CHAPTER 1

INTRODUCTION

First Come First Serve Algorithm is OS based Computer graphics Project. This Project as name suggest demonstrate the working of First Come First Serve Algorithm or FCFS Algorithm. The objects are drawn using the GLUT functions. This project has been developed using Code::Blocks IDE on Windows 10 operating system with OpenGL package.

1.1 Computer Graphics

Graphics provides one of the most natural means of communicating within a computer since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. 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.

Computer graphics started with the display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers themselves. It has grown to include the creation, storage, and manipulation of models and images of objects. These models come from a diverse and expanding set of fields, and include physical, mathematical, engineering, architectural, and even conceptual structures, natural phenomena, and so on. Computer graphics today is largely interactive. The user controls the contents, structure, and appearance of the objects and of their displayed images by using input devices, such as keyboard, mouse, or touch-screen. Due to close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics. The advantages of the interactive graphics are many in number. 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 data rapidly and efficiently. In many design, implementation, and construction processes today, the information pictures can give is virtually indispensable. Scientific visualization became an important field in the 1980s when the scientists and engineers realized that they could not interpret the prodigious quantities of data produced in supercomputer runs without summarizing the data and highlighting trends and phenomena in various kinds of graphical representations.

1.2 OpenGL Interface

OpenGL is an application program interface (API) offering various functions to implement primitives, models and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate system and frames. OpenGL offers translation, rotation and scaling of objects.

Most of our applications will be designed to access OpenGL directly through functions in three libraries. They are:

1. Main GL: Library has names that begin with the letter gl and are stored in a library usually referred to as GL.
2. OpenGL Utility Library (GLU): This library uses only GL functions but contains code for creating common objects and simplifying viewing.
3. OpenGL Utility Toolkit (GLUT): This provides the minimum functionality that should be accepted in any modern windowing system.

1.2.1OpenGL Pipeline Architecture

Many OpenGL functions are used specifically for drawing objects such as points, lines, polygons, and bitmaps. Some functions control the way that some of this drawing occurs (such as those that enable antialiasing or texturing). Other functions are specifically concerned with frame buffer manipulation. The topics in this section describe how all of the OpenGL functions work together to create the OpenGL processing pipeline. This section also takes a closer look at the stages in which data is actually processed, and ties these stages to OpenGL functions.

The following diagram details the OpenGL processing pipeline. For most of the pipeline, you can see three vertical arrows between the major stages. These arrows represent vertices and the two primary types of data that can be associated with vertices: color values and texture coordinates. Also note that vertices are assembled into primitives, then into fragments, and finally into pixels in the frame buffer. This progression is discussed in more detail in Vertices, Primitives, Fragments, and Pixels.

