**Chapter 1**

**INTRODUCTION**

The dominant characteristics of this new computer and communication technologies have become dominant forces in our life. Activities as wide ranging as filmmaking, publishing, banking and education continue to undergo revolutionary changes as these technologies alter the ways in which we conduct our daily activities. The combination of computers , networks, and the complex visual system, through the computer graphics has led to new way of displaying information, seeing virtual worlds and communicating with people and machines.

Computer graphics is concerned with all aspects of producing pictures or images using a computer. The field has begun humbly almost 50 years ago, with a display of few lines on a cathode ray tube, now we can create images by computer that are indistinguishable from photographs of real objects. We routinely train pilots with simulated airplanes, generating graphical displays of virtual environment of real time. Feature length movies made entirely by computer have been successful, both critically and financially. Massive multiplayer games can involve tens of thousands of concurrent participants.

**1.1 GLUT:**

It is complete API written by mark kilogram, which lets us create windows and handle the messages. It exists for several platforms that a program, which uses GLUT, can be compiled on many platforms without any changes in the code.

**ADDITONAL LIBRARIES:**

There are two libraries functions that we have learnt in OpenGL.

**1 .glu :**It is a set of utility functions, they are easy way of doing things that is tedious with raw OpenGL.

**2. glx :**It allows you to open up X window and link it up with OpenGL so that it will draw to that window.

**1.2 HOW OpenGL WORKS?**

To be hardware independent OpenGL provides its own data types, they all begin with “GL”. Eg: GLfloat, Glint. There are also many symbolic constants that all begin with ”GL”. Eg: GL\_POINTS, GL\_LINES. Finally commands have prefix ”gl”. Eg: glVertex3f (f, i, d).

There are utility library called as “GLU”. Here the prefixes are “GLU\_”, ”glu”. GLUT commands begin with “glut” it is same for every library. OpenGL is a widely accepted standard for developing graphics applications. Fortunately, OpenGL is easy to learn and it possesses most of the characteristics of other graphics systems. Graphics API’s such as OpenGL, developed as a way to provide application programmers with the access to hardware features that were being provided by the latest graphics hardware.

**Application of Computer Graphics:**

 The development of computer graphics has been driven both by needs of the user community and by advancement in technology. The applications of computer graphics of computer graphics are many and varied, we can divide them into four categories :

1. Display of information
2. Design
3. Simulation and Animation
4. User interface

**1. Display of Information:**

Medical imaging possess interesting and important data analysis problem. Modern imaging technologies such as computed tomography (CT), magnetic resonance imaging(MRI), ultrasound, and position emission tomography(PET), generate 3D data that must be subjected to algorithmic manipulation to provide useful information.

The field of scientific visualization provides graphical tools that help the researchers interpret the fast quantity of data that generate.

**2. Design:**

Professions such as engineering and architecture are concerned with design. Starting with a set of specifications, engineers and architects seek a cost effective and esthetical solution that satisfies the specifications.

Design is an interactive process. Design problem are either over determined such that they possess no solution that satisfies all criteria, much less an optimal solution, or undetermined, such that they have multiple solutions that satisfies the design criteria.

**3. Simulation and Animation:**

Graphics system evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators. One of the most important use has been in training of pilots. The use of special PLSI chips has led to a generation of arcade, games as sophisticated as flight simulators.

The simulators can be used for designing the robot, planning its path, and simulating its behaviour in complex environment. The success of flight simulators led to the use of computer graphics for animation in TV, motion pictures and advertising industries. Entire animated movies can now be made by computers at a cost less than that of movies made with traditional ways.

**4.User interface:**

Our interaction with computers has become dominated by visual paradigm that includes icons, menus and pointing devices such as mouse. From users perspectives, winding system such as X and window system, Microsoft windows.

**1.3 OPENGL:**

* Graphics rendering API.
* High Quality colour images composed of geometric primitives
* Operating system independent
* OpenGL is an application programmer’s interface (API) that allows programmers to write programs that access graphics hardware.

