## Chapter 1

## INTRODUCTION

**1.1: Problem Statement**

This project represents a prospect at sea or of the sea, or a picture representing a scene at sea; a marine view. This is a view of the sea shore with boats moving and day turning into night. To anyone having a cursory look these projects look very primitive and unprofessional. While animating 3D objects is a complicated task doing some image editing and displaying them may do the trick in getting a professional look. This project is completely designed using various OpenGL functions and also the concepts of Computer Graphics. In this project some polygons and shading them has worked its magic and we have a beautiful sunrise scene. In this project various functions are used draw the various objects such as ship, boat, etc, and functions for user interface. This c++ graphics project, has been aimed of showing a simple sea view. This project also includes movement of boats from one point to another during both day view and night view and it also includes the movement of moon during the night view.

**1.2: Scope**

This project is entirely developed using the OpenGL and Computer Graphics functions. 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. This project is developed to create a sea view using various objects and functions. This project uses some built-in functions which are present in the OpenGL library and some functions are user-defined which are written for performing various tasks. Finally, after building the entire project it has to be compiled and run using Visual Studio.

**1.3: Project Description**

In this project various functions are used draw the various objects such as ship, boat, etc, and functions for user interface. Functions used to draw the respective objects with their syntax is shown below:

* void day();
* void night();
* void drawsea();
* void mountain();
* void drawtree();
* void drawship();
* void redrawboat();
* void drawsmallboat();
* void circle\_draw(GLint h,GLint k,GLint r);
* void plotpixels(GLint h,GLint k,GLint x,GLint y);

This project is designed to show the use of 3D shapes and transformation. It is a graphics package encoded in Dev C++ with OpenGL and the project basically deals with providing the graphical interface between user and the system. The project is about moving an Object using keyboard.

The above project can be analyzed as follows:

* This project is implemented using ‘C++’ and ‘OpenGL’ built in functions.
* When the user presses the ‘X’ key the boats move in the +ve x-direction.
* When the user presses the ‘Y’ key the boats move in the -ve x-direction.
* When the user presses the ‘S’ key to stop the boats.
* When the user presses the ‘R’ key to restart the boats.
* When the user presses the ‘Z’ key the moon moves in the +ve y-direction.
* When the user presses the ‘W’ key the moon moves in the -ve y-direction

## Chapter 2

## HARDWARE AND SOFTWARE REQUIREMENTS

* 1. **Hardware Requirements**
* Processor : Intel 386 onwards Compatible Hardware.
* RAM : 128Mb RAM
* Hard Disk : 40 GB
* Monitor : 1024\*786 display resolution
* Keyboard : Standard 101 Key board
  1. **Software Requirements**
* Operating System : UBNTU 10.10, WINDOWS…
* Language Tool : C/C++ using OpenGL
* Compiler : gcc version 4.5.1
* Libraries : Supporting OpenGL & 32-bit color Resolution
* Documentation : MS-Word.

## Chapter 3

## DESIGN

This project is developed using Dev C++ with OpenGL as an academic project using keyboard and mouse. This project is implemented by making use of extensive use of library functions offered by graphic package of ‘OPENGL’. A number of user defined functions also have been used and a summary of those functions follows this list of standard library functions.

**3.1: List of Standard library functions**

* **glBegin (GLenum mode);**

It defines the type of primitives that the vertices define. Each subsequent execution of glVertex3f pecifies the x,y,z coordinates of a location in space.

* **glEnd (void);**

It ends the list of vertices.

* **glClear (GLbitfield mask);**

It is used to make the screen solid and white.

* **glClearColor (GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);**

The background color is set with this API where the last argument specifies a degree of transparency.

* **glEnable (GLenum cap);**

We enable different algorithms by this function call.

* **glutInit( );**

This function defines the interaction between the windowing system and OpenGL.

**Declaration:**

glutInit(&argc,argv);

* **glutInitDisplayMode( );**

Here we specify RGB color system and also double buffering.

