VISVESVARAYA TECHNOLOGICAL UNIVERSITY,

BELAGAVI - 590 018

Mini Project Report

On

"ROTATION OF FAN"

A report submitted in partial fulfilment of the requirements for

COMPUTER GRAPHICS AND VISUALIZATION LABORATORY (17CSL68)

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

AMRUTHA M	**4AL17CS005**
APOORVA H P	**4AL17CS011**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

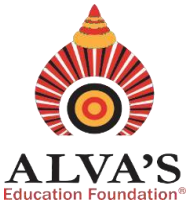
MOODBIDRI-574225, KARNATAKA

2019 – 2020

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI D.K. -574225

KARNATAKA



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Mini Project entitled **"ROTATION OF FAN"** has been successfully completed by

AMRUTHA M	**4AL17CS005**
APOORVA H P	**4AL17CS011**

in the partial fulfillment for the award of Degree of Bachelor of Engineering in Computer and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2019-2020. It is certified that all corrections/suggestions indicated have been incorporated in the report. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the award of Bachelor of Engineering Degree.

Ms. SHILPA	**Dr. Manjunath Kotari,**
Mini Project Guide	**HOD CSE**

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude we acknowledge all those whose guidance and encouragement served as beacon of light and crowned the effort with success.

The selection of this mini project work as well as the timely completion is mainly due to the interest and persuasion of our mini project coordinator **Ms. Shilpa, Assistant professor,** Department of Computer Science & Engineering. We will remember her contribution for ever.

We sincerely thank, **Dr. Manjunath Kotari**, Professor and Head, Department of Computer Science & Engineering who has been the constant driving force behind the completion of the project.

We thank our beloved Principal **Dr. Peter Fernandes,** for his constant help and support throughout.

We are indebted to **Management of Alva's Institute of Engineering and Technology, Mijar, Moodbidri** for providing an environment which helped us in completing our mini project.

Also, we thank all the teaching and non-teaching staff of Department of Computer Science & Engineering for the help rendered.

AMRUTHA M **4AL17CS005**

APOORVA H P **4AL17CS011**

ABSTRACT

In this miniproject we have demonstrated the Rotation of Fan by using OpenGL and the model is designed by pure C++ code. OpenGL is a freely available Computer Graphics operating system. Here we use header files of <GL/glut.h>, OpenGL is an interactive environment. When the program is executed the fan would be rotating at its normal speed and also a mouse click event is added which leads to two options. The first option is for modifying the speed and second is for quitting. The first option further has two more options i.e., fast and slow. On selecting fast option the rotation speed of the fan will increase and on selecting slow option the rotation speed of the fan will decrease.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Computer Graphics is a complex and diversified technology. To understand the technology it is necessary to subdivide it into manageable Parts. This can be accomplished by considering that the end product of computer graphics is a picture. The picture may, of course be used for a large variety of purposes; e.g., it may be an engineering drawing, an exploded parts illustration for a service manual, a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, or a single frame from an animated movie. The picture is the fundamental cohesive concept in computer graphics.

Consider how: Pictures are represented in computer graphics.

- Pictures are prepared for presentation.
- Previously prepared pictures are presented.
- Interaction with the picture is accomplished.

Here "picture" is used in its broadest sense to mean any collection of lines, points, text, etc. displayed on a graphics device.

1.1 COMPUTER GRAPHICS SOFTWARE

The totality of computer graphics software encompasses the concepts from data structures, from data base design and management, from the psychology, ergonometric of the man-machine interface, from programming languages and operating system.

Numerous computer graphics standards can currently be grouped following general categories.

- First is the graphics application interface, where ideas are translated into a form that is understandable by a computer system. Current representative standards are the GKS, GKS-3D, and the Programmer's Hierarchical Interactive Graphics Standards (PHIG).
- The Second is concerned with the storage and transmission of data between graphics systems and between graphics-based computer aided design and computer aided

manufacturing systems. The current standard in this area is the Initial Graphics Exchange Specification (IGES).

