EXERCISE - 10: CREATING A 3D SCENE IN C++ USING OPENGL

AIM:
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]

ALGORITHM:

```
 1.   Set ammbience, diffuse, specular and shininess values of material
      and light source
 2.   Enable lighting so that the renderer can see light, turn LIGHT0
      on, enable depth test for rendering depth and enable texture
 3.   Initialise frame buffer using glutInit()
 4.   Set display mode as single buffer for 2D graphics with RGB colour
      using glutInitDisplayMode()
 5.   Set output window size as 640, 640 pixels using glutWindowSize()
 6.   Create the output window using glutCreateWindow()
 7.   Call function to draw using glutDisplayFunc()
 8.   Set visibility of faces using depth test parameter using
      glEnable()
 9.   In the display function
     9.1.   Set background colour (RGB, opacity) as white using
            glClear()
     9.2.   Clear frame buffer using  glClear()
     9.3.   Set matrix mode to manipulate matrix values using
            glMatrixMode()
     9.4.   Load identity matrix using glLoadIdentity()
     9.5.   Set camera position
     9.6.   Set background color and matrixmode
     9.7.   Push matrix
     9.8.   Enable and disable GL_TEXTURE_2D
     9.9.   Add objects to the animation by pushing matrix, setting
            color and shininess using glMaterialfv()
    9.10.   Set transformations
    9.11.   Create the object
    9.12.   Pop the matrix
    9.13.   Repeat the same for the other
    9.14.   End using glEnd()
    9.15.   Flush frame buffer using glFlush()
```

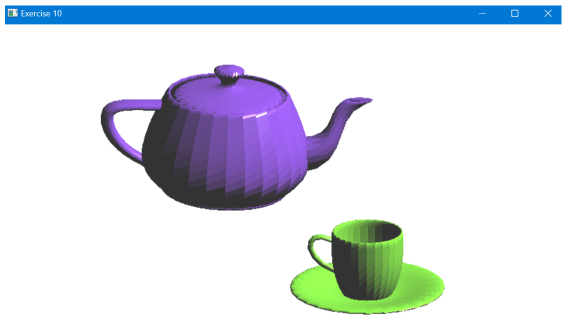
10. Refresh the screen repeatedly while calling the function to draw using glutMainLoop()

