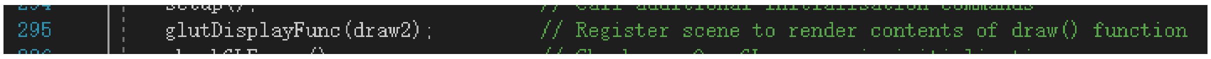
Computer Graphics | Group 9 | Assignment 1

Jialu Gan, Ethan Aron Phipps, Jinhao Wang



```
295     glutDisplayFunc(draw2);          // Register scene to render contents of draw() function
```

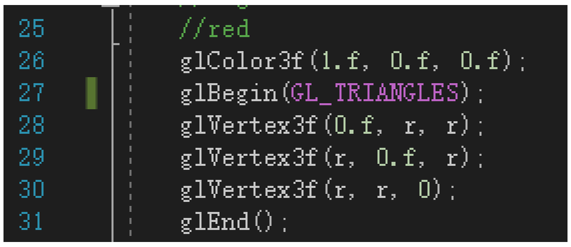
To view the different parts of the assignment, please modify the argument for glutDisplayFunc() on line 295 to draw for the first part, and “draw2” for the second part.

1 – Modelling

This starts with modelling a tetrahedron with red, yellow, green, and blue faces to distinguish between them (line 24 ~ line 55), and then creates a cube from translucent triangles (line 58 ~ line 143).

The coordinates of the 4 vertices of the tetrahedron are: (0, 0, 0); (100, 0, 100); (100, 100, 0); (0, 100, 100).

The coordinates of the 8 vertices of the cube are: (0, 0, 0); (100, 0, 0); (100, 0, 100); (0, 0, 100); (0, 100, 0); (100, 100, 0); (100, 100, 100); (0, 100, 100).



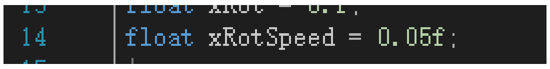
```
25      //red
26      glColor3f(1.f, 0.f, 0.f);
27      glBegin(GL_TRIANGLES);
28      glVertex3f(0.f, r, r);
29      glVertex3f(r, 0.f, r);
30      glVertex3f(r, r, 0);
31      glEnd();
```

Take the code for one face of the tetrahedron as an example:

glColor3f() sets the colour of the current face (which is red); Three glVertex3f() specify coordinates of 3 vertices of the triangle which OpenGL is going to draw (by using GL_TRIANGLES) in counter-clockwise.

Same method is deployed for other faces of the tetrahedron (4 triangle faces in total) and the cube (6 faces, 12 triangles in total).

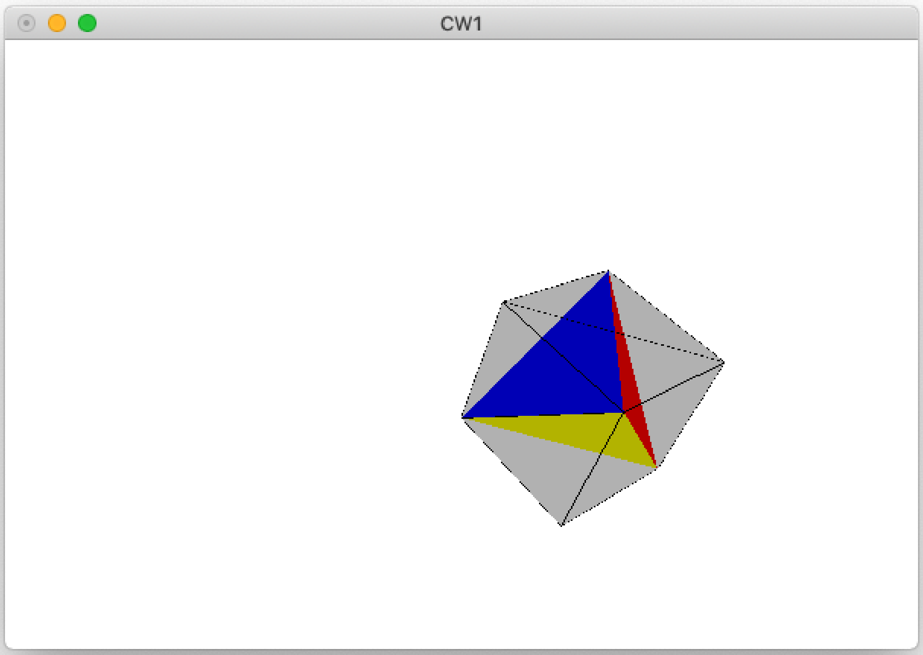
For the purpose of showing where the triangles are, lines 145 to 198 create an outline on the triangles. When using GL_LINE_LOOP to draw the wireframe of the cube, there is no worry about counter-clockwise or clockwise when specifying vertices, since drawing lines has nothing to do with face culling.