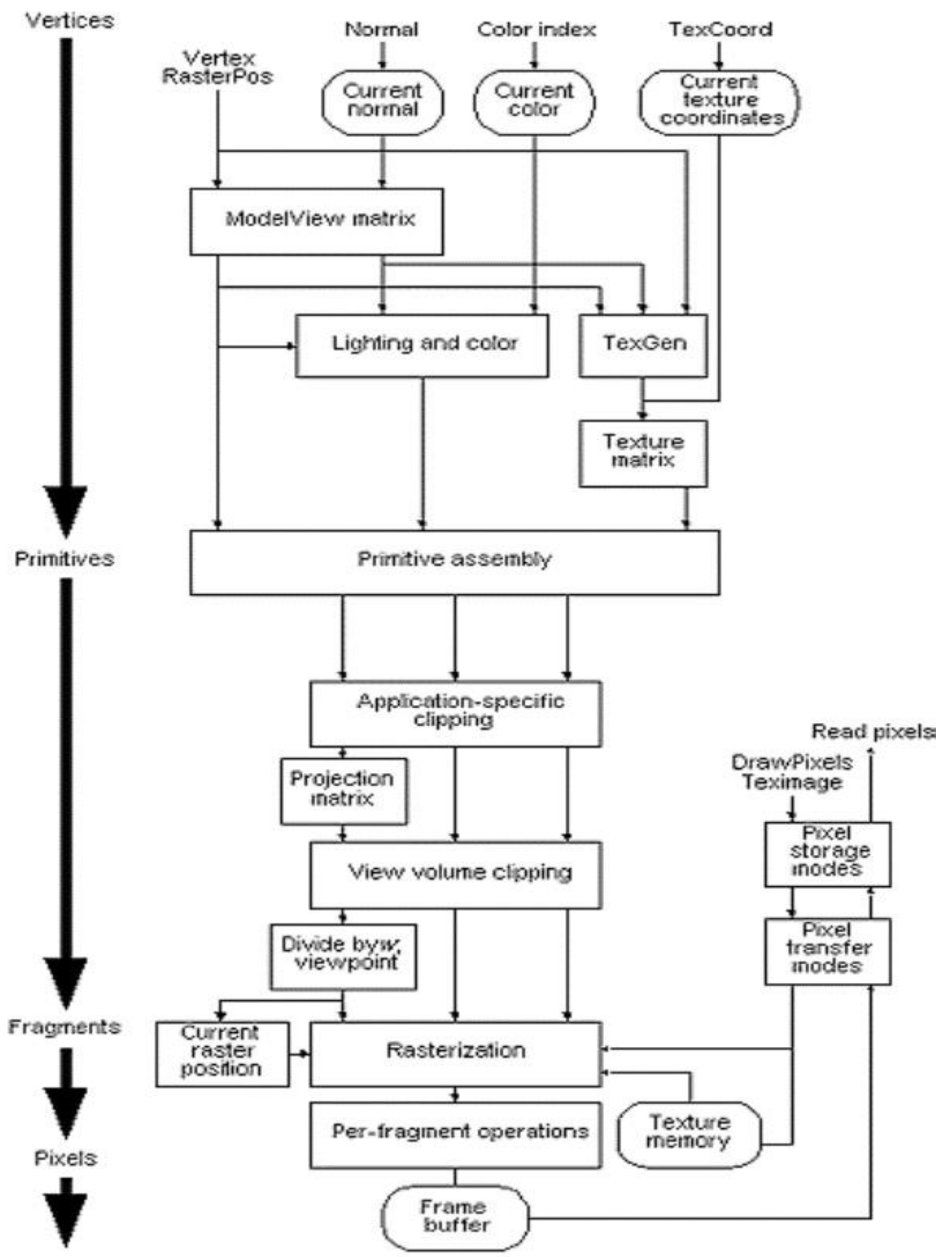

Figure. 1.1 OpenGL Pipeline Architecture

1.2.2 OpenGL Overview

- OpenGL (Open Graphics Library) is the interface between a graphic program and graphics hardware. It is streamlined. In other words, it provides low-level functionality. For example, all objects are built from points, lines and convex polygons. Higher level objects like cubes are implemented as six four-sided polygons.

- OpenGL supports features like 3-dimensions, lighting, anti-aliasing, shadows, textures, depth effects, etc.
- It is system-independent. It does not assume anything about hardware or operating system and is only concerned with efficiently rendering mathematically described scenes. As a result, it does not provide any windowing capabilities.
- It is a state machine. At any moment during the execution of a program there is a current model transformation.
- It is a rendering pipeline. The rendering pipeline consists of the following steps:
  - Defines objects mathematically.
  - Arranges objects in space relative to a viewpoint.
  - Calculates the color of the objects.
  - Rasterizes the objects.

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.

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. 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.

OpenGL serves two main purposes:

- 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.

Chapter 2
SYSTEM REQUIREMENTS SPECIFICATIONS

2.1 HARDWARE REQUIREMENTS

- Microprocessor : 1.0 GHz and above CPU based on either AMD or INTEL Microprocessor Architecture
- Main memory : 2 GB 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

2.2 SOFTWARE REQUIREMENTS

- Programming language – C/C++ using OpenGL
- Operating system – Windows 8.1 Pro
- Compiler/IDE - Code::Blocks IDE
- Graphics library – GL/glut.h ● OpenGL 2.0.

2.3FUNCTIONAL REQUIREMENTS ➢
OpenGL APIs:

If we want to have a control on the flow of program and if we want to interact with the window system then we use OpenGL API'S. Vertices are represented in the same manner internally, whether they are specified as two-dimensional or three-dimensional entities, everything that we do are here will be equally valid in three dimensions. Although OpenGL is easy to learn, compared with other APIs, it is nevertheless powerful. It supports the simple three dimensional programs and also supports the advanced rendering techniques.

➢ GL/glut.h:

We use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system. The application program uses only GLUT functions and can be recompiled with the GLUT library for other window system. OpenGL makes a heavy use of macros to increase code readability and avoid the use of magic numbers. In most implementation, one of the include lines.

➢ Code::Blocks Integrated Development Environment (IDE):

Code::Blocks is a free, open-source cross-platform IDE that supports multiple compilers including GCC, Clang and Visual C++. It is developed in C++ using wxWidgets as the GUI toolkit. Using a plugin architecture, its capabilities and features are defined by the provided plugins. Currently, Code::Blocks is oriented towards C, C++, and FORTRAN. It has a custom build system and optional Make support. Code::Blocks is being developed for Windows, Linux, and MacOS and has been ported to FreeBSD, OpenBSD and Solaris.

CHAPTER 3
ABOUT THE PROJECT

3D Home Architect is a property designing program. 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.

.The objects are drawn using the GLUT functions. This project has been developed using Code::Blocks IDE on Windows 10 operating system with OpenGL package.

3.1 Overview

3D Home Architect was introduced by Broderbund in the 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.

The narrative mode (also known as the mode of narration) is the set of methods the author of a literary, theatrical, cinematic, or musical story uses to convey the plot to the 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 audience.

3.2 User interface

- Mouse is used to interact with the program.
- The user can provide any input(through mouse) and see the changes in Clock on those input.
- The user can see the changes in clock on a predefined example by pressing the right key in mouse.

3.3 Objective

.

3.4 Benefits

- **Simplicity:** The project is built with the help of many standard library functions of the glut package. Hence the readability of the project is good**.**
- **Usability:** Since the project is developed on a Windows platform, it has good usability since many systems have implemented Windows today.
- **Flexibility:** It is easy to add new features to the project since all the objects are independent of their existence in the project. Hence the flexibility of the project is beneficial.

Chapter 4

SYSTEM DESIGN

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

- **Purpose and Scope**

  This section provides a brief description of the Systems Design Document's purpose and scope.

