**Chapter 1**

**INTRODUCTION**

* 1. **Computer Graphics**
* Computer graphics are pictures and movies created using computers - usually referring to image data created by a computer specifically with help from specialized graphical hardware and software.
* Important topics in computer graphics include user interface design, sprite graphics, vector graphics, 3D modeling, shaders, GPU design, and computer vision.
* Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
* Graphics provide a so natural means of communicating with the computer that they have become widespread.
* Computer graphics can be broadly classified into two types

a) Non Interactive Computer Graphics.

b) Interactive Computer Graphics.

* Interactive graphics is the most important means of producing pictures since the invention of photography and television .
* A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.

* 1. **OpenGL**
* OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2D and 3D graphics application programming interface (API), bringing thousands of applications to a wide variety of computer platforms.
* OpenGL fosters innovation and speeds application development by incorporating a broad set of rendering, texture mapping, special effects, and other powerful visualization functions. Developers can leverage the power of OpenGL across all popular desktop and workstation platforms, ensuring wide application deployment.
* OpenGLAvailable Everywhere: Supported on all UNIX® workstations, and shipped standard with every Windows 95/98/2000/NT and Mac OS PC, no other graphics API operates on a wider range of hardware platforms and software environments.
* OpenGL’s main purpose is to render two and three dimensional objects into a frame buffer. These objects are described as sequences of vertices or pixels.
* A sophisticated library that provides these features could certainly be built on top of OpenGL

.

* The libraries are

AGL, GLX, WGL- interaction between OpenGL and windowing system.

GLU (OpenGL Utility Library)-part of OpenGL.

GLUT (OpenGL Utility Toolkit)-portable windowing API.

* 1. **Project – Hover**

The literal meaning of hover is to navigate to a place and stand still in the air. Hover is selected as the name for the project due to the similar function that is incorporated in the project.

Computer Graphics is an excellent tool to visualize objects on a computer screen. The whole universe is made up of matter and every matter has its own size and shape.

Humans identify the objects based on its color, shape, size, surface of material etc…, it will be very helpful if the same can be established with the help of a computer. Project Hover aims at providing a way for identification of an object by a computer by using OpenGL API’s.

Hover makes it possible to identify a 3-D object just by moving the mouse cursor on the object of interest. The object gets highlighted and the name of the object will be displayed at the bottom of the screen.

Hover also provides a way for displaying 3-D object in filled and outlined modes. It also provides an excellent way to simulate the use of lighting and material models provided by OpenGL.

Hover can be used as a tool for school children to make them learn about various geometrical objects and day to day objects.

**Chapter 2**

**SYSTEM REQUIREMENTS**

**2.1 Hardware Requirements**

* Hard disk : 40GB Hard disk or higher.
* Operating System : Windows/Linux/Mac.
* Processor : Intel Pentium dual core/AMD Athlon or higher.
* Memory : 1GB or higher.
* Monitor : Mono/Color.
* Keyboard : Low profile, dispatch able type.
* Mouse : optical
* Graphics Card : 1280x800 display with qualified hardware accelerated

OpenGL Graphics card.

**2.2 Software Requirements**

* Integrated Development Environment: Microsoft Visual Studio 2008 or higher.
* OpenGL Interface.
* Windows 32-Bit Operating System.

**Library Files- OpenGL Utility Library(GLU)**

* OPENGL32.LIB
* GLU32.LIB
* GLUT32.LIB

**Header Files- OpenGL Utility Toolkit (GLUT)**

* GL.H
* GLU.H
* GLU.H
* GLUT.H

**DLL Files**

* OPENGL32.DLL
* GLU32.DLL
* GLUT32.DLL

**Chapter3**

**System Design**

This project shows the use of OpenGL navigation, lighting and material models.The clear gesture of stages is shown.The design contains only the names of each stage.This project provides the user to identify the objects easily.

**Figure 3.1 : Design of Hover**

**Chapter 4**

**IMPLEMENTATION**

**Header Files:**

**#include<stdio.h>**

**stdio.h** which stands for “standard input/output header”, is the header in the C standard library that contains macros, constants and declaration functions used for various input and output operations.

**#include<stdlib.h>**

**Stdlib.h**is the header of the general purpose standard library of c programming language.

**#include<stdarg.h>**

**stdarg.h** is a header in the C standard library of the C programming language that allows functions to accept an indefinite number of arguments.

**Functions:**

**glutReshapeFunc Function**

glutReshapeFunc sets the reshape callback for the current window.

Syntax:

voidglutReshapeFunc(void (\*func)(int width, int height));

Parameters:

func:The new reshape callback function.

**glutDisplayFunc Function**

glutDisplayFunc sets the display callback for the currentwindow.

Syntax:

voidglutDisplayFunc(void (\*func)(void));

func:The new display callback function.

**glScalef Function**

Alters the current matrix by a scaling of (sx, sy, sz). TYPE here is GLfloat.

Here in the above considered example we use scaling to minimize the length of the curve at each iteration. For this curve we use the scale factor to be 3 units because we substitute a line by 4 lines in each iteration.

Syntax:

voidglScalef(TYPE sx, TYPE sy, TYPE sz);

**glTranslatef Function**

Alters the current matrix by a displacement of (x, y, z). TYPE here is GLfloat.

We need to translate to display the new position of the line from the old position and also to go out to the beginning of the next side while drawing.

Syntax:

voidglTranslatef(TYPE x, TYPE y, TYPE z);

**glRotatef Function**

Alters the current matrix by a rotation of angle degrees about the axis(dx, dy, dz). TYPE here is GLfloat.

Syntax:

voidglRotatef(TYPE angle, TYPE dx, TYPE dy, TYPE dz);

**glLoadIdentity Function**

sets the current transformation matrix to an identity matrix.

Syntax:

voidglLoadIdentity();

**glPushMatrix Function**

Pushes to the matrix stack corresponding to the current matrix mode.

Void glPushMatrix();

**glPopMatrix Function**

Pops from the matrix stack corresponding to the current matrix mode.

**gluOrtho2D Function**

Defines a two-dimensional viewing rectangle in the plane z=0.

void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);

**glutSwapBuffers Function**

swaps the front and back buffers.

voidglutSwapBuffers()

**glutInit Function**

Initializes GLUT< the arguments from main are passed in and can by the application.

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

**glutCreateWindow Function**

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

voidglutCreateWindow(char \*title)

**glLightf Function**

glLight — set light source parameters

voidglLightf(GLenum light, GLenumpname, GLfloatparam);

**glMaterialf function**

glMaterial — specify material parameters for the lighting model

voidglMaterialf(GLenum face, GLenumpname, GLfloatparam);

**glutMainLoop Function**

Cause the program to enter an event –processing loop.it should be the statement in main.

voidglutMainLoop**()**

**glutPostRedisplay Function**

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

voidglutPostRedisplay()

**glutMouseFunc Function**

glutMouseFunc sets the mouse callback for the currentwindow.

voidglutMouseFunc(void (\*func)(int button, int state,int x, int y));

**Menu Functions:**

**intglutCreateMenu(void (\*func)(int value));**

glutCreateMenu creates a new pop-up menu.

