Chapter 1

INTRODUCTION

Computer graphics generate the sophisticated images in 3D or 2D form in real world. It involves user interface design, sprite graphics, vector graphics, 3D modeling, and GPU design and computer vision. It uses the methodology of geometry, optics and physics. It is widely used in animation, movies, advertising, video games, and graphics design.

Graphics are virtual presentation on the surface such as computer. It is a combination of text and illustration. It may consist of the deliberate selection, creation or arrangement of typography alone as in a brochure, filter, poster, website or book without any other element.

1.1 OpenGL

Open Graphics Library (OpenGL) is a cross language, cross platform application programming interface (API), for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU) to achieve hardwareaccelerated rendering.

Silicon Graphics Inc., (SGI) started developing OpenGL in 1991 and released it in January 1992, application use it extensively in the fields of Computer Aided Deisgn (CAD), virtual reality, Scientific visualization, information visualization, flight simulation and video games. It is managed by non-profit technology consortium Khronos Group.

OpenGL has mainly two applications:

- To hide the differing capabilities of hardware platforms by requiring that all implementation support the full OpenGL feature set.
- To hide the complexity of interfacing with different 3D accelerators by presenting programmers with a single uniform API.

The basic primitives that are facilitated by OpenGL are rasterized, points, lines and polygon.

1.2 OpenGL Library

OpenGL consists of three library function through which the applications can access the OpenGL directly. They are GL (OpenGL), GLU (OpenGL Utility) and GLUT (OpenGL Utility Tollkit). GL library functions will begin with the latter gl and is stored in the library and is usually referred as GL.

GLU

GL
OpenGL
Application
Program

Frame
Buffer

GLUT

Xlib, Xtk

GLX

Fig 1.1: OpenGL Library

GLU is a computer graphics library for OpenGL. It consists of a number of functions that use the base OpenGL library to provide higher level drawing routine from the more primitives routine that OpenGL provides. It is usually distributed with base OpenGL package. GLU is not implemented in the embedded version of OpenGL package. All the GLU functions begin with the letter glu.

GLU functions are used in mapping screen and world coordinates, generation of texture mipmaps, drawing of quadric surfaces, interpretation of OpenGL error codes, an extended range of transformation routines for setting up viewing volumes and simple positioning of the camera, generally in more human friendly terms than the routines presented by OpenGL. It also provides additional primitives for use in OpenGL applications including sphere, cylinder and disks.

GLUT is a window system independent toolkit for writing OpenGL programs. It implements a simple windowing API for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT is designed for constructing small and medium sized OpenGL programs but not for full features toolkit so large sophisticated user interface are better off using native window system toolkit. GLUT is simple, easy and small. It has C, C++, FORTRAN and ADA programming bindings. Current Version of GLUT API is 3. Current source code distribution is GLUT 3.1.

The toolkit supports:

- Multiple windows for OpenGL library
- Callback driven event processing
- Sophisticated input devices
- An idle routine and timers
- Support for bitmap and stroke fonts
- A simple, cascading pop-up menu facility.

1.3 OpenGL Architecture

The OpenGL architecture is structured as a state-based pipeline. Below is a simplified diagram of this pipeline. Commands enter the pipeline from left.

Fig 1.2: OpenGL architecture

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

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

The second stage is the 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 addresses 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 be subjected to a series of conditional tests and modifications, such as blending or z-buffering.

Parts of the frame buffer may be fed back into the pipeline as pixel rectangles.

Texture memory may be used in the rasterization process when texture mapping is enabled.

1.4 Red/Blue Stereo

It is a red/blue filter stereo glass. It shows the effects that can be seen via glasses.

Even the people with some eye defect can be able to view some objects in red/blue via their left or right eye. The objects can be viewed with switching of either stereo or mono views. There is a right button attached to it to perform the action with menu come out as right mouse button clicked. The solar system consists of only earth and sun. More planets can be added to it.