- **Project Executive Summary**

  This section provides a description of the project from a management perspective and an overview of the framework within which the conceptual system design was prepared. If appropriate, include the information discussed in the subsequent sections in the summary.

- **System Overview**

  This section describes the system in narrative form using non-technical terms. It should provide a high-level system architecture diagram showing a subsystem breakout of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high level context diagram for the system and subsystems, if applicable. Refer to the requirements trace ability matrix (RTM) in the Functional Requirements Document (FRD), to identify the allocation of the functional requirements into this design document.

- **Design Constraints**

  This section describes any constraints in the system design (reference any trade-off analyses conducted such, as resource use versus productivity, or conflicts with other systems) and includes any assumptions made by the project team in developing the system design.

- **Future Contingencies**

  This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans.

- **DocumentOrganizatiom clockLayout()**

I've used this function to print the clock layout i.e. clock dial and the markings on the clock. If we observe clearly, the clock has hours marking each separated by 30 degrees and each hour is divided into 5 markings each making an angle of 6 degrees. So, iterating the markings for every 30 degrees gives hours and iterating markings with 6 degrees give minutes markings on the clock.

secHand()

It is clear from the name that this gonna do something with the seconds hand. This function is going to get the present second from the system clock and incline the line according to a particular angle.

minHand()

This function fulfills the task of moving the minutes hand based on the system clock. The minutes hand must be inclined 6 degrees for every minute passing.

hrHand()

This function is going to print an inclined hours line. The function is designed to get the present hour and also the no. of elapsed minutes from the system clock and incline the line according to a particular angle.

main()

The first lines in main are graphic initialization, you must change the path

"c:\turboc3\bgi\" to your compiler's BGI file path otherwise program will not work. Coming to the while loop, the while loop iterates for every 100 milliseconds reprinting all the functions. This program is like getting the static picture of clock every second and combining all the pictures to make a moving analog clock.

Chapter 5

IMPLEMENTATION

5.1 User Defined Functions

- **frontscreen(void):** this function creates the front screen of the project.
- **mydisplay(void):** this function on the value of the flag bit either calls front screen() or disp().
- **strin():**this function displays the text on the screen.
- **constr():**this function displays the constraints on the front screen .
- **backgrnp():**this function displays the string “enter the no. of process” on the front screen.
- **backgrat():**this function displays the string “enter the arrival time of process” on the front screen.
- **backgrbt():**this function displays the string “enter the Burst time of process” on the front screen.
- **drawrec(int p,int q,int r ,ints)**:this function draws rectangles for the gnatt chart.
- **drawscale():**this function draws the scale for the gnatt chart.
- **avg():**calculates the average turn around time and average waiting time.
- **gchart():**displays the gnatt chart on the screen.
- **example()**:sets the process structure with the predefined values.
- **table():**displays the process table on the screen.
- **key1(unsigned char k ,int x,int y):**handles all the keyboard operations. **disp():**calls the backgrnp();

5.2 OpenGL Functions

 glColor3f(float, float, float) - This function will set the current drawing color.

- **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 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 top left corner of the window in pixels.
- **glutInitCreateWindow (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 glutKeyboardFunc(void(*func) (void))-**This function is called every time when you press key to add a balloon with a three letter word starting from the key pressed or to specify which operation is going on in the sum option.
- **void glutDisplayFunc (void (*func) (void))-**Register the display function func that is executed when the window needs to be redrawn.
- **glutPostRedisplay ( )** - which requests that the display callback be executed after the current callback returns.
- **glutMainLoop () -** Cause the program to enter an event-processing loop.It should be the last statement in main function.
- **glBegin(), glEnd()** – delimit the vertices of a primitive or a group of like primitives.
- **glOrtho** - This function defines orthographic viewing volume with all parameters measured from the centre of projection.
- **glFlush()** - force execution of GL commands in finite time ● **GLfloat, GLint** – datatypes.
- **glVertex()** – specify a vertex.

Chapter 6

TESTING

6.1 Introduction to Testing

Verification and validation is a generic name given to checking processes, which ensures that the software conforms to its specifications and meets the demands of users.

- **Validation**

  Are we building the right product? Validation involves checking that the program has implanted meets the requirement of the users.

- **Verification**

  Are we building the product right? Verification involves checking that the program confirms to its specification.

6.2 Stages in the Implementation of Testing

 Unit testing

Each individual unit is tested for correctness. These individual components will be tested to ensure that they operate correctly.

- **Module Testing**

  A module is a collection of dependent components such as a function. A module encapsulates related components so can test without other system modules.

- **Sub-system Testing**

  This phase involves testing collection of modules, which have been integrated into sub-systems. Sub-systems may be independently designed and implemented.

- **System testing**

  The sub-systems are integrated to make up the entire system. The errors that result from unanticipated interaction between sub-systems and system components are removed.

- **Acceptance testing**

  This is the final stage in the testing process before the system is tested for operational use. Any requirement problem or requirement definition problem revealed from acceptance testing are considered and made error free.

 Test plan

Careful planning is needed to the most of testing and controlled testing cost.

6.3 Results

Several errors were detected and rectified and the whole project is working as it should have to work with proper output and high efficiency.

