback be executed after the current callback returns.

**Void MouseFunc (void (\*func) void)):-**This function is invoked when mouse keys are pressed. This function is used as an alternative to the previous function i.e., it is used to move the object(car) to right or left in our program by clicking left and right button respectively.

#### Void glutMainLoop ()

Cause the program to enter an event-processing loop. It should be the last statement in main function.

**Chapter 6**

### RESULTS & SNAPSHOTS

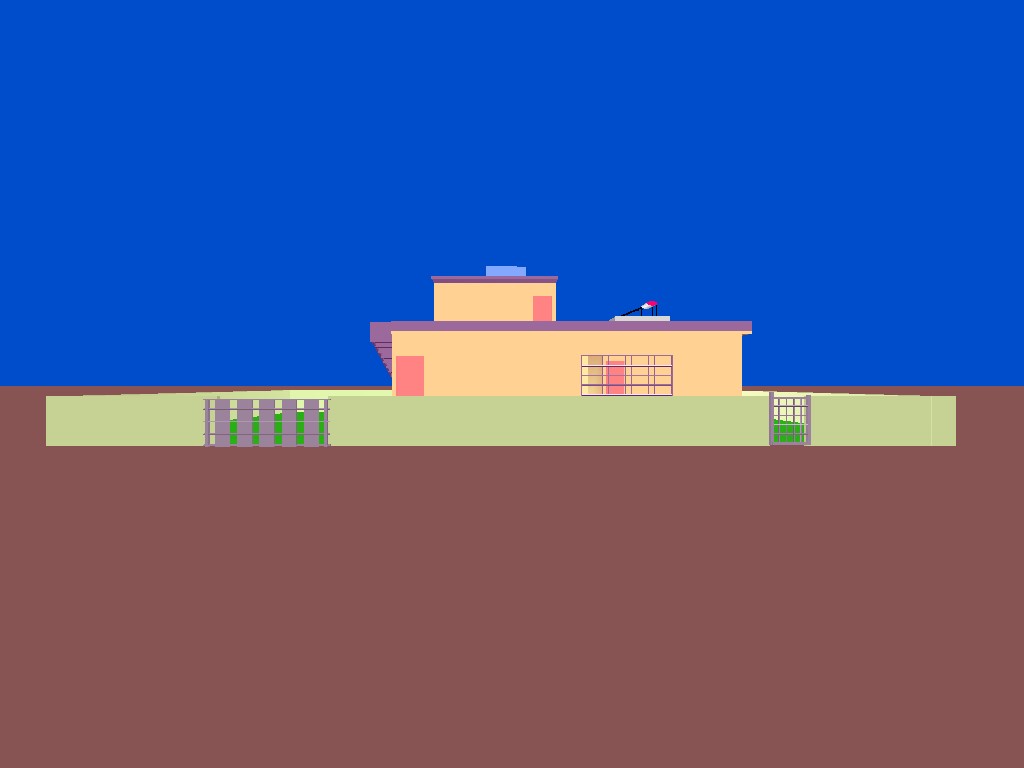


Figure 6.1 After Run the Code

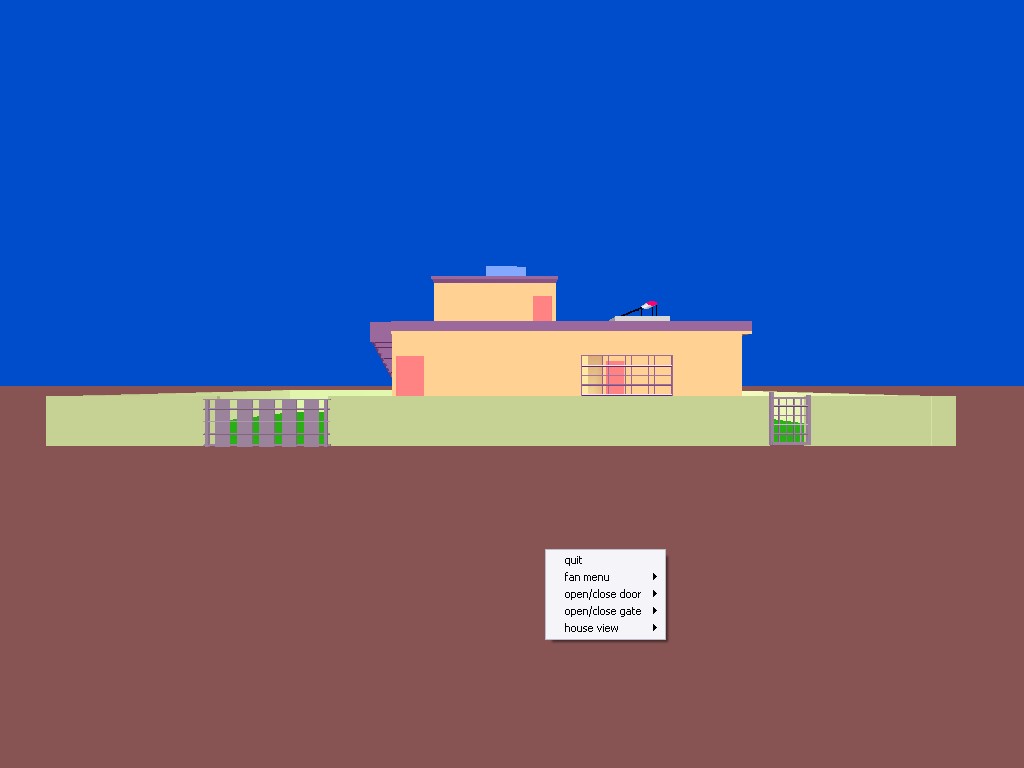


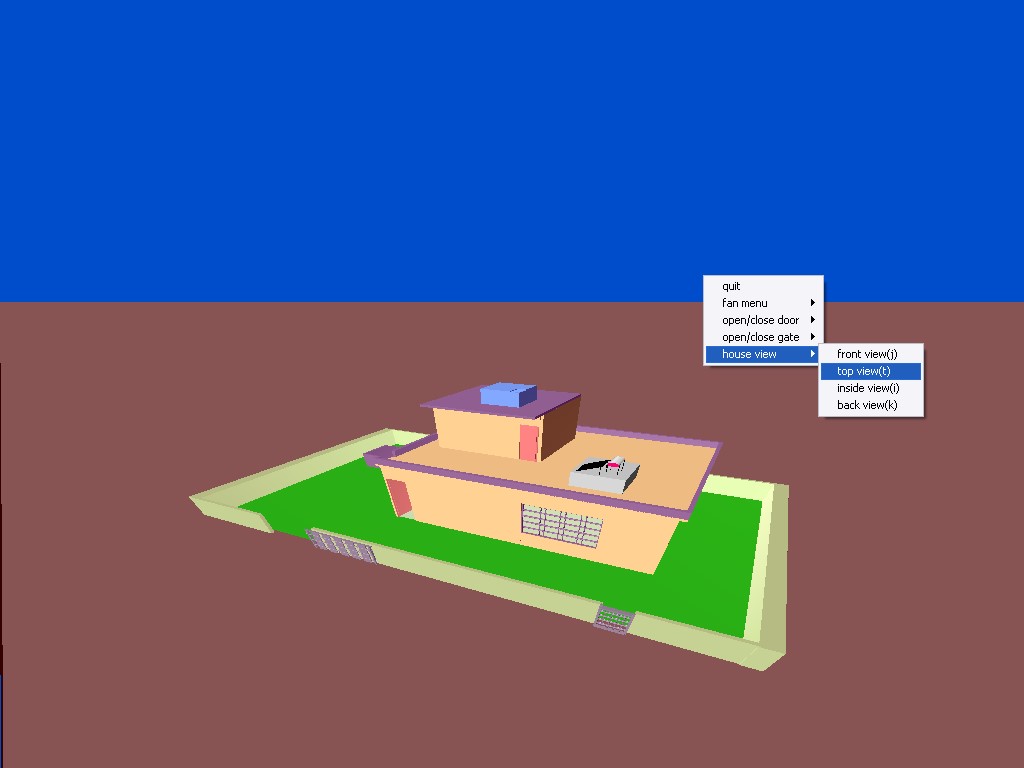
Figure 6.2 After Right Click it’s Showing options