**Declaration:**

glutInitDisplayMode(GLUT\_RGB|GLUT\_DOUBLE);

* **glutInitWindowSize( );**

This function specifies a window in the top left corner of the display.

**Declaration:**

glutInitWindowSize(900,700);

* **glutCreateWindow( );**

This creates an OpenGL window using the glut function where the title at the top of the window is given by the string inside the parameter of the above function.

**Declaration:**

glutCreateWindow(“Use of 3D shapes and Transformation”);

* **glutDisplayFunc( );**

Graphics are sent to screen through a function called the display call back, here the function named ‘func’ will be called whenever the windowing system determines that OpenGL window needs to be redisplayed.

**Declaration:**

glutDisplayFunc(display);

* **glutKeyboardFunc( );**

This function handles keyboard events.

**Declaration:**

glutKeyboardFunc(myKey);

* **glColor3f( );**

In RGB color we have three attributes to set. The first is clear color which is set to black and we can select the rendering color for points by setting the state of variable to black by using the following function call.

**Declaration:**

glClearColor(0.0,0.0,0.0,0.0);

glColor(0.0,0.0,0.0);

* **Void myKey( );**

This is the keyboard callback function. Within the callback function function, we define the actions that we want to take place if the specified event occurs.

**Declaration:**

Void myKey(unsigned char key,int x,int y) {

Case ‘Q’: exit(0);

Break}

* **Void display(void)**

This function is the callback function for ‘DispalyFunc’. This function is mandatory because GLUT requires that all programs have a display callback.

* **Void myReshape(int w,int h)**

This function is used whenever the window is resized. In our programs we need the required display callback for drawing because the primitives are generated when a mouse event occurs.

## Chapter 4

## IMPLEMENTATION

**4.1: Code Snippets**

**4.1.1: void day( )**

In this function **mountain( ),drawsea( ),drawtree( ),drawboat( ),drawship( )** functions are called to create a day view.

void day()

{

//to set the sky color and ground color

glColor3f(.8, .37, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(2200, 1800);

glColor3f(.7, .37, 0.0);

glVertex2f(0, 1800);

glColor3f(.4, .6, 0.0);

glVertex2f(0, 0);

glColor3f(.4, .6, 0.0);

glVertex2f(2200, 0);

glEnd();

// to draw the mountains

mountain();

// to draw the sea

drawsea();

// to draw the sun

glPointSize(9.0);

glColor3f(.9, 0.2, 0.0);

circle\_draw(1400, 1600, 100);

glColor3f(.95, 0.2, 0.0);

circle\_draw(1400, 1600, 90);

glColor3f(1.0, 0.25, 0.0);

circle\_draw(1400, 1600, 70);

glColor3f(1.1, 0.3, 0.0);

circle\_draw(1400, 1600, 50);

glColor3f(1.15, 0.35, 0.0);

circle\_draw(1400, 1600, 30);

glColor3f(1.18, .38, 0.0);

circle\_draw(1400, 1600, 10);

drawtree();

//to draw street light for DAY

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1400, 0);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1425, 0);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1425, 250);

glVertex2f(1400, 250);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1410, 280, 30);

circle\_draw(1410, 280, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1600, 200);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1625, 200);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1625, 350);

glVertex2f(1600, 350);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1610, 380, 30);

circle\_draw(1610, 380, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1800, 380);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1825, 380);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1825, 525);

glVertex2f(1800, 525);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1810, 555, 30);

circle\_draw(1810, 555, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(2000, 500);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(2025, 500);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(2025, 650);

glVertex2f(2000, 650);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(2010, 680, 30);

circle\_draw(2010, 680, 10);

drawsmallboat();

drawboat();