The objects that are used to show effects are dodecahedron, icosahedrons, teapot and solar system. The objects are switched from one to another using glTranslate function and can rotate about the axis using glRotate function. The objects are implemented using glSolid function. The functions glMaterial and glLight used to show effects.

Chapter 2

LITERATURE SURVEY

Stereo 3D is also known as Anaglyph 3D which means of encoding each eye's image using filters of different colors. This images contains two colored images one for each eye. When it is viewed through color coded anaglyph/stereo glasses each of the two images reaches the eye it's intended for revealing an integrated stereoscopic image. The visual cortex of the brain fuses this into perception of 3D scene or composition. Stereo images have seen a recent resurgence due to presentation of images and video on the web, Blu-ray Discs, CD's and even in print. Video games, theatrical films and DVD's can be shown in stereo 3D process. Practical images for science or design where depth perception is useful, include presentation of full scale and microscopic stereographic images.

The first method to produce stereo images was developed in 1852 by Wilhelm Rollmann. He first illustrated the principle of stereo using red and blue lines on the black field in 1853 with red and blue glasses to perceive the effect but this was for line drawing only. Later in 1858 Joseph D"Almeida began projecting 3D magic lantern slideshow using red and green filter with red and green googles.

Popular professional programs such as Adobe Photoshop provide the basic digital tools for processing of anaglyphs. They do not provide instruction for anaglyph in their basic documentation. Various websites offer free instruction related to 3D for photoshop. Simple, low cost programs, dedicated to anaglyph creation, are also available. In recent simple practice, the left eye image is filtered to remove blue & green. The right eye image is filtered to remove red. The two images are usually positioned in the compositing phase in close overlay registration. Blu-ray Disc anaglyph techniques have more recently been supplanted by the Blu-ray 3D format, which uses Multiview Video Coding(MVC) to encode full stereoscopic images.

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Hardware Requirement

RAM

- Input device: Keyboard, Mouse

- Output device: Monitor

Processor: Pentium Processor
: 512MB

3.2 Software Requirement

- Package : OpenGL
- Programming Language: C
- Operating System: Ubuntu , Windows

Chapter 4

DETAIL DESIGN

The design of the project basically includes the standard OpenGL functions which are necessary. The concepts of the particles are thoroughly executed so that a layman also could get a detail understanding of the project. The basic concept of computer graphics such as colors, menus, display functions, reshape functions, mouse functions, keyboard functions and much more are implemented to design the source code. The use of complex concepts is avoided. Thus the source code is easy to understand and also easy to implement.

4.1 Algorithm

Algorithm is a step by step procedure to solve a given problem.

4.1.1 Algorithm for main () function

```
Step 1: [Initialization]
glutInit(argc, argv)
glutInitDisplayMode()
glutInitWindowSize()
glutInitWindowPosition()
init()
step 2: [Create Window]
glutCreateWindow()
step 3: [Register following Functions]
glutVisibilityFunc()
```

```
glutDisplayFunc()
glutReshapeFunc()
glutKeyboardFunc()
glutCreateMenu()
glutAddMenuEntry()
glutAttachMenu()
step 4: [Terminate]
End of main() algorithm.
```

4.1.2 Algorithm for Display()

```
Step 1: [Clear the buffer]
glClear()
Step 2: [Color components in frame buffer]
glColorMask()
Step 3: [Translate the identity matrix]
glTranslatef()
Step 4: [Draw dodecahedron, icosahedrons, teapot and solar system]
Switch (solid)
Case 1:
//DRAW DODECAHEDRON
glTranslated()
```

```
glRotated()
glPushMatrix()
glscalef()
glutSolidDodecahedron()
glPopMatrix()
break()
Case 2:
//DRAW ICOSAHEDRON
glTranslated()
glRotated()
glutSolidIcosahedron()
break()
Case 3:
//DRAW TEAPOT
glRotated()
glTranslated()
glRotated()
glSolidTeapot()
break()
```

```
Case 4:
//DRAW SOLAR SYSTEM
glutSolidSphere()

//Draw Sun

glRotated()
glPushMatrix()
glRotated()
glutSolidTorus();
glPopMatrix()
glRotated()
glTranslatef()
glutSolidSphere()

//Draw Earth

break()
```

