Chapter 1

INTRODUCTION

1.1 Introduction to Computer Graphics

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. The field began humbly almost 50 years ago, with the display of few lines on a cathode-ray tube (CRT); now it can create images by computer that are indistinguishable from photographs of real objects. Computer Graphics became 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 and science had to relay on expensive equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, find computer graphics in a diverse area such as science, engineering, medicine, business, government, art, entertainment, education and training. So computer graphics can be considered as a generalized tool for drawing and creating pictures which simulates the real world situations within a small computer window.

1.1.1 OpenGL

OpenGL (Open Graphics Library) is a graphics software system, which has become a widely accepted standard for developing graphics applications. It is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. The interface consists of over 250 different function calls which can be used to draw complex three-dimensional scenes from simple primitives. It is widely used for games, animation, CAD/CAM, medical imaging, and other applications that need a framework for visualizing and manipulating complex, three-dimensional shapes. OpenGL programs are highly portable and produce consistent results on any supported platform.

OpenGL serves two main purposes:

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

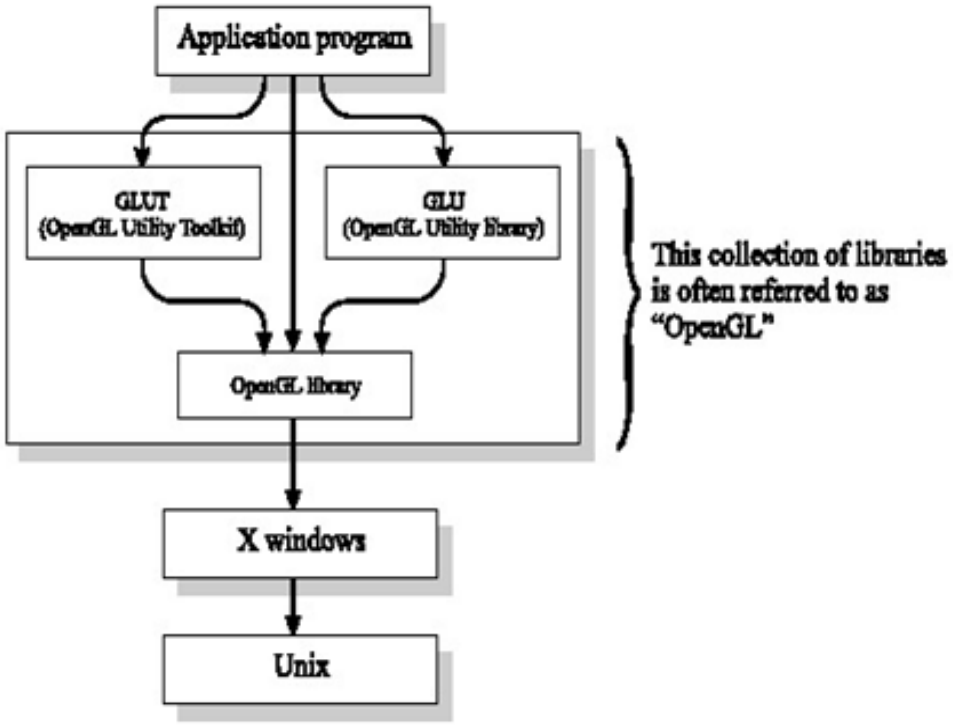


Figure 1.1 OpenGL three libraries

"OpenGL" is actually a set of three libraries: OpenGL itself, and the supporting libraries GLU and GLUT.

- GLUT provides the facilities for interaction that OpenGL lacks. It provides functions for managing windows on the display screen, and handling input events from the mouse and keyboard. It provides some rudimentary tools for creating Graphical User Interfaces (GUIs). It also includes functions for conveniently drawing 3D objects like the platonic solids, and a teapot. All GLUT function names start with "glut".

Figure 1.1 shows the relationships between OpenGL, GLU, and GLUT. As you can see, it's helpful to think of "layers" of software, where each layer calls upon the facilities of software in a lower layer. However, somewhat confusingly, when most people say "OpenGL", what they really mean is "OpenGLplus GLUplus GLUT". It's a slightly lazy terminology, but we'll use it too.

To interface with the window system and to get input from external devices into our programs, we need at least one more library. For each major window system there is a system-specific library that provides the "glue" 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, we use a readily available library called the OpenGL Utility Toolkit (GLUT), which provides the minimum functionality that should be expected in any modern windowing system.

1.1.2 Computer Graphics Technology

The Department of Computer Graphics Technology prepares visually oriented students who are interested in creating and managing the production of computer graphics for a wide range of industry. Students work in computer labs developing their graphics skills, techniques, concepts, and management ability through individual and team-based projects. After successful completion of the pre-technical graphics curriculum, students can select to specialize in one of four signature areas in interactive multimedia, technical animation, manufacturing graphics, or construction graphics.

1.1.3 Advantages of Computer Graphics

Computer graphics is used today in many different areas of industry, business, government, education, entertainment, and, most recently, the home. The list of applications is enormous and is growing rapidly as computers with graphics capabilities become commodity products. Let us look at some of these applications:-

- User interface
- Interactive plotting in business, science, and technology
- Office automation and electronic publishing
- Computer-aided drafting and design
- Simulation and animation for scientific visualization and entertainment
- Art and Commerce
- Process control
- Cartography

1.2 About Project

1.2.1 Aim

- Our aim is to draw attention of users towards computer graphics.
- Our aim is to create a domain which is simple in use.

1.2.2 Benefits

- Simplicity: Our project is easy and simple to use.
- Usability: It is easy to use and implement.

- Flexibility: It is very flexible since it is easy to add new features to it.
- Portable: It offers portability. It can be run any way where any time.
- Self-learning: Coding of project itself is understandable to others.

1.2.3 Constraints

- Project gets executed only when glut is installed.
- Output will differ if there is fault in graphic system.
- May produce flickering images if colored buffer is not cleared properly.
- We need to initialize glut library in order to interact with window system.

1.2.4 Applications

- Can be used as demonstration package for the function like translation and rotation.
- Same idea of this project can also be implemented to animated movies.

Chapter 2

REQUIREMENT SPECIFICATION

The system requirement is the total configuration of the system that system should satisfy to run the designed project as desired. This condition tells about how portable our project is.

2.1 Classification of System Requirements

They are classified into two categories**:-**

- Hardware Requirements
- Software Requirements

2.1.1 Software Requirements

- Operating system : Windows
- Software used : Code Blocks
- API used : OpenGL

2.1.2 Hardware Requirements

- Processor : AMD processor
- Main memory : 512 MB RAM
- Hard disk : 1GB

Chapter 3

SYSTEM DESIGN

3.1 Design of the System

The development of this project would improve the user's knowledge about computer graphics and OpenGL. This project provides good understanding of OpenGL prominent functions like Translation and Rotation.

Keeping the factors of usability in mind we have developed this project to provide ease of use. This project will allow user to interact with it through the use of devices like keyboard.

This project will demonstrate the use of three dimensional propeller in OpenGL. Our project has detail information about simulation of the propeller in different forms as well as regaining its original shape.

The project is mainly divided into two functions. They are as follows:

- Translation
- Rotation

Translation

Translation is an operation that displaces points by fixed distance in a given direction. A translation takes a point and maps that point into another point. We use translation in order to shift from one position to other.