- A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

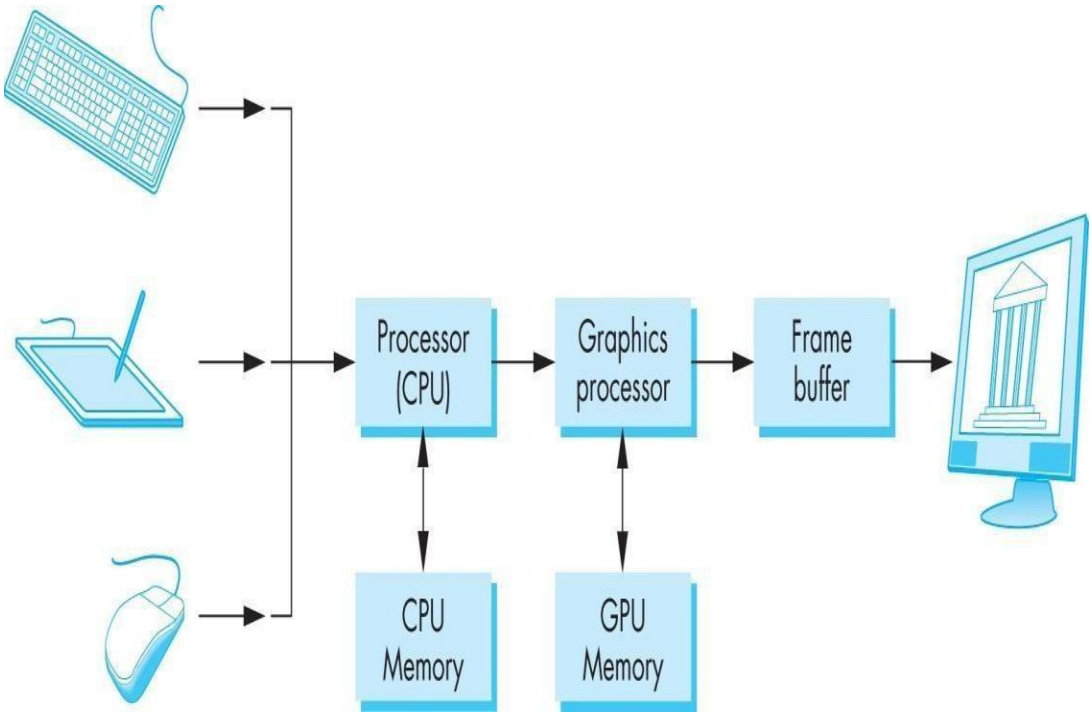


Fig 1.1 Simple Graphic System

1.2 BRIEF HISTORY OF OPENGL

OpenGL was developed by Silicon Graphics and is popular in the video games industry where it completes with Direct3D on Microsoft Windows IrisGL, a proprietary graphics API, is the precursor of OpenGL. It is developed by Silicon Graphics Inc. (SGI).IrisGL was used as the starting point of an open standard for computer graphics that would save time porting applications by avoiding direct hardware access. After SGI cleaned up IrisGL and opened up the standard to other companies, OpenGL was born. In 1992 the OpenGL Architectural Review Board (OpenGL ARB) was established. The OpenGL ARB is a group of companies that maintain and update the OpenGL standard. In 2003 the first OpenGL (Exchange Specifications) ES specification was released. OpenGL ES is a subset of OpenGL designed for mobile phones, embedded devices and

video game systems. In 2004 the OpenGL 2.0 specification was released, including the GLSL (OpenGL Shading Language) specification. In August 2008, the OpenGL 3.0 specification was released.

What is OpenGL?

- OpenGL is a Software Interface to Graphics Hardware.
- OpenGL is designed as a streamlined, hardware-independent interface to beimplemented on many different hardware platforms.
    - It consists of about 250 Distinct Commands.
    - It is Hardware-independent Interface.
    - No command for windows or user input handling.
    - Does not include low-level I/O management.
    - It is mid-level, device independent, portable graphics subroutine package.
    - It's developed primarily by SGI.
    - It consists of 2D/3D graphics, lower-level primitives.
    - OpenGL performs several processing steps on the data to convert it to pixels to form the final desired mage in the frame buffer.
    - OpenGL is network-transparent.

Design

OpenGL serves two main purposes:

- To hide the complexities of interfacing with different 3D accelerations, by presenting the programmer with a single, uniform API.
- To hide the different capabilities of hardware platforms, by require that all implementations support the full OpenGL feature set (using software emulation if necessary).

OpenGL's basic operation is to accept primitives such as points, lines and polygons, and convert them into pixels. This is done by a graphics pipeline known as the OpenGL state machine. Most OpenGL commands either issue primitives to the graphics pipeline, or configure how the pipeline processes these primitives .Prior to the introduction of OpenGL 2.0, each stage of the pipeline

performed a fixed function and was configurable only within tight limits. OpenGL 2.0 offers several stages that are fully programmable using GLSL.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. This contrasts with descriptive (aka scene graph oriented mode) API's, where a programmer only need to describe a scene and can let the library manage the details of rendering it. OpenGL's low level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

OpenGL has historically been influenced on the development of 3D accelerators, promoting a base level of functionality that is now common in consumer-level hardware: Rasterized points, lines and polygons as basic primitives.

Simplified version of the Graphics Pipeline Process; excludes a number of features like blending, VBO's and logic ops.

- A transform and lighting pipeline.
- Z-buffering
- Texture mapping
- Alpha blending