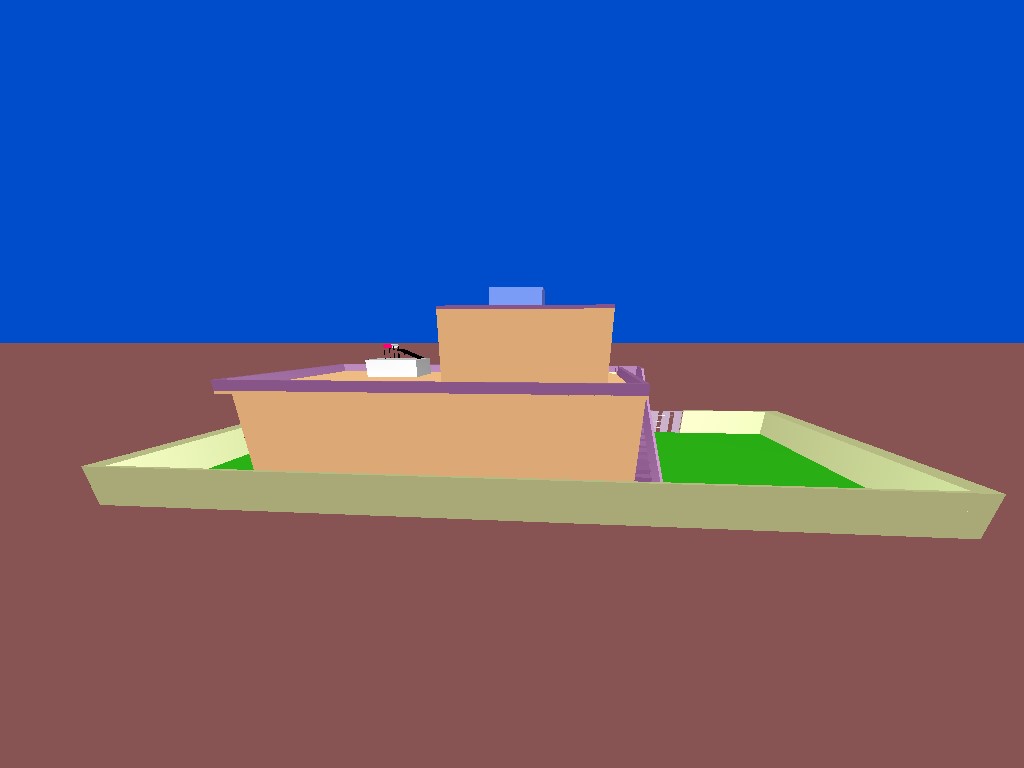
Figur 6.3 After Selected inner view of house



