EX-10 - Creating a 3D Scene in C++ using OpenGL

26/10/2021

Venkataraman Nagarajan, CSE - C
18500192

AIM

To create a 3D scene in C++.

SPECIFICATION

Write a C++ program using Opengl to draw atleast four 3D objects. Apply lighting and texture and render the scene. Apply transformations to create a simple 3D animation. [*Use built-in transformation functions*]

OpenGL Functions to use:

- $glShadeModel()$
- $glMaterialfv()$
- $glLightfv()$
- $glEnable()$
- $glGenTextures()$
- $glTexEnvf()$
- $glBindTexture()$
- $glTexParameteri()$
- $glTexCoord2f()$

PROGRAM - 01

3D Projections

```
#pragma warning(disable : 4996)
#include <GL/glut.h>
#include <GL/glu.h>
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <errno.h>

int INC = 1;

const int WINDOW_WIDTH = 500;
const int WINDOW_HEIGHT = 500;


void myInit() {
    glClearColor(1, 1, 1, 1.0);
    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_LIGHT0);
    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) {
        std::cout << errno << "\n";
        return 0;
    }

    width = 474;
    height = 395;
    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);
    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, data);
    free(data);
    std::cout << texture << "\n";
    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);

    // "Silver" texture will be applied to the teapot alone
    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[] = {0.46, 0.26, 0.2, 1.0};
    glMaterialfv(GL_FRONT, GL_DIFFUSE, cube_color);
    glScalef(4, 1.5, 1.0);
    glTranslatef(0.4, -1.0, 0.0);
    glutSolidCube(1.0);
    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);
    glTranslatef(1.7, -0.2, 0.0);
    glutSolidTeapot(0.7);
    glDisable(GL_TEXTURE_2D);
    glPopMatrix();

    // Ramp
    glPushMatrix();
    GLfloat ramp_color[] = {0.6, 0.44, 0.39, 1.0};
    glMaterialfv(GL_FRONT, GL_DIFFUSE, ramp_color);
    glRotatef(45, 0, 0, 1);
    glTranslatef(-1.2, -0.2, 0);
    glScalef(3.4, 0.2, 1.0);
    glutSolidCube(1.0);
    glPopMatrix();

    // Sphere
    glPushMatrix();
    GLfloat ball_color[] = {0.59, 0.1, 0.55, 1.0};
    glMaterialfv(GL_FRONT, GL_DIFFUSE, ball_color);
    glRotatef(-0.1 * state, 0, 0, 1);
    glTranslatef(-2.5 - 0.25 * state, -2, 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(WINDOW_WIDTH, WINDOW_HEIGHT);
    glutCreateWindow("3D Scene");
    myInit();
    glutDisplayFunc(sceneDemo);
    glutReshapeFunc(reshape);
    glutMainLoop();
    return 0;
}
```

SAMPLE I/0

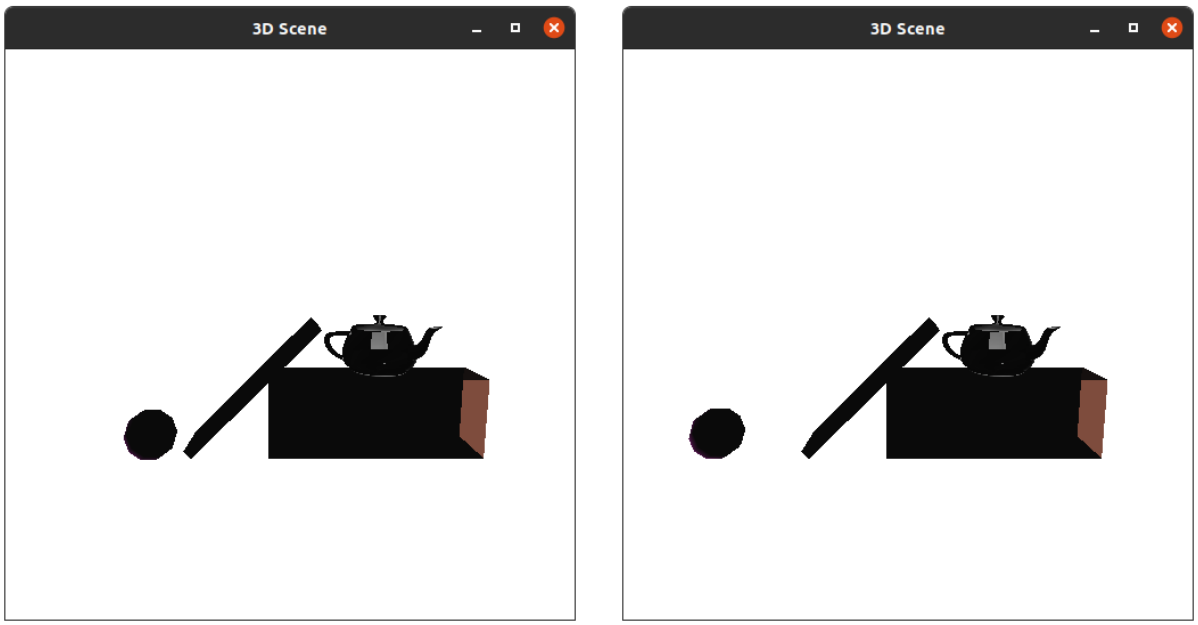

Figure 1: Animated version of balling rolling in the scene

RESULT

The code to create a 3d scene is written and output is verified.