In our project glTranslate produces a translation by x, y, z. The current matrix is multiplied by this translation matrix, with the product replacing the current matrix. If the matrix mode is either GL_MODELVIEW or GL_PROJECTION**,** all objects are drawn after a call to glTranslate or glTranslated**.** Use glPushMatrix and glPopMatrix to save and restore the untranslated coordinate system. To perform translation we can use following function call

void glTranslatef(GLfloat x, GLfloat y, GLfloat z);

Again we can translate the object in following direction:

- X-axis
- Y-axis
- Z-axis

Rotation

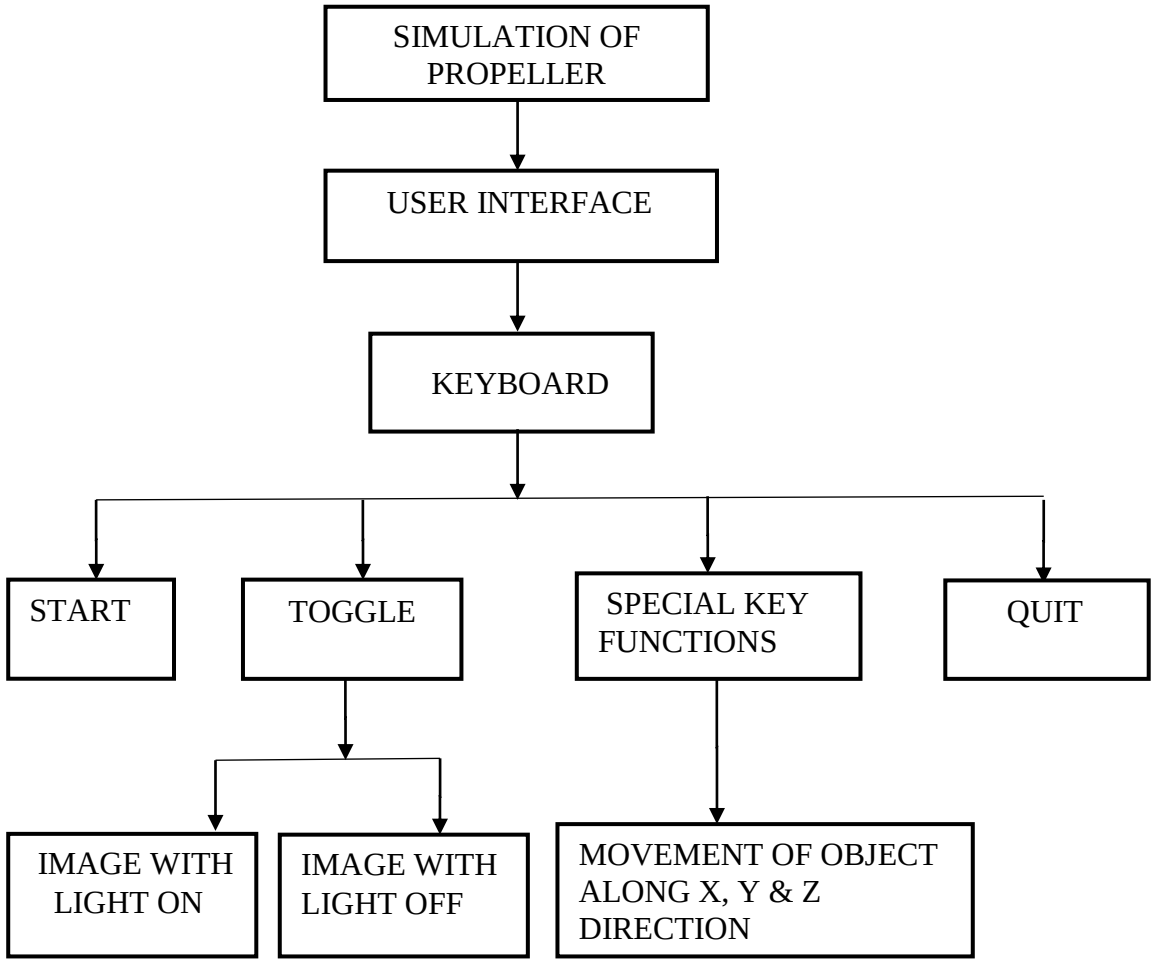
Rotation is an operation that rotates the object about fixed point in any given direction. A rotated function takes four parameters first is to accept the angle according to which it has to perform rotation and remaining three parameters for direction of rotation. In our project we use rotation to rotate the propeller along with the cone about origin in the desired direction. Following are the direction in which we rotate cone and propeller.

- Left direction
- Right direction
- Up direction
- Down direction

To perform the rotation we use the following function

glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z);

where angle specifies how many degree we want to rotate around an axis and x, y, z specifies the axis to rotate respectively and below figure shows us the Design of the project



Chapter4

SYSTEM IMPLEMENTATION

4.1 Pseudocode

The execution of the program starts from the main function. It calls various inbuilt and user defined functions, the pseudo code is as follows:

- **Void DrawTopTriangleSet()** : Draw the top triangles of propeller.
- **Void DrawTopTriangleSetNormalVector() :** Draw the top triangles of propeller with normal vectors.
- **Void DrawBottomTriangleSet() :** Draw the bottom triangles of propeller.
- **Void DrawBottomTriangleSetNormalVector() :** Draw the bottom triangles of propeller with normal vectors.
- **Void DrawBackTriangleSet() :** Draw the back triangles of propeller.
- **Void DrawBackTriangleSetNormalVector() :** Draw the back triangles of propeller with normal vectors.
- **Void DrawInsideTriangleSet() :** Draw the inside triangles of propeller.
- **Void writemessage() :** Writes the message required to make the movements of propeller.
- **Void reshape(int w, int h) :** Reshapes the current window size.
- **Void display() :** The function is called by GLUT to display the text.
- **Void keyboard(unsigned char key, int x, int y) :** Alphabet specific keys for movement of propeller.
- **Void SpecialKeys(unsigned char key, int x, int y) :** Cursor key functions to move propeller.
- **Void mouse(int btn, int state, int x, int y) :** Mouse buttons used to quit the program.

4.2 Functions

The functions that are used in the program are discussed below. This section contains brief description of all the headers and functions. These functions are as follows:

4.2.1 Headers Defined

The in-built are defined in the OpenGL library. Some of the headers that are used as follows:

- **#include<stdio.h>**: to take input from standard input and write to standard output.
- **#include<stdlib.h>**: to include standard library functions.
- **#include<GL/glut.h>**: to include glut library files.

4.2.2 Inbuilt Functions

OpenGL functions used in the code are as follows:

- **glClearColor() :** This function call sets the present RGBA clear color used when clearing the color buffer.
  General syntax: void glClearColor(GLclampf red, GLclampf green, GLclampf blue, GLclampf alpha);
- **glFlush() :** This function call forces any buffered openGL command to execute.
  General syntax : void glFlush();
- **glutInit() :** This function call initializes glut library.
  General syntax : void glutInit(int *argc, char **argv);
  The arguments from main are passed in and can be used by an application.
  Ex : glutInit(&argc, argv);
- **glutCreateWindow() :** This function call creates a window on the display, the string title can be used to label the window.
  Ex : glutCreateWindow("SIMULATION OF PROPELLER");
- **glutInitDisplayMode() :** This function call request a display with the properties that are specified in m mode. The value is determined by the logical OR operation of options including the color model.
  General syntax : void glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
- **glutInitWindowSize() :** This function call specifies the initial height and width of the window in pixels.
  General syntax : void glutInitWindowSize(int width, int height);
  Ex : glutInitWindowSize(500, 500);
- **glutInitWindowPosition() :** this function call specifies the initial position of top left corner of the window in pixels.

General syntax : void glutInitWindowPosition(int x, int y);

Ex : glutInitWindowPosition(100, 100);

- **glutDisplayFunc() :** This function call registers the display function *func i.e executed when the window needs to be redrawn.