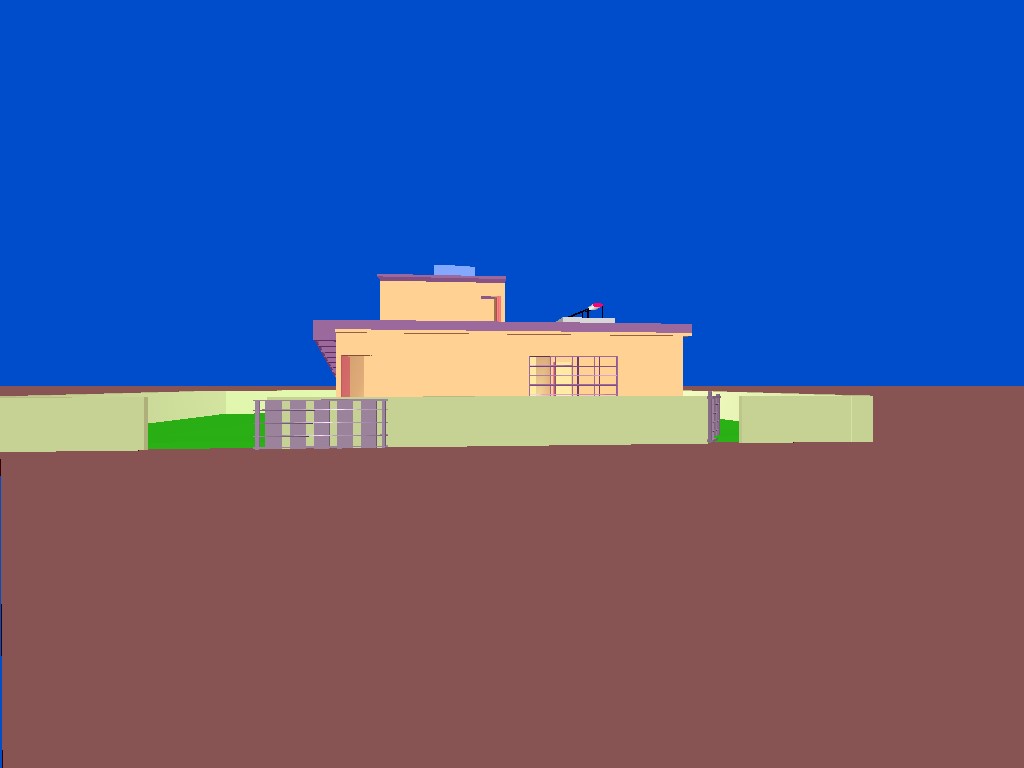
Figur 6.4 Selecting Main door to open



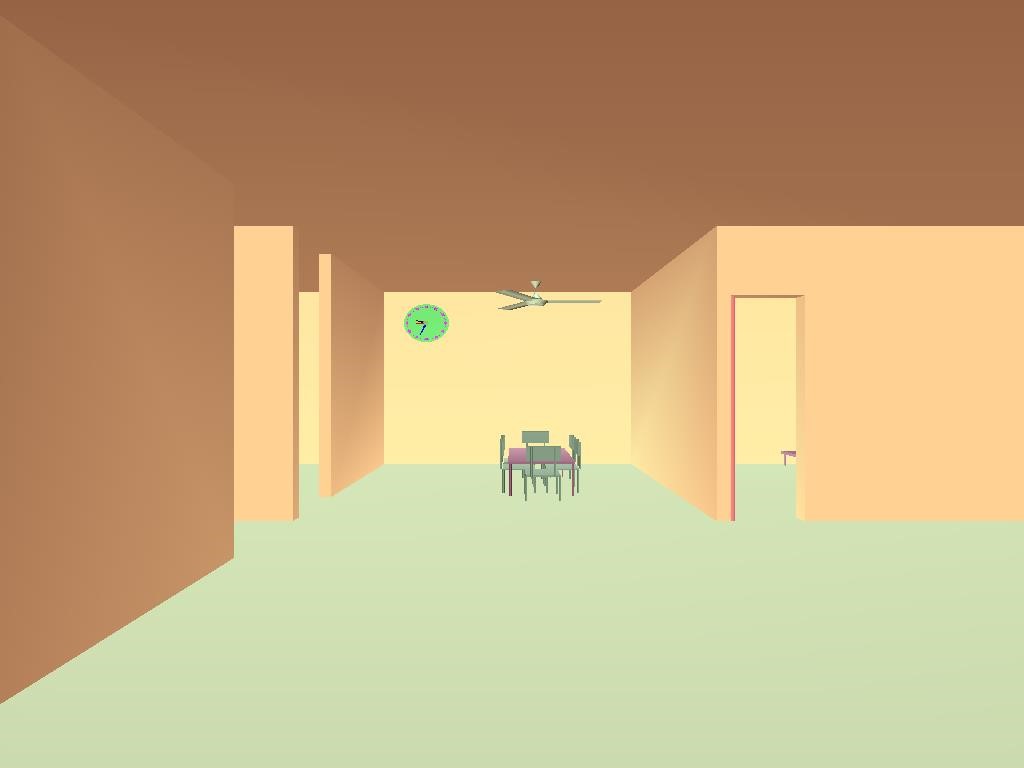
Figur 6.5 House Top view is showing



Figur 6.6 Back View of HOUSE



Figur 6.7 All the door’s opened



Figur 6.7 Inside Door open & it’s showing time

**Chapter 7**

### CONCLUSION AND FUTURE SCOPE

#### 7.1 CONCLUSION

The 3D House has been tested under Windows XP and has been found to provide ease of use and manipulation to the user. The 3D house created for the Windows XP operating system can be used to draw lines, boxes, circles, ellipses, and polygons. It has a very simple and aesthetic user interface.

We found designing and developing this 3D House as a very interesting and learning experience. It helped us to learn about computer graphics, design of Graphical User Interfaces, interface to the user, user interaction handling and screen management. The graphics editor provides all and more than the features that have been detailed in the university syllabus.