Test case id	Test case	Steps to execute the test case	Expected result	Actual result	Verdict (Pass/Fail)
1	Front screen	Press any key	Remains in the front screen	Remains in the front screen	Pass
2	Front screen	Right click	Menu should appears	Menu appears	Pass
3	Front/other Screen	Select option from menu.	Menu should appear	Date and time appears	Pass
4	Front/other Screen	Select about option from menu	Displays about window.	Displayed About window.	Pass
5	Front/other Screen	Select Exit option,	Program exited	Program exited	Pass

Table 1.Testing for the '3D house' Project,

Chapter 7

SNAPSHOTS

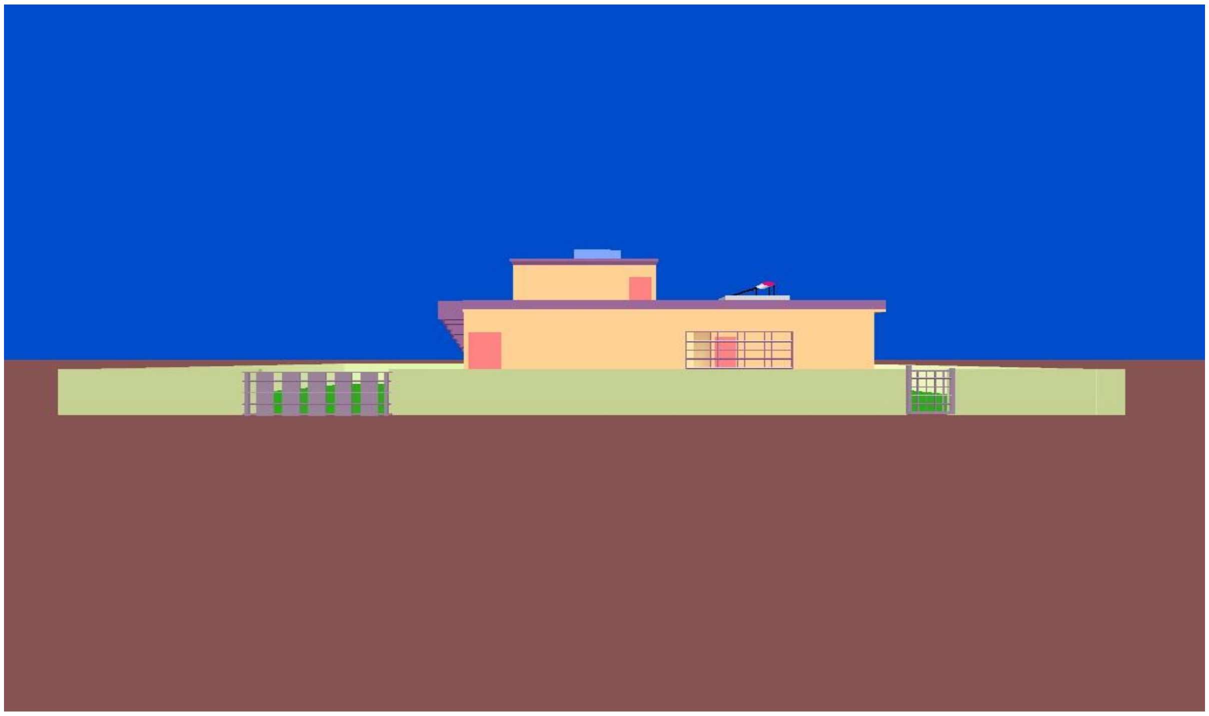

Figure 6.1 After Run the Code

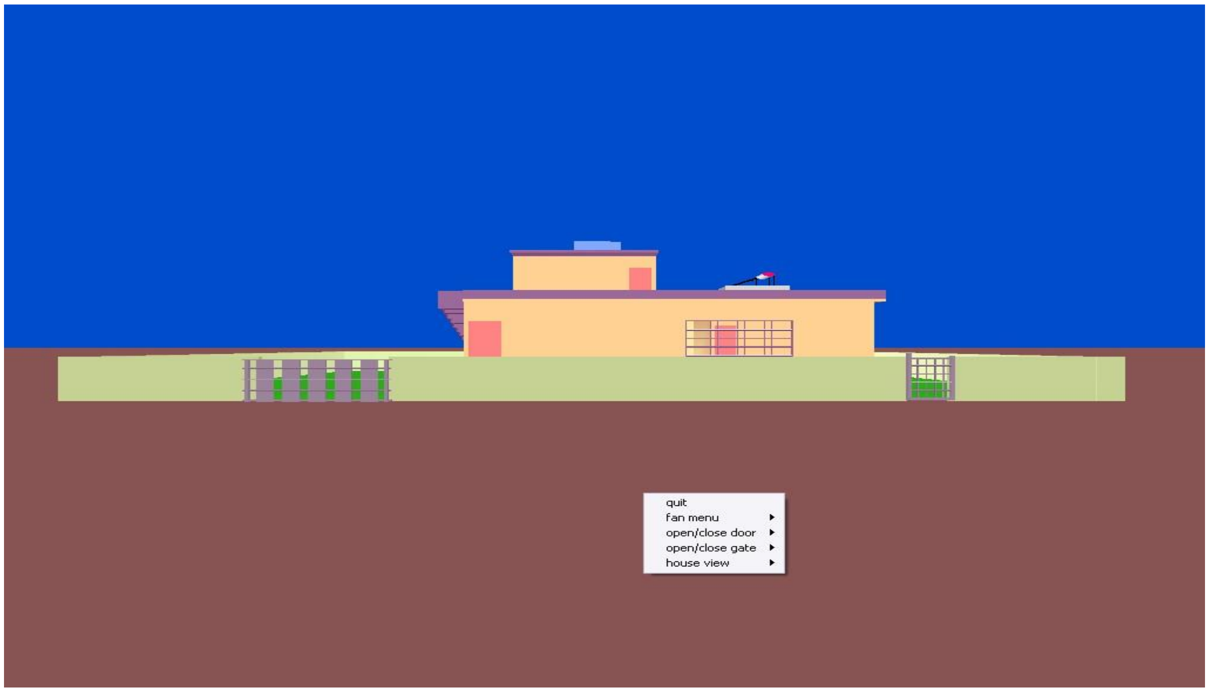

Figure 6.2 After Right Click it's Showing options

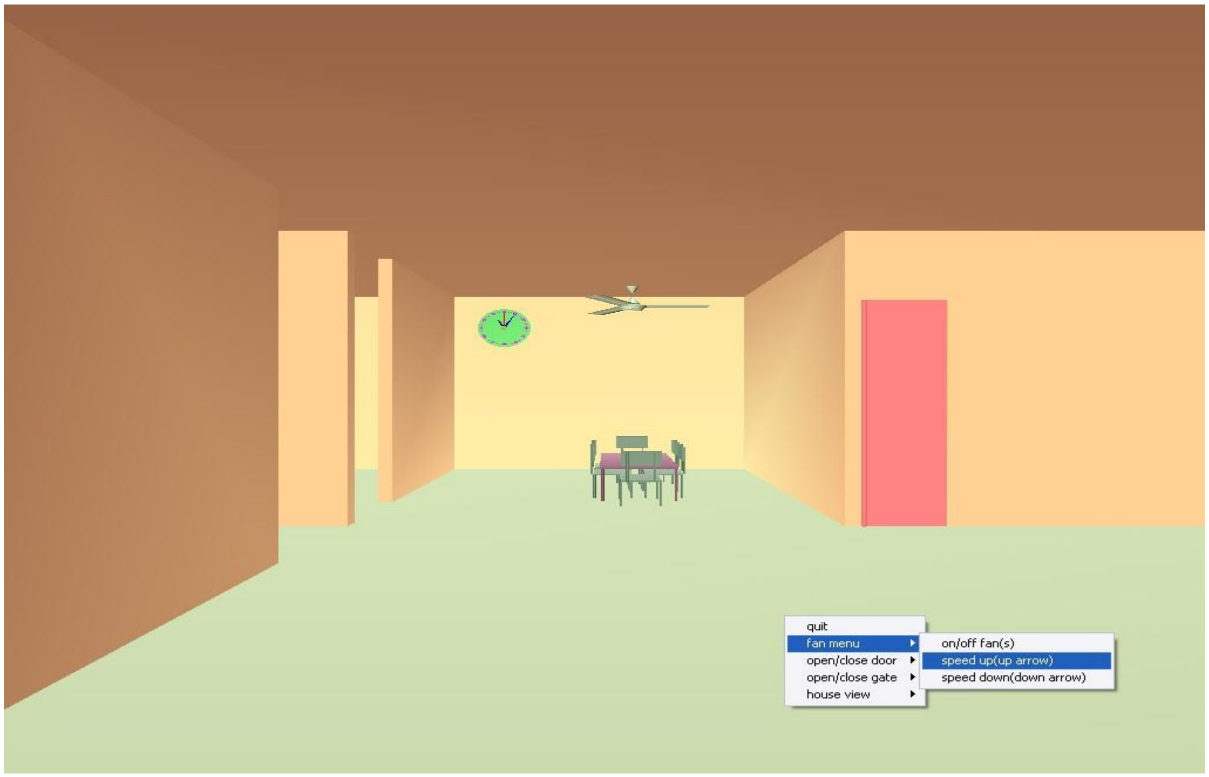

Figure 6.3 After Selected inner view of house

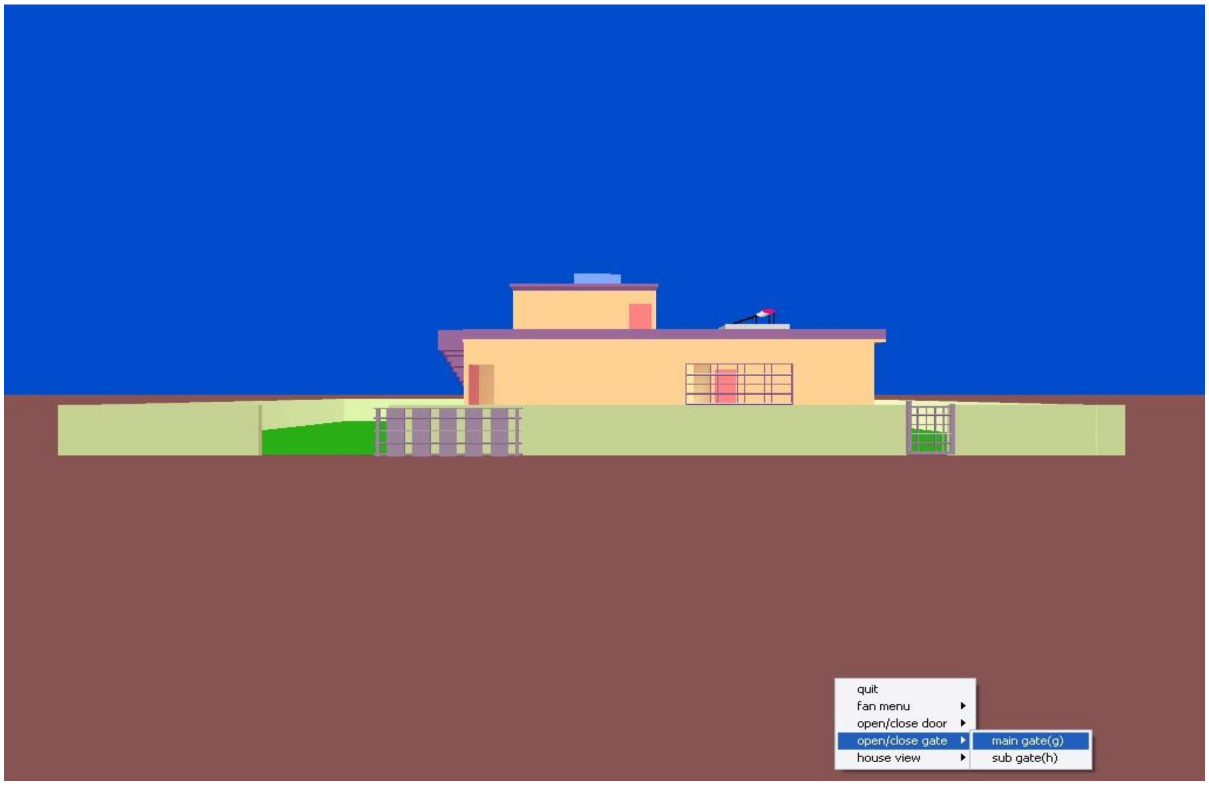

Figure 6.4 Selecting Main door to open

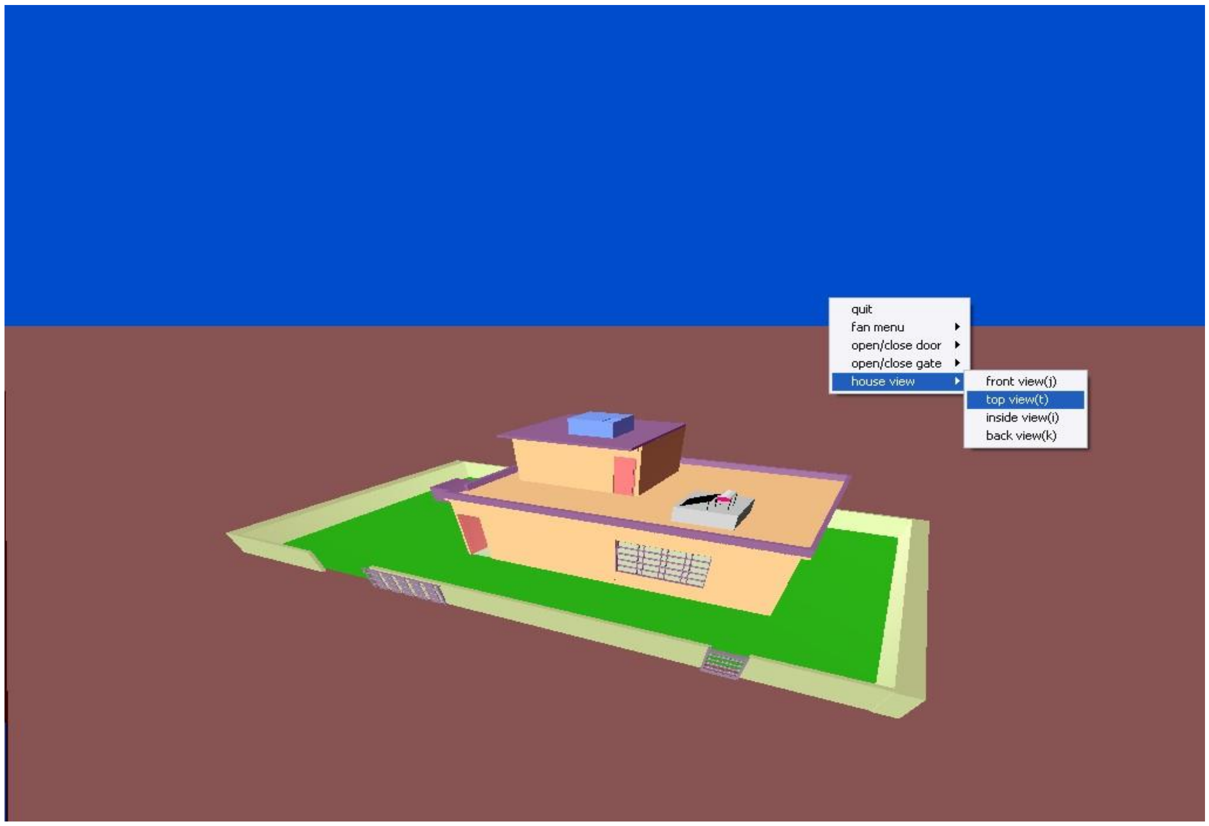