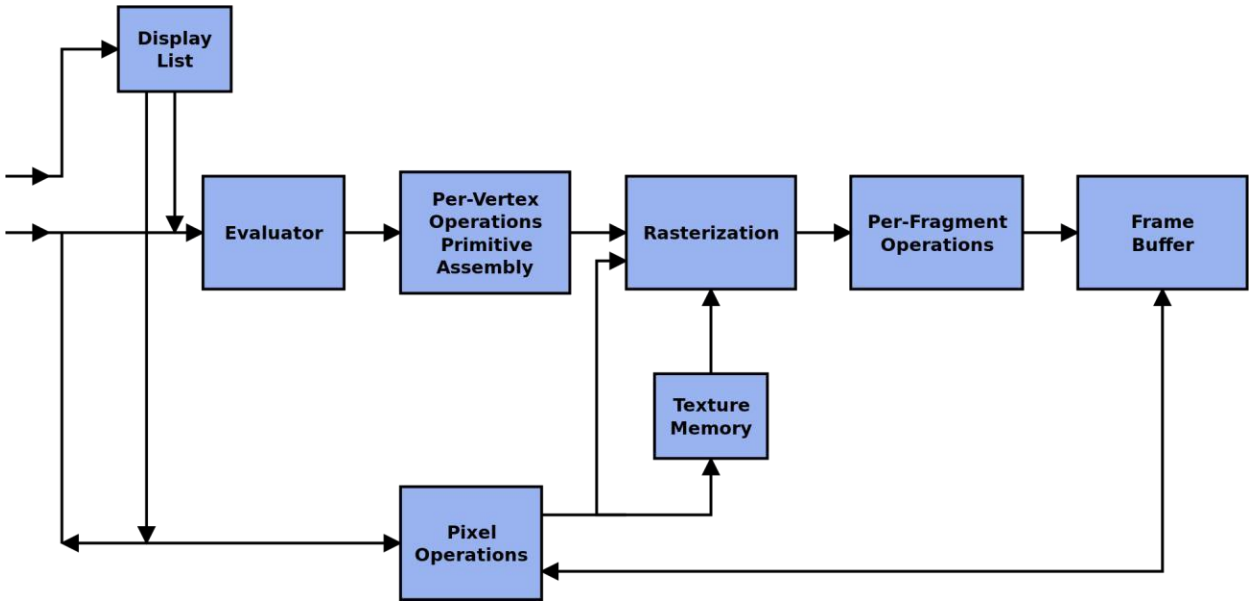


Fig 1.2 Graphics System

A brief description of the process in the graphics pipeline could be:

- Evaluation, if necessary, of the polynomial functions which define certain inputs, like NURBS surfaces, approximating curves and the surface geometry.
- Vertex operations, transforming and lighting them depending on their material. Also clipping non visible parts of the scene in order to produce the viewing volume.
- Rasterization or conversion of the previous information into pixels. The polygons are represented by the appropriate color by means of interpolation algorithms.
- Per-fragment operations like updating values depending on incoming and previously stored depth values or color combination among others.

Lastly fragments are inserted into Frame buffer. Many modern 3D accelerators provide functionality far above this basic line, but these features are enhancements of the basic pipeline rather than radical revisions of it.

1.3 HEADER FILES

- Most of the applications directly use functions from three libraries. The main GL library have names beginning with letters gl usually referred as GL. The second GLU library contains code for creating common objects and simplifying viewing.
- For X window system, the library used to interface with the window system is GLX and gets the input from the external devices into our programs.
- GLUT uses GLX and X libraries.

  #include<GL/glut.h>

  Or

  #include<GLUT/glut.h>

- GLUT a window system independent toolkit for writing OpenGL programs. It implements a simple API for OpenGL. GLUT makes it easy to learn and explore OpenGL programming .GLUT provides portable API that works across all PC and OS platforms.
- GLUT constructs small to medium sized OpenGL programs. It's not a fully featured toolkit so larger applications requiring sophisticated user interfaces are better off using native window system toolkits .GLUT is simple, easy and small.

- The GLUT library has C, C++, FORTRAN and ADA programming bindings. The GLUT source code distribution is portable to nearly all OpenGL implementations and platforms. The current version is 3.7. Additional releases of the library are not anticipated. GLUT is not open source.

The OpenGL interface

Our application will be designed to access OpenGL directly through functions in three libraries namely: gl, glu, and glut.

