**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview of computer graphics**

Computer Graphics become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

Although early application in engineering & science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today people find Computer Graphics in a diverse area such as science, engineering, medicine, business, industry, government, art, entertainment, education and training. Computer graphics as generalized tool for drawing and creating pictures and simulate the real world situations within a small computer window.

**1.2 History**

William fetter was credited with coning the term Computer Graphics in 1960, to describe his work at Boeing. One of the first displays of computer animation was future world (1976), which included an animation of a human face and hand-produced by Carmel and Fred Parkle at the University of Utah.

There are several international conferences and journals where the most significant results in computer-graphics are published. Among them are the SIGGRAPH and Euro graphics conferences and the association for computing machinery (ACM) transaction on Graphics journals.

**1.3 Applications of computer graphics**

Nowadays Computer Graphics used in almost all the areas ranges from science, engineering, medicine, business, industry, government, art, entertainment, education and training.

**1.3.1 CG in the field of CAD**

Computer Aided Design methods are routinely used in the design of buildings,

Automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

**1.3.2 CG in presentation Graphics**

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

**1.3.3 CG in computer Art**

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist’s paintbrush program (lumena), other pain packages, desktop packages, mathematics packages, animation packages that provide facility for designing object motion. Ex: cartoons design is an example of computer art which uses CG.

**1.3.4 Entertainment**

Computer graphics methods are now commonly used in making motion pictures, music, videos, games and sounds. Sometimes graphics objects are combined with the actors and live scenes.

**1.3.5 Education and Training**

Computer generated models of physical financial, economic system is often as education aids. For some training application special systems are designed. Ex: specialized system is simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

**1.3.6 Image Processing**

Although the methods used in CG image processing overlap, the 2 areas are concerned with fundamentally different operations. In CG a computer is used to create picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans, TV scans.

**1.4 User interface**

It is common for software packages to provide a graphical interface. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing.

* 1. Problem statement

Learning computer graphics as a part of academics and implementing it as a project on OpenGL made us to implement simulation of windmill. i.e. how does the process of making the visualization of the simulation of windmill and how to generate electrical energy using wind?

* 1. Objectives of the project
* To create a visualization of the simulation of windmill.
* To apply the visual method of learning about the simulation of windmill.
* Illustrating about the keyboard and mouse interaction.
* To provide reference to learn about the simulation of windmill.

1.7Organization of the report

This section deals with the Introduction and organization of the project report. Chapter 2 discusses the basic concepts OpenGL. Chapter 3 discusses the basic concept and working principle. Chapter 4 gives information about the design and implementation Chapter 5 include results and snapshot and. Chapter 6 gives the conclusion and future enhancement of the project.

.

**CHAPTER 2**

**INTRODUCTION TO OPENGL**

**2.1 Introduction**

OpenGL is a software interface to graphics hardware. This interface consists of about 150 distinct commands that you use to specify the objects and operations needed to produce interactive three-dimensional applications. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms.

Most of our application will be designed to access OpenGL directly through functions in three libraries. Functions in the main GL (or OpenGL in windows) library have names that begin with the letters gl and are stored in a library usually referred to as GL (or OpenGL in windows). The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library but application programmers prefer not to write the code repeatedly. The GLU library is available in all OpenGL implementations; functions in the GLU library begin with letters glu.

To interface with the window system and to get input from external devices into our programs, developer needs at least one more library. For each major window system there is a system-specific library that provides the “glu” between the window system and OpenGL. For the X window system, this library is called GLX, for windows, it is wgl, and for the Macintosh, it is agl. Rather than using a different library for each system, here it uses a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

Fig 2.1 shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.

OpenGL application Program

GLU

GL

GLUT

GLX

Xlib, Xtk

Frame Buffer

**Fig 2.1: Library organization of OpenGL**

**2.2 OpenGL command syntax**

OpenGL commands use the prefix **gl**and initial capital letters for each word making up the command name. Similarly, OpenGL defined constants begin with GL\_, use all capital letters, and use underscores to separate words (like GL\_TRUE).

Some extraneous letters are appended to some command names (for example, the 3f in glColor3f() and glVertex3f()). It's true that the Color part of the command name glColor3f() is enough to define the command as one that sets the current color. However, more than one such command has been defined so as to use different types of arguments. In particular, the 3 part of the suffix indicates that three arguments are given; another version of the Color command takes four arguments. The f part of the suffix indicates that the arguments are floating-point numbers. Having different formats allows OpenGL to accept the user's data.

