CGV ASSIGNMENT

Approximating sphere:

Code:

```
#include <stdio.h>
// #include <stdlib.h>    n = atoi(argv[1]);
#include <GL/glut.h>
#include <math.h>
typedef float point[4];
point v[] = {{0.0, 0.0, 1.0}, {0.0, 0.943, -0.33}, {-0.816, -0.471, -0.33}, {0.816, -0.471, 0.33}};
static GLfloat theta[] = {0.0, 0.0, 0.0};
int n;
int mode;

void triangle(point a, point b, point c)
{
   if (mode == 0)
      glBegin(GL_LINE_LOOP);
   else
      glBegin(GL_POLYGON);
   if (mode == 1)
      glNormal3fv(a);
   glVertex3fv(a);
   if (mode == 2)
      glNormal3fv(b);
   glVertex3fv(b);
   if (mode == 3)
```

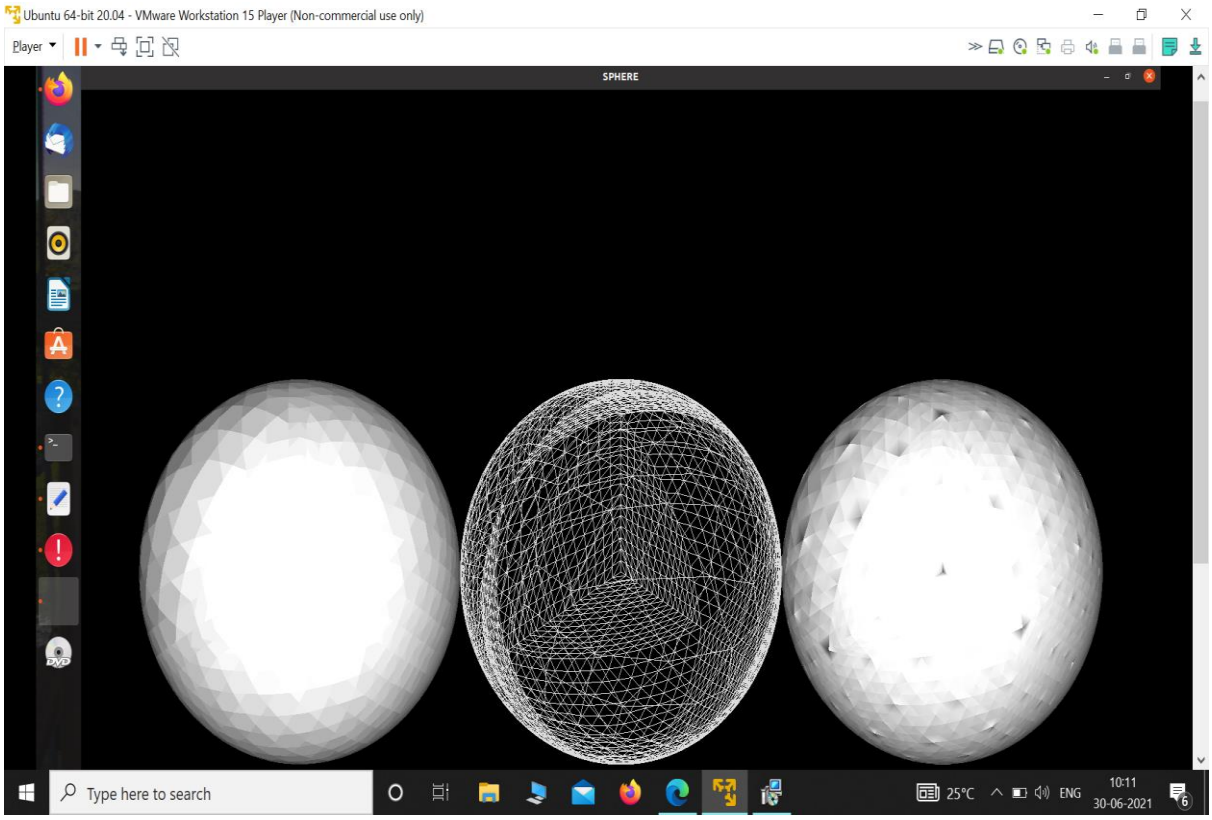
```
        glNormal3fv(c);
    glVertex3fv(c);
    glEnd();
}

void normalize(point p)
{
    double d = 0.0;
    int i;
    for (i = 0; i < 3; i++)
        d += p[i] * p[i];
    d = sqrt(d);
    if (d > 0.0)
        for (i = 0; i < 3; i++)
            p[i] /= d;
}

 Void divide_triangle(point a, point b, point c, int m)
{
    GLfloat v1[3], v2[3], v3[3];
    int j;
    if (m > 0)
    {
        for (j = 0; j < 3; j++)
            v1[j] = a[j] + b[j];
        normalize(v1);
        for (j = 0; j < 3; j++)
            v2[j] = a[j] + c[j];
        normalize(v2);
```

