Ex. No: 10
Date : 10/11/2022

Karthik D
195001047

Creating a 3D Scene using OpenGL

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

Algorithm

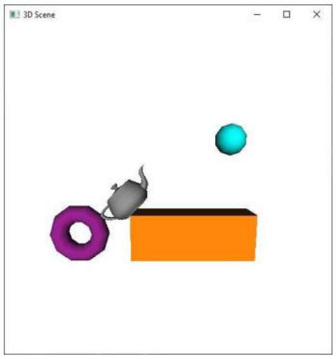
- Use glBindTexture to bind the texture to the 3d object
- Use glShadeMode select flat or smooth shading.
- The glMaterialfv function assigns values to material parameters
- The glLightfv function sets the value or values of individual light source parameters.
- glGenTexture specifies an array in which the generated texture names are stored

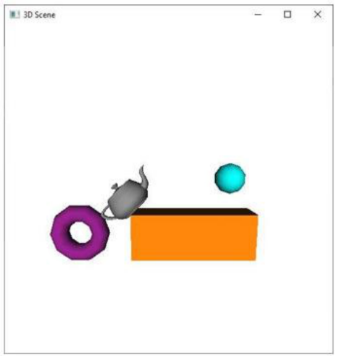
Program

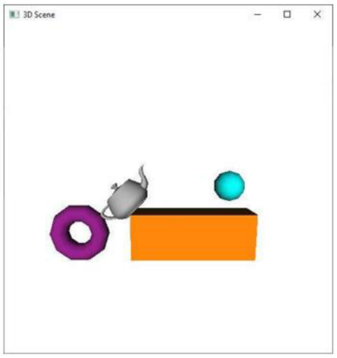
ex10.cpp

```
#include <iostream>
#include <cstring>
#include <GL/glut.h>
#include <math.h>

using namespace std;

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_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)
        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);

    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);
    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.jpg");
    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);
    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();
    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();
    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();
    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);
}

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







Learning Outcomes

- Learned to apply textures, lighting 3d objects
- Learned to create a scene using three 3d objects