#### 7.2 FUTURE ENHANCEMENTS

These are the features that are planned to be supported in the future

* Support for multiple canvases
* Support for pattern filling
* Support for 3d transformations
* Support for transparency of layers

#### 7.3 LIMITATIONS

As with all types of [parallel projection,](http://en.wikipedia.org/wiki/Parallel_projection) objects drawn with isometric projection do not appear larger or smaller as they extend closer to or away from the viewer. While advantageous for [architectural drawings](http://en.wikipedia.org/wiki/Architectural_drawing) where measurements need to be taken directly, the result is a perceived distortion, as unlike [perspective projection,](http://en.wikipedia.org/wiki/Perspective_projection) it is not how our eyes or photography normally work. It also can easily result in situations where depth and altitude are difficult to gauge, as is shown in the illustration to the right.

#### APPENDIX

**Step 1: initialisation** void myinit(void)

{

glMatrixMode(GL\_PROJECTION); glLoadIdentity();

glFrustum(-1.0,1.0,-1\*w/h,1\*w/h,1,200.0); glMatrixMode(GL\_MODELVIEW); glLoadIdentity();

//defining new quadric object

Cylinder = gluNewQuadric(); //to set drawing style

gluQuadricDrawStyle( Cylinder, GLU\_FILL); //to set automatic normals gluQuadricNormals( Cylinder,GLU\_SMOOTH); Disk = gluNewQuadric(); gluQuadricDrawStyle( Disk, GLU\_FILL); gluQuadricNormals( Disk, GLU\_SMOOTH); GLfloat gam[]={0.2,0.2,0.2,1}; glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT,gam);

}

**Step 2: set material property** void matprop(GLfloat amb[],GLfloat dif[],GLfloat spec[],GLfloat shi[])

{

glMaterialfv(GL\_FRONT\_AND\_BACK,GL\_AMBIENT,amb); glMaterialfv(GL\_FRONT\_AND\_BACK,GL\_DIFFUSE,dif); glMaterialfv(GL\_FRONT\_AND\_BACK,GL\_SPECULAR,spec); glMaterialfv(GL\_FRONT\_AND\_BACK,GL\_SHININESS,shi);

}

**Step 3:to create earth** void earth(void)

{

GLfloat ambient[]={1,0,0,1};

GLfloat specular[]={0,1,1,1};

GLfloat diffuse[]={.5,.5,.5,1};

GLfloat shininess[]={50}; matprop(ambient,diffuse,specular,shininess);

GLfloat lightIntensity[]={.7,.7,.7,1}; GLfloat light\_position[]={2,5,-3,0}; glLightfv(GL\_LIGHT0,GL\_POSITION,light\_position); glLightfv(GL\_LIGHT0,GL\_DIFFUSE,lightIntensity); glPushMatrix();

glTranslated(0,-.25,0); glScaled(10000,.5,1000000); glutSolidCube(1.0); glPopMatrix(); glFlush();

}

**Step 4: To crete a room** void room()

{

GLfloat ambient1[]={1,0,1,1};

GLfloat specular1[]={1,1,1,1};

GLfloat diffuse1[]={0.5,0.5,0.5,1};

GLfloat mat\_shininess[]={50}; matprop(ambient1,diffuse1,specular1,mat\_shininess); glPushMatrix(); glTranslated(.5,4,.5);

//roof

glPushMatrix(); glTranslated(-.02\*4,.7\*3.9,-.02\*4); glScaled(.6+.02,1.5,.5+.02+.1); wall(0.08); glPopMatrix();

GLfloat ambient2[]={1,0,0,1};

GLfloat specular2[]={1,1,1,1};

GLfloat diffuse2[]={1,1,.7,1};

GLfloat shininess1[]={50}; matprop(ambient2,diffuse2,specular2,shininess1);

//left wall glPushMatrix(); glTranslated(0,0,-.02); glScaled(1,.7,.5); glRotated(90.0,0,0,1); wall(0.08);

glPopMatrix(); //right wall glPushMatrix(); glTranslated(2.4,0,-.02); glScaled(1,.7,.5);

glRotated(90.0,0,0,1); wall(0.08); glPopMatrix(); //back wall glPushMatrix(); glTranslated(-.08,0,0); glScaled(.62,.7,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //front wall glPushMatrix(); glTranslated(-0.08,0,2); glScaled(.5,.7,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //wall above the room door glPushMatrix(); glTranslated(1.9,.7\*3,2); glScaled(.11,.7\*.25,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix();

GLfloat ambient[]={1,0.5,.5,1};