**2.3 Pixel operation**

Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components. Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the Rasterization step. If pixel data is read from theframe buffer, pixel-transfer operations (scale, bias, mapping, and clamping) are performed. Then these results are packed into an appropriate format and returned to an array in system memory.

There are special pixel copy operations to copy data in the frame buffer to other parts of the frame buffer or to the texture memory. A single pass is made through the pixel transfer operations before the data is written to the texture memory or back to the frame buffer.

**2.4 Texture assembly**

An OpenGL application may wish to apply texture images onto geometric objects to make them look more realistic. Some OpenGL implementations may have special resources to accelerate texture performance. There may be specialized, high-performance texture memory.

**2.5 Rasterization**

Rasterization is the conversion of both geometric and pixel data into *fragments*. Each fragment square corresponds to a pixel in the frame buffer. Line and polygon stipples, line width, point size, shading model, and coverage calculations to support initializing are taken into consideration as vertices are connected into lines or the interior pixels are calculated for a filled polygon. Color and depth values are assigned for each fragment square.

Fragment

Processor

Rasterizer

Clipper and

Primitive Assembler

Vertex Processor

VerticesPixel

**Figure 2.2 Block diagram showing Rasterization**

**2.6Immediate mode and display lists**

All data, whether it describes geometry or pixels, can be saved in a *display list* for current or later use. When a display list is executed, the retained data is sent from the display list just as if it were sent by the application in immediate mode

#### Transforming to Window Coordinates

Before clip coordinates can be converted to window *coordinates*, they are normalized by dividing by the value of w to yield *normalized device coordinates*. After that, the viewport transformation applied to these normalized coordinates produces window coordinates. One can control the viewport, which determines the area of the on-screen window that displays an image, with glDepthRange() and glViewport().

#### Matrix Transformations

Vertices and normals are transformed by the modelview and projection matrices before they're used to produce an image in the frame buffer. You can use commands such as glMatrixMode(), glMultMatrix(), glRotate(), glTranslate(), and glScale() to compose the desired transformations, or you can directly specify matrices with glLoadMatrix() and glLoadIdentity(). Use glPushMatrix() and glPopMatrix() to save and restore modelview and projection matrices on their respective stacks.

* The basic model for OpenGL command interpretation is immediate mode, in which a command is executed as soon as the server receives it; vertex processing, for example, may begin even before specification of the primitive of which it is a part has been completed. Immediate mode execution is well-suited to interactive applications in which primitives and modes are constantly altered. In OpenGL, the fine-grained control provided by immediate mode is taken as far as possible: even individual lighting parameters (the diffuse reflectance color of a material, for instance) and texture images are set with individual commands that have immediate effect. While immediate mode provides flexibility, its use can be inefficient if unchanging parameters or objects must be re-specified. To accommodate such situations, OpenGL provides display lists. A display list encapsulates a sequence of OpenGL commands (all but a handful of OpenGL commands may be placed in a display list), and is stored on the server. The display list is given a numeric name by the application when it is specified; the application need only name the display list to cause the server to effectively execute all the commands contained within the list. This mechanism provides a straightforward, effective means for an application to transmit a group of commands to the server just once even when those same commands must be executed many times.

**2.6.1 Display List Optimization**

Accumulating commands into a group for repeated execution presents possibilities for optimization. Consider, for example, specifying a texture image. Texture images are often large, requiring a large, and therefore possibly slow, data transfer from client to server (or from the server to its graphics subsystem) whenever the image is re-specified. For this reason, some graphics subsystems are equipped with sufficient storage to hold several texture images simultaneously. If the texture image definition is placed in a display list, then the server may be able to load that image just once when it is specified. When the display list is invoked (or re-invoked), the server simply indicates to the graphics subsystem that it should use the texture image already present in its memory, thus avoiding the overhead of re-specifying the entire image. Examples like this one indicate that display list optimization is required to achieve the best performance. In the case of texture image loading, the server is expected to recognize that a display list contains texture image information and to use that information appropriately. This expectation places a burden on the OpenGL implementer to make sure that special display list cases are treated as efficiently as possible. It also places a burden on the application writer to know to use display lists in cases where doing so could improve performance. Another possibility would have been to introduce special commands for functions that can be poor performers in immediate mode. But such specialization would clutter the API and blur the clear distinction between immediate mode and display lists.