  General syntax : void glutDisplayFunc(void(*func)(void));

  Ex : glutDisplayFunc(display);

- **glutPostRedisplay() :** This function call requests the display call back after the current call back returns.

  General syntax : void glutPostRedisplay();

  Ex : glutPostRedisplay();

- **glutMainLoop():** This function call causes the program to enter an event processing loop. It should be the last statement in main.

- **glClear() :**Clears buffer to preset values. Specifies BITWISE OR of masks that indicate the buffers to be cleared.

  General syntax : void glClear(GLbitfield mask);

  Ex : glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

- **glBegin() :**Specifies the primitive or primitives that will be created from vertices presented between glBegin() and glEnd().

  General syntax : void glBegin(GLenum mode);

  Ex : glBegin(GL_TRIANGLE_STRIP);

- **glTranslate() :**It produces a translation by (x,y,z). The current matrix is multiplied by this translation matrix, with the product replacing the current matrix.

  General syntax : void glTranslatef(GLfloat x, GLfloat y, GLfloat z);

  Ex : glTranslated(0, 0, -1.5);

- **glutKeyboardFunc() :**It is a user interactive function which displays snapshots on hitting the appropriate keyboard keys onto the display screen.

  Ex : glutKeyboardFunc(keys);

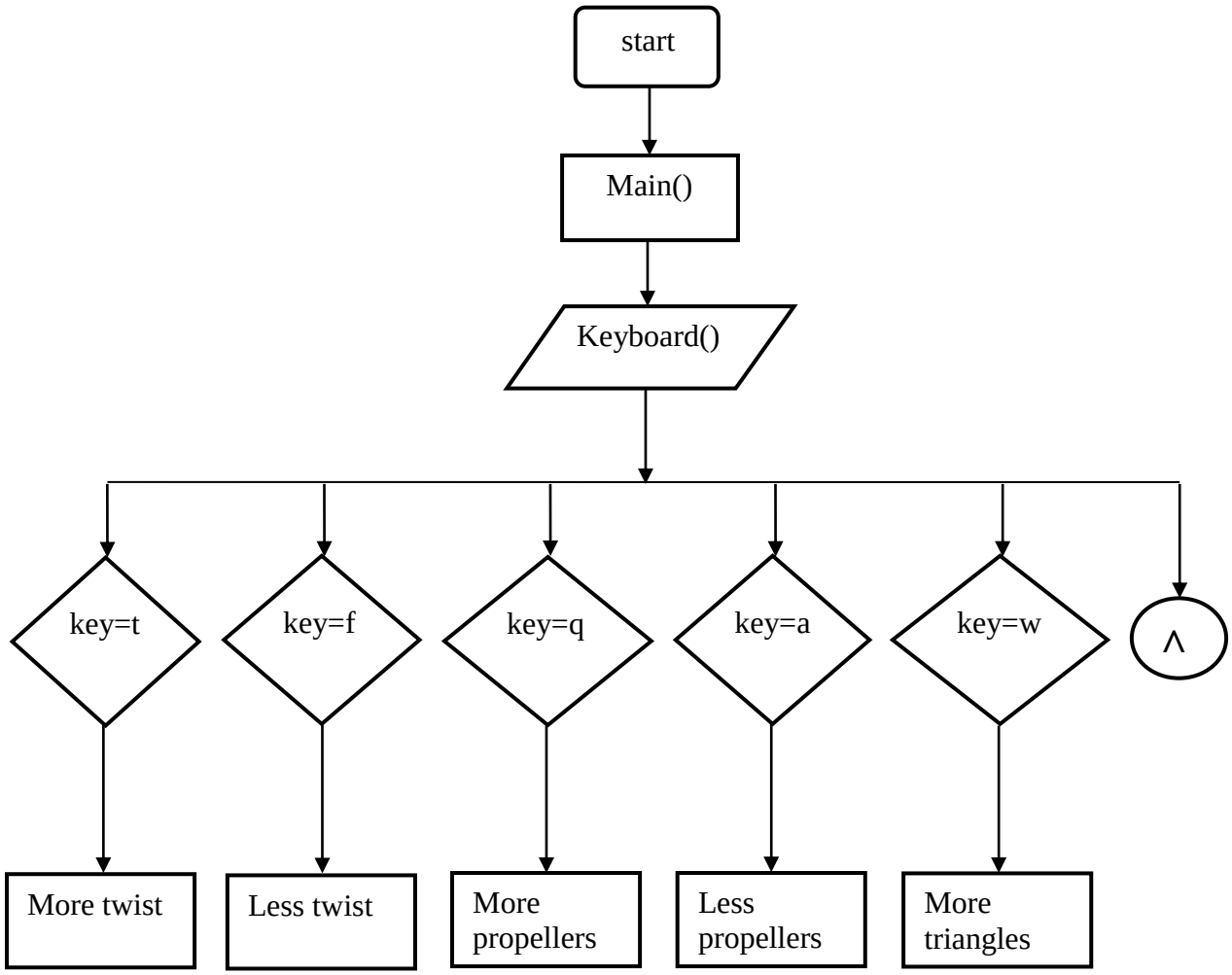
- **glutMouseFunc() :** Used to set mouse callback for the current window and quit the program.

  Ex : glutMouseFunc(mouse);

- **glVertex() :** Function commands are used within the glBegin/glEnd to specify points, line and polygon vertices.

  General syntax : void glVertex3f(GLint x, GLint y,GLint z);

4.3 Flowchart



4.4 Coding

Display Functions:

```
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   cpos[0] = zoom * cos(beta) * sin(alpha);
   cpos[1] = zoom * sin(beta);
   cpos[2] = zoom * cos(beta) * cos(alpha);
   gluLookAt(cpos[0], cpos[1], cpos[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
```

```
if (lightSource == true){
   glLightfv(GL_LIGHT0, GL_POSITION, lpos);
   glMaterialfv(GL_FRONT, GL_EMISSION, white);
   glPushMatrix();
   glTranslatef(lpos[0], lpos[1], lpos[2]);
   glutSolidSphere(0.1, 10, 8);
   glPopMatrix();
   glMaterialfv(GL_FRONT, GL_EMISSION, black);
}
glMaterialfv(GL_FRONT, GL_EMISSION, black);
glMaterialfv(GL_BACK, GL_EMISSION, black);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
glMaterialfv(GL_BACK, GL_AMBIENT_AND_DIFFUSE, red);
glPushMatrix();
glTranslated(0, 0, -1.5);
glutSolidCone(1, 2, 50, 50);
glPopMatrix();
glMaterialfv(GL_FRONT, GL_EMISSION, black);
glMaterialfv(GL_BACK, GL_EMISSION, black);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
glMaterialfv(GL_BACK, GL_AMBIENT_AND_DIFFUSE, red);
glBegin(GL_POLYGON);
glNormal3f(0, 0, 1);
for (int i = 0; i <= 360; i++)
{ glVertex3f(cos(i*PI / 180) * 1, sin(i*PI / 180) * 1, -1.5); }
glEnd();
glPopMatrix();
glPushMatrix();
glTranslated(3,0,0);
glRotatef(rotateConstant, 0, 0, 1);
DrawTopCubeSet();
for (int i = 0; i < numberOfObj; i++){
glPushMatrix();
glRotatef(i * 360 / numberOfObj, 0, 0,1 );
glScaled(.2,2,4);
glutSolidCube(.7);
glRotatef(i * 360 / numberOfObj, 0, 0, 1);
glPopMatrix();    }
glMaterialfv(GL_FRONT, GL_EMISSION, black);
glMaterialfv(GL_BACK, GL_EMISSION, black);
glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
glMaterialfv(GL_BACK, GL_AMBIENT_AND_DIFFUSE, red);
glPushMatrix();
```

```
    glTranslated(0, 0, -1.5);
    glutSolidCone(1, 2, 50, 50);
    glPopMatrix();
    glMaterialfv(GL_FRONT, GL_EMISSION, black);
    glMaterialfv(GL_BACK, GL_EMISSION, black);
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, red);
    glMaterialfv(GL_BACK, GL_AMBIENT_AND_DIFFUSE, red);
    glBegin(GL_POLYGON);
    glNormal3f(0, 0, 1);
    for (int i = 0; i <= 360; i++){
    glVertex3f(cos(i*PI / 180) * 1, sin(i*PI / 180) * 1, -1.5);}
    glEnd();
    glPopMatrix();
    glutSwapBuffers();
    glFlush();
}
```