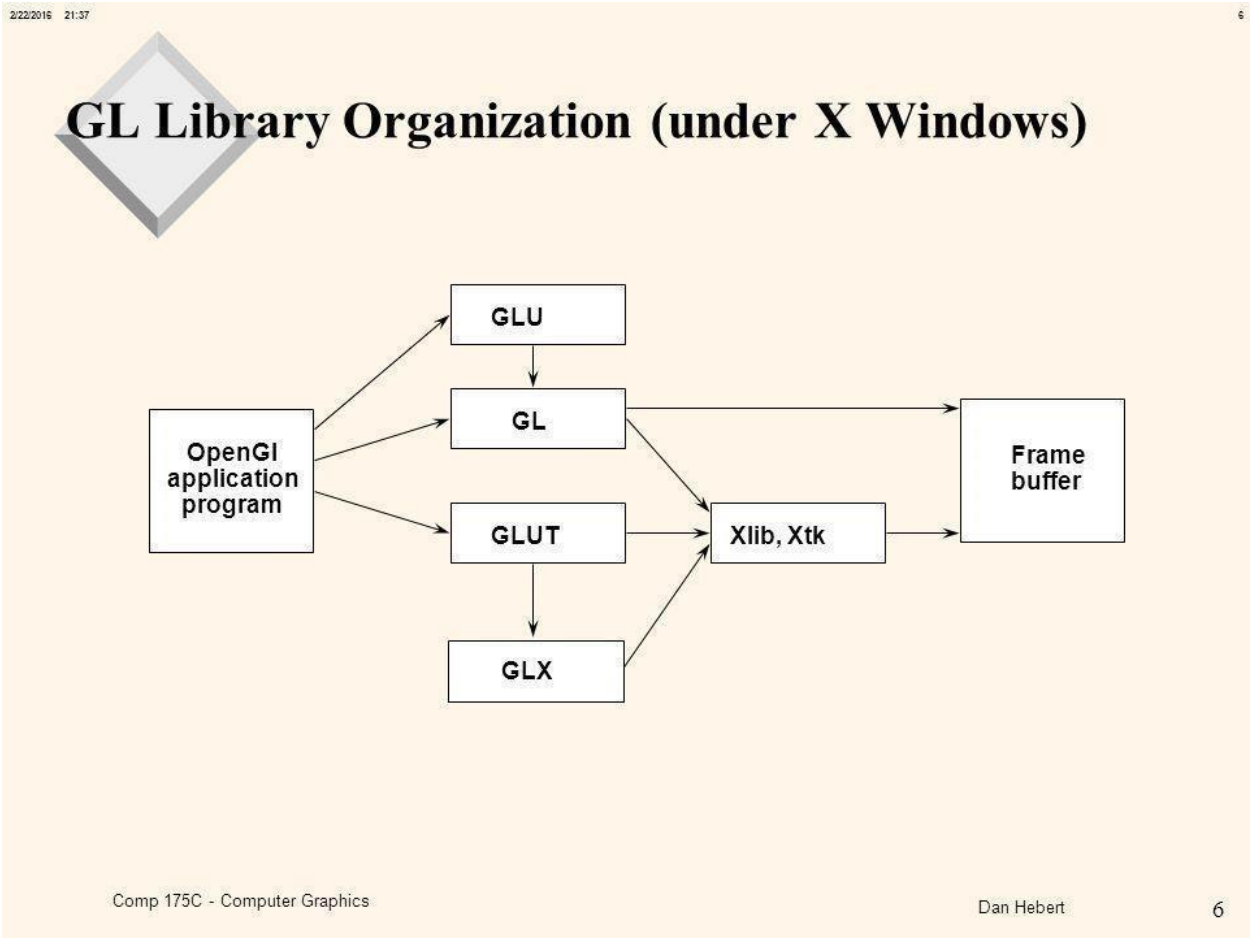


Fig 1.3 Library Organization

OpenGL support libraries:

- GLUT - The OpenGL utility toolkit.
- SDL - The Simple Direct Media Layer.
- GLU - Some additional functions for OpenGL programs.
- GLEE - The OpenGL Easy Extension Library.
- GLEW - The OpenGL Extension Wrangler Library.
- GLUI - A GUI toolkit made with GLUT.
- GLFW - A Portable framework for OpenGL application development.
- GLM - C ++ mathematics toolkit for OpenGL based on the GLSL specification.
- SFML- Simple and Fast Multimedia Library.
- Glut - The OpenGL Utility & Auxiliary Library.

1.4 USES OF COMPUTER GRAPHICS:

Graphics is used in many different areas of industry, business, government**,** education, entertainment etc.

User Interfaces:

Word-processing, spreadsheet and desktop-publishing programs are typical applications of such user-interface techniques.

Interactive plotting in Business, Science and Technology:

The common use of graphics is to create 2D and 3D graphs of mathematical, physical and economic functions, histograms, and bar and pie charts.

Computer Aided Drafting and Design (CAD):

In CAD, interactive graphics is used to design components and systems of mechanical, electrical and electronic devices including structures such as buildings, automobile bodies, aero planes, ship hulls etc.

Simulation and Animation for Scientific Visualization and Entertainment:

Computer-produced animated movies are becoming increasing popular for scientific and engineering visualization. Cartoon characters will increasingly be modeled in the computer as 3D shape descriptions whose movements are controlled by computer commands.

2D Graphics:

These editors are used to draw 2D pictures (line, rectangle, circle and eclipse) alter those with operations like cut, copy and paste. These may also support features like translation, rotation etc.

3D Graphics:

These editors are used to draw 2D pictures (line, rectangle, circle and eclipse). These may also support features like translation, rotation etc.

1.5 ADVANTAGES

- Scientific visualization.
- Information visualization.
- Computer vision.
- Image processing.
- Computational geometry.
- Computational topology.
- Applied mathematics.

CHAPTER 2

SYSTEM ANALYSIS

Aim of this project is to design a 3D fan using opengl, where the speed of rotation of the fan can be controlled by the user. This demo displays the Rotation of fan which is designed and implemented using C++ program language and API's provided by OpenGL. It has been developed to undergo user modifications. The user modification can be specification and movement with several other features. It is our first insight into the vast world of interactive graphics.

Scope - The user can modify the speed of the fan either by increasing its speed or by decreasing the speed. It is user friendly and intuitive to user. It supports the user interaction through mouse. This project has been implemented efficiently to obtain the optimized results and also various functions and features that are made available by OpenGL software package have been utilized effectively.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 SOFTWARE REQUIREMENTS:

API : OpenGL

Platform : Windows

Language Tool : C++

Compile : Microsoft Visual Studio 2010

3.2 HARDWARE REQUIREMENTS

Processor : Pentium PC

RAM : 512MB

Hard Disk : 20GB (approx.)

Monitor : Standard Monitor

Keyword : Standard Keyboard

CHAPTER 4

SYSTEM DESIGN

There is a fan which is rotating at its normal speed whose speed can be modified by either increasing the speed or by decreasing the speed.

Mouse Interaction

Right Click: When we click the right mouse button, menu options are displayed to modify the speed of the fan and quit from the menu.