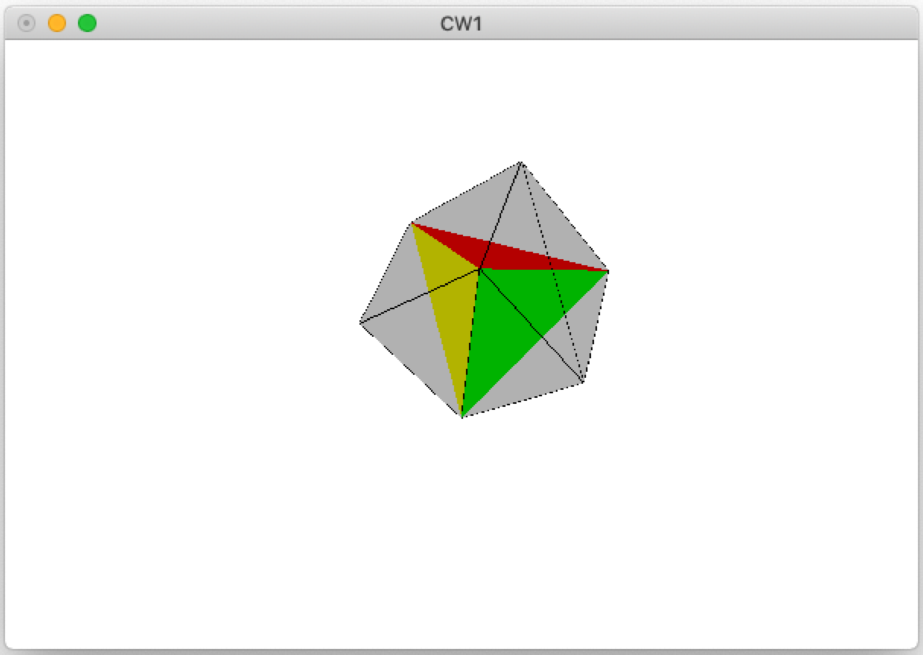


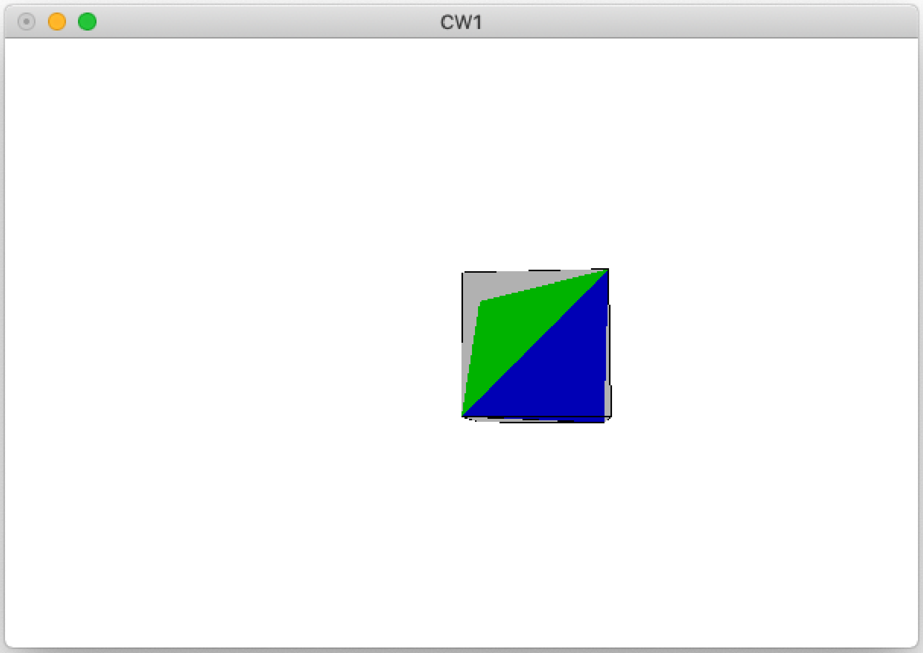
```
14    float xRotSpeed = 0.05f;
```

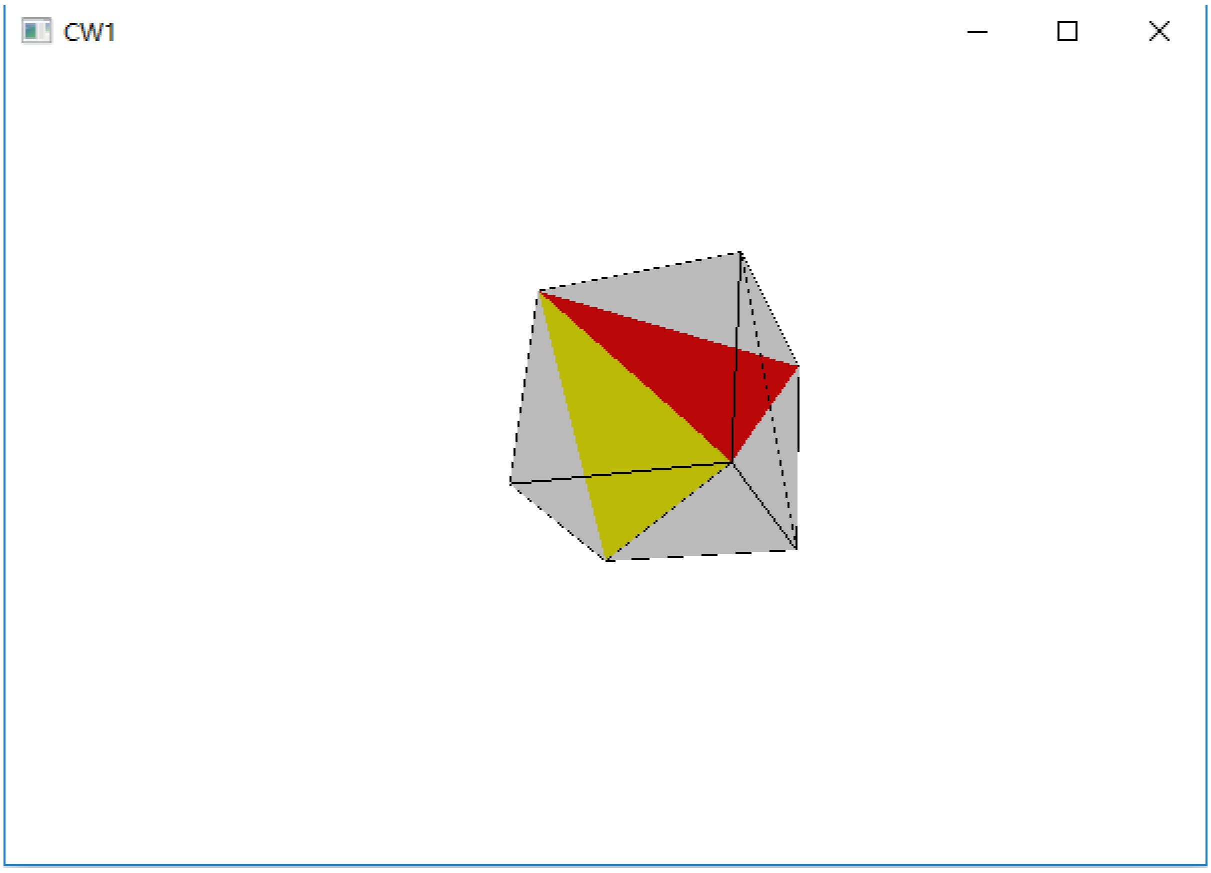
To change the speed of the cube’s rotation, modify the variable xRotSpeed on line 14 in MyScene.h.

Screenshots

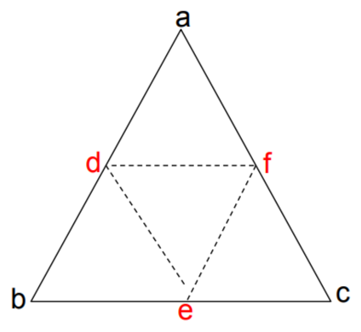




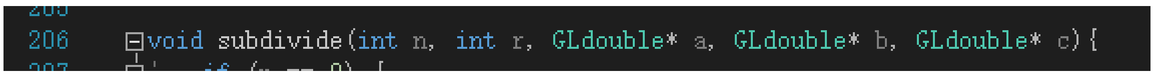

CW1


CW1

2 – Surface Subdivision

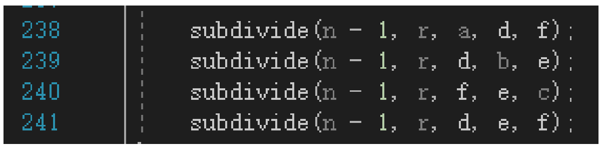


This performs subdivision on each face of the octahedron n times by running a subdivision function on each face individually, which subdivides that face n times before moving onto the next face. The subdivision function locates the midpoints of each edge of the face and moving these points until they have the same distances to origin point along their original directions. Then, the algorithm uses those points to run the subdivision function again on the four equal triangles that can be created from those midpoints and the points we already know.



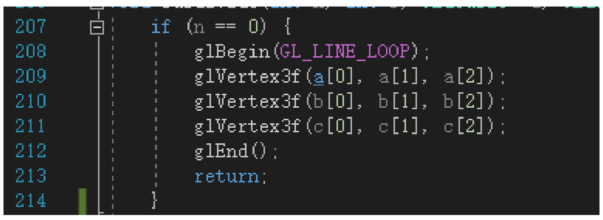
```
206  void subdivide(int n, int r, GLdouble* a, GLdouble* b, GLdouble* c){
```

The subdivision function is called "subdivide()" which takes 5 arguments: "n" represents times of iteration, the bigger it is, the rounder the sphere will be; "r" represents the radius of the sphere (which should be a constant value for a single program execution); "GLdouble* a, b, c" store the coordinates of the current triangle vertices (point a, b ,c).



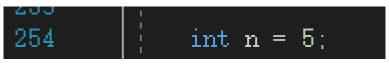
```
238         subdivide(n - 1, r, a, d, f);
239         subdivide(n - 1, r, d, b, e);
240         subdivide(n - 1, r, f, e, c);
241         subdivide(n - 1, r, d, e, f);
```

In each iteration, one triangle face can be subdivided into 4 new triangle faces. Therefore, we pass the corresponding coordinates and call subdivide() 4 times in itself, each call represents a new subdivision operation on that new face.



```
207     if (n == 0) {
208         glBegin(GL_LINE_LOOP);
209         glVertex3f(a[0], a[1], a[2]);
210         glVertex3f(b[0], b[1], b[2]);
211         glVertex3f(c[0], c[1], c[2]);
212         glEnd();
213         return;
214     }
```

When n equals 0, it means the iteration is finished. Use lines to connect all current points.

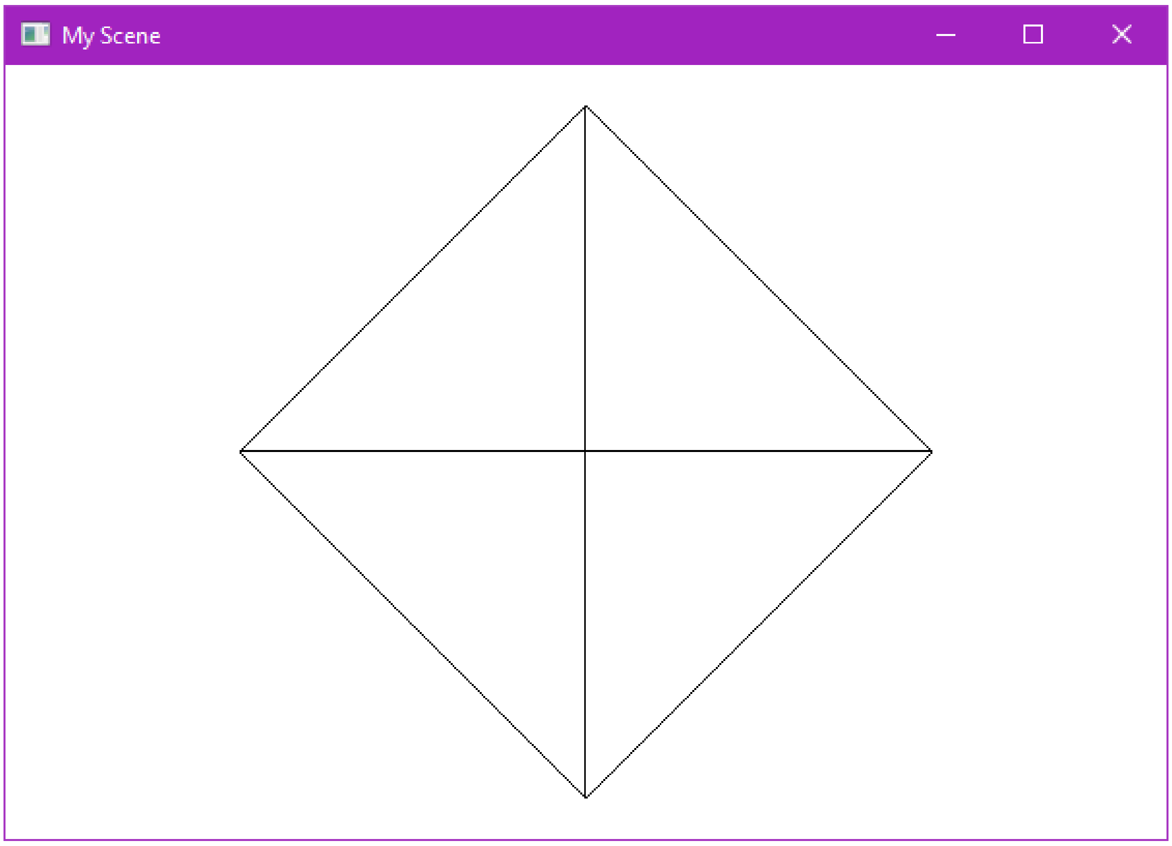


```
254     int n = 5;
```

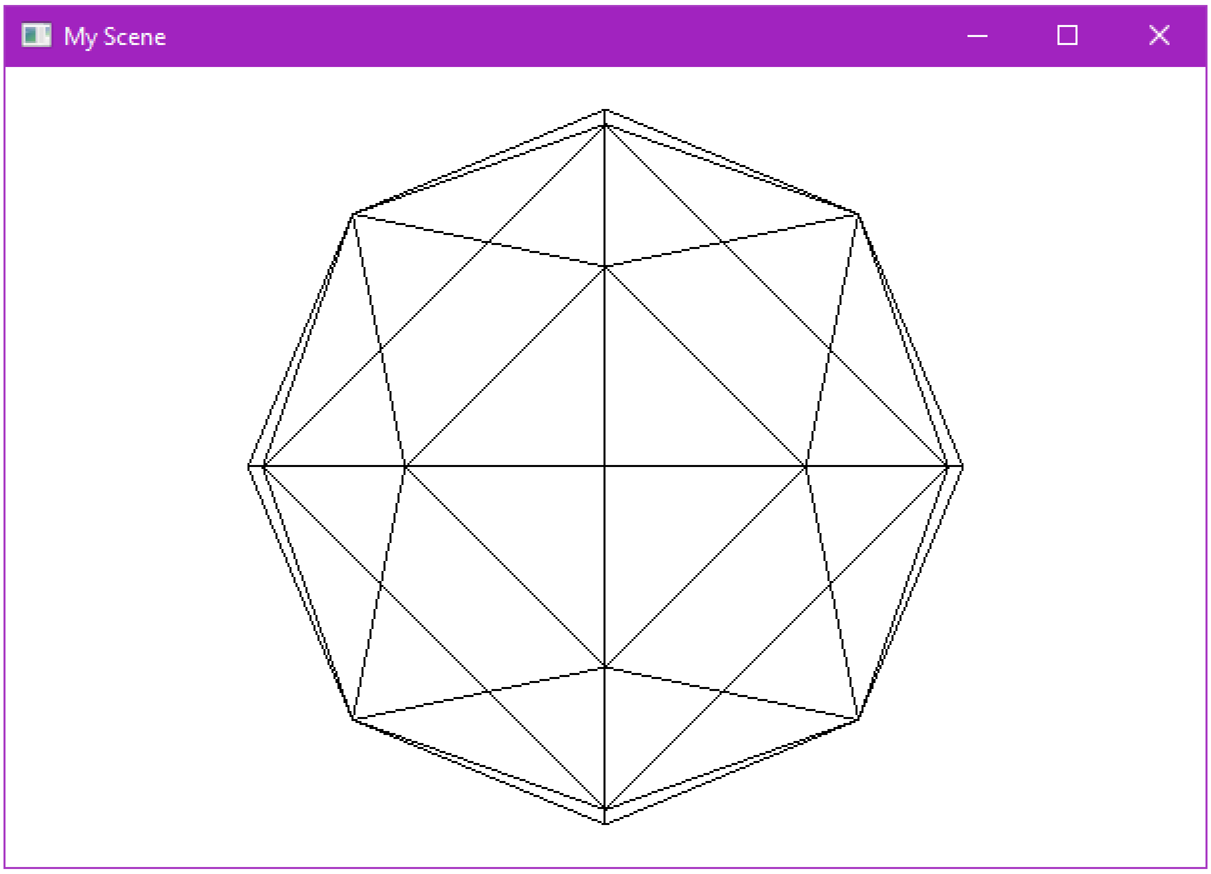
To alter the value n, modify line 254.

Screenshots