Mouse Function:

```
void mouse(int button,int state,int x,int y)
{
if(button==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
exit(0);
if(button==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
exit(0);
if(button==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
exit(0);
}
```

Main Functions:

```
int main(int argc, char** argv)
{
    writemessage();
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(1200, 800);
    glutInitWindowPosition(0, 0);
    glutCreateWindow(argv[0]);
    glClearColor(0.0, 0.0, 1.0, 1.0);
    glEnable(GL_DEPTH_TEST);
```

```
    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
    glEnable(GL_LIGHT0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(0.0, 5.0, 10.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0);
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboard);
    glutSpecialFunc(specialkey);
    glutMouseFunc(mouse);
    glutMainLoop();
    return 0;
}
```

Chapter 5

TESTING

5.1 Introduction to Testing

Verification and validation is a generic name given to checking processes, which ensures that the software confirms to its specifications and meets the demands of users.

Validation

Validation involves checking that the program has implanted meets the requirement of the users.

Verification

Verification involves checking that the program confirms to its specification.

5.2 Test Cases

Test case Id	Test case description	Input	Actual output	Expected output	Remarks
1	Initial view of propeller	Execute the program	Displays solid cone propeller and a light source, refer Figure 6.1	Has to display solid cone propeller and a light source	Pass
2	Propeller with more twist	Press Key 't'	Displays propeller with more twist, refer Figure 6.2	Has to display propeller with more twist	Pass
3	Propeller with less twist	Press Key 'f'	Displays propeller with less twist, refer Figure 6.3	Has to display propeller with less twist	Pass

4	More number of propellers	Press Key 'q'	Displays more number of propellers, refer Figure 6.4	Has to display more number of propellers	Pass
5	Less number of propellers	Press Key 'a'	Displays less number of propellers, refer Figure 6.5	Has to display less number of propellers	Pass
6	Propeller with more triangles	Press Key 'w'	Displays more number of triangles, refer Figure 6.6	Has to more number of triangles	Pass
7	Propeller with less triangles	Press Key 's'	Displays less number of triangles, refer Figure 6.7	Has to display less number of triangles	Pass
8	Propeller view with toggle of light source	Press Key '0'	Displays light source on and off, refer Figure 6.8	Has to switch on and off of light source	Pass
9	Propeller with normal vectors	Press Key 'n'	Display normal vectors, refer Figure 6.9	Has to display normal vectors	Pass

10	Rotation of propeller	Press Key 'r'	Displays rotation of propeller	Propeller has to rotate	Pass
11	Left view of propeller	Press Left arrow	Displays left view of propeller, refer Figure 6.11	Has to display left view of propeller	Pass
12	Right view of propeller	Press Right arrow	Right view of propeller will be displayed, refer Figure 6.12	Has to display right view of propeller	Pass
13	Top view of propeller	Press Upward arrow	Top view of propeller will be displayed, refer Figure 6.13	Has to display top view of propeller	Pass
14	Bottom view of propeller	Press Downward arrow	Bottom view of propeller will be displayed, refer Figure 6.14	Has to display bottom view of propeller	Pass
15	Propeller view along X-axis	Press Key 'X'	Propeller view along X-axis will be displayed.	Has to display propeller view along X-axis	Pass

Table 5.1 Test case table

Chapter6

RESULT AND SNAPSHOTS

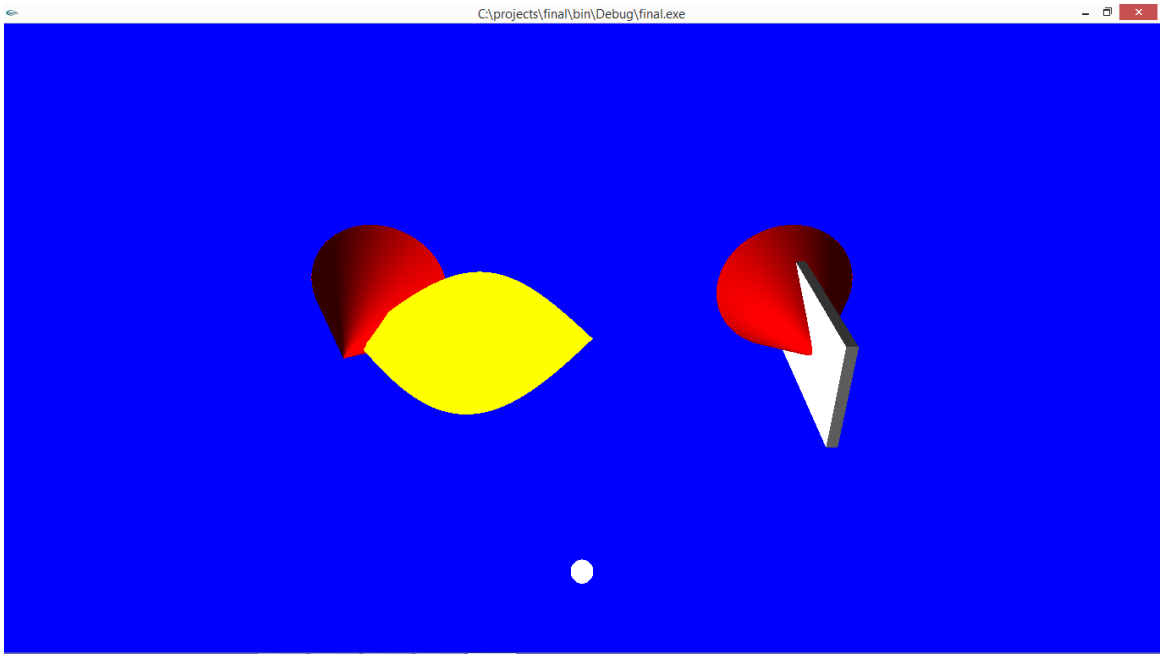


Figure6.1 Initial view of propeller

Above Figure 6.1 illustrates the initial view of propeller containing solid cone, a propeller triangle and a light source.