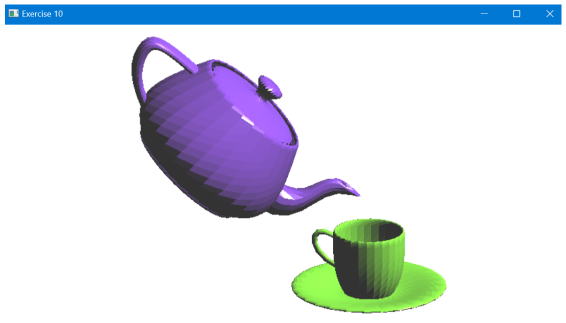
CODE:
```
#include <gl/freeglut.h>
#include <Windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>

GLfloat black[] = { 0.0, 0.0, 0.0, 1.0 };
GLfloat white[] = { 1.0, 1.0, 1.0, 1.0 };
GLfloat direction[] = { 1.0, 1.0, 1.0, 0.0 };


float teapot_rotate = 0.2, teapot_rotate_direction = 1, teapot_posx =
-0.5, teapot_posy = 1.0, teapot_xplace = 0, teapot_yplace = 0;
float teaspoon_posx = 0.75, teaspoon_posy = 2.5, teaspoon_yplace = 0;
float sugar1_posx = 0.65, sugar1_posy = 2.5, sugar2_posx = 0.8,
sugar2_posy = 2.75, sugar1_yplace = 0, sugar2_yplace = 0;
float teacup_rotate = 0;
#define red {0xff, 0x00, 0x00}
#define magenta {0xff, 0, 0xff}
GLubyte texture[][3] = {
    red, magenta,
    magenta, red,
};

void display() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);

    glPushMatrix();

    glEnable(GL_TEXTURE_2D);
    glDisable(GL_TEXTURE_2D);

    // Add a teacup to the scene.
    glPushMatrix();
    GLfloat teacup_color[] = { 0.482, 1, 0.161, 0.0 };
    GLfloat teacup_mat_shininess[] = { 100 };
    glMaterialfv(GL_FRONT, GL_DIFFUSE, teacup_color);
    glMaterialfv(GL_FRONT, GL_SHININESS, teacup_mat_shininess);
    glTranslatef(0.75, -0.25, 0.0);
    glRotatef(teacup_rotate, 0, 1, 0);
    glutSolidTeacup(1.0);
    glPopMatrix();

    // Add a teapot to the scene.
    glPushMatrix();
    GLfloat teapot_color[] = { 0.486, 0.212, 0.871, 0.0 };
    GLfloat teapot_mat_shininess[] = { 100 };
    glMaterialfv(GL_FRONT, GL_DIFFUSE, teapot_color);
    glMaterialfv(GL_FRONT, GL_SHININESS, teapot_mat_shininess);
    glTranslatef(teapot_posx, teapot_posy, 0.0);
    glRotatef(teapot_rotate, 0, 0, 1);
    glutSolidTeapot(0.75);
    glPopMatrix();

    // Add a sugar cubes to the scene.
    GLfloat sugar_color[] = { 1, 1, 1, 0.0 };
    GLfloat sugar_mat_shininess[] = { 50 };

    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_DIFFUSE, sugar_color);
    glMaterialfv(GL_FRONT, GL_SHININESS, sugar_mat_shininess);
    glTranslatef(sugar1_posx, sugar1_posy, 0.0);
    glRotatef(-45.0, 0, 0, 1);
    glutSolidCube(0.1);
    glPopMatrix();

    glPushMatrix();
    glMaterialfv(GL_FRONT, GL_DIFFUSE, sugar_color);
    glMaterialfv(GL_FRONT, GL_SHININESS, sugar_mat_shininess);
    glTranslatef(sugar2_posx, sugar2_posy, 0.0);
    glRotatef(45.0, 0, 0, 1);
    glutSolidCube(0.1);
    glPopMatrix();

    // Add a teaspoon to the scene.
```

```
    glPushMatrix();
    GLfloat teaspoon_color[] = { 0.2, 0.2, 0.2, 0.0 };
    GLfloat teaspoon_mat_shininess[] = { 100 };
    glMaterialfv(GL_FRONT, GL_DIFFUSE, teaspoon_color);
    glMaterialfv(GL_FRONT, GL_SHININESS, teaspoon_mat_shininess);
    glTranslatef(teaspoon_posx, teaspoon_posy, 0.0);
    glRotatef(135, 0, 1, 0);
    glRotatef(-60, 1, 0, 0);
    glutSolidTeaspoon(1.25);
    glPopMatrix();


    if (teapot_rotate_direction == 1 && teapot_rotate > -45.0)
teapot_rotate -= 0.5;
    if (teapot_rotate_direction == 1 && teapot_rotate <= -45.0)
teapot_rotate_direction = -1;
    if (teapot_rotate_direction == -1 && teapot_rotate < 0)
teapot_rotate += 0.5;
    if (teapot_rotate_direction == -1 && teapot_rotate >= 0)
teapot_rotate_direction = 0;

    teacup_rotate -= 0.2;

    if (teapot_rotate_direction == 0) {
        if (teapot_posx > -1.25 && teapot_xplace == 0) teapot_posx -=
0.05;
        if (teapot_posx <= -1.25) teapot_xplace = 1;
        if (teapot_posy > 0 && teapot_yplace == 0) teapot_posy -=
0.05;
        if (teapot_posy <= -1) teapot_yplace = 1;
    }

    if (teapot_rotate_direction == 0) {
        if (sugar1_posy > -0.5 && sugar1_yplace == 0) sugar1_posy -=
0.05;
        if (sugar1_posy <= -0.5) sugar1_yplace = 1;
        if (sugar2_posy > -0.5 && sugar2_yplace == 0) sugar2_posy -=
0.05;
        if (sugar2_posy <= -0.5) sugar2_yplace = 1;
    }
    if (sugar1_yplace == 1 && sugar2_yplace == 1) {
        if (teaspoon_posy > -0.25 && teaspoon_yplace == 0)
teaspoon_posy -= 0.05; //-0.5
        if (teaspoon_posy <= -0.5) teaspoon_yplace = 1;
    }

    glutSwapBuffers();
}

void reshape(GLint w, GLint h) {
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    GLfloat aspect = GLfloat(w) / GLfloat(h);
    glLoadIdentity();
    glOrtho(-2.5, 2.5, -2.5 / aspect, 2.5 / aspect, -10.0, 10.0);
}

void init() {
    glClearColor(1, 1, 1, 1);

    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, white);
    glMaterialfv(GL_FRONT, GL_SPECULAR, white);
    glMaterialf(GL_FRONT, GL_SHININESS, 30);

    glLightfv(GL_LIGHT0, GL_AMBIENT, black);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, white);
    glLightfv(GL_LIGHT0, GL_SPECULAR, white);
    glLightfv(GL_LIGHT0, GL_POSITION, direction);

    glEnable(GL_LIGHTING);                // so the renderer considers
light
    glEnable(GL_LIGHT0);                  // turn LIGHT0 on
    glEnable(GL_DEPTH_TEST);              // so the renderer considers
depth

    glShadeModel(GL_FLAT);

    glEnable(GL_TEXTURE_2D);
    glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
    glTexImage2D(GL_TEXTURE_2D,
        0,                      // level 0
        3,                      // use only R, G, and B components
        2, 2,                   // texture has 2x2 texels
        0,                      // no border
        GL_RGB,                 // texels are in RGB format
        GL_UNSIGNED_BYTE,       // color components are unsigned bytes
        texture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glRotatef(20.0, 1.0, 0.0, 0.0);
}

void sceneDemo(int v)
{
    glutPostRedisplay();
    glutTimerFunc(1000 / 24, sceneDemo, 0);
}

// The usual application statup code.
int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowPosition(80, 80);
    glutInitWindowSize(800, 600);
    glutCreateWindow("Exercise 10");
    glutReshapeFunc(reshape);
    glutDisplayFunc(display);
    glutTimerFunc(1000, sceneDemo, 0);
    init();
    glutMainLoop();
}
```