**voidglutSetMenu(int menu);**

glutSetMenu sets the current menu*;*

**intglutGetMenu(void);**

glutGetMenu returns the identifier of the current menu*.*

**voidglutAddSubMenu(char \*name, int menu);**

glutAddSubMenu adds a sub-menu trigger to the bottom of the current menu.

**voidglutAttachMenu(int button);**

The button to attach a menu**.**

**glPloygonMode function:**

glPolygonMode — select a polygon rasterization mode

voidglPolygonMode( GLenumface,GLenum mode);

**Chapter 5**

**SOURCE CODE**

/\*Code implements the highlighting of objects\*/

#include <stdlib.h>

#include <stdarg.h>

#include <stdio.h>

#include <GL/glut.h>

#define BUFSIZE 512

#define TORUS 1

#define TETRAHEDRON 2

#define ICOSAHEDRON 3

#define SPHERE 4

enum {

LIGHT\_OFF, LIGHT\_RED, LIGHT\_WHITE, LIGHT\_GREEN

} LightValues;

GLfloat red\_light[] =

{1.0, 0.0, 0.0, 1.0}, green\_light[] =

{0.0, 1.0, 0.0, 1.0}, white\_light[] =

{1.0, 1.0, 1.0, 1.0};

GLfloat left\_light\_position[] =

{-1.0, 0.0, 1.0, 0.0}, right\_light\_position[] =

{1.0, 0.0, 1.0, 0.0};

GLuint selectBuf[BUFSIZE];

int W = 500, H = 500;

GLfloat x, y;

int locating = 0;

int theObject = 0;

int menu\_inuse = 0;

int mouse\_state = 0;

char \*objectNames[] =

{"Nothing", "Torus", "Tetrahedron", "Icosahedron","sphere"};

int shade\_model = GL\_SMOOTH;

char \*left\_light, \*right\_light;

char \*ico\_material, \*sphere\_material, \*torus\_material;

void output(GLfloat x, GLfloat y, char \*format,...)

{

va\_list args;

char buffer[200], \*p;

va\_start(args, format);

vsprintf(buffer, format, args);

va\_end(args);

glPushMatrix();

glTranslatef(x, y, 0);

for (p = buffer; \*p; p++)

glutStrokeCharacter(GLUT\_STROKE\_ROMAN, \*p);

glPopMatrix();

}

/\* Initialize material property and light source. \*/

void myinit(void)

{

GLfloat light\_ambient[] =

{0.2, 0.2, 0.2, 1.0};

GLfloat light\_diffuse[] =

{1.0, 1.0, 1.0, 1.0};

GLfloat light\_specular[] =

{1.0, 1.0, 1.0, 1.0};

GLfloat light\_position[] =

{1.0, 1.0, 1.0, 0.0};

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient);

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular);

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position);

glEnable(GL\_LIGHT0);

glDepthFunc(GL\_LESS);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_LIGHTING);

glLightfv(GL\_LIGHT0, GL\_POSITION, left\_light\_position);

glLightfv(GL\_LIGHT0, GL\_SPECULAR, white\_light);

glLightfv(GL\_LIGHT1, GL\_POSITION, right\_light\_position);

glLightfv(GL\_LIGHT1, GL\_SPECULAR, white\_light);

glEnable(GL\_LIGHTING);

glEnable(GL\_DEPTH\_TEST);

glEnable(GL\_NORMALIZE);

glLineWidth(1.0);

glMatrixMode(GL\_PROJECTION);

gluPerspective( /\* degrees field of view \*/ 50.0,

/\* aspect ratio \*/ 1.0, /\* Z near \*/ 1.0, /\* Z far \*/ 10.0);

glMatrixMode(GL\_MODELVIEW);

gluLookAt(0.0, 0.0, 5.0, /\* eye is at (0,0,5) \*/

0.0, 0.0, 0.0, /\* center is at (0,0,0) \*/

0.0, 1.0, 0.); /\* up is in positive Y direction \*/

glTranslatef(0.0, 0.0, -1.0);

glSelectBuffer(BUFSIZE, selectBuf);

glNewList(TORUS, GL\_COMPILE);

glutSolidTorus(0.275, 0.85, 10, 15);

glEndList();

glNewList(TETRAHEDRON, GL\_COMPILE);

glutSolidTetrahedron();

glEndList();

glNewList(ICOSAHEDRON, GL\_COMPILE);

glutSolidIcosahedron();

glEndList();

glNewList(SPHERE,GL\_COMPILE);

glutSolidSphere(0.75,25,25);

glEndList();

}

void highlightBegin(void)

{

static GLfloat blue[4] =

{0,1,1,1};

glPushAttrib(GL\_LIGHTING\_BIT | GL\_CURRENT\_BIT);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, blue);

glColor3f(0,0,1);

}

void highlightEnd(void)

{

glPopAttrib();

}

void draw(void)

{

glPushMatrix();

glScalef(1.3, 1.3, 1.3);

glRotatef(20.0, 1.0, 0.0, 0.0);

glLoadName(2);

glPushMatrix();

if (theObject == 2)

highlightBegin();

glTranslatef(-0.75, -0.75, 0.0);

glRotatef(79.0, 1.0, 1.0, 0.0);

glCallList(TETRAHEDRON);

if (theObject == 2)

highlightEnd();

glPopMatrix();

glLoadName(1);

glPushMatrix();

if (theObject == 1)

highlightBegin();

glTranslatef(-0.75, 0.5, 0.0);

glRotatef(90.0, 1.0, 0.0, 0.0);

glCallList(TORUS);

if (theObject == 1)

highlightEnd();

glPopMatrix();

glLoadName(3);

glPushMatrix();

if (theObject == 3)

highlightBegin();

glTranslatef(0.75, 0.0, -1.0);

glCallList(ICOSAHEDRON);

if (theObject == 3)

highlightEnd();

glPopMatrix();

glLoadName(4);

glPushMatrix();

if (theObject == 4)

highlightBegin();

glTranslatef(1, -1.85, -1.5);

glCallList(SPHERE);

if (theObject == 4)

highlightEnd();

glPopMatrix();

glPopMatrix();

glPushAttrib(GL\_ENABLE\_BIT);

glDisable(GL\_DEPTH\_TEST);

glDisable(GL\_LIGHTING);

glMatrixMode(GL\_PROJECTION);

glPushMatrix();

glLoadIdentity();

gluOrtho2D(0, 3000, 0, 3000);

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glLoadIdentity();

glPopMatrix();

glMatrixMode(GL\_PROJECTION);

glPopMatrix();

glPopAttrib();

glutSwapBuffers();

}

void myortho(void)

{

if (W <= H)

glOrtho(-2.5, 2.5, -2.5 \* (GLfloat) H / (GLfloat) W,

2.5 \* (GLfloat) H / (GLfloat) W, -10.0, 10.0);

else

glOrtho(-2.5 \* (GLfloat) W / (GLfloat) H,

2.5 \* (GLfloat) W / (GLfloat) H, -2.5, 2.5, -10.0, 10.0);

}

/\* processHits() prints out the contents of the

\* selection array.