4.1 DESIGN

Details of graphics routine implemented: This mini project designed extensively uses the OpenGL graphics library functions. So it would be very much suitable to brief about it at this context. The most commonly used functions of the graphics library in our project are:-

- Call back functions
- GL functions
- GLUT functions

Advantage:

- Easy to understand
- Easy to implement
- Fast processing

4.2 CALL BACK FUNCTIONS:

glutDisplayFunc( ) : Graphics are sent to the screen through a function called the display callback and registered with the window system. Here the function name will called whenever the windowing system determines that the OpenGL indo needs to be redisplayed.

4.3 GL FUNCTIONS:

glMatrixMode( ) : glMatrixMode specifies which matrix will be affected by subsequent transformation. Modes can be GL_MODEL_VIEW, GL_PROJECTION OR GL_TEXTURE.

glLoadIdentity( ) : This Function replaces the current matrix with the identity matrix.

glBegin( ), glEnd( ): This function delimits the vertices or a primitive or a group of liking Primitives.

glClear( ): Clears the buffers to present values.

glEnable( ): This enables the server-side GL capabilities.

glClearColor( ): Sets the present RGBA clear color used when clearing the color buffer.

glFlush( ): Forces any buffered openGL commands to execute.

glPushMatrix( ): Pushes to the matrix stack corresponding to the current matrix mode.

glPopMatrix( ): Pops from the matrix stack corresponding to the current matrix mode.

glTranslate[fd]( ): Alters the current matrix by a displacement of(x, y, z).

4.4 GLU FUNCTIONS:

gluOrtho2D ( ): This function defines a 2-D orthographic projection matrix.

4.5 GLUT FUNCTIONS:

glutCreateWindow ( ): This creates a top-level window. The name will be providing to the window system as the window's name. The intent is that the window system will label the window with the same.

glutMainLoop ( ): This enters the GLUT event processing loop. His routine should be called most once in a GLUT program called

glutDisplayFunc ( ): Sets the display callback for the current window.

glutPostRedisplay ( ): Requests that the display call back be executed after the current call back returns.

glutMouseFunc ( ): Sets the mouse callback for the current window.

glutCreateMenu ( ): Creates a new pop-up menu and returns a unique small integer identifier. The range of allocated identifiers starts at one. The menu identifier range is separate from the window identifier range.

glutAddMenuEntry ( ): Adds a menu entry to the bottom of the current menu. The string nameWill be displayed for the newly added menu entry. If the menu entry is selected by the user, the menu's callback will be called passing valueas the callback's parameter.

4.6 GEO PRIMITIVE NAME AND MEANINGS

GL_COLOR_BUFFER_BIT: Indicates the buffers currently enabled for color writing.

GL_PROJECTION: Captures properties intrinsic to the camera itself.

GL_MODELVIEW: Defines how your objects are transformed in your world coordinate frame.

CHAPTER 5

IMPLEMENTATION

5.1 OVERVIEW

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL. It is also used in video games, where it competes with Direct3D on Windows platforms that produce 2D and 3D graphics .with OpenGL, you must build up your desired model from a small set of *geometric primitives* - points, lines, and polygons.

5.2 USER INTERFACE

The project which we have done uses OpenGL functions and is implemented using C++. Our project is to develop a two-player game with two pads and a ball. Player tries to hit the moving ball using his pad, hence preventing the it from hitting the wall. If he fails to hit the ball he will lose the game. The package must have user friendly interface with keyboard and mouse.

5.3 STRUCTURE

- void glutDisplayFunc(void (*func)(void));
- void glutKeyboardFunc(void (*func)(unsigned char key, int x, int y));
- glutPostRedisplay(void);
- glutReshapeFunc(void (*func)(int width, int height));