//to draw road DAY

glPointSize(8.0);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(2200, 700);

glColor3f(0.05, 0.05, 0.0);

glVertex2f(2200, 400);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1700, 0);

glColor3f(0.05, 0.07, 0.0);

glVertex2f(1400, 0);

glEnd();

glColor3f(1.0, 1.0, 1.0);

glBegin(GL\_LINES);

glVertex2f(2200, 550);

glVertex2f(1600, 0);

glVertex2f(2200, 560);

glVertex2f(1590, 0);

glEnd();

}

**4.1.2: void night( )**

In this function **mountain2( ),drawsea2( ),drawtree2( ),drawboat( ), drawship( )** functions are called to create a night view.

void night()

{

// to set the background color to black

glColor3f(0.0, 0.0, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(2200, 1800);

glVertex2f(0, 1800);

glVertex2f(0, 0);

glVertex2f(2200, 0);

glEnd();

//to draw the moon

glPointSize(9.0);

glColor3f(1.0, 1.0, 1.0);

circle\_draw(1400, 1600 + s, 100);

circle\_draw(1400, 1600 + s, 90);

circle\_draw(1400, 1600 + s, 70);

circle\_draw(1400, 1600 + s, 50);

circle\_draw(1400, 1600 + s, 30);

circle\_draw(1400, 1600 + s, 10);

glPointSize(5.0);

//to draw the stars

glBegin(GL\_POINTS);

glVertex2f(900, 1700);

glVertex2f(800, 1700);

glVertex2f(700, 1100);

glVertex2f(200, 1500);

glVertex2f(1800, 1700);

glVertex2f(1700, 1200);

glVertex2f(2000, 1500);

glEnd();

//glColor3f(.25,.25,.25);

mountain2();

glColor3f(.7, .7, .7);

drawsea2();

//to draw street light for NIGHT

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1400, 0);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1425, 0);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1425, 250);

glVertex2f(1400, 250);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1410, 280, 30);

circle\_draw(1410, 280, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1600, 200);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1625, 200);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1625, 350);

glVertex2f(1600, 350);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1610, 380, 30);

circle\_draw(1610, 380, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(1800, 380);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1825, 380);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(1825, 525);

glVertex2f(1800, 525);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(1810, 555, 30);

circle\_draw(1810, 555, 10);

glColor3f(0.10, 0.10, 0.0);

glBegin(GL\_POLYGON);

glVertex2f(2000, 500);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(2025, 500);

glColor3f(0.10, 0.10, 0.0);

glVertex2f(2025, 650);

glVertex2f(2000, 650);

glEnd();

glColor3f(1.0, 01.0, 01.0);

circle\_draw(2010, 680, 30);

circle\_draw(2010, 680, 10);

//to draw tree and boat

drawtree2();

drawsmallboat();

drawboat();

}

**4.1.3: void display( )**

We first clear the color and buffer and **night( )** and **day( )** are called based on the ‘*glob’* value.

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

if (glob == 1)

night();

if (glob == 0)

day();

redrawboat();

redrawboat1();

glFlush();

glutSwapBuffers();

}

**4.1.4: int main( )**

In this we will set RGB colour mode, set depth buffer for hidden surface removal and double buffering. The window size is set to specific value.

int main(int argc, char\*\* argv)

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(2200, 1800);

glutInitWindowPosition(0, 0);

glutCreateWindow("SEA VIEW");

glutKeyboardFunc(key);

glutDisplayFunc(display);

glutCreateMenu(mainmenu);

glutAddMenuEntry("QUIT", 1);

glutAddMenuEntry("DAY VIEW", 2);

glutAddMenuEntry("NIGHT VIEW", 3);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

init();

glutMainLoop();

return 0;

