**UCS1712 – Graphics and Multimedia Lab**

**Ex-12: Creating a 3D Scene using OpenGL**

**Name:** Satheesh Kumar G R

**Reg. No:** 185001136

**AIM**

To write a C++ program using Opengl to draw at least four 3D objects. Apply lighting and texture and render the scene. Apply transformations to create a simple 3D animation

**CODE:**

#pragma warning(disable : 4996)

#include <GL/glut.h>

#include <GL/glu.h>

#include <stdlib.h>

#include <stdio.h>

int INC = 1;

void initialize(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0); glShadeModel(GL\_SMOOTH);

GLfloat light\_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };

GLfloat light\_position[] = { 0, 0, 1, 0 }; glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse); glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position); glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT; glEnable(GL\_DEPTH\_TEST);

}

GLuint LoadTexture(const char\* filename)

{

GLuint texture; int width, height;

unsigned char\* data;

FILE\* file;

file = fopen(filename, "rb");

if (file == NULL) return 0; width = 1024;

height = 512;

data = (unsigned char\*)malloc(width \* height \* 3);

//int size = fseek(file,);

fread(data, width \* height \* 3, 1, file); fclose(file);

for (int i = 0; i < width \* height; ++i)

{

int index = i \* 3; unsigned char B, R; B = data[index];

R = data[index + 2];

data[index] = R; data[index + 2] = B;

}

glGenTextures(1, &texture);

glBindTexture(GL\_TEXTURE\_2D, texture); glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR\_MIPMAP\_NEAREST);

data);

}

glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR); glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT); glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT); gluBuild2DMipmaps(GL\_TEXTURE\_2D, 3, width, height, GL\_RGB, GL\_UNSIGNED\_BYTE,

free(data); return texture;

void drawScene(int state)

{

if (state == 0)

INC = 1;

else if (state == 10) INC = -1;

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

GLuint texture;

texture = LoadTexture("silver.bmp"); glBindTexture(GL\_TEXTURE\_2D, texture); glLoadIdentity();

gluLookAt(0.0, 1.0, 7.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glMatrixMode(GL\_MODELVIEW);

// Cube glPushMatrix();

GLfloat cube\_color[] = { 1, 0.5, 0.0, 0.0 }; glMaterialfv(GL\_FRONT, GL\_DIFFUSE, cube\_color); glScalef(4, 1.5, 1.0);

glTranslatef(0.2, -1.0, 0.0); glutSolidCube(1.0); glPopMatrix();

// Torus glPushMatrix();

GLfloat torus\_color[] = { 0.59, 0.1, 0.55, 1.0 }; glMaterialfv(GL\_FRONT, GL\_DIFFUSE, torus\_color); glTranslatef(-3, -1.5, 0.0);

glutSolidTorus(0.3, 0.7, 10, 10); glPopMatrix();

// Teapot glPushMatrix(); glEnable(GL\_TEXTURE\_2D);

GLfloat teapot\_color[] = { 0.7, 0.7, 0.7, 0.0 }; GLfloat mat\_shininess[] = { 100 }; glMaterialfv(GL\_FRONT, GL\_DIFFUSE, teapot\_color); glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess); glRotatef(45, 0, 0, 1);

glTranslatef(-1.2, 0.8, 0.0); glutSolidTeapot(0.7); glDisable(GL\_TEXTURE\_2D); glPopMatrix();

// Sphere glPushMatrix();

GLfloat ball\_color[] = { 0.0, 1, 1, 1.0 }; glMaterialfv(GL\_FRONT, GL\_DIFFUSE, ball\_color); glTranslatef(2, 2.1 - 0.25 \* state, 0);

glutSolidSphere(0.5, 10, 10); glPopMatrix();

glutSwapBuffers();

glutTimerFunc(1000 / 60, drawScene, state + INC);

}

void reshape(int w, int h)

{

glViewport(0, 0, (GLsizei)w, (GLsizei)h); glMatrixMode(GL\_PROJECTION); glLoadIdentity();

gluPerspective(75, 1, 1, 20); glMatrixMode(GL\_MODELVIEW);

}

void sceneDemo() {

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

glutTimerFunc(1000 / 60, drawScene, 0);

}

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

{

glutInit(&argc, argv); glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB); glutInitWindowSize(500, 500); glutCreateWindow("3D Scene");

initialize(); glutDisplayFunc(sceneDemo); glutReshapeFunc(reshape); glutMainLoop();

return 0;

}

**OUTPUT:**