\*/

void processHits(GLint hits, GLuint buffer[])

{

GLuint depth = ~0;

unsigned int i, getThisName;

GLuint names, \*ptr;

GLuint newObject;

ptr = (GLuint \*) buffer;

newObject = 0;

for (i = 0; i < hits; i++) { /\* for each hit \*/

getThisName = 0;

names = \*ptr;

ptr++; /\* skip # name \*/

if (\*ptr <= depth) {

depth = \*ptr;

getThisName = 1;

}

ptr++; /\* skip z1 \*/

if (\*ptr <= depth) {

depth = \*ptr;

getThisName = 1;

}

ptr++; /\* skip z2 \*/

if (getThisName)

newObject = \*ptr;

ptr += names; /\* skip the names list \*/

}

if (theObject != newObject) {

theObject = newObject;

glutPostRedisplay();

}

}

/\* ARGSUSED \*/

void locate(int value)

{

GLint viewport[4];

GLint hits;

if (locating) {

if (mouse\_state == GLUT\_ENTERED) {

(void) glRenderMode(GL\_SELECT);

glInitNames();

glPushName(-1);

glMatrixMode(GL\_PROJECTION);

glPushMatrix();

glLoadIdentity();

viewport[0] = 0;

viewport[1] = 0;

viewport[2] = W;

viewport[3] = H;

gluPickMatrix(x, H - y, 5.0, 5.0, viewport);

myortho();

glMatrixMode(GL\_MODELVIEW);

draw();

glMatrixMode(GL\_PROJECTION);

glPopMatrix();

glMatrixMode(GL\_MODELVIEW);

hits = glRenderMode(GL\_RENDER);

} else {

hits = 0;

}

processHits(hits, selectBuf);

}

locating = 0;

}

void passive(int newx, int newy)

{

x = newx;

y = newy;

if (!locating) {

locating = 1;

glutTimerFunc(1, locate, 0);

}

}

void entry(int state)

{

mouse\_state = state;

if (!menu\_inuse) {

if (state == GLUT\_LEFT) {

if (theObject != 0) {

theObject = 0;

glutPostRedisplay();

}

}

}

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

draw();

glPushAttrib(GL\_ENABLE\_BIT);

glDisable(GL\_DEPTH\_TEST);

glDisable(GL\_LIGHTING);

glDisable(GL\_LINE\_SMOOTH);

glMatrixMode(GL\_PROJECTION);

glPushMatrix();

glLoadIdentity();

gluOrtho2D(0, 3000, 0, 3000);

glMatrixMode(GL\_MODELVIEW);

glPushMatrix();

glLoadIdentity();

output(80, 2800, "Automatically names object under mouse");

output(80, 100, "Located: %s.", objectNames[theObject]);

glPopMatrix();

glMatrixMode(GL\_PROJECTION);

glPopMatrix();

glMatrixMode(GL\_MODELVIEW);

glPopAttrib();

glutSwapBuffers();

}

void myReshape(int w, int h)

{

W = w;

H = h;

glViewport(0, 0, W, H);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

myortho();

glMatrixMode(GL\_MODELVIEW);

}

void polygon\_mode(int value)

{

switch (value) {

case 1:

glEnable(GL\_LIGHTING);

glDisable(GL\_BLEND);

glEnable(GL\_DEPTH\_TEST);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

break;

case 2:

glDisable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 1.0);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

glEnable(GL\_LINE\_SMOOTH);

glEnable(GL\_BLEND);

glDisable(GL\_DEPTH\_TEST);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

break;

case 3:

glDisable(GL\_LIGHTING);

glColor3f(1.0, 1.0, 1.0);

glPointSize(2);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_POINT);

glEnable(GL\_POINT\_SMOOTH);

glEnable(GL\_BLEND);

glDisable(GL\_DEPTH\_TEST);

glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA);

break;

}

glutPostRedisplay();

}

void mstatus(int status, int newx, int newy)

{

if (status == GLUT\_MENU\_NOT\_IN\_USE) {

menu\_inuse = 0;

passive(newx, newy);

} else {

menu\_inuse = 1;

}

}

void light\_select(GLenum which, int value, char \*\*label)

{

glEnable(which);

switch (value) {

case LIGHT\_OFF:

\*label = "off";

glDisable(which);

break;

case LIGHT\_RED:

\*label = "red";

glLightfv(which, GL\_DIFFUSE, red\_light);

break;

case LIGHT\_WHITE:

\*label = "white";

glLightfv(which, GL\_DIFFUSE, white\_light);

break;

case LIGHT\_GREEN:

\*label = "green";

glLightfv(which, GL\_DIFFUSE, green\_light);

break;

}

glutPostRedisplay();

}

void left\_light\_select(int value)

{

light\_select(GL\_LIGHT0, value, &left\_light);

}

void right\_light\_select(int value)

{

light\_select(GL\_LIGHT1, value, &right\_light);

}

void material(int dlist, GLfloat \* ambient, GLfloat \* diffuse,

GLfloat \* specular, GLfloat shininess)

{

glNewList(dlist, GL\_COMPILE);

glMaterialfv(GL\_FRONT, GL\_AMBIENT, ambient);

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, diffuse);

glMaterialfv(GL\_FRONT, GL\_SPECULAR, specular);

glMaterialf(GL\_FRONT, GL\_SHININESS, shininess);

glEndList();

}

void main\_menu(int value)

{

if (value == 666)

exit(0);

glShadeModel(shade\_model = value);

glutPostRedisplay();

}

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

{

int submenu;

int left\_light\_m, right\_light\_m, torus\_m, sphere\_m, ico\_m;

glutInit(&argc, argv);

glutInitWindowSize(W, H);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

#define LIGHT\_MENU\_ENTRIES() \

glutAddMenuEntry("Disable", LIGHT\_OFF); \

glutAddMenuEntry("Red", LIGHT\_RED); \

glutAddMenuEntry("White", LIGHT\_WHITE); \

glutAddMenuEntry("Green", LIGHT\_GREEN);

glutCreateWindow("hover");

myinit();

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

left\_light\_m = glutCreateMenu(left\_light\_select);

LIGHT\_MENU\_ENTRIES();

right\_light\_m = glutCreateMenu(right\_light\_select);