GLfloat specular[]={1,1,1,1};

GLfloat diffuse[]={1,0.5,0.5,1}; matprop(ambient,diffuse,specular,mat\_shininess);

//door

glPushMatrix(); glTranslated(2.3,0,(2-.05)); glRotated(-tro,0,1,0);

glTranslated(-2.3,0,-(2-.05)); glPushMatrix(); glTranslated(1.927,0,2); glScaled(.09,.525,1); glRotated(-90.0,1,0,0); wall(0.02); glPopMatrix();

glPushMatrix(); glTranslated(2.3,0,2-.05); glScaled(.6,.7,.8); glRotated(-90,1,0,0);

gluCylinder(Cylinder, 0.05, 0.05, 3, 16, 16); glPopMatrix(); glPopMatrix(); glPopMatrix();

}

**Step 5: to create a fan** void fan(void)

{ glPushMatrix(); glTranslated(2.5,1.9,0); glScaled(.5,.5,.5);

GLfloat mat\_ambient[]={.5,0,0,1};

GLfloat mat\_specular[]={0,1,1,0};

GLfloat mat\_diffuse[]={.8,1,.8,1};

GLfloat mat\_shininess[]={50}; glMaterialfv(GL\_FRONT,GL\_AMBIENT,mat\_ambient); glMaterialfv(GL\_FRONT,GL\_DIFFUSE,mat\_diffuse); glMaterialfv(GL\_FRONT,GL\_SPECULAR,mat\_specular); glMaterialfv(GL\_FRONT,GL\_SHININESS,mat\_shininess); if(flag==-1)

{

glPushMatrix(); fanbottom(); glPopMatrix();

} else { angle+=speed; glPushMatrix(); glTranslated(1,0,1); glRotated(angle,0,1,0); glTranslated(-1,0,-1); fanbottom(); glPopMatrix();

}

glPushMatrix(); glTranslatef(1,3.3,1); glRotated(-90,1,0,0);

gluCylinder(Cylinder, .1, 0.005, .25, 16, 16); glPopMatrix(); glPushMatrix();

glTranslatef(1,4,1); glRotated(90,1,0,0);

gluCylinder(Cylinder, .006, 0.006, .6, 16, 16); glPopMatrix();

glPushMatrix();

glTranslatef(1,3.96,1); glRotated(90,1,0,0);

gluCylinder(Cylinder, .1, 0.005, .25, 16, 16); glPopMatrix(); glPopMatrix(); if(flag==1) glutPostRedisplay();

}

**Step 6: to create a Clock** void myclock()

{

GLfloat mat\_ambient[]={.4,.8,.4,1};

GLfloat mat\_specular[]={1,1,1,1};

GLfloat mat\_diffuse[]={0.4,.8,.4,1}; GLfloat mat\_shininess[]={50};

matprop(mat\_ambient,mat\_diffuse,mat\_specular,mat\_shininess); int hour\_ticks,sec\_ticks; glPushMatrix(); glTranslated(2,3.2,-.02); glScaled(.03,.06,.03); glPushMatrix(); // Draw clock face glTranslatef( 0, 0, 1.0); gluDisk(Disk, 0, 7, 32, 16);

glPopMatrix();

GLfloat mat\_ambien[]={1,0,0,1}; matprop(mat\_ambien,mat\_diffuse,mat\_specular,mat\_shininess); glPushMatrix(); glTranslatef( 0, 0, 1.95); gluDisk(Disk, 0, .8, 32, 16); glPopMatrix();

GLfloat ambient[]={0,0,0,1};

GLfloat specular[]={1,1,1,1}; GLfloat diffuse[]={0,0,0,1};

matprop(ambient,diffuse,specular,mat\_shininess);

// Draw hour hand glPushMatrix(); glColor3f(1.0, 0.5, 0.5); glTranslatef( 0, 0, 1.5);

glRotatef(-(360/12)\*(newtime->tm\_hour+newtime->tm\_min/60.0),

0.0,0.0,1.0);

glRotatef( -90, 1.0, 0.0, 0.0); gluCylinder(Cylinder, 0.45, 0, 4, 16, 16); glPopMatrix();

GLfloat ambient1[]={0,0,1,1};

GLfloat specular1[]={1,1,1,1}; GLfloat diffuse1[]={0,0,1,1};

matprop(ambient1,diffuse1,specular1,mat\_shininess);