**2.7 Advantages of using OpenGL**

* Industry standard: An independent consortium, the OpenGL Architecture Review Board, guides the OpenGL specification. With broad industry support, OpenGL is the only truly open, vendor-neutral, multiplatform graphics standard.

Stable: OpenGL implementations have been available for more than seven years on a wide variety of platforms. Additions to the specification are well controlled, and proposed updates are announced in time for developers to adopt changes.

* Backward compatibility requirements ensure that existing applications do not become obsolete.
* Reliable and portable: All OpenGL applications produce consistent visual display results on any OpenGL API-compliant hardware, regardless of operating system or windowing system.
* Evolving: Because of its thorough and forward-looking design, OpenGL allows new hardware innovations to be accessible through the API via the OpenGL extension mechanism. In this way, innovations appear in the API in a timely fashion, letting application developers and hardware vendors incorporate new features into their normal product release cycles.

**CHAPTER 3**

**BASIC CONCEPT AND WORKING PRINCIPLE**

Computer graphics are graphics created using computers and more generally the representation and manipulation of pictorial data by a computer. In this project, weare representing layout of simulation of windmill containing the fans and power house that is when wind is moving, fans also rotate according to the direction of the wind for generating electrical power.

It will also contain the representation of clockwise and anticlockwise in rotation according to speed of wind.

Program will take input keys as command to show the movements of these objects

Prominent movements of objects that we are concerned in this project are:

1. Rotation of turbine blades on a simple principle that is by using wind.
2. Revolution of windmill is generating the electrical power by result of air (wind).
3. Revolution of wind turns 2 or 3 propeller-like blades around the rotor which spins a generator to create electricity.

For each movement of object to start there will be separate key input from key board and mouse which will cause object to start rotating along specific axis(clockwise and anticlockwise).

The main object of our project is to illustrate arbitrary motion of wind revolution and

Turbine blades rotation for electrical power with visualization of speed variation of wind in day mode and night mode.

**3.1 Working Principle of the Simulation of Windmill**

In this visualization, user can interacts with the application to perform transformation against wind. Users can do the revolution and rotation against fan turbine blades in simulation of windmill.

User can interact with the application by keyboard function and mouse function as follows:

**Options Actions**

* D Press D for Day Mode
* N Press N for Night Mode
* No wind There is no wind Flowing
* Wind CWWind flowing in the clockwise direction
* Wind AWCWind flowing in the anticlockwise direction
* Fast Wind CWWind flowing faster in the clockwise direction
* Fast Wind ACWWind flowing faster in the anticlockwise direction
* Quit Leave the program

**CHAPTER 4**

**DESIGN AND IMPLEMENTATION**

**4.1 Design**

The whole program has been implemented in C language. The bottom line of the design is Translation function, mouse interaction and keyboard interaction are included and some minor functions are used to print the text on the screen. In this project, ellipse is the major part and it is created by using simple polygon. By using the polygon, one can easily create the ellipse.

**4.2 Implementation**

The OpenGL provides very powerful translation facilities which relive the programmers by allowing them to concentrate on their jobs rather than focusing on how to implement these operations

**4.2.1 Built in functions**

* **void glutInitDisplayMode(unsigned int mode)**

*mode* – specifies the display mode. Use the *mode* to specify the colour mode

And the number of type of buffers.

* **void glutInitWindowSize(int width, int height)**

*Width* - The width of the window.

*Height* - The height of the window.

* **void glutWindowPosition(int x, int y)**

*x* -Window X location in pixels.

*y* -Window Y location in pixels.

glutWindowPositionset the initial window position and size respectively.

* **intglutCreateWindow(char \*name);**

*name* - ASCII character string for use as window name.

glutCreateWindowcreates a top-level window.

* **void glutDisplayFunc(void (\*func)(void));**

*func* - The new display callback function.

glutDisplayFunc sets the display callback for the current window*.*

* **void glutMouseFunc(void (\*func)(int button, intstate,int x, int y));**

*func--* The new mouse callback function.

glutMouseFunc sets the mouse callback for the *current window*.

* **void glutKeyboardFunc(void (\*func)(char key,intwidth,int height));**

*func--* The new keyboard callback function.

glutKeyboardFunc sets the keyboard callback for the *current window*.