OpenGL provide a set of commands to render a two dimensional scene. That means you provide data in OpenGL usable form and OpenGL will show you this data in scene (render it).it is developed by many companies and it is free for use. You can develop OpenGL applications without licensing.

OpenGL based on the state variables, there are many values for example, the colour, you can specify a colour once and draw several polygons, lines and whatever you want. Then there are no classes like in directX. However it is logically structured. Before we come to the commands themselves, here are another thing.

To be hardware independent, OpenGL provides its own data types . They all begin with “GL”. For ex: GLfloat, GLint and so on. There are also many symbolic constants; they all begin with “GL” for ex: GL\_POINTS, GL\_LINES. Finally the commands have the prefix “gl”. There is utility library called GLU, here the prefixes are “GLU” & “glu”. It is same for every library. You want to know which library coexist with the one called before. There are libraries for every system. Windows as the wgl\*functions, UNIX system glx\* and so on.

It is very important thing to know, that there are two important matrices, which affect the transformation from 3D world to 2D screen. The projection matrix and the model view matrix. The projection matrix contains information, how a vertex can be mapped tom the screen. This contains, whether the projection shall be isometric or a form of perspective, how wide the field of view is and so on. Into the other matrix you put information, how the objects are moved, where the viewer is and so on.

**The OpenGL pipeline:**



**Fig1.3.1: the OpenGL rendering pipeline**

Commands may be either be accumulated in display lists are processed immediately through pipeline architecture. Display lists allow for greater optimization and command reuse, but not all commands can be put in display lists.

The first stage is evaluator. This stage effectively takes any polynomial evaluator commands and evaluates them into their corresponding vertex and attributes commands

The second stage is per-vertex operations, including transformations, lighting, primitive assembly, clipping, projection, and viewport mapping. The third stage is rasterization. This stage produces fragments, which are series of frame buffer address and values from the viewport mapped primitives as well as bitmaps and pixel rectangles.

The fourth stage is the per-fragment operations. Before fragments go to the frame buffer, they may subjected to a series of conditional tests and modifications, such as blending or Z-buffering.

**1.4 PROJECT INTRODUCTION**

 A windmill is a mill that converts the wind energy into rotational energy by means of vanes called as blades. Our project shows a simulation of a 3D windmill. glutMouseFunc is used for mouse function . Left mouse button is used to increase the speed of windmill. Right mouse button is used to decrease the speed. glutKeyboardFunc is used for keyboard functions. Key ‘x’ is used to view the top view of the windmill. Key ‘X’ is used to retain to front view. Key ‘y’ used to view side view. Key ‘Y’ is used to retain to the front view. Key ‘f’ is used to decrease the size of view. Key ‘n’ is used to increase the size of view. Key ‘r’ or ‘R’ is used to reset the view. When the wind mill rotates, the electricity is generated and 3 different color light glows.

**Chapter 2**

 **SYSTEM REQUIREMENTS**

**2.1 Software Requirements:**

* An MS\_DOS based operating system like windows XP
* A visual C/C++ compiler is required for compiling the source code to make a executable file which can be directly executed.
* The built in graphics and dynamic link libraries like glut and glut32, and header file like glut.h to create 2D &3D layout.

**2.2 Hardware Requirements:**

* **Processor :** intel 486/Pentium processor and above
* **Processor speed :** 500MHz or above
* **RAM :** 64 MB or above
* **Storage space :** 2 MB or above
* **Monitor Resolution :**colour monitor with minimum resolution of 640\*480

**Chapter 3**

**PROJECT DESIGN**

**3.1 PURPOSE**

Demonstration of simulation of a 3D windmill .