4.2 Data Flow Diagram

Dataflow is a term used in computing.Dataflow is a software paradigm based on the idea of disconnecting computational actors into stages that can execute concurrently. Dataflow can also be called stream processing or reactive programming. These ideas are all highly inter-related. There have been multiple data-flow/stream processing languages of various forms. Data-flow hardware is an alternative to the classic Von Neumann architecture. The most obvious example of data-flow programming is the subset known as reactive programming with spreadsheets.

Main()

VisibilityFunc

ReshapeFunc()

MenuFunc()

()

DisplayFunc()

IdleFunc()

OUTPUT
Screen

KeyboardFunc()

INPUT

INPUT

Keyboard

MOUSE

Fig 4.1: Dataflow diagram of Red/Blue Stereo

Chapter 5

IMPLEMENTATION

Implementation is the realization of an application or execution of the plan, idea, model, design, specification, standard, and algorithm.

5.1 Library functions used in OpenGL

Library functions are the functions which are stored in OpenGL library. These are functions which are accessed at the run time.

5.1.1 glutPostRedisplay()

It marks the current window as needing to be redisplayed. The next iteration through glutMainLoop, the window's display callback will be called to redisplay the window's normal plane.

5.1.2 glutSolidDodecahedron()

It display a solid 12 sided regular solid centered at the modeling coordinates origin with a radius of sqrt (3).

5.1.3 glutSolidIcosahedron()

It display a 20 sided regular solid centered at the modeling coordinates origin with the radius of 1.0.

5.1.4 glutSolidTeapot()

It displays a solid teapot. Both surface normal and texture coordinates for the teapot are generated with OpenGL evaluators.

5.1.5 glBeign() and glEnd()

These functions are used to delimit the vertices of the primitive or a group of like primitives.

5.1.6 glutSolidSphere()

It consists of 3 parameters i.e., radius, slices(lines of longitude), stacks(lines of latitude). It display a sphere centered at the modeling coordinates origin of the specified radius. The sphere is subdivided around the Z-axis into slices and along the Z-axis into stacks.

5.1.7 glutSwapBuffers()

It swaps the buffers of the current window if double buffered. Specially it promotes the contents of the back buffer of the layer in use of the current window to become the contents of the front buffer.

5.1.8 glClear()

This function is used to clear the buffer to preset values. It accepts parameters such as GL_COLOR_BUFFER_BIT (buffer currently enabled for color writing) and GL_DEPTH_BUFFER_BIT (depth buffer) which are separated by bitwise OR operator.

5.1.9 glColorMask()

This function is used to enable or disable writing of frame buffer color components.

5.1.10 glEnable() and glDisable()

These functions are used to enable or disable OpenGL capabilities.

5.1.11 glFlush()

This function is used to forces execution of OpenGL function in finite time.

5.1.12 glFrontFace()

This function defines front facing and back facing polygons. GL_CW and GL_CCW are accepted as parameters. The default value of this function is GL_CCW.

5.1.13 glFrustum()

This function multiplies the current matrix by a perspective matrix. It accepts six parameters such as left, right, top, bottom, zNear(distance of the near-depth clipping plane and must be positive) and zFar(distance of the far-depth clipping plane and must be positive).

5.1.14 glLoadIdentity()

This function replaces the current matrix with the identity matrix.

5.1.15 glLightfv()

This function returns light source parameter values. It accepts the parameters such as follows:

➢GL_AMBIENT: contains 4 floating points that specify the ambient RGBA intensity of the light which are mapped directly.