5.4 ALGORITHM

Step 1 : Start

Step 2 : Fan running at normal speed

Step 3 : Right click to get the options of modify speed and quit

Step 4 : If we choose modify we get

A) fast - fan begins to rotate fast

B) slow - fan begins to rotate slow

Step 5 : Quit

5.5 PSEUDOCODE

```
// fan.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include<stdio.h>
#include<stdlib.h>
#include<GL/glut.h>
///Libraries to Create Custom Graphics
#include"const.h"
#include"cgl.h"
#include"draw.h"
#include"tablefan.h"
static int submenu_id,menu_id,value = 0;
void disp();
void idle();
void speed(int num);
void displayStars(int);
void speed(int num)
 {
        if(num == 3)
                exit(0);
        else if(num == 1)
                TABLEFAN_SPEED = 8.0;
        else
```

```
                TABLEFAN_SPEED = 1.0;
 }
 int main()
{

        glutInitWindowSize(SCREEN_WIDTH, SCREEN_HEIGHT);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_DEPTH);
        glutCreateWindow("CG Demo");
        glutDisplayFunc(disp);
        submenu_id = glutCreateMenu(speed);
        glutAddMenuEntry("fast",1);
        glutAddMenuEntry("slow",2);
        menu_id = glutCreateMenu(speed);
        glutAddSubMenu(" modify speed" ,submenu_id);
        glutAddMenuEntry("quit",3);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        //glutPassiveMotionFunc(passiveMouse);
        glutKeyboardFunc(keyboard);
        glutIdleFunc(idle);
        // Stars will be displayed with some delay
        //glutTimerFunc(100, displayStars, 1);
        glutMainLoop();
}
void disp()
{
        /// Intialize Window Properties
          CGL_Init3D(-HALF_SCREEN_WIDTH, HALF_SCREEN_WIDTH, -
          HALF_SCREEN_HEIGHT, HALF_SCREEN_HEIGHT, 0, 10);
        //Draw Stars
        glPushMatrix();
        glTranslatef(-1, -1, 0.2);
        glScalef(-5, 5, 1);
        //drawStars();
        glPopMatrix();
```

```
        // Draw Table fan
        glDepthMask(GL_TRUE);
        glPushMatrix();
        // Set the Position & Rotation
        glRotatef(20,1, 1, 0);
        glTranslatef(0.2, 0, 0);
        // Set the size of the Table fan
        glScalef(0.8, 0.8, 0.8);
        drawTableFan();
        glPopMatrix();
        // Draw Instruction (aka Breaking News)
        if (INSTRUCTION_POS > 4)INSTRUCTION_POS = -1;
        else INSTRUCTION_POS += 0.0005;
        STRING instruction;
        instruction.text = "Press R to show/hide stars";
        glPushMatrix();
        drawTextUsing(instruction, INSTRUCTION_POS, -0.95, -0.61);
        glPopMatrix();
        glDepthMask(GL_TRUE);
        glFlush();
        glutSwapBuffers();
}
void displayStars(int val)
{
        glutTimerFunc(150, displayStars, 10);
        //STARS_DISPLAY_NOW = true;
        glutPostRedisplay();
}
void idle()
{
        glutPostRedisplay();
}
```

CHAPTER 6

RESULTS

6.1 SNAPSHOTS

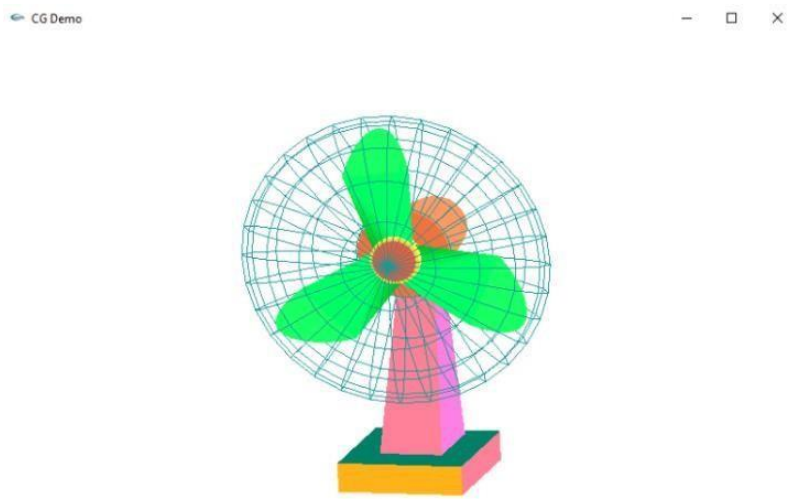


Fig 6.1.1

The Fig 6.1 represents the initial speed of the fan when code is executed.

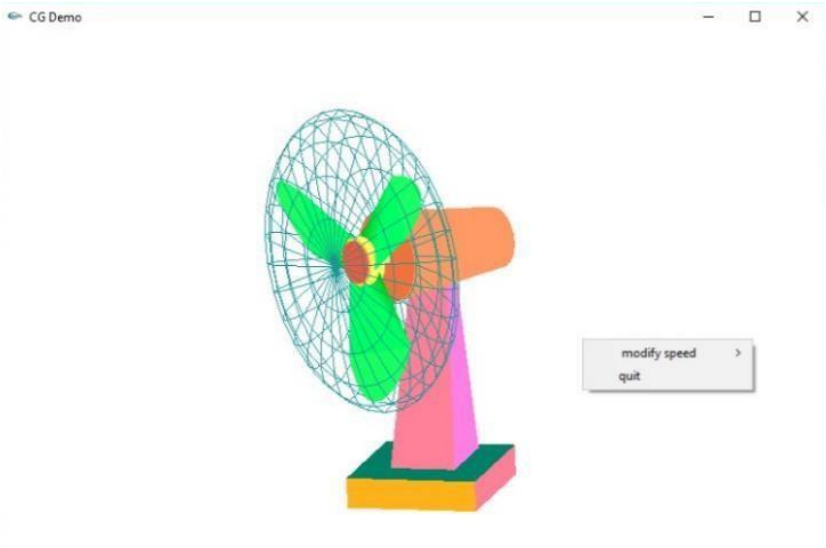


Fig 6.1.2

The fig 6.1.2, displays the menu option on right click.