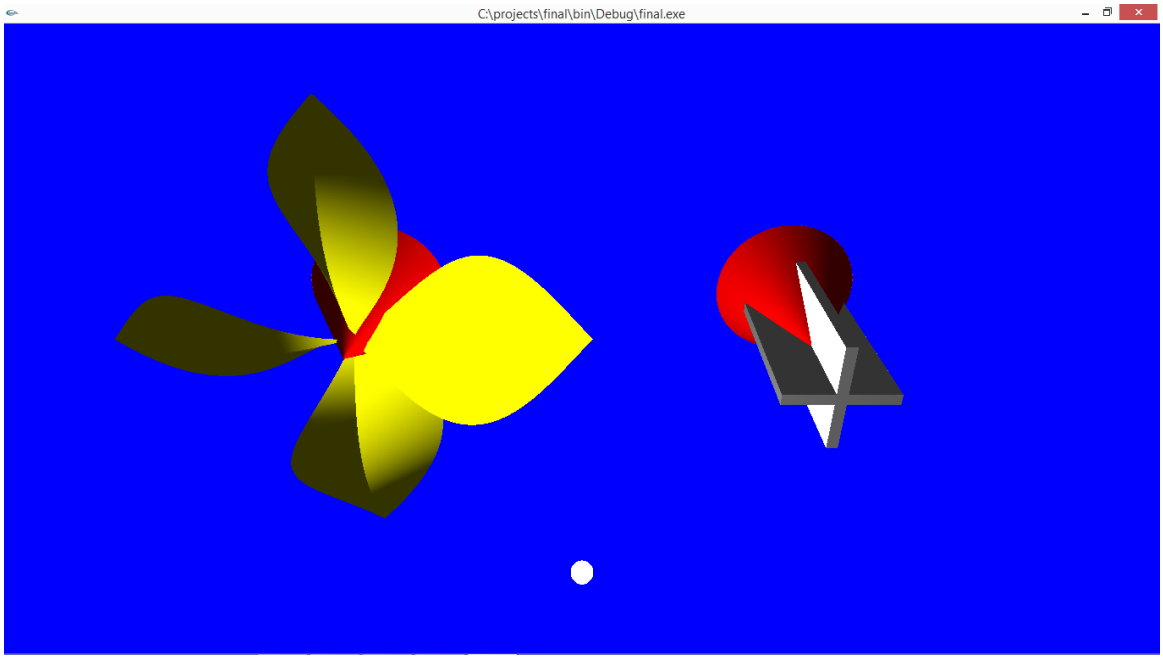


Figure 6.2 Propeller with more twist

Above Figure 6.2 illustrates the more twist of propeller that is obtained by pressing key 't' in keyboard.

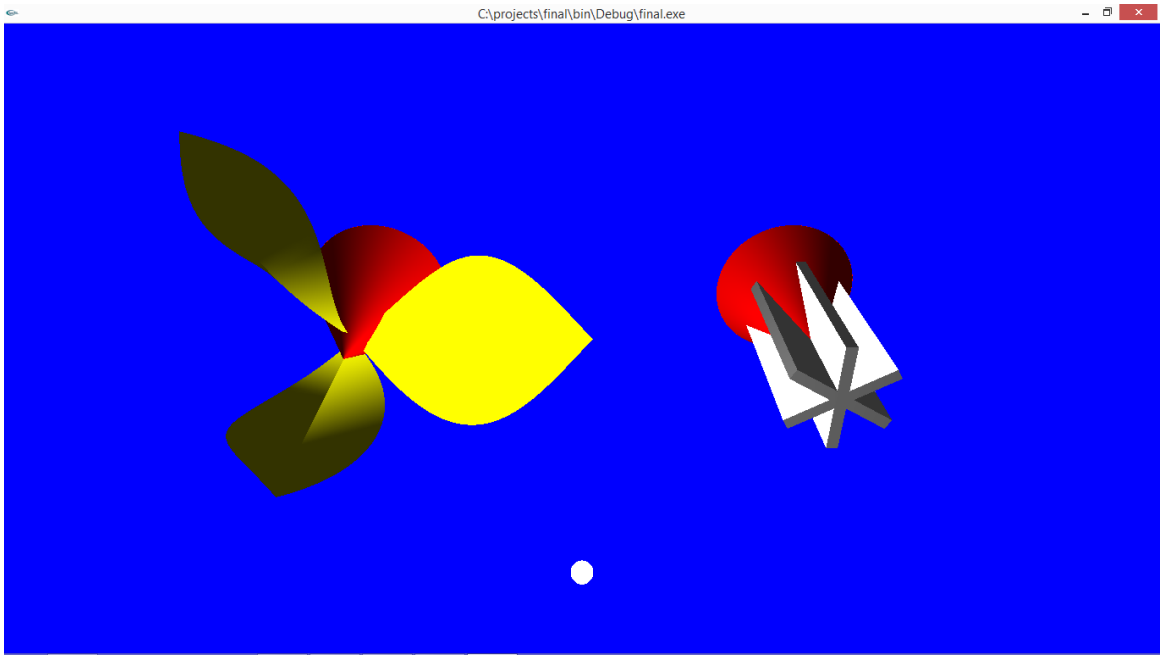


Figure 6.3 Propeller with less twist

Above Figure 6.3 illustrates the less twist of propeller that is obtained by pressing key 'f' in keyboard.

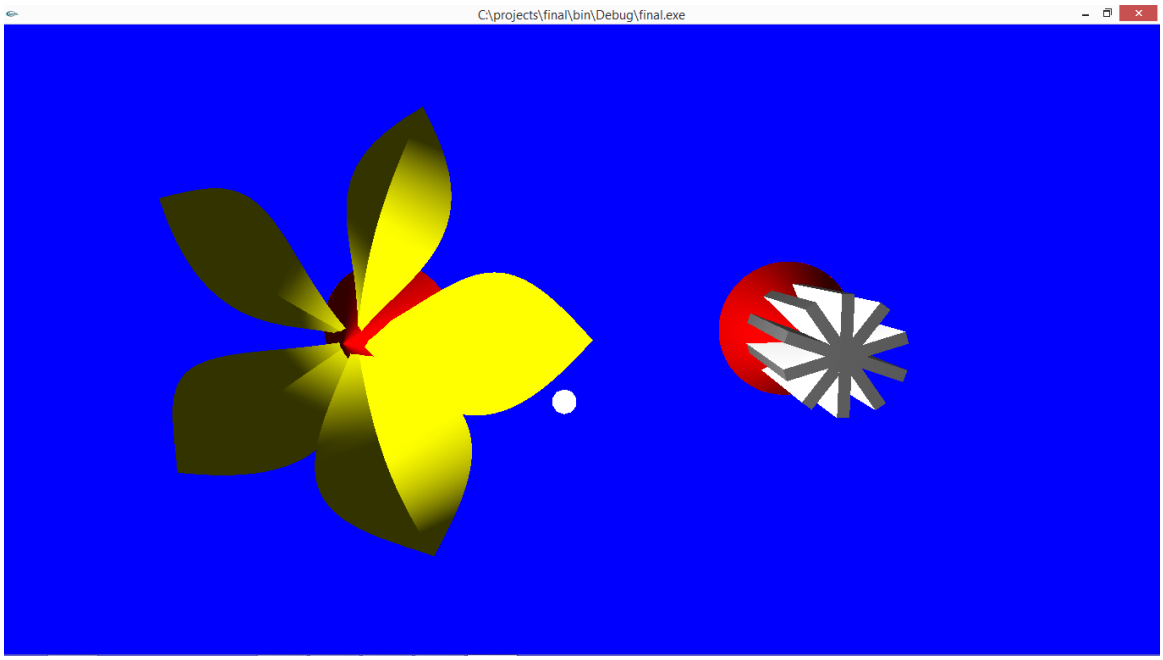


Figure 6.4 More number of propeller

Above Figure 6.4 illustrates the more number of propellers that is obtained by pressing key 'q' in keyboard.