LIGHT\_MENU\_ENTRIES();

submenu = glutCreateMenu(polygon\_mode);

glutAddMenuEntry("Filled", 1);

glutAddMenuEntry("Outline", 2);

glutAddMenuEntry("Points",3);

glutCreateMenu(main\_menu);

glutAddMenuEntry("Quit", 666);

glutAddSubMenu("Polygon mode", submenu);

glutAddMenuEntry("Smooth shading", GL\_SMOOTH);

glutAddMenuEntry("Flat shading", GL\_FLAT);

glutAddSubMenu("Left light", left\_light\_m);

glutAddSubMenu("Right light", right\_light\_m);

glutAttachMenu(GLUT\_RIGHT\_BUTTON);

glutPassiveMotionFunc(passive);

glutEntryFunc(entry);

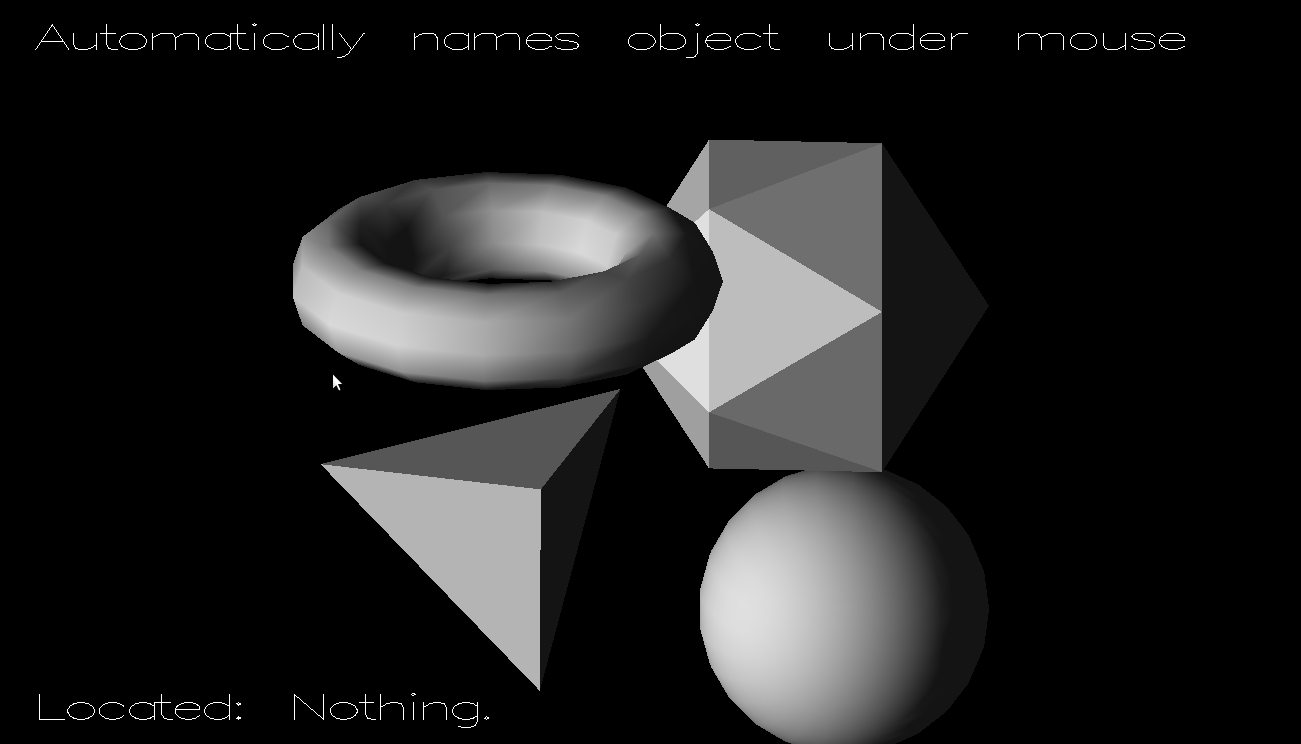
glutMenuStatusFunc(mstatus);

glutMainLoop();

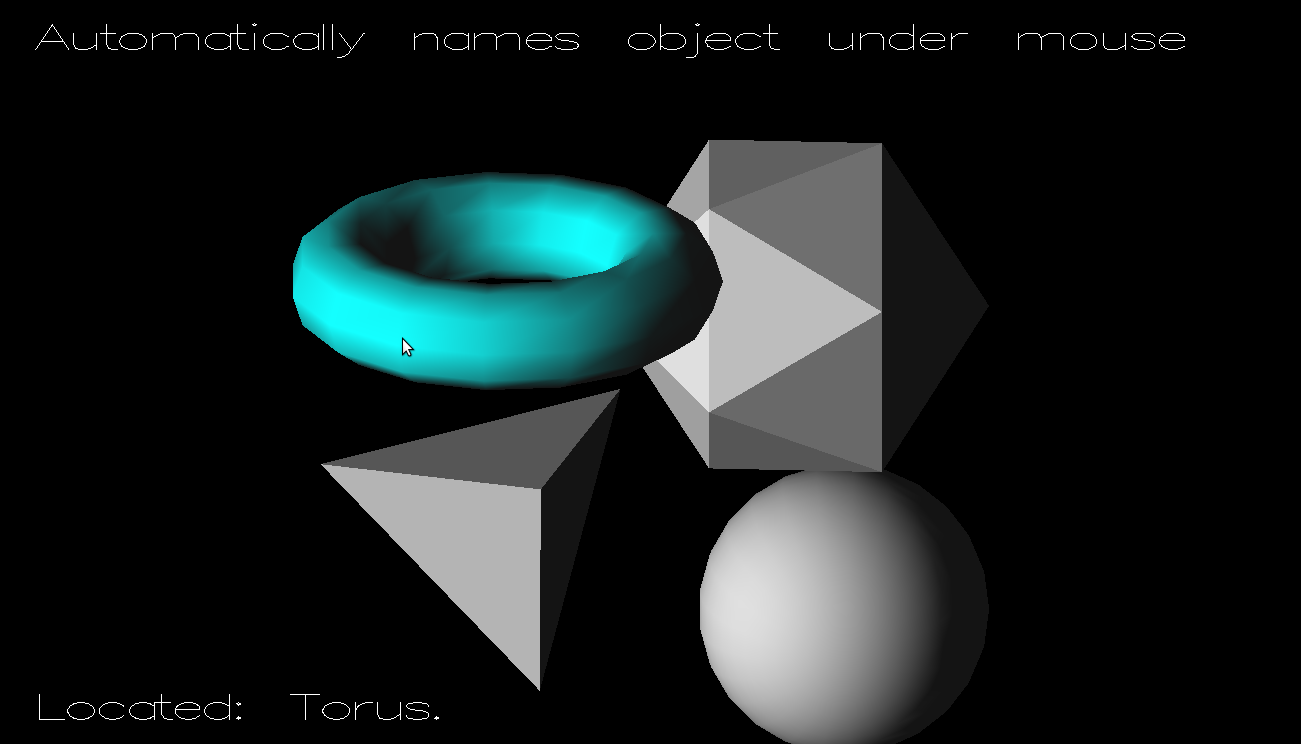
return 0;