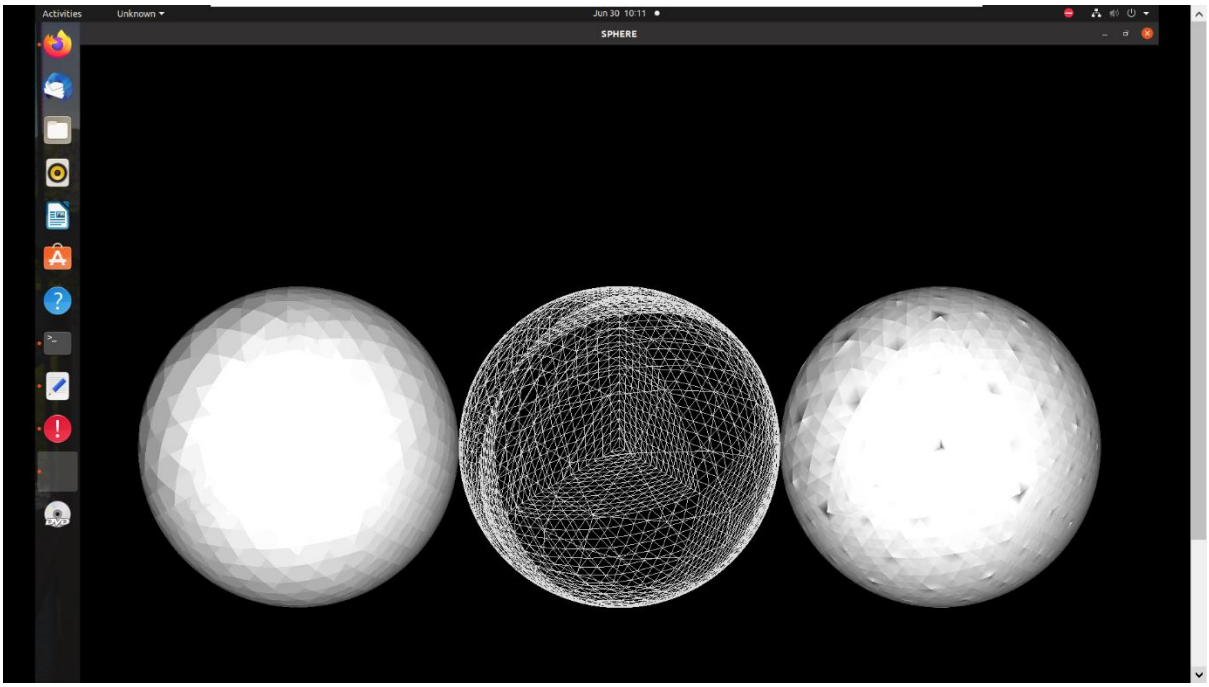
```
        for (j = 0; j < 3; j++)
            v3[j] = c[j] + b[j];
        normalize(v3);
        divide_triangle(a, v2, v1, m - 1);
        divide_triangle(c, v3, v2, m - 1);
        divide_triangle(b, v1, v3, m - 1);
        divide_triangle(v1, v2, v3, m - 1);
    }
    else
        triangle(a, b, c);
}

void tetrahedron(int m)
{
    divide_triangle(v[3], v[2], v[1], m);
    divide_triangle(v[0], v[3], v[1], m);
    divide_triangle(v[0], v[1], v[2], m);
    divide_triangle(v[0], v[2], v[3], m);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    mode = 0;
    tetrahedron(n);
    mode = 1;
    glTranslatef(-2.0, 0.0, 0.0);
    tetrahedron(n);
```

```
    mode = 2;
    glTranslatef(4.0, 0.0, 0.0);
    tetrahedron(n);
    glFlush();
}

void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-2.5, 2.5, -20.0 * (GLfloat)h / (GLfloat)w, 2.5 * (GLfloat)h / (GLfloat)w, -10.0,
10.0);
    else
        glOrtho(-2.5 * (GLfloat)w / (GLfloat)h, 2.5 * (GLfloat)w / (GLfloat)h, -2.5, 2.5, -10.0,
10.0);
    glMatrixMode(GL_MODELVIEW);
    display();
}

void myinit()
{
    GLfloat mat_Specular[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_ambient[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_shininess[] = {100};
    GLfloat light_ambient[] = {0.0, 0.0, 0.0, 1.0};
    GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
    GLfloat light_Specular[] = {1.0, 1.0, 1.0, 1.0};
```

```
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_Specular);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_Specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glClearColor(0.0, 0.0, 0.0, 0.0);
    glColor3f(1.0, 1.0, 1.0);
}

int main(int argc, char **argv)
{
    n = 5;
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(900, 900);
    glutCreateWindow("SPHERE");
    myinit();
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Output:


Ubuntu 64-bit 20.04 - VMware Workstation 15 Player (Non-commercial use only)
Player
SPHERE
Type here to search
25°C
ENG
10:11
30-06-2021


Activities
Unknown
Jun 30 10:11
SPHERE