}

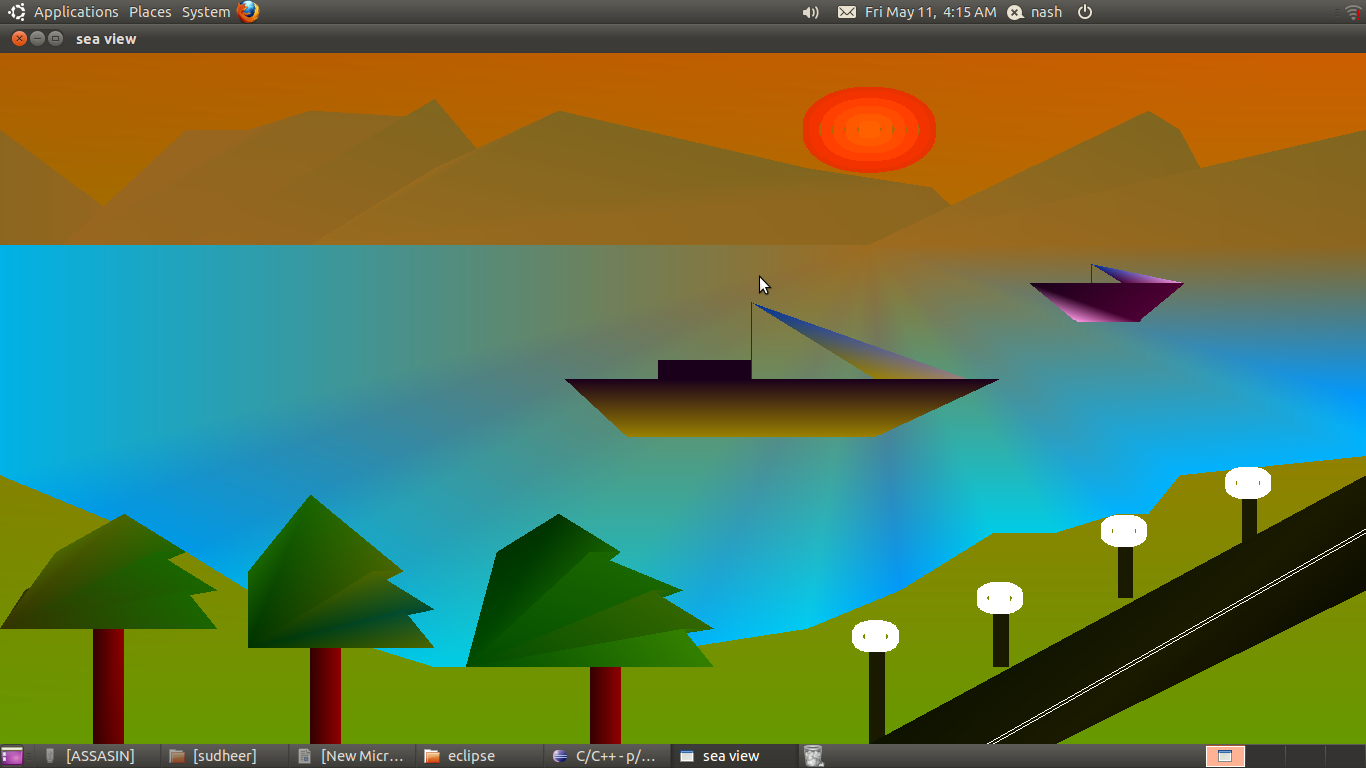
**Chapter 5**

**RESULT AND DISCUSSION**

**5.1: Screen Shots**

**5.1: Day View:**