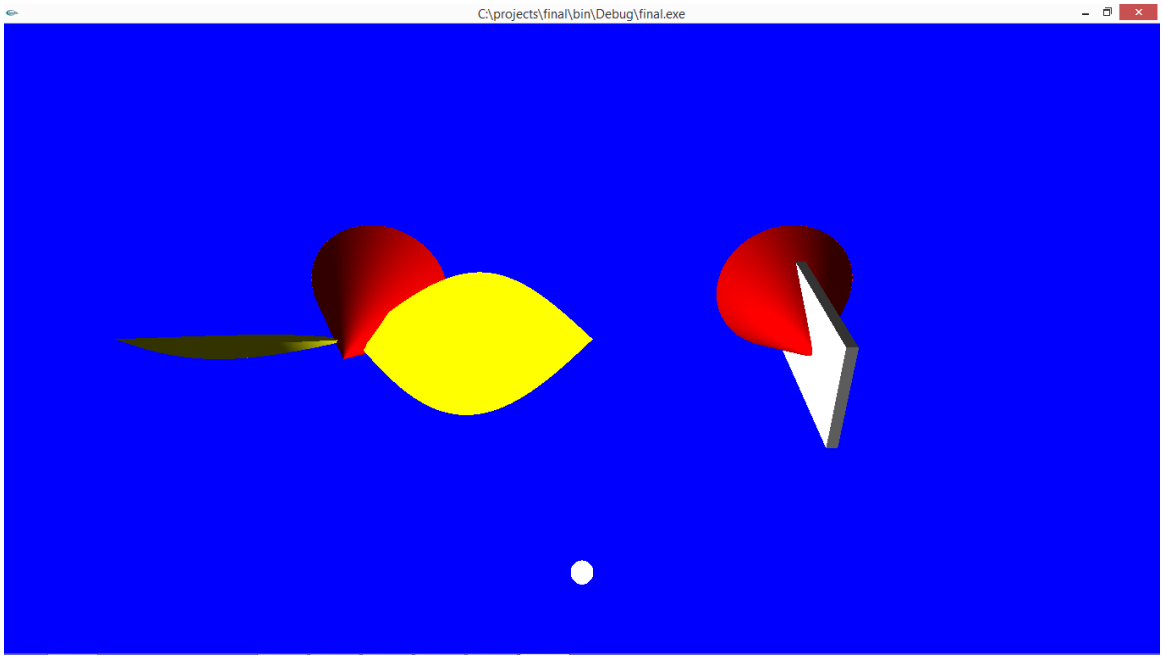


Figure 6.5 Less number of propeller

Above Figure 6.5 illustrates the less number of propellers that is obtained by pressing key 'a' in keyboard.

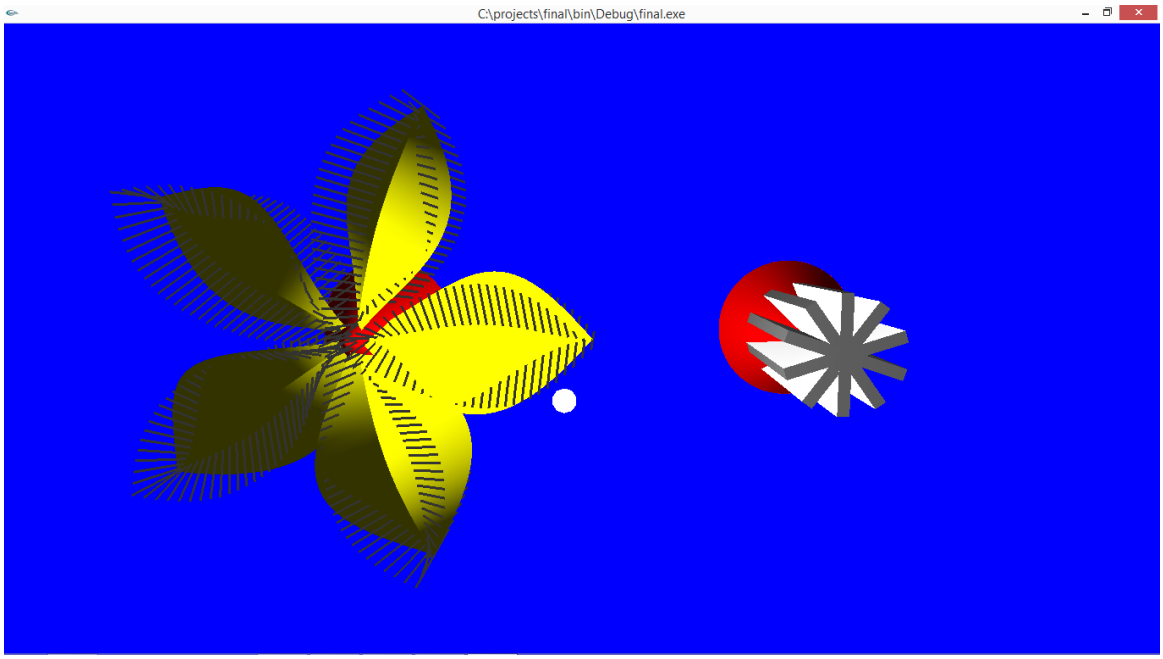


Figure 6.6 Propeller with more triangles

Above Figure 6.6 illustrates the propeller with more triangles that is obtained by pressing key 'w' in keyboard.

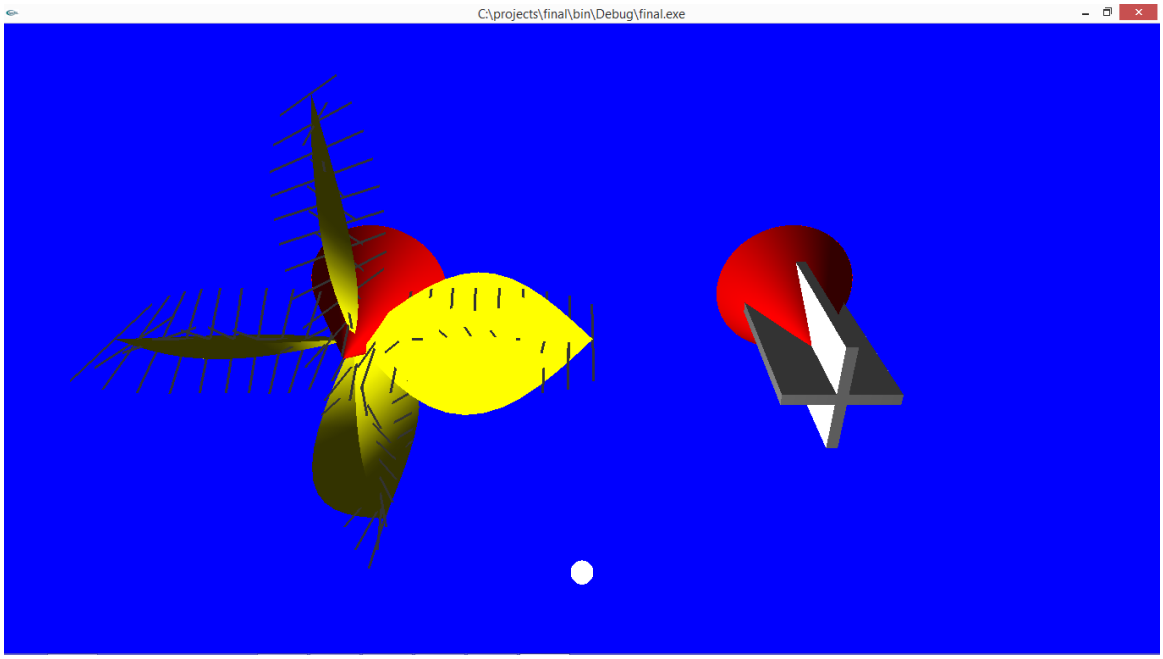


Figure 6.7 Propeller with less triangles

Above Figure 6.7 illustrates the propeller with less triangles that is obtained by pressing key's' in keyboard.