Figure 6.5 House Top view is showing

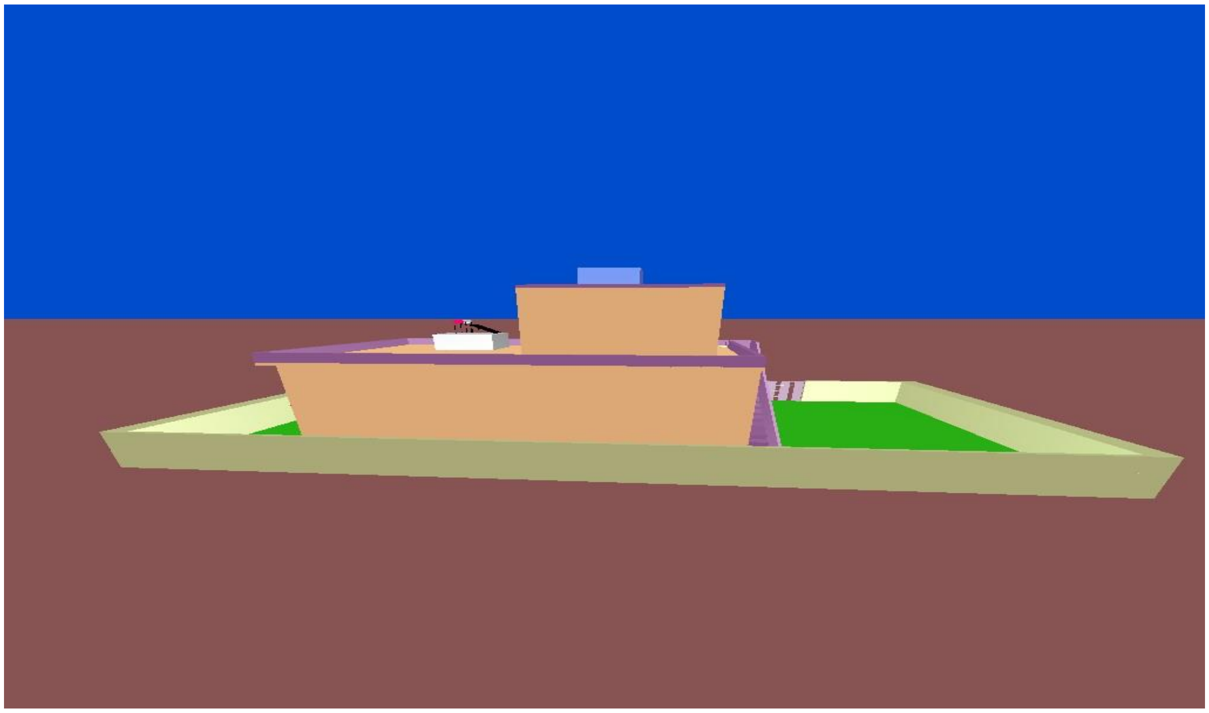

Figure 6.6 Back View of HOUSE

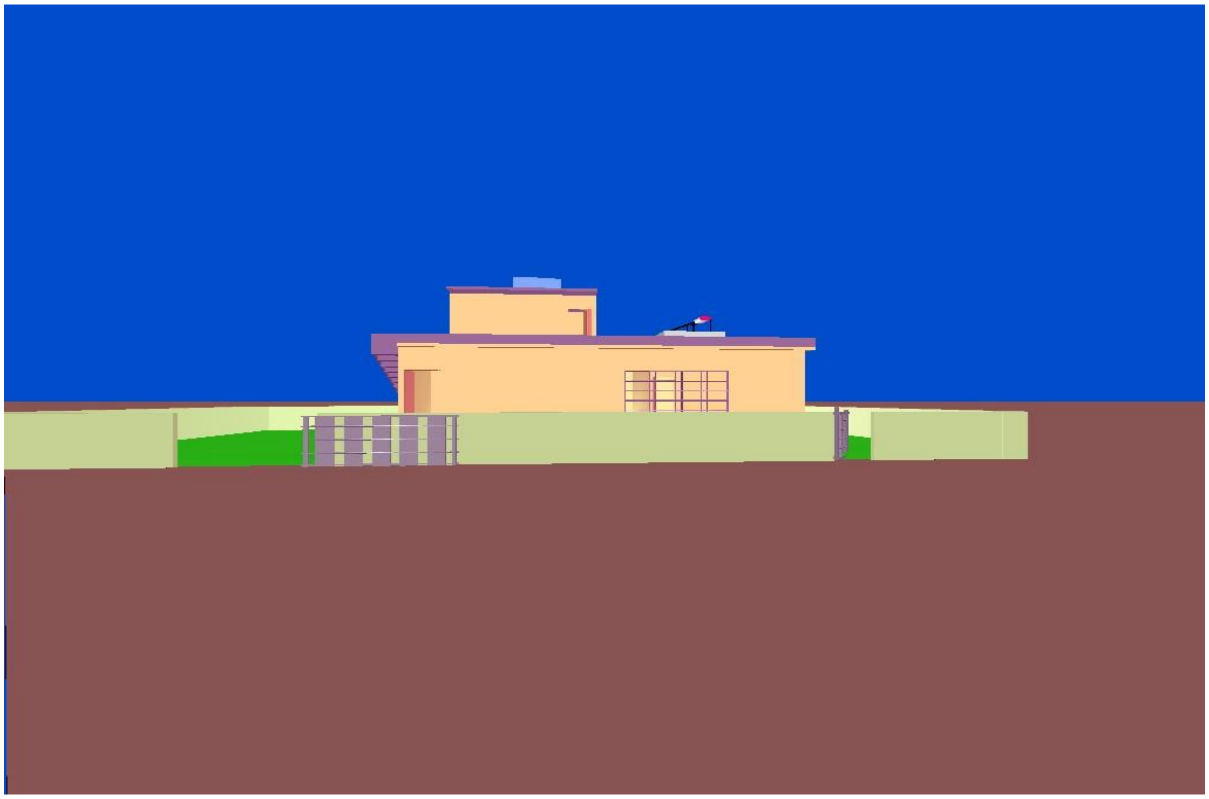

Figure 6.7 All the door's opened

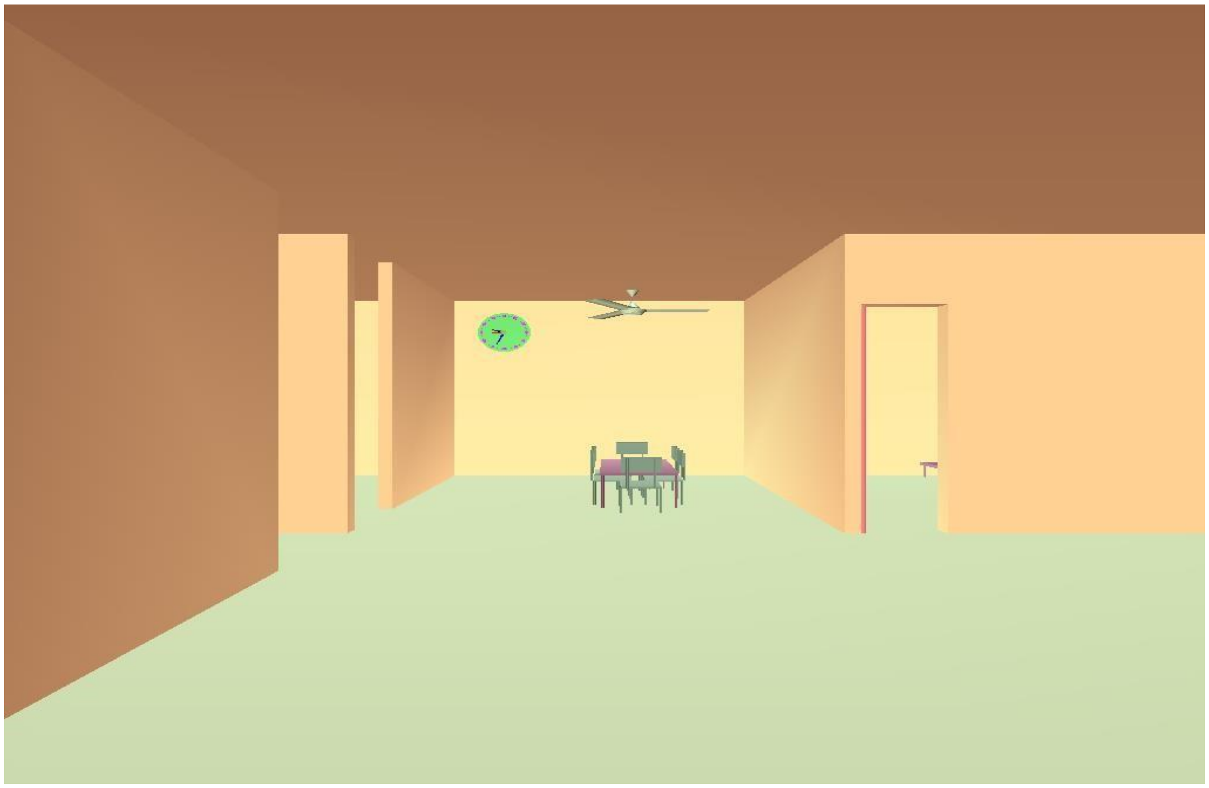

Figure 6.7 Inside Door open & it's showing time

Chapter 8

CONCLUSION AND FUTURE ENHANCEMENT

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.

This package is very useful for the user since it provides the basic information about various OpenGL functions and its component utilities. This is an interactive project which has user friendly interaction given through keyboard. Thus this project meets the basic requirements successfully and is flexible in all respects to one and all.

FUTURE ENHANCEMENT:

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

- Support for pattern filling
- Support for 3d transformations
- Support for transparency of layers
- Keyboard interaction can be implemented.
- 3d visualization can be implemented.
- The project can be further developed to change in clock and also we can give the background effect as well.

BIBLOGRAPHY

Books:

[1] Edward Angel's Interactive Computer Graphics Pearson Education 5th Edition

[2] Interactive computer Graphics --A top down approach using open GL--by Edward Angle

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

[4] Donald D Hearn and M.Pauline Baker, "Computer Graphics with OpenGL", 3rd Edition.

Websites for reference

- Opengl.org
- Stackoverflow.com
- Wikipedia.org