**3.2 DESCRIPTION**

 A windmill is a mill that converts the wind energy into rotational energy by means of vanes called as blades. Our project shows a simulation of a 3D windmill. glutMouseFunc is used for mouse function . Left mouse button is used to increase the speed of windmill. Right mouse button is used to decrease the speed. glutKeyboardFunc is used for keyboard functions. Key ‘x’ is used to view the top view of the windmill. Key ‘X’ is used to retain to front view. Key ‘y’ used to view side view. Key ‘Y’ is used to retain to the front view. Key ‘f’ is used to decrease the size of view. Key ‘n’ is used to increase the size of view. Key ‘r’ or ‘R’ is used to reset the view. When the wind mill rotates, the electricity is generated and 3 different color light glows.

**2.3 FLOW CHART**

MAIN

INITIALIZE CALLBACK FUNCTIONS

MAIN SCREEN DISPLAYED

MOUSE BUTTON

 KEYBOARD KEYS

LEFT BUTTON

RIGHT BUTTON

 X or x

 Y or y

 f

 n

 TOP VIEW

 SIDE VIEW

 DECREASE IN SIZE

 INCREASE IN SIZE

EVENTS

 **Fig2.3.1: Flowchart for Simulation of 3D Windmill**

**Chapter 4**

**IMPLEMENTATION**

 This program has been developed and implemented using OpenGL interface. In corporate this facility includes glut.h header files.

* **Void glBegin(glEnum mode)**

Initiates a new primitive of mode and starts the collection of vertices. Values of mode include GL\_POITS, GL\_LINES and GL\_POLYGON.

* **Void glEnd()**

Terminates a list of vertices.

**4.1 Attributes**

* **Void glColor3[b I f d ub us ui](TYPE r,TYPEg,TYPE b)**

Sets the present RGB colors. Valid types are byte(b),int (i), float (f), double (d), unsigned byte(ub), unsigned short(us) and unsigned int (ui).

1. **Void glClearColor(GLclampfg,Glclampfg,Glclampfb, Glclampfa)**

Sets the present RGBA clear color used when clearing the color buffer. Variable of Glclampf are floating point numbers between 0.0 and1.0.

1. **Void glPointSize(Glfloat size)**

Sets the point size attribute in pixels.

**4.2 Working with the window system**

* **Void glFlush()**

Forces any buffered openGL commands to execute.

* **Void glutInit(int argc,char \*\*argv)**

Initializes GLUT. The arguments from main are passed in and can be used by the application.

* **Void glutCreateWindow(char\*title)**

Creates a window on the display. The string title can be used by the window. The return value provides reference to the windows that can be used when there are multiple windows.

* **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 colormodel(GLUT\_RGB, GLUT\_INDEX) and buffering (GLUT\_SINGLE,GLUT\_DOUBLE).

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

Specifies the initial height and width of the window in pixels.

* **void glutInitWindowPosition(intx,int y)**

Specifies the initial position of top-left corner of the window in pixels.

* **void glViewport(intx,inty,Glsizei height)**

Specifies a width x height viewport in pixels whose lower left corner is at(x,y) Measured from origin of the window.

* **void glutMainLoop()**

Causes the program to enter an event processing loop. It should be the lat statement in main.

* **Void glutDisplayFunc(void(\*func)(void))**

Requests that the display callback be executed after the current callback returns.

* **void glutSwapBuffers()**

Swaps the front and back buffers.

**4.3 Interaction**

* **Void glutReshapeFunc(void \*f(intwidth,int height)**

Registers the reshape callback function f. The callbackfunc returns the height and width of the new window. The reshape callback invokes display callback.

* **Void glutKeyboardFunc(void \*f(char key,intwidth,int height)**

Register the keyboard callbackfunc f. The callback function returns the ASCII ode of the key pressed and the position of the mouse.

* **Void glutIdleFunc(void (\*f)(void))**

Registers the display callback function f that is executed whenever there are no other events to be handled.

* **Void glutMouseFunc(void \*f(intbutton,intstate,intx,int y))**

Registers the mouse callbackfunction f. The callback function returns the button (GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON, GLUT\_RIGHT\_BUTTON), the state of the button after the event (GLUT\_UP,GLUT\_DOWN). And the position of the mouse relative to top left corner of the window.