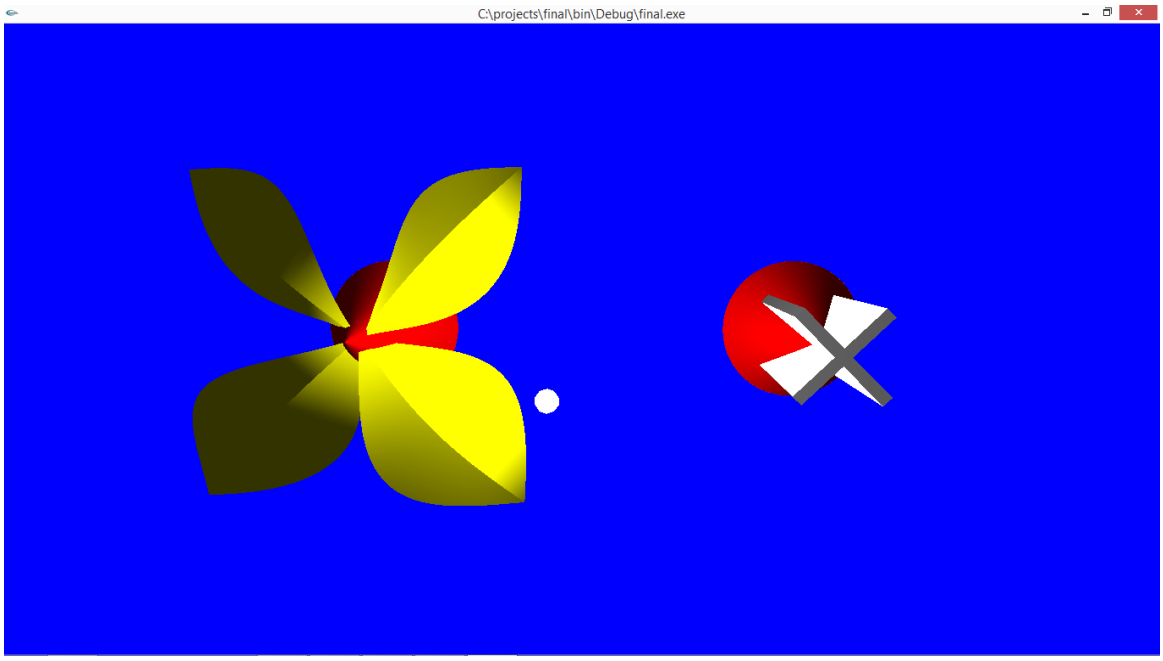


Figure 6.8 Rotation of propeller

Above Figure 6.10 illustrates the rotation of propeller that is obtained by pressing key 'r' in keyboard.

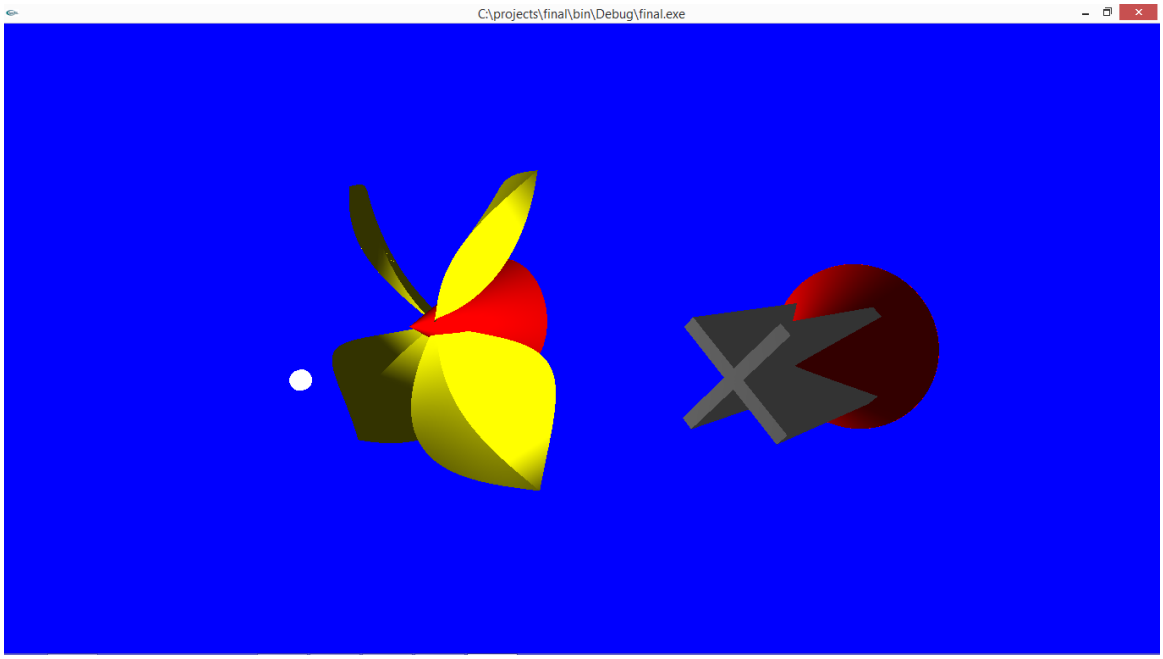


Figure 6.9 Left view of the propeller

Above Figure 6.11 illustrates the left view of the propeller that is obtained by moving the light source towards left using left arrow key in keyboard.

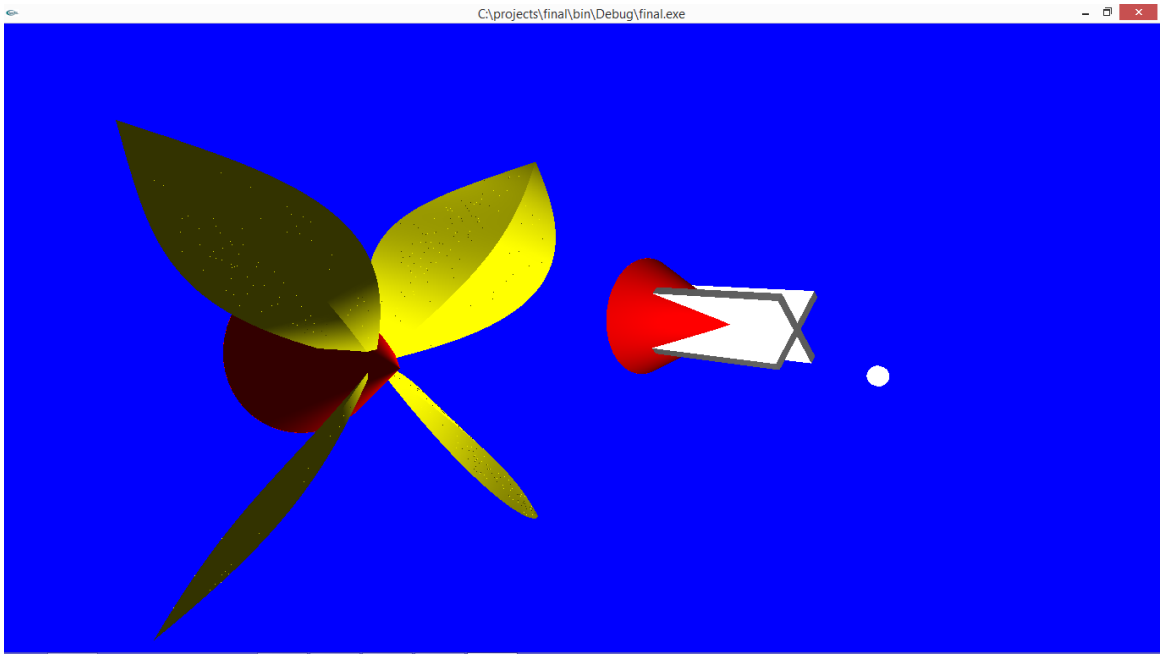


Figure 6.10 Right view of the propeller

Above Figure 6.12 illustrates the right view of the propeller that is obtained by moving the light source towards right using right arrow key in keyboard.