}

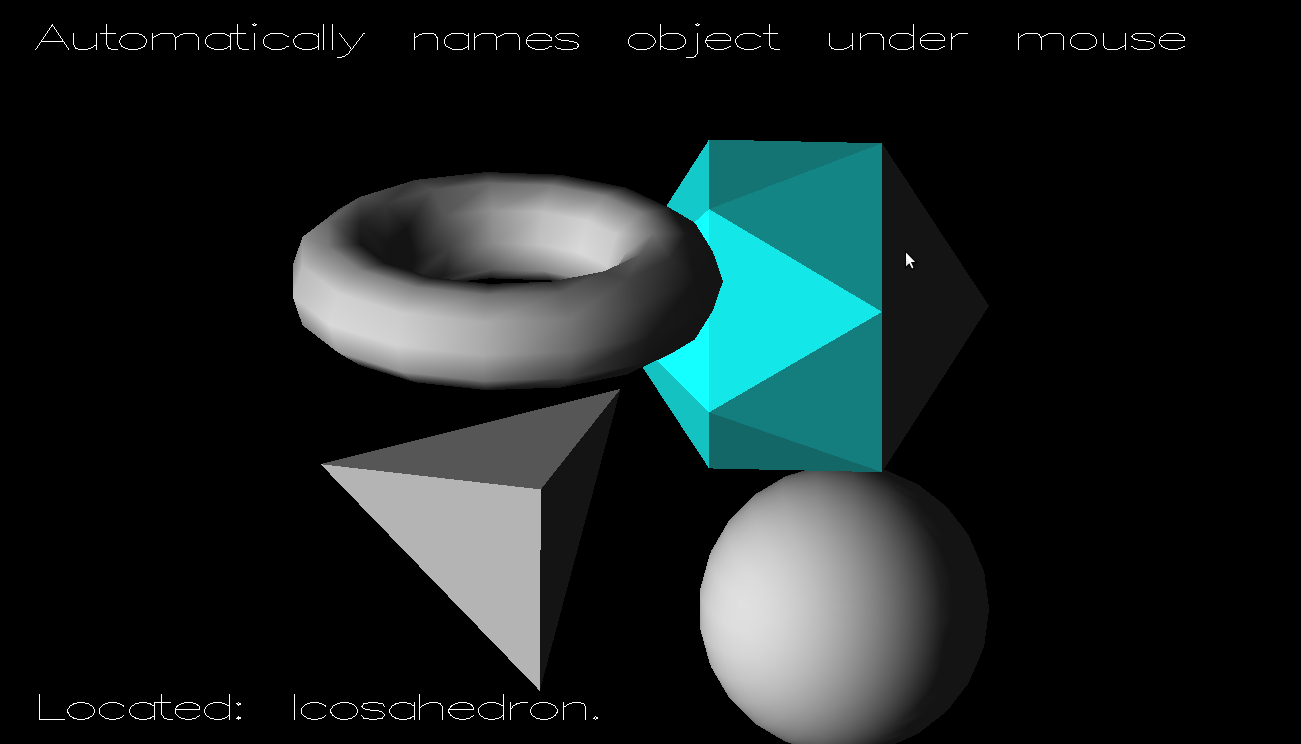
**Chapter 6**

**SNAP SHOTS**

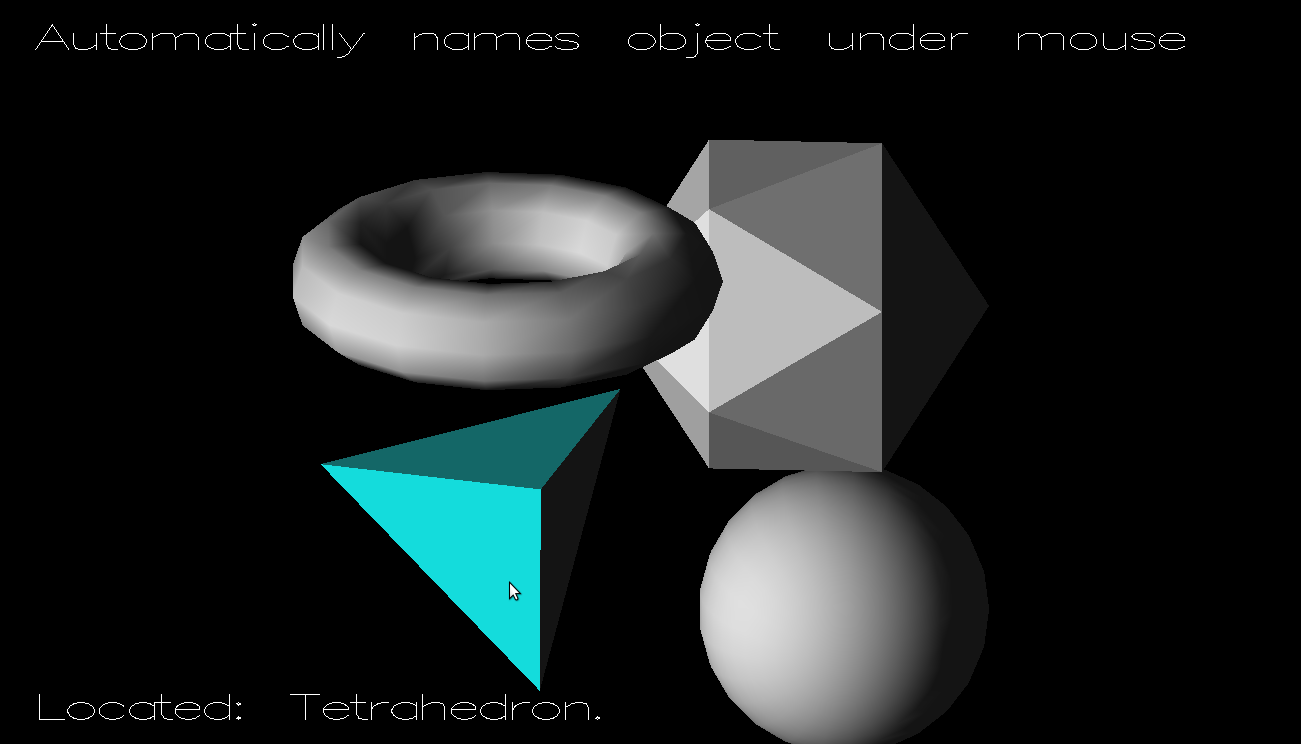
**Figure 6.1:Location of the cursor outside of the objects**