**4.4 Transformation**

* **void glMatrixmode(Glenum mode)**

Specifies which matrix will be affected by subsequent transactions. Mode can be GL\_MODEL\_VIEW,GL\_PROJECTION or GL\_TEXTURES.

* **Void glLoadIdentity()**

Sets the current transformation matrix to an identity matrix.

* **void glPushMatrix()**
* **void glPopMatrix()**

Pushes to and pops from the matrix stack corresponding tothe current matrix mode.

* **void glRotate**[**fd](TYPE x, TYPE y,TYPE z)**

Alters the current matrix by a rotation of the angle degree ahou the axis(dx,dy,dz). TYPE is either Glfloat or Gldouble.

* **void glTranslate[fd](TYPE x, TYPE y,TYPE z)**

Alters the current matrix by a displacement of(x,y,z). TYPE is of either Glfloat or Gldouble.

* **Void glScale[fd]](TYPE x, TYPE y,TYPE z)**

Alters the current matrix by scaling of(sx,sy,sz). TYPE is either Glfloat or Gldouble.

**4.5 Viewing**

* **Void glOrtho2D(Gldoubleleft,Gldoubleright,Gldoublebottom,Gldouble top)**

 Defines a two dimentional viewing rectangle in the planes z=0.

* **Void gluperspective(Gldoublefov, Gldouble aspect, Gldouble near, Gldouble far)**

Defines a perspective viewing volume using the y direction field of view fov measured in degrees, the aspect ratio of the front clipping plane, and the near and far distances.

**4.6 Lighting**

* **Void glLight[if](Glenumface,Glenumparam,TYPE value)**

Sets scalar and vector parameter param for light source light.

* **Void Materialfv(Glenumface,Glenumparam,TYPE \*value)**

Sets parameter for face (GL\_FRONT,GL\_BACK,GL\_FRONT\_AND\_BACK)

**4.7 SOURCE CODE**

**Chapter 5**

**SNAPSHOTS**

**Chapter 6**

**TESTING**

# **6.1 TESTING**

Software testing is a critical element of the ultimate review of specification design and coding. Testing of software leads to the uncovering of errors in the software functional and performance requirements are met. Testing also provides a good indication of software reliability and software quality as a whole. The result of different phases of testing are evaluated and then compared with the expected results. If the errors are uncovered they are debugged and corrected. A strategy approach to software testing has the generic characteristics:

* Different testing techniques are appropriate at different points of time.
* Testing and debugging are different activities, but debugging must be accommodated in the testing strategy.

Following three approaches of debugging were used:

* Debugging by Induction
* Debugging by Deduction
* Backtracking

**6.2 TEST PLANS**

In this test plan all major activities are described below:

* Unit testing.
* Integration testing.
* Validation testing.
* System testing.

.

**Table 6.1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Slno.** | **Test case Description** | **Expected result** | **Actual Result** | **Remarks** |
| 1. |  |  |  |  |
| 2. |  |  |  |  |

 **Table 5.3: Test case for keyboard**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.no** | **Test case description** | **Expected result** | **Actual result** | **Remarks** |
|  |  |  |  |  |
| 2. |  |  |  |  |
| 3. |  |  |  |  |
| 4. |  |  |  |  |

 **CONCLUSIONS**

An attempt has been made to develop a openGL which meets necessary requirements of the user successfully. Since it is user friendly it enables user to interact efficiently and easily.

The development of mini project has given us a good exposure to openGL by which we have learnt some of the techniques which helps in the development of animation and gaming.

Hence it is helpful for us to even take this field as our career too and develop some other features in openGL and provide as token of contribution to the graphics world.

**FUTURE ENHANCEMENT**

This project has been designed using C++, which works on the windows platform. The project can be designed using other languages and better graphical interfaces. The following features could have been incorporated.

**REFERENCES**

**BIBLIOGRAPHY**

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

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

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

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

Edition.

**WEBSITES**

[5] www.OpenGLRedbook.com

[6] www.OpenGLsimpleexamples.com

[