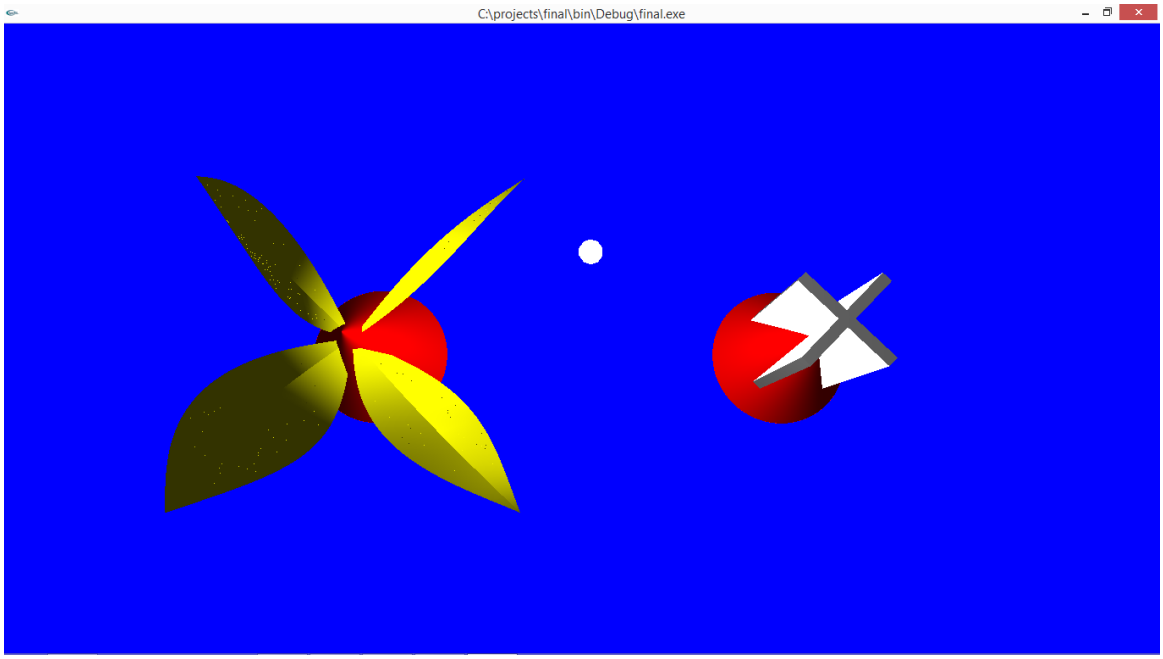


Figure 6.11 Top view of the propeller

Above Figure 6.13 illustrates the top view of the propeller that is obtained by moving the light source towards up using up arrow key in keyboard.

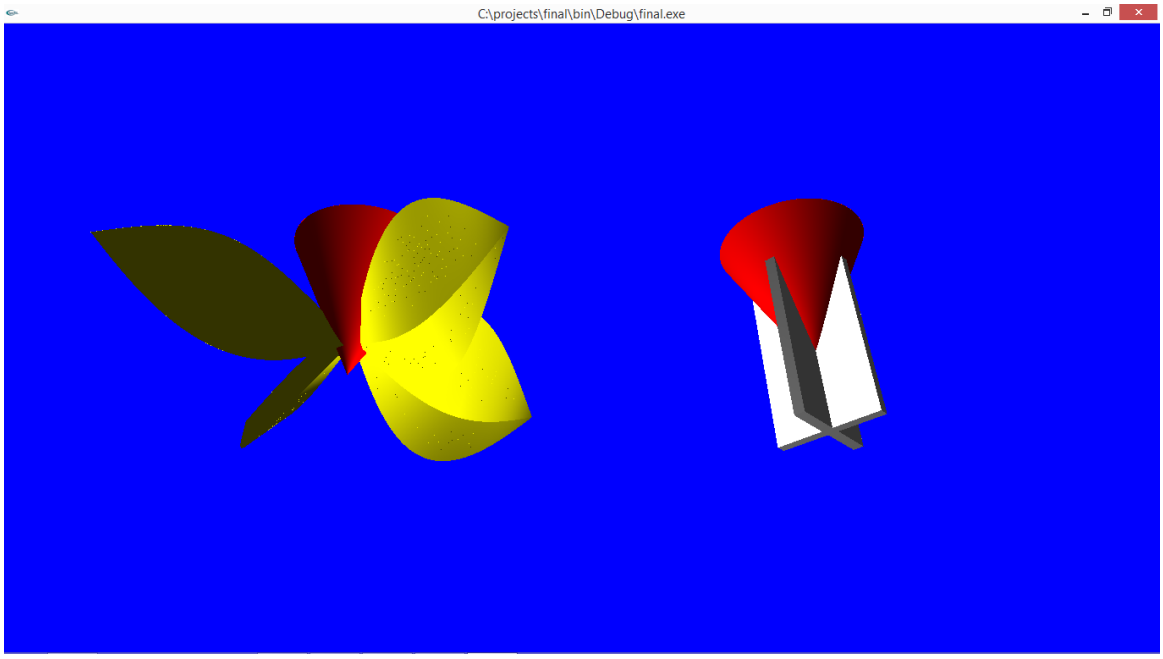


Figure 6.12 Bottom view of the propeller

Above Figure 6.14 illustrates the bottom view of the propeller that is obtained by moving the light source towards down using down arrow in keyboard.

Chapter 7

CONCLUSION

The project has involved the designing of propeller. It is implemented using some in-built functions which are provided in the standard graphics package. We have exhibited the view of propeller and its movements. This project draws a propeller and its movements. The project consist of propeller's different models like propeller with normal vectors, propeller with more and less triangles. The propeller rotate about its axis, moves down, right and left. We have light source which can move in all x, y, z direction, which illuminates the specific part of propeller where light falls. We can zoom in and out the scene. We can also switch on and off the light source. We can start and stop animation by left and right mouse key button respectively. The movements are done with the help of keyboard and mouse functions which have been successfully implemented.

Chapter 8

FUTURE ENHANCEMENT

We have implemented graphical functions like movement of the propeller, light source and also its rotation. We look forward to implement this in future along with some additional functions and algorithms.

- Giving the option of allowing the user to put any image of his choice onto the tiles. This again involves Texture mapping and image processing.
- Implementing different levels with increasing levels of difficult
- Giving the option of following user to display the inside triangles of the propeller and propeller in wire form
- Giving the option of following user to change the color of propeller and to modify its background

Chapter 9

REFERENCES

TEXT BOOKS

1. Edward Angel, 2009, Interactive computer graphics: A top-down Approach With OpenGL, 5th edition, Addison-Wesley

2. Blind, J.F., and M.E. Newell, 1976, Texture and Reflection in Computer Generated Images, CACM, 19(10)

3. Hearn, D., and M.P Baker, Computer Graphics, Second Edition, Prentice-Hall, Englewood Cliffs, NJ, 2010.

WEBSITES

http://www.stackoverflow.com

http://www.opengl.org/resouces/libraries/glut

http://www.opengl.org

PERSONAL DETAILS:

NAMES: SWAPNIL POWAR
SWARAJ PAUL

USN : 1DT15CS421
1DT14CS102

SEMESTER AND SECTION: 6TH SEM, B SEC

DEPARTMENT: COMPUTER SCIENCE AND ENGINEERING

COLLEGE: DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT

EMAIL IDs: powarswap@gmail.com
paulswaraj15@gmail.com