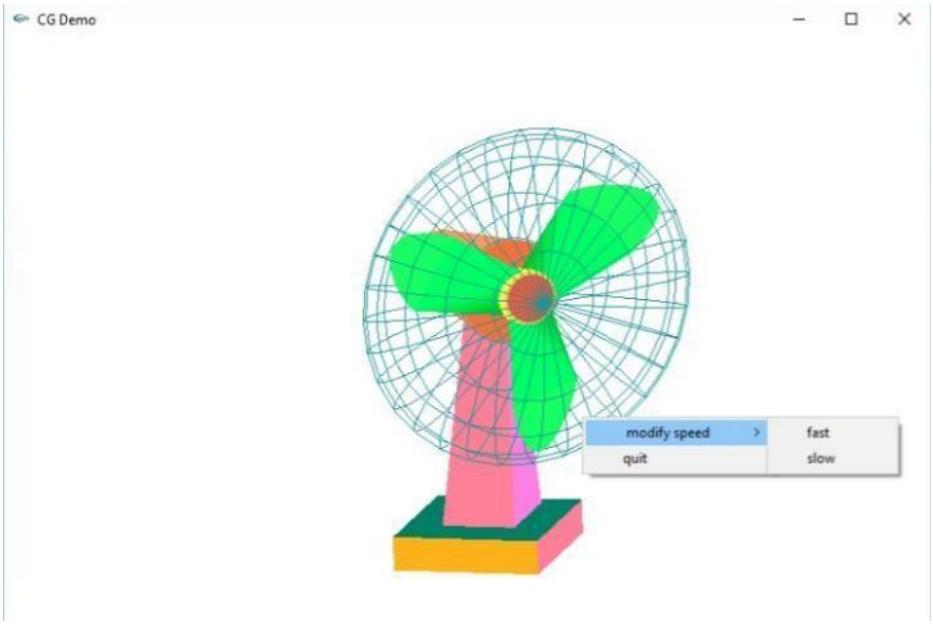


Fig 6.1.3

The fig 6.1.3, represents the options provided by first menu that is fast and slow

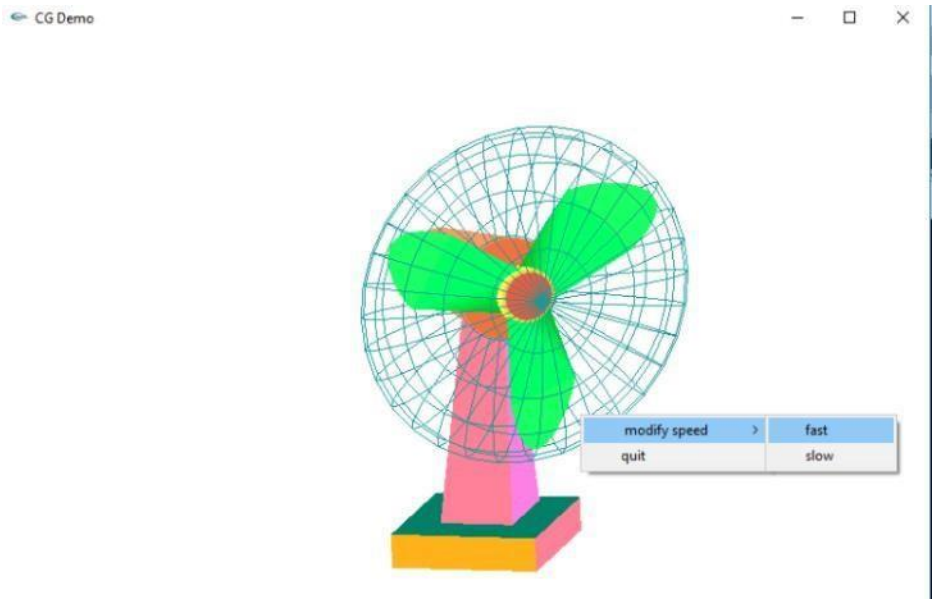


Fig 6.1.4

The fig 6.1.4, indicates the fast option choosen by the user.

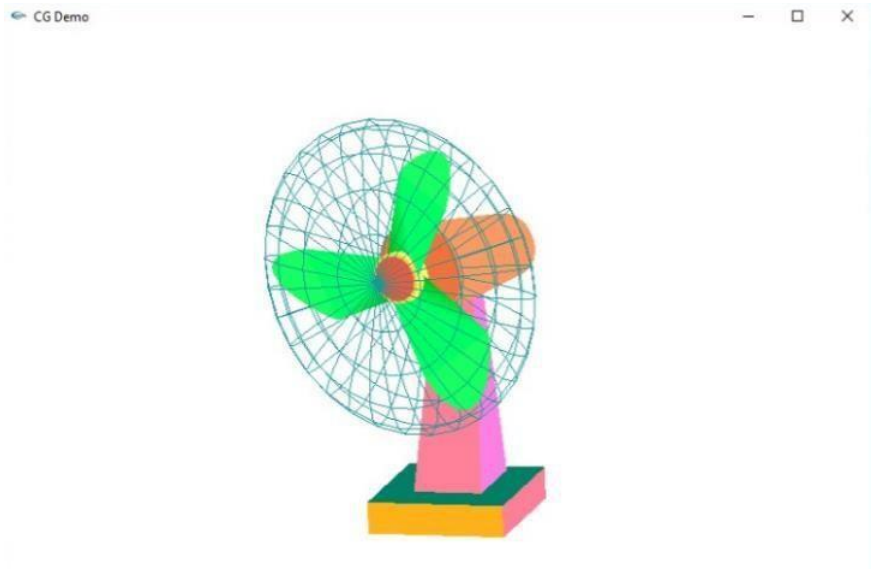


Fig 6.1.5

The fig 6.1.5 represents the speed of the fan when it is in fast mode.