➢GL_DIFFUSE: contains 4 floating points that specify the diffuse RGBA intensity of the light which are mapped directly.

➢GL_SPECULAR: contains 4 floating points that specify the specular RGBA intensity of the light which are mapped directly.

5.1.16 glMaterialfv()

This function specifies material parameters for lighting model. It accepts the parameters such as

➢GL_AMBIENT: It contains 4 floating point values that specify the ambient RGBA reflectance of the material. Integer values are mapped linearly but floating point values are mapped directly.

➢GL_DIFFUSE: It contains 4 floating point values that specify the diffuse RGBA reflectance of the material. Integer values are mapped linearly but floating point values are mapped directly.

➢GL_SPECULAR: It contains 4 floating point values that specify the specular RGBA reflectance of the material. Integer values are mapped linearly but floating point values are mapped directly.

5.1.17 glMatrixMode()

This function specifies which matrix is the current matrix. It accepts parameters such as GL_MODEL_VIEW (applies subsequent matrix operation to the model view matrix stack) and GL_PROJECTION (applies subsequent matrix operation to the projection matrix stack).

5.1.18 glViewport()

This function sets viewport. It accepts the parameters such as x, y, width, height.

5.1.19 glOrtho()

This function multiplies the current matrix by an orthographic matrix. It accepts 6 parameters such as left, right, top, bottom, near, far.

5.1.20 glPushMatrix() and glPopMatrix()

These functions are used to push and pop the current matrix stack.

5.1.21 glRotated()

This function multiplies the current matrix by a rotation matrix. It accepts parameters such as angle, x, y, z.

5.1.22 glTranslated() and glTranslatef()

These functions multiply the current matrix by a translation matrix. It accepts parameters such as x, y, z.

5.1.23 glScaled() and glScalef()

These functions multiply the current matrix by a general scaling matrix. It accepts parameters such as x, y, z.

5.1.24 glutInit()

It is used to initialize the GLUT library. It passes two command line arguments “argc and argv”.

5.1.25 glutInitDisplayMode()

It sets the initial display mode which is used when creating top level windows, sub-windows, and overlaps to determine the OpenGL display mode for to be created window or overlap. The display mode involves (GLUT_RGB, GLUT_DOUBLE).

5.1.26 glutCreateWindow()

It creates a top level window. The name of the window is send as a parameter through this function.

5.2 Implementation of user defined functions

These are the functions which are used by the user to call their functions during run time.

5.2.1 glutVisibilityFunc()

It sets the visibility callback function. GLUT considers a window as visible if pixels of the window are visible or pixels of any descendent window are visible on the screen.

5.2.2 glutDispalyFunc()

It sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed the display callback for the window is called.

5.2.3 glutReshapeFunc()

It sets the reshape callback for the current window. It is triggered when a window is reshaped.

5.2.4 glutKeyboardFunc()

It sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback.

5.2.5 glutCreateMenu()

It 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.

5.2.6 glutAddMenuEntry()

It adds a menu entry to the bottom of the current window.

5.2.7 glutAttachMenu()

It attaches a mouse button to the current window to the identifier. By attaching a menu identifier to a button, the named menu will be poped up when the user presses the specified button. Here we are using right button (GLUT_RIGHT_BUTTON).

Chapter 6

TESTING AND SNAPSHOTS

6.1 Testing process

The most widely used testing process consists of different stages. The testing process of this assignment is based on four stages. They are:

6.1.1 Unit Testing

Unit testing finds problems early in the development cycle. This includes both bugs in the programmer's implementation and missing parts of the specification for the unit. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. It provides a strict, written contract that the piece of code must satisfy which results in affords several benefits. The cost of finding a bug before coding begins or when the code is first written is considerably lower than the cost of detecting, identifying, and correcting the bug later; bugs may also cause problems for the end-users of the software. Some argue that code that is impossible or difficult to test is poorly written, thus unit testing can force developers to structure functions and objects in better ways.

