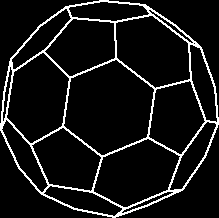


| **TITLE**: Write a program to draw “Buckyball” using openGL library. |
| --- |

**AIM:**

Write a program to draw “Buckyball” using openGL library.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

You should see a rotating Buckyball rendered in 3D on your screen, with different colors for each face.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* **https://www.opengl.org/**
* **https://learnopengl.com/**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Algorithm/ Pseudocode for each process:**

**Initialize OpenGL:**

**· Set up the display mode.**

**· Initialize the window using glutCreateWindow().**

**· Set up the viewport, projection, and modelview matrices.**

**· Define Vertices for Buckyball:**

**· A Buckyball can be represented using a truncated icosahedron, which consists of 60 vertices and 32 faces (12 pentagonal and 20 hexagonal).**

**· Draw the Buckyball:**

**· Use triangle strips or polygons to define the faces of the Buckyball.**

**· For each face, use the vertices you defined to form pentagons and hexagons.**

**· Use glBegin(GL\_POLYGON) for each polygon face.**

**· Rendering:**

**· Set color and lighting for better visualization.**

**· Use glutMainLoop() to start the rendering loop.**

**· User Interaction:**

**· Include keyboard or mouse interaction to rotate the Buckyball or zoom in/out.**

**Implementation details:**

**#include <GL/glut.h>**

**#include <cmath>**

**GLfloat xRotated, yRotated, zRotated;**

**void display(void);**

**void idle(void) {**

**xRotated += 0.01;**

**yRotated += 0.01;**

**zRotated += 0.01;**

**display();**

**}**

**void myinit() {**

**GLfloat mat\_specular[] = { 1.0, 1.0, 1.0, 1.0 };**

**GLfloat mat\_shininess[] = { 50.0 };**

**GLfloat light\_position[] = { 1.0, 1.0, 1.0, 0.0 };**

**glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);**

**glMaterialfv(GL\_FRONT, GL\_SHININESS, mat\_shininess);**

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

**glEnable(GL\_LIGHTING);**

**glEnable(GL\_LIGHT0);**

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

**}**

**void display(void) {**

**glClearColor(1, 1, 1, 1);**

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

**glMatrixMode(GL\_MODELVIEW);**

**glLoadIdentity();**

**glTranslatef(0.0, 0.0, -5.0);**

**glColor3f(0.9, 0.3, 0.2);**

**glRotatef(xRotated, 1.0, 0.0, 0.0);**

**glRotatef(yRotated, 0.0, 1.0, 0.0);**

**glRotatef(zRotated, 0.0, 0.0, 1.0);**

**glutSolidDodecahedron();**

**glFlush();**

**}**

**void myReshape(GLsizei w, GLsizei h) {**

**glViewport(0, 0, w, h);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**glOrtho(-1.5, 1.5, -1.5 \* (GLfloat)h / (GLfloat)w, 1.5 \* (GLfloat)h / (GLfloat)w, -10.0, 10.0);**

**glMatrixMode(GL\_MODELVIEW);**

**glLoadIdentity();**

**}**

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

**glutInit(&argc, argv);**

**glutInitWindowSize(640, 480);**

**glutInitWindowPosition(200, 200);**

**glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGBA | GLUT\_DEPTH);**

**xRotated = 30.0;**

**yRotated = 50.0;**

**glutCreateWindow("Bucky ball");**

**glutDisplayFunc(display);**

**glutReshapeFunc(myReshape);**

**myinit();**

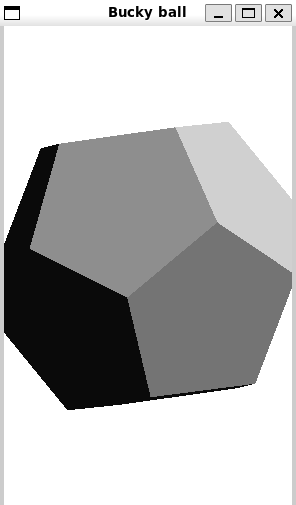
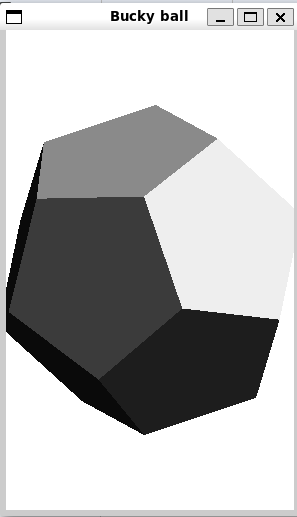
**glutIdleFunc(idle);**