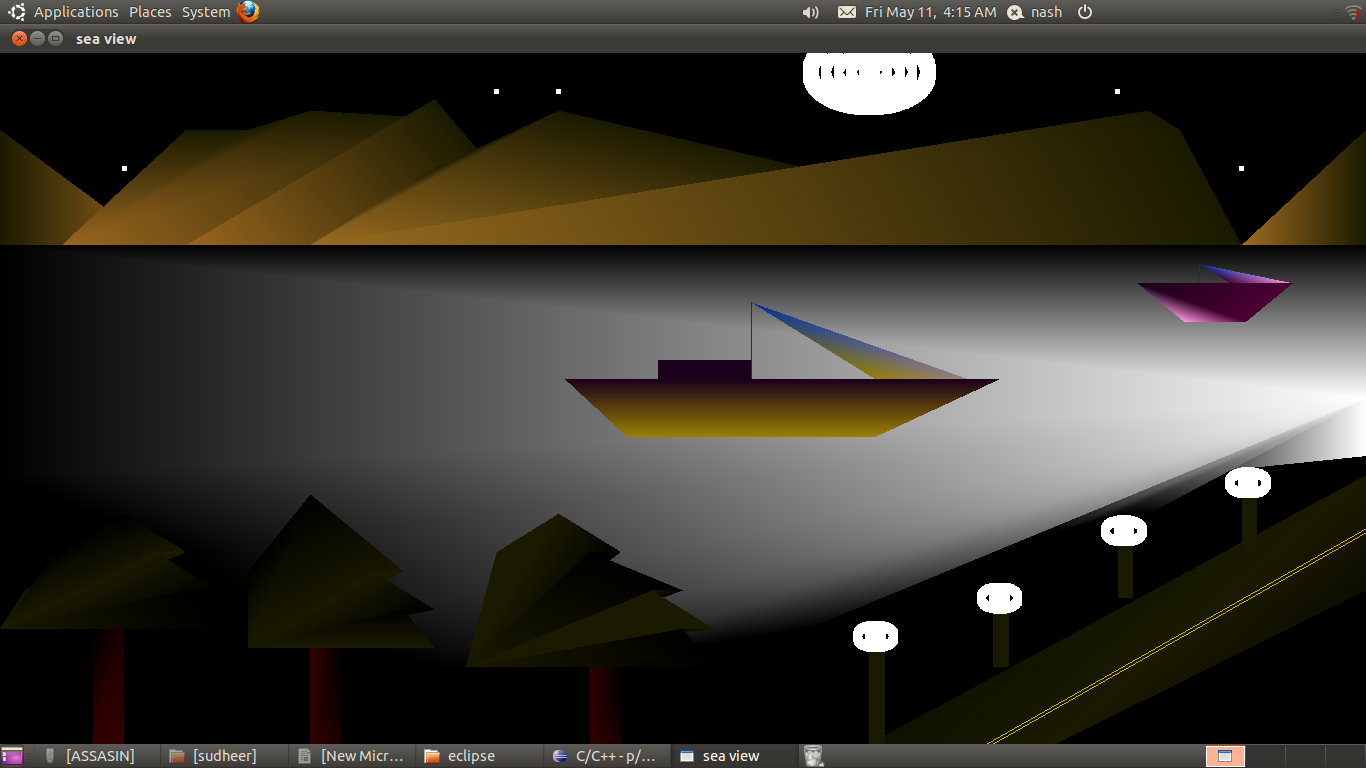
Fig 5.1 represents an instance of the sea view project. It is an image of sea view during day.