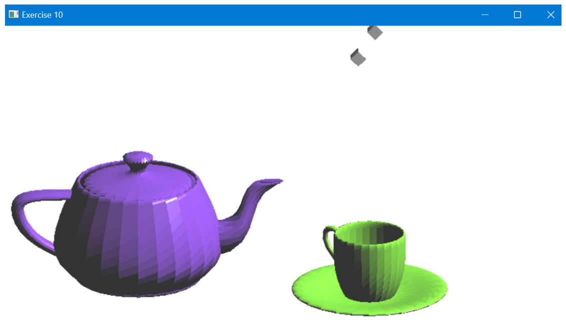
OUTPUT:

Frame 1 **Frame 2**

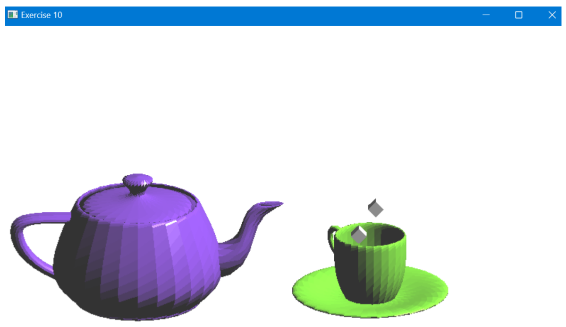




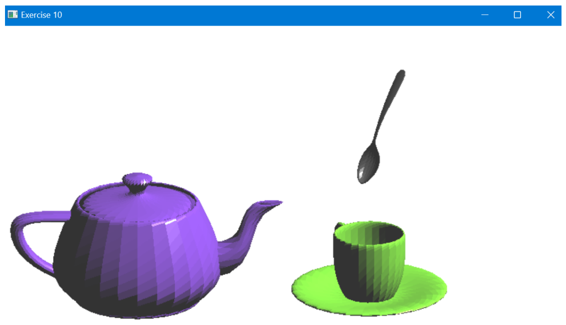
Frame 3



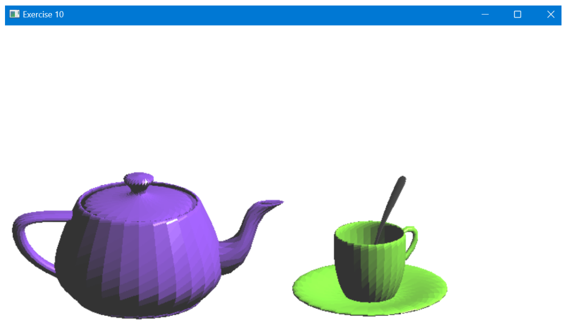
Frame 4



Frame 5



Frame 6



LEARNING OUTCOMES:

1. Learnt to adjust the parameters of the frames
2. Learnt to plot points and mark coordinates
3. Learnt to plot points in 3-Dimensions
4. Learnt to animate object in 3D
5. Learnt to add interactions between object