* **void glViewport(intx,inty,GLsizeiwidth,GLsizei height);**

Specifies a width\*height viewport in pixels whose lower-left corner is at (x,y)

Measured from the origin of the window.

* **void glutMainLoop(void);**

glutMainLoop enters the GLUT event processing loop.

* **glPushMatrix() and glPopMatrix()**

Pushes to and pops from the matrix stack corresponding to the current matrix

mode.

* **void glFlush()**

Forces any buffered openGL commands to execute.

**,**

* **void glClear(GL\_COLOR\_BUFFER\_BIT)**

To make the screen solid and black.

* **gluquadricObj\* gluNewQuadric()**

Returns a pointer to a new quadric object.

**4.2.2 User defined function**

* **void background1()**

This function is to displays a background image on screen. Here we

Used two dimension for background image. It doesn’t access any parameter.

* **void title()**

This function is used to display all titles which are used in project.

* **void road()**

This function is used to display a road by using polygon function.

* **void clouds()**

This function draws a clouds in project.

* **void powerstation()**

This function used to create a PowerStation in this project.

* **void mykey(unsinged char,int,int)**

This function used to handle interaction of keyboard.

* **void spinclockwise()**

This function is used to rotate a fan as clockwise in day mode as well as night

mode according to values.

* **void anticlockwise()**

This function is used to rotate a fan as anticlockwise day mode as well as night modeaccording to values.

* **void menu()**

This function is used to create a menu having different options for user interface.

**4.3 Hardware and Software requirements**

**Hardware requirements:**

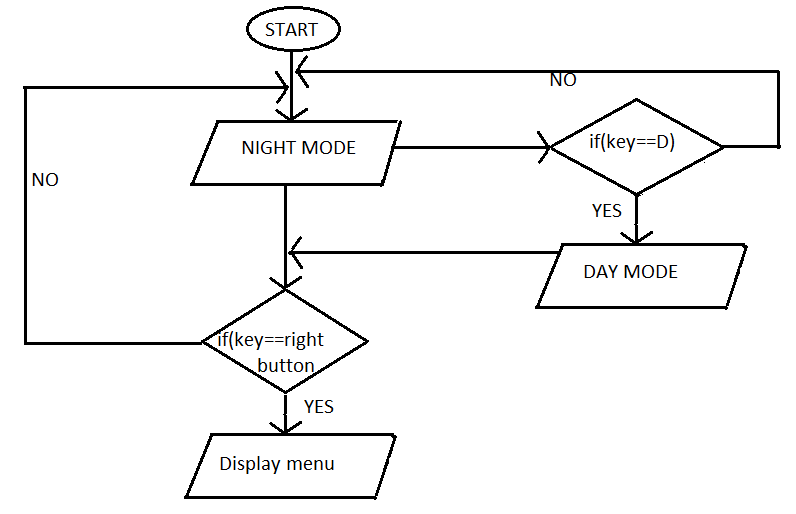
* Processor : Intel Pentium-4, Intel Pentium Dual Core, Intel Pentium Core2duo
* Hard disc : 40GB and above
* RAM : 256MB and above
* Monitor : CRT or TFT

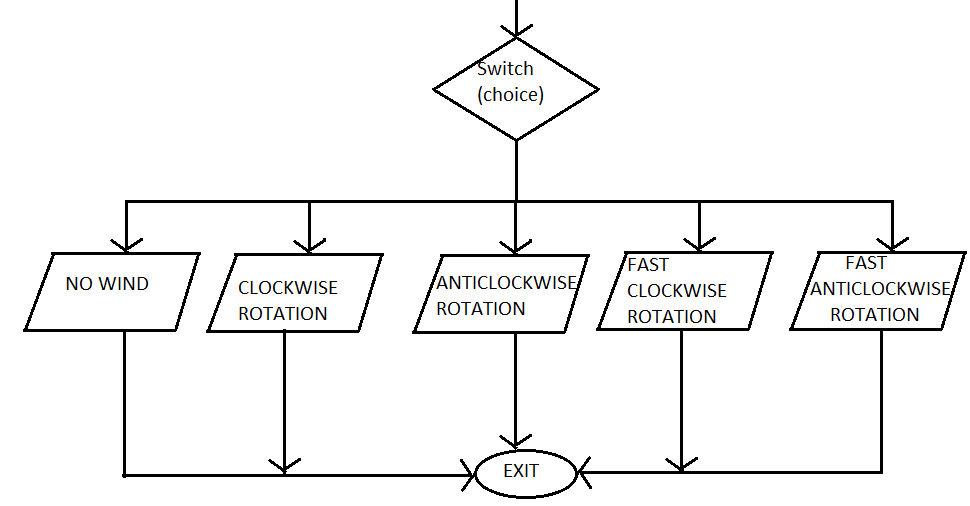
**Software requirements:**

* Operating system: Windows 2000, Windows XP, Windows VISTA. Windows 7
* Application: Visual studio 6.0.