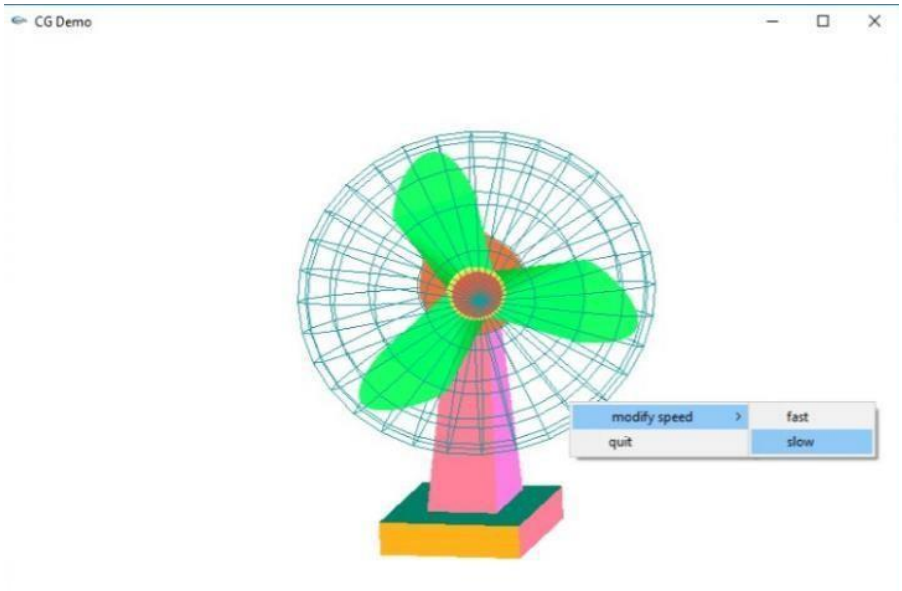


Fig 6.1.6

In fig 6.1.6, the user selects the slow option.

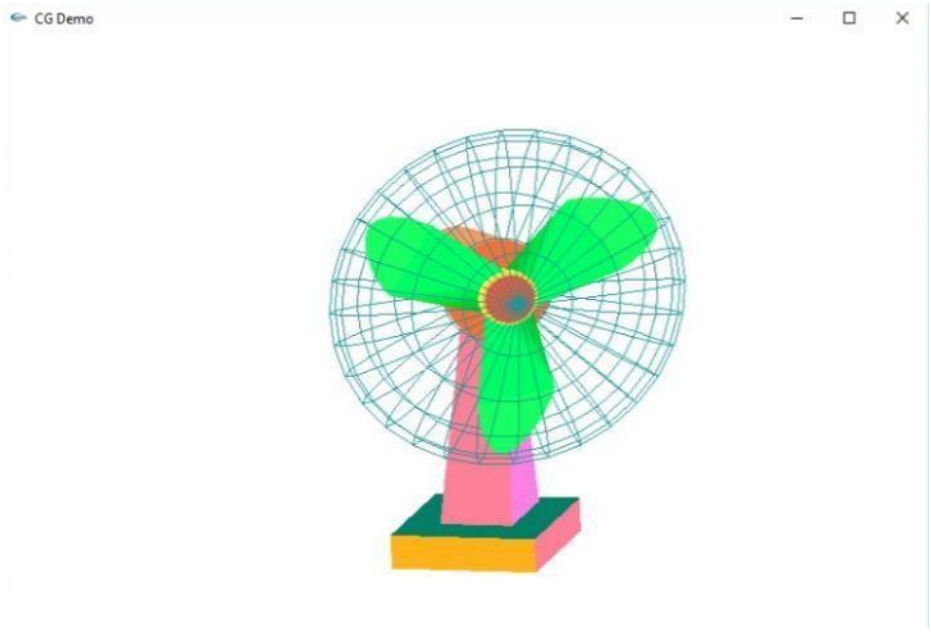


Fig 6.1.7

In fig 6.1.7, the fan is rotating in slow speed.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

This project involving OpenGL was completed successfully using Microsoft Visual C++ 2010.

There are many advantages of Computer Graphics and OpenGL. The OpenGL remain standard programmer's interface both for writing and developing application programs. OpenGL is supported on all platforms like Microsoft Windows, Apple Macintosh operating system etc…OpenGL can be implemented by using both C and C++ languages.

This project has focused on the design and development of a useful, intuitive and user-friendly Three-Dimensional Geometric Modeling application. The prototype application has succeeded in providing this interface to the user. Clearly many useful features have been realized without compromising the interface and a foundation has been built on which to add other powerful features.

This prototype is of more intuitive and user friendly. This improved interface allows the user to view object from several angles and to see a rendering of the model at all times. This allows the user to easily move objects in space and to immediately see the effects the movement has on the rendered scene.

7.2 FUTURE ENHANCEMENT

This project is a user friendly application that gives an idea about the rotating blades of different primitives in 3D. But still there is room for improvement like following:

- We can give the option of switches like on and off conditions.
- We can also allow the user to choose the speed as per his convenience.
- The GUI can be enhanced by using the vast features provided by Visual C++.

REFERENCES

1. Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011.
2. Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition, Pearson Education, 2008.
3. Gl programming sites- http://www.glprogramming.com
4. OpenGL source: http://www.opengl.org