****

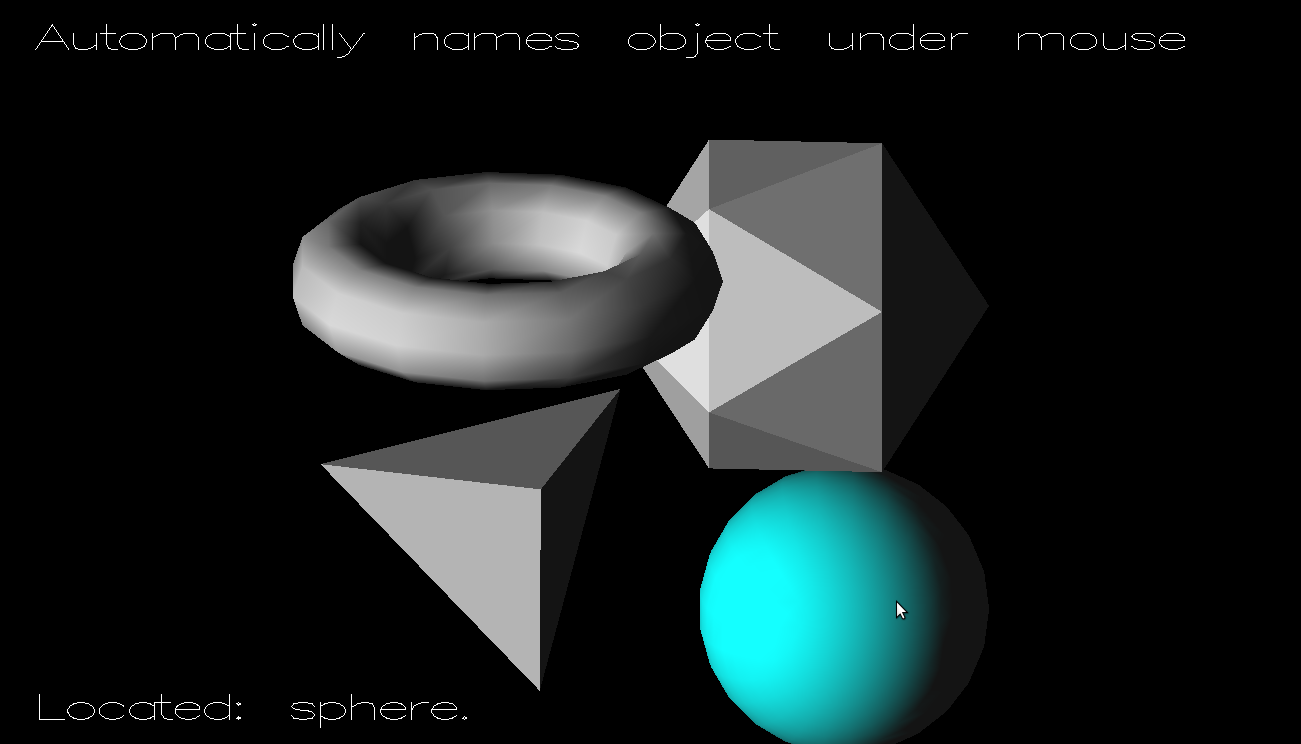
**Figure 6.2: Navigation of cursor on Torus Object and its highlighting.**

****

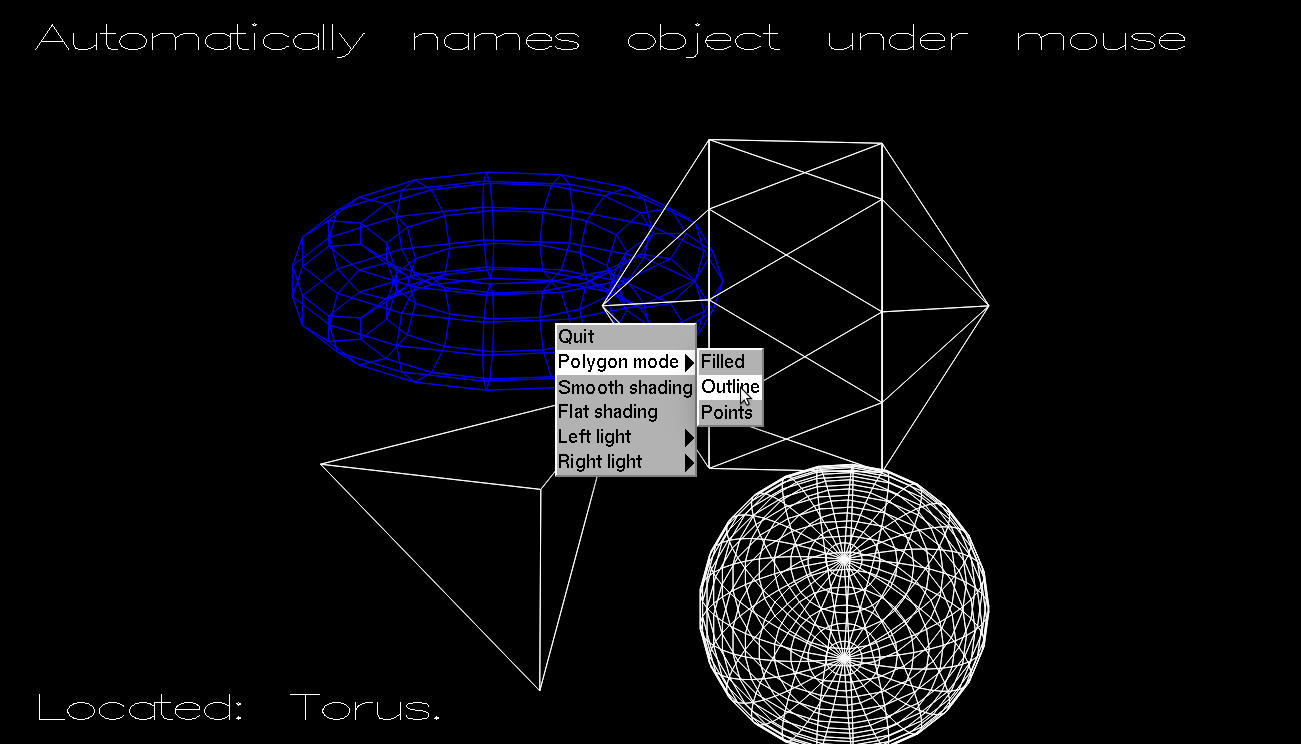
**Figure 6.3: Navigation of cursor on Icosahedron Object and its highlighting.**