****

**Fig 5.1: Day View**

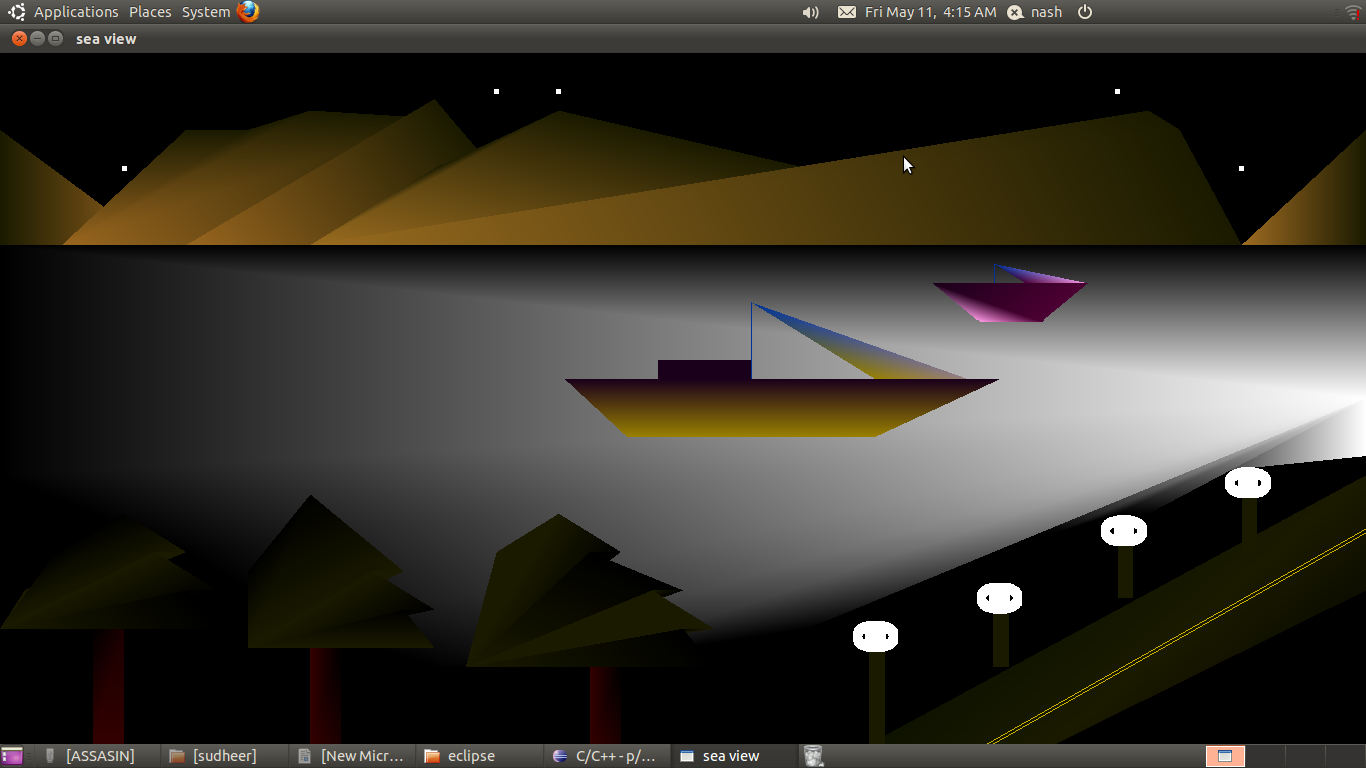
**5.2: Night View with Moon:**

Fig 5.2 represents an instance of the sea view project. It is an image of sea view during night when the moon is visible in the night sky.

**Fig 5.2: Night View with Moon**

**5.3: Night View without Moon:**

Fig 5.3 represents an instance of the sea view project. It is also an image of sea view during night when the moon is invisible in the night sky.



**Fig 5.3: Night View without Moon**

## Chapter 6

## CONCLUSION AND FUTURE ENHANCEMENTS

An attempt has been made to develop an OpenGL graphics package which meets all the necessary requirements that were set out. It is user friendly and provides an easy interaction for the user. The user can very easily use this tool to draw or manipulate a drawing. The interfaces and mouse driven and the user can select a function by clicking on an option representing that function. I finally conclude that this graphics package satisfies all requirements and provides good entertainment.

**REFERENCES**

**Reference Books**

1. **Interactive computer graphics a top-down approach with OpenGL-** Edward Angel, 5th Edition, Addition -Wesley 2009
2. **Computer graphics using Open GL–**F S Hill, Jr.2nd edition, Pearson education2001
3. **OpenGL shading language -**Randi J Rost, John M. Kessenich - 2006
4. **OpenGL game programming** **-**Kevin Hawkins, Dave Astle - 2001**.**
5. **Computer graphics with OpenGL -**Donald Hearn, M. Pauline Baker - 2004.

**Reference Links**

1. **www.sulaco.co.za/opengl.html**
2. **www.opengl.org/**
3. **www.khronos.org/opengl/**
4. **developer.apple.com**