**4.4 Flow Chart**

The following diagram shows the flow chart of the project. It shows the flow of control throughout the program. This is very helpful for the user to know the usage of the simulation of windmill. By seeing this flow chart user can easily analyze the implementation.

****

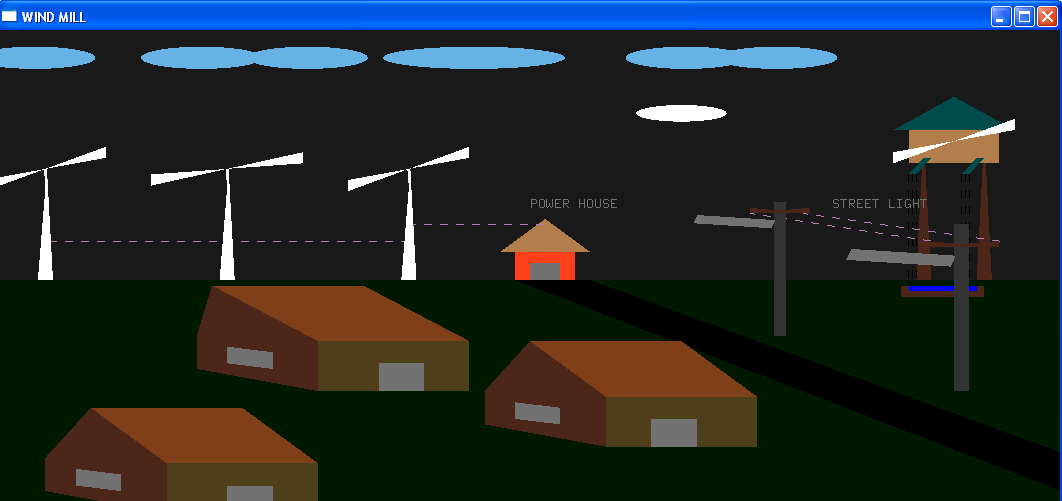
****

**Fig3.2 Flowchart**

**CHAPTER 5**

**RESULTS AND SNAPSHOTS:**

**Fig 5.1:This Snapshot showing night mode of project.**



**Fig 5.2 :Snapshot showing day mode of project along with menu options.**

******

**Fig 5.3:Snapshot of rotation of clockwise**

****

**Fig 5.4:Snapshot of rotation of anti-clockwise**

******

**CONCLUSION & FUTURE SCOPE**

**Conclusion:**

Designing and implementing project in graphics is a great experience. Electric power generation through windmill simulation is about to show how we efficiently make use of renewable energy using OpenGL functions. This project consists of many user defined function such as increasing windmill fan speed clockwise and anti-clockwise and movement of clouds. Depicting day mode as well as night mode.

This application is like open source where anyone can design and add their own codes, do more actions with it. Even more features and good graphics can be added to this application to make it even more interesting and provide good graphical user interface. Developing this application in compatible with touch screen will provide even more comfortable and easy use. Providing background sounds to different actions in the project will make it more real and understandable. Addition of some more in-built 2D and 3D models of geometric figures is a feasible idea. Designing this type of application is very interesting.

**Future Scope:**

Further development in the project can be done by illustrating graphically some other conditions other than what we have illustrated and some more advanced techniques that are recently used can be implemented. It can be made still more attractive using implementation of wind mill and make it user interactive. The project is widely open to anyone who wishes to improve it further on graphical way.

**REFERENCES**

# Interactive Computer Graphics a Top-Down Approach with OpenGL – Edward Angel, 5th edition, Addison-Wesley, 2008.

# Introduction to the Design and Analysis of Algorithm – AnanyLevetin, 3rd edition, 2007.

# Computer Graphics using OpenGL- F.S.Hill,Jr. 2nd edition, Pearson education, 2001.

# [www.aw.com/cssupport](http://www.aw.com/cssupport)

# [www.opengl.org](http://www.opengl.org)