****

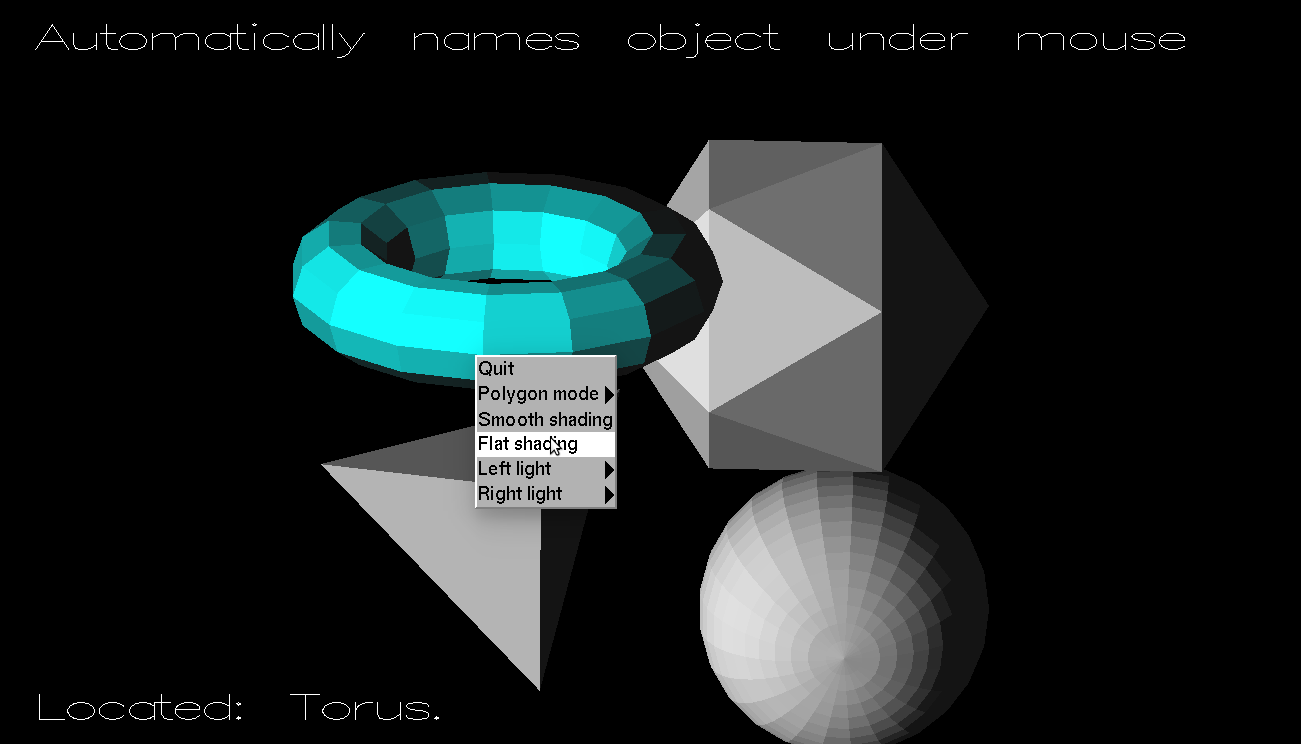
**Figure 6.4:Navigation of cursor on Tetrahedron Object and its highlighting.**

****

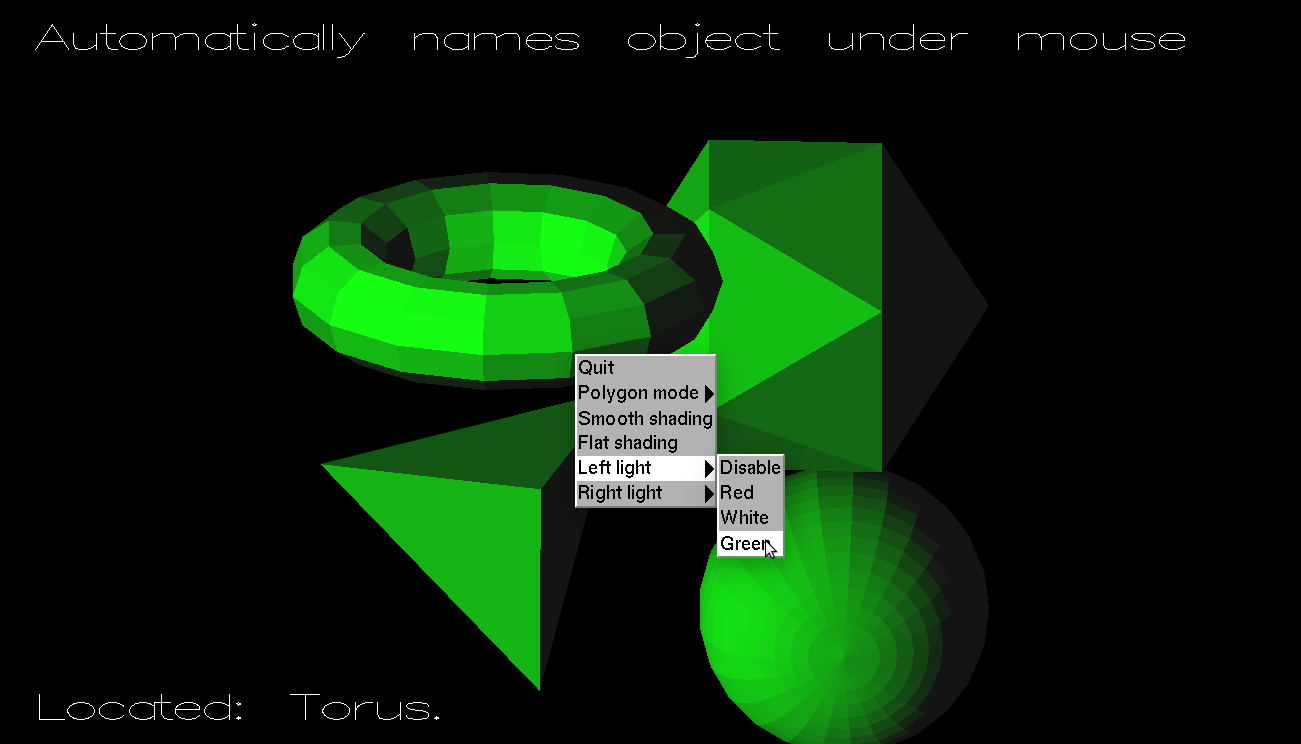
**Figure 6.5: Navigation of cursor on Sphere Object and its highlighting.**