6.1.2 Integration testing

Integration testing is the phase in software testing in which the individual software modules are combined and tested as a group. It occurs after unit testing. It takes as its input modules that have been unit tested, groups them, applies tests defined in an integration test plan and delivers as its output the integrated system ready for system testing. The goal of integration testing is to verify functionality, performance, reliability requirement placed on major design items.

In this application the transformation such as rotation, scaling, translate and other functions are combined. The functions such as viewport, model viewport, pushing and poping of the matrix etc., are combined and tested for functionality and performance.

6.1.3 System Testing

The combination of different integrated components the system form the entire system. The testing is used to validate the functionality of the system and to find errors in the system. It also used to define the functional requirement and non-functional requirement. It is used to check the interaction between the different components in the system. The initial component of the system is to create objects to perform stereo effects. The next component is to define the effects which are needed to show the effects and so on. All these components are combined to form a system to perform the stereo effects.

a) Black-Box Testing

Black-box testing is a method of testing that examines the functionality of an application without peering into its internal structure. This method of the test can be applied to virtually every level of testing. It typically comprises most if not all higher level testing but can also dominate unit testing as well.

b) White-Box Testing

White-Box testing is a method of testing that tests internal structures as opposed to its functionality. In white-box testing an internal perspective of the system as well as programming skills is used to design test cases. The tester chooses input to exercise paths through the code and determine the appropriate outputs. White-box testing can be applied to all levels of testing.

6.2 Snapshots

Fig 6.1: Dodecahedron with stereo effects

Initially the output of the stereo starts from the Dodecahedron

Fig 6.2:Icosahedron with stereo effects

Next object with stereo effect is icosahedron which is seen with red color with the effect of blue color.

Fig 6.3: Teapot with stereo effect

The next image with the stereo effect is teapot. We can see the effect of the stereo in blue color along with red.

Fig 6.4: Solar system (sun and the earth) with stereo effect

The next object is solar system which consists of sun and earth rotating the sun with the effect of red/blue.

Fig 6.5: Dodecahedron with mono effects

The next output is dodecahedron with mono effects i.e., it is visible only in red color.

Fig 6.6: Icosahedron with mono effect

The next object is icosahedrons with mono effect.

Fig 6.7: Teapot with mono effect

The next object is teapot with mono effects.

Fig 6.8: Solar system with mono effects

The next object is solar system with mono effects.

Chapter 7

CONCLUSION

Stereo-anaglyph (red/blue) digital image models can be produced quickly and inexpensively as a complement to existing maps and imagery. Producing threedimensional anaglyphs from aerial photography and oblique ground photography are with their individual challenges. The challenge therefore resides in the eyes of the artist and his/her ability to identify possible obstacles to rendering quality anaglyphic imagery. The relatively inexpensive cost of producing and projecting these images is dwarfed by the excitement students derive from interacting together in a group interpretation of topography.

Chapter 8

FUTURE SCOPE

Today the most prominent use of stereoscopic imaging in education is inside the world of Virtual Reality. The use of stereoscopic imagery in Virtual Reality is easy to grasp. Virtual Reality has two parts: creation of stereoscopic imagery and its display. Virtual Reality requires fast computer to create stereoscopic imagery, to generate the information for each view and also to data to create these views. The simplest way to view the stereoscopic images in Virtual Reality environment is to use Head Mounted Display with a separate screen for each eye. Video can be passively projected using polarized process or actively viewed using LCD shutter glasses. The LCD shutter glasses alternately open and close each eye in synchronous with the display so that each eye sees a different signals.

REFERENCES

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL, Edward Angel, 5th Edition, Pearson Education, 1008.
[2] Computer Graphics-OpenGL Version, Donald Hearn and Pauline Baker, 3rd edition, Pearson Education.
[3] www.openglproject.in
[4] www.opengl.org
[5] Hydro.csumb.edu/essp%20360/images/geomorph3d/3d101.htm