// Draw minute hand glPushMatrix(); glColor3f(1.0, 0.5, 1.0); glTranslatef( 0, 0, 1.25);

glRotatef( -(360/60) \* newtime->tm\_min, 0.0, 0.0, 1.0); glRotatef(-90, 1.0, 0.0, 0.0); gluCylinder(Cylinder, 0.4, 0, 6, 16, 16); glPopMatrix();

GLfloat ambient2[]={1,0,0,1};

GLfloat specular2[]={1,1,1,1}; GLfloat diffuse2[]={1,0,0,1};

matprop(ambient2,diffuse2,specular2,mat\_shininess);

// Draw second hand glPushMatrix(); glTranslatef( 0, 0, 1);

glRotatef(-(360/60) \* newtime->tm\_sec, 0.0, 0.0, 1.0);

glRotatef( -90, 1.0, 0.0, 0.0); gluCylinder(Cylinder, 0.3, 0, 6, 16, 16); glPopMatrix();

GLfloat ambient3[]={1,1,1,1};

GLfloat specular3[]={1,1,1,1};

GLfloat diffuse3[]={1,0,1,1}; matprop(ambient3,diffuse3,specular3,mat\_shininess); for(hour\_ticks = 0; hour\_ticks < 12; hour\_ticks++)

{

glPushMatrix();// Draw next arm axis. glTranslatef(0.0, 0.0, 1); glRotatef( (360/12) \* hour\_ticks, 0.0, 0.0, 1.0); glTranslatef( 6.0, 0.0, 0.0); glutSolidCube(.8); glPopMatrix();

}

for(sec\_ticks = 0;sec\_ticks<60;sec\_ticks++)

{

glPushMatrix(); glTranslatef(0.0,0.0,1.1); glRotatef((360/60)\*sec\_ticks,0.0,0.0,1.0); glTranslatef(6.0, 0.0, 0.0); glutSolidCube(0.25); glPopMatrix();

}

glPopMatrix();

}

**step 7: to creat a house** void house(void)

{

GLfloat mat\_ambient[]={1,0,0,1};

GLfloat mat\_specular[]={1,1,1,1};

GLfloat mat\_diffuse[]={1,1,.7,1}; GLfloat mat\_shininess[]={50};

matprop(mat\_ambient,mat\_diffuse,mat\_specular,mat\_shininess);

GLfloat lightIntensity4[]={.7,.7,.7,.7}; GLfloat light\_position4[]={3,1,.5,1}; glLightfv(GL\_LIGHT6,GL\_POSITION,light\_position4); glLightfv(GL\_LIGHT6,GL\_DIFFUSE,lightIntensity4); glEnable(GL\_LIGHT6); glPushMatrix(); glTranslated(0,.15,0);

//roof glPushMatrix();

glTranslated(-.02\*4,3.9,-.01\*4-.25); glScaled(1.5+.05,1.5,1.1); wall(0.08); glPopMatrix();

GLfloat ambient2[]={1,0,0,1};

GLfloat specular2[]={1,1,1,1};

GLfloat diffuse2[]={.7,1,0.8,1}; GLfloat shininess[]={50};

matprop(ambient2,diffuse2,specular2,shininess);

//floor

glPushMatrix(); glTranslated(-.02\*3,-0.05,-.01\*4); glScaled(1.5+.01,1.5,1); wall(0.08); glPopMatrix();

GLfloat ambient1[]={1,0,0,1};

GLfloat specular1[]={1,1,1,1};

GLfloat diffuse1[]={1,1,.7,1}; GLfloat shininess1[]={50};

matprop(ambient1,diffuse1,specular1,shininess1);

//left wall

glPushMatrix(); glRotated(90.0,0,0,1); wall(0.08); glPopMatrix(); //right wall glPushMatrix(); glTranslated(6,0,0);

glRotated(90.0,0,0,1); wall(0.08); glPopMatrix(); //back wall glPushMatrix(); glTranslated(-.08,0,0); glScaled(1.5+.02,1,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix();