****

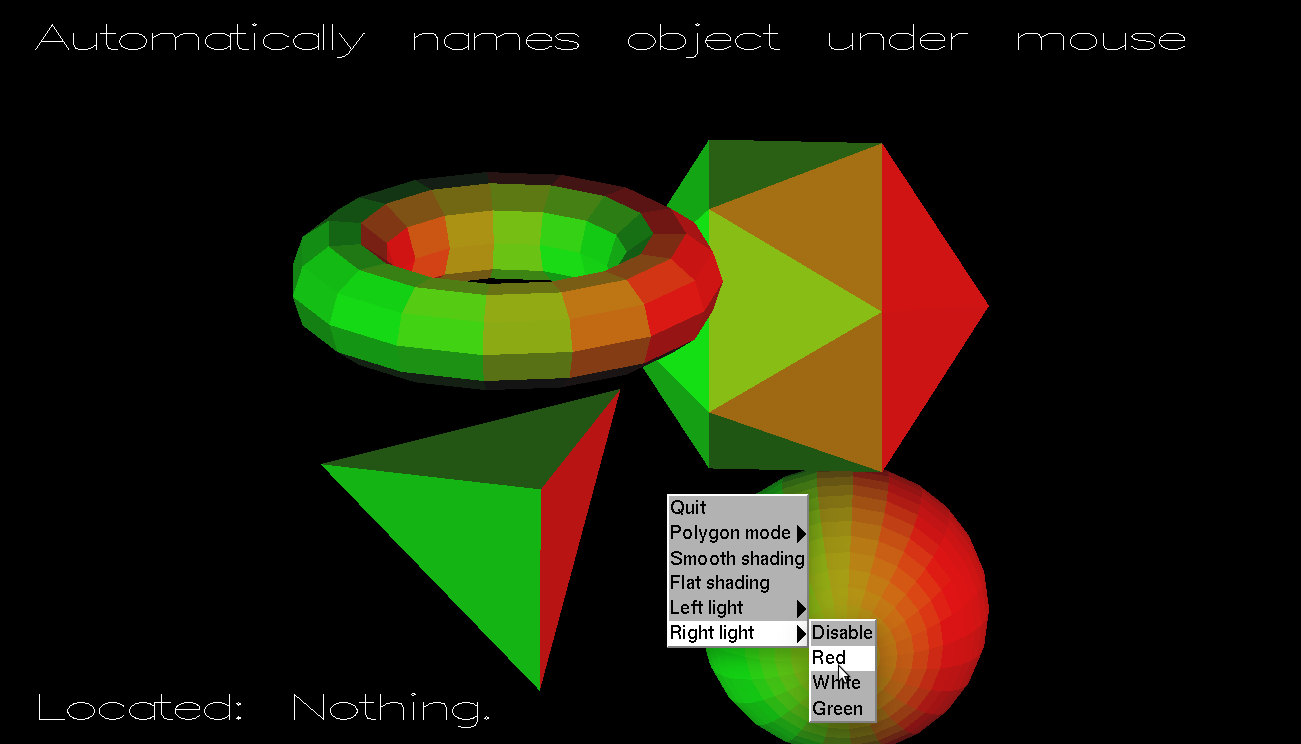
**Figure 6.6:The outlined view of the objects.**

****

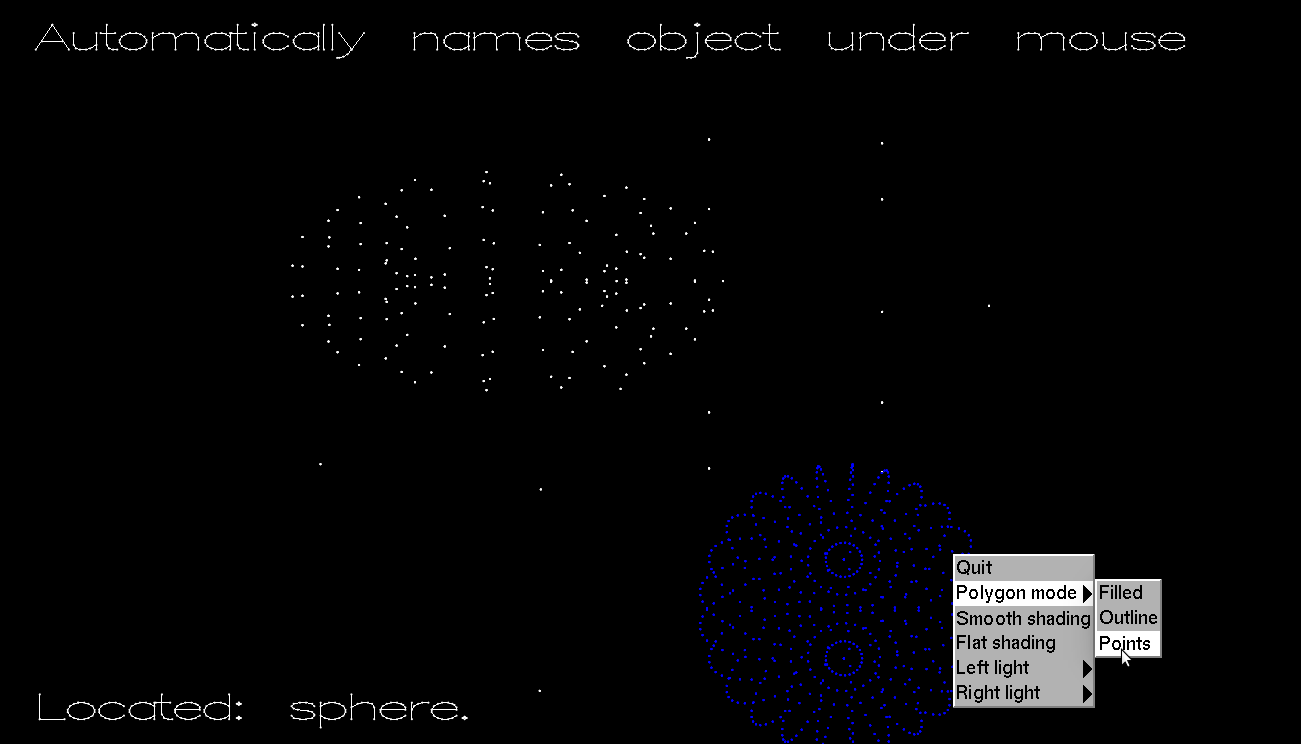
**Figure 6.7 Flat shading of objects.**

****

**Figure 6.8 Green light placed at left side.**

****

**Figure 6.9 Red light from right side.**

****

**Figure 6.10 Point mode of objects.**

**Chapter 7**

**CONCLUSION**

This project clearly explains the use of mouse navigation, lighting, material properties usage in OpenGL.The primary goal of the project is to provide a way to identify 3-D objects just by hovering/navigating the mouse cursor over them.This project also aims at providing a way to display the 3-D objects in different modes like filled and Outline modes.More graphics functions can be used to make it more attractive and self-explanatory.

**BIBLIOGRAPHY**

**Book References:**

* Interactive Computer Graphics: A top-down approach with OpenGL,2nd Edition by Edward Angel
* OpenGL Programming Guide(Addison-Wesley Publishing Company), 2nd Edition

**Website References:**

* <https://www.opengl.org>
* <http://www.mesa3d.org>