**glutMainLoop();**

**return 0;**

**}**

**Output(s) (Screen Shot):**

**  
  
**

**Conclusion and discussion:**

**In this experiment, we successfully used OpenGL in C++ to render a Buckyball, a geometric shape represented by a truncated icosahedron with pentagons and hexagons. The process included defining vertices and faces, applying transformations like rotation, and rendering the object with color. This project improved our understanding of OpenGL's capabilities for drawing complex 3D structures. We also gained practical experience in handling 3D transformations and object visualization. Overall, this experiment enhanced our skills in 3D graphics programming using OpenGL.**

**Date: 9/10/24 Signature of faculty in-charge**

**Post Lab**

**Draw The 5-, 11-, and 17-rosettes. using OpenGL.**

**#include <GL/glew.h>**

**#include <GLFW/glfw3.h>**

**#include <cmath>**

**const float PI = 3.14159265358979323846f;**

**void drawRosette(int n, float radius, int num\_points = 100) {**

**glBegin(GL\_LINE\_LOOP);**

**for (int i = 0; i < num\_points; ++i) {**

**float angle = 2.0f \* PI \* i / num\_points;**

**float r = radius \* (1 + 0.5f \* sin(n \* angle));**

**float x = r \* cos(angle);**

**float y = r \* sin(angle);**