//room vertical wall glPushMatrix(); glTranslated(4,0,0); glScaled(1,1,.5); glRotated(90.0,0,0,1); wall(0.08); glPopMatrix(); //room horizantal wall glPushMatrix(); glTranslated(4.4,0,2); glScaled(.4,1,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //wall above the room door glPushMatrix(); glTranslated(4,3,2); glScaled(.11,.25,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //left room horizantal wall glPushMatrix(); glTranslated(0,0,2); glScaled(.4,1,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //lroom vertical wall glPushMatrix();

glTranslated(1.6,0,0); glScaled(1,1,.35); glRotated(90.0,0,0,1); wall(0.08); glPopMatrix(); //entrance room right wall glPushMatrix(); glTranslated(1.6,0,2.59); glScaled(1,1,.35); glRotated(90.0,0,0,1); wall(0.08);

glPopMatrix(); //wall above main door glPushMatrix(); glTranslated(-0.02,3,4); glScaled(.13,.27,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix();

//wall right to the main door glPushMatrix(); glTranslated(.48,0,4); glScaled(.68,1,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //wall right to the window glPushMatrix(); glTranslated(4.8,0,4); glScaled(.3,1,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); //wall below the window glPushMatrix(); glTranslated(3.2,0,4); glScaled(.4,.25,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix();

//wall above the window glPushMatrix(); glTranslated(3.2,3.03,4); glScaled(.4,.25,1); glRotated(-90.0,1,0,0); wall(0.08); glPopMatrix(); room(); watertank(); terece(); steps();

window(); fan();

cot(.6,.9,.06,.35,.009); diningtable(); myclock(); solar();

GLfloat ambient[]={1,0.5,.5,1}; GLfloat specular[]={1,1,1,1}; GLfloat diffuse[]={1,.5,.5,1}; matprop(ambient,diffuse,specular,mat\_shininess);

//main door glPushMatrix(); glTranslated(0,0,4); glRotated(maino,0,1,0); glTranslated(0,0,-4); glPushMatrix(); glTranslated(0,0,4); glScaled(.12,.75,1); glRotated(-90.0,1,0,0); wall(0.04); glPopMatrix();

glPushMatrix();

glTranslated(0,0,4); glScaled(.5,1,.2); glRotated(-90,1,0,0);

gluCylinder(Cylinder, 0.05, 0.05, 3, 16, 16); glPopMatrix(); glPopMatrix();

//bolow room door glPushMatrix(); glTranslated(4,0,(2-.025)); glRotated(romo,0,1,0); glTranslated(-4,0,-(2-.025)); glPushMatrix();

glTranslated(4,0,2); glScaled(.099,.75,1); glRotated(-90.0,1,0,0); wall(0.01); glPopMatrix(); glPushMatrix();

glTranslated(4.01,0,2-.025); glScaled(.5,1,.6); glRotated(-90,1,0,0);

gluCylinder(Cylinder, 0.05, 0.05, 3, 16, 16); glPopMatrix(); glPopMatrix();

glPopMatrix(); glFlush();

}

**step 8: to create a fan** void fan\_menu(int m)

{

switch(m)

{ case 1:

flag\*=-1; glutPostRedisplay();

break;

case 2:if ( speed < 30.0)

{

speed+=5;

}

break; case 3: if (speed>0)

{

speed-=5;

}

break;

}

}

**Step 9: to creat a gate** void gate\_menu(int m)

{

switch(m)

{

case 1:

if(mgo==0) mgo=1; else mgo=0; break; case 2:

if(sgo==0) sgo=50; else sgo=0; break;

}

}

**Step 10: To creat a house view** void house\_view(int m)

{

switch(m)

{

case 1: view[0]=2.8; view[1]=2; view[2]=4.8; look[0]=2.8; look[1]=2; look[2]=1; break; case 2: view[0]=6; view[1]=12; view[2]=10; look[0]=2; look[1]=8; look[2]=2; break; case 3: view[0]=2; view[1]=2; view[2]=12.9; look[0]=3; look[1]=2; look[2]=3; break; case 4: view[0]=1; view[1]=6; view[2]=-7; look[0]=2; look[1]=4; look[2]=2; break;

}

}

#### BIBLIOGRAPHY

1. Edward Angel’s Interactive Computer Graphics Pearson Education 5th Edition

1. Interactive computer Graphics --A top down approach using open GL--by Edward Angle

1. Jackie .L. Neider, Mark Warhol, Tom.R.Davis, "OpenGL Red Book", Second Revised Edition, 2005.

1. Donald D Hearn and M.Pauline Baker, "Computer Graphics with OpenGL",

3rd Edition.