**Chapter 5**

**IMPLEMENTATION**

**5.1 Built-in functions**

1. **glRasterPos3f( x, y, -1)**

Specifies the raster position for pixel operations.

1. **glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, st[i])**

Renders a bitmap charater using OpenGL from the specified array of characters, and in the specified font style.

1. **glutPostRedisplay()**

Marks the current window as needing to be redisplayed.

1. **glutTimerFunc( 100, TimeEvent, 1)**

Registers a timer callback to be triggered in a specified number of milliseconds.

1. **glClearColor (0.0, 0.0, 0.0, 0.0)**

Specifies clear values for the color buffers.

1. **glShadeModel (GL\_SMOOTH)**

Select flat or smooth shading. Specifies a symbolic value representing a shading technique. Accepted values are GL\_FLAT and GL\_SMOOTH.

1. **glEnable(GL\_DEPTH\_TEST)**

Enables the OpenGL capabilities, Specifies the conditions under which the pixels will be drawn.

1. **glLightfv(GL\_LIGHT1 ,GL\_AMBIENT, LightAmbient)**

Returns light source parameter values.

1. **gluQuadricDrawStyle( Cylinder, GLU\_FILL)**

Specifies the draw style required for quadrics.

1. **glPushMatrix()** and **glPopMatrix()**

Push and pop the current matrix stack.

1. **glTranslatef() and glRotatef()**

Multiplies current matrix by Translation and Rotation matrix respectively.

1. **glMatrixMode (GL\_PROJECTION)**

Specifies which matrix is the current matrix.

1. **glLoadIdentity()**

Replaces current matrix with identity matrix.

1. **gluLookAt()**

Defines a viewing transformation.

1. **glutSwapBuffers()**

Swaps the buffers of the current window if double buffered.

1. **glViewport()**

Sets the viewport.

1. **glutInitDisplayMode (GLUT\_DOUBLE | GLUT\_RGB)**

Sets the initial display mode.

1. **glutInitWindowSize (500, 500)** and **glutInitWindowPosition (50, 50)**

Set the initial window size and position respectively.

1. **glutCreateWindow()**

Creates a top level window with the window name as specified.

1. **glutAddMenuEntry()**

 Adds a menu entry to the bottom of the current menu.

1. **glutAttachMenu(GLUT\_RIGHT\_BUTTON)**

Attaches a mouse button for the current window to the identifier of the current menu**.**

1. **glutDisplayFunc(display)**

Sets the display callback for the current window.

1. **glutReshapeFunc(reshape)**

Sets the reshape callback for the current window.

1. **glutMainLoop()**

Enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

**5.2 Source Code**

#include <GL/glut.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

// define glu objects

int about\_int=0;

GLUquadricObj \*Cylinder;

GLUquadricObj \*Disk;

struct tm \*newtime;

time\_t ltime;

int M\_TWOPI=0;

GLfloat rx, ry, rz, angle;

// lighting

GLfloat LightAmbient[]= { 0.5f, 0.5f, 0.5f, 1.0f };

GLfloat LightDiffuse[]= { 0.5f, 0.5f, 0.5f, 1.0f };

GLfloat LightPosition[]= { 5.0f, 25.0f, 15.0f, 1.0f };

GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };

static int light\_state = 1; // light on = 1, light off = 0

static int view\_state = 1; // Ortho view = 1, Perspective = 0

void Sprint( float x, float y, char \*st)

{

int l,i;

l=strlen( st );

glRasterPos3f( x, y, -1);

for( i=0; i < l; i++)

{

glutBitmapCharacter(GLUT\_BITMAP\_TIMES\_ROMAN\_24, st[i]);

}

}

static void TimeEvent(int te)

{

rx = 30 \* cos( angle );

ry = 30 \* sin( angle );

rz = 30 \* cos( angle );

angle += 0.01;

if (angle > M\_TWOPI) angle = 0;

glutPostRedisplay();

glutTimerFunc( 100, TimeEvent, 1);

}

void init(void)

{

glClearColor (0.0, 0.0, 0.0, 0.0);

glShadeModel (GL\_SMOOTH);

glEnable(GL\_DEPTH\_TEST);

// Lighting is added to scene

glLightfv(GL\_LIGHT1 ,GL\_AMBIENT, LightAmbient);

glLightfv(GL\_LIGHT1 ,GL\_DIFFUSE, LightDiffuse);

glLightfv(GL\_LIGHT1 ,GL\_POSITION, LightPosition);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT1);

Cylinder = gluNewQuadric();

gluQuadricDrawStyle( Cylinder, GLU\_FILL);

gluQuadricNormals( Cylinder, GLU\_SMOOTH);

gluQuadricOrientation( Cylinder, GLU\_OUTSIDE);

gluQuadricTexture( Cylinder, GL\_TRUE);