**glVertex2f(x, y);**

**}**

**glEnd();**

**}**

**void renderScene() {**

**glClear(GL\_COLOR\_BUFFER\_BIT);**

**glColor3f(1.0f, 1.0f, 1.0f); // Set color to white**

**//Draw 5-rosette**

**drawRosette(5, 0.5f);**

**// Translate to the right for 11-rosette**

**glPushMatrix();**

**glTranslatef(1.5f, 0.0f, 0.0f);**

**drawRosette(11, 0.5f);**

**glPopMatrix();**

**// Translate to the right for 17-rosette**

**glPushMatrix();**

**glTranslatef(3.0f, 0.0f, 0.0f);**

**drawRosette(17, 0.5f);**

**glPopMatrix();**

**}**

**int main() {**

**// Initialize GLFW**

**if (!glfwInit()) {**

**return -1;**

**}**

**// Create a windowed mode window and its OpenGL context**

**GLFWwindow\* window = glfwCreateWindow(800, 600, "Rosettes", nullptr, nullptr);**

**if (!window) {**

**glfwTerminate();**

**return -1;**

**}**

**glfwMakeContextCurrent(window);**

**glewInit();**

**while (!glfwWindowShouldClose(window)) {**

**renderScene();**

**glfwSwapBuffers(window);**

**glfwPollEvents();**

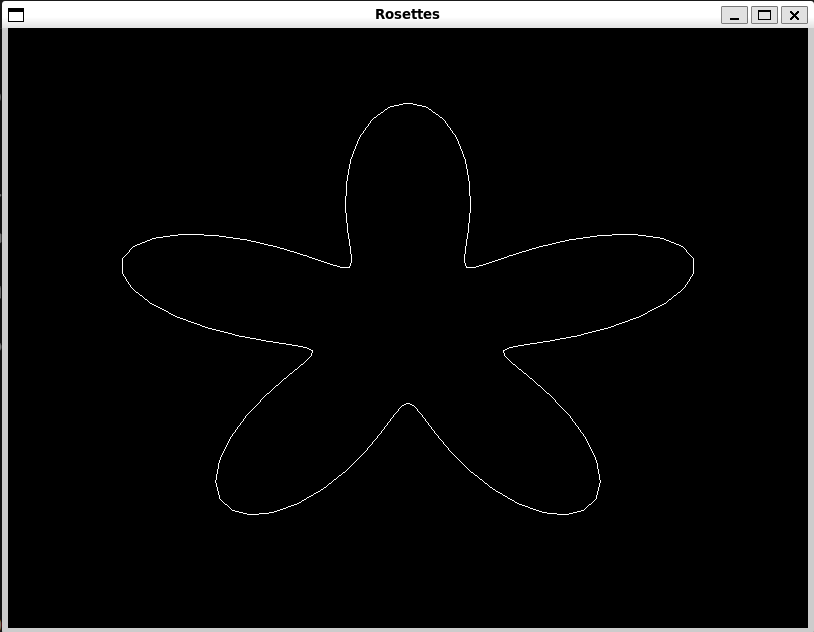
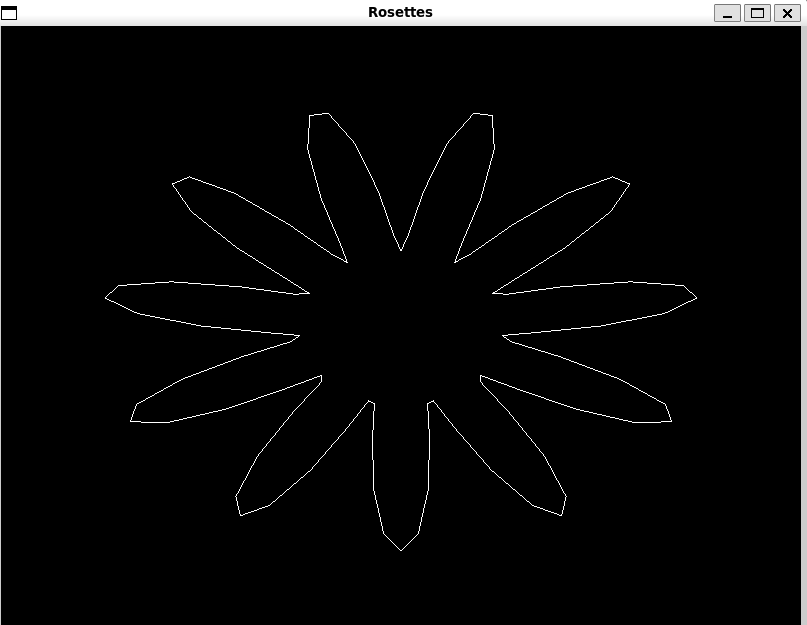
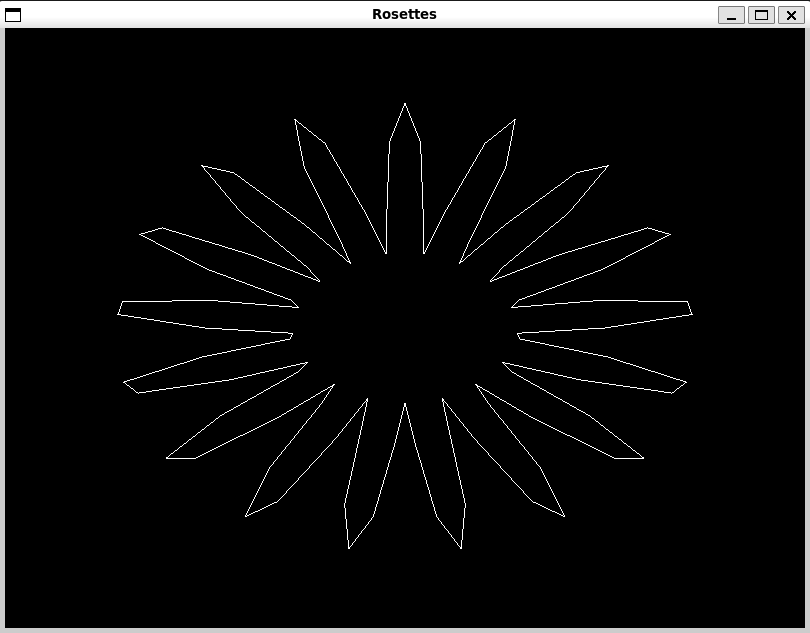
**}**

**glfwDestroyWindow(window);**

**glfwTerminate();**

**return 0;**

**}**

**5 rosettes  
  
  
  
11rosettes  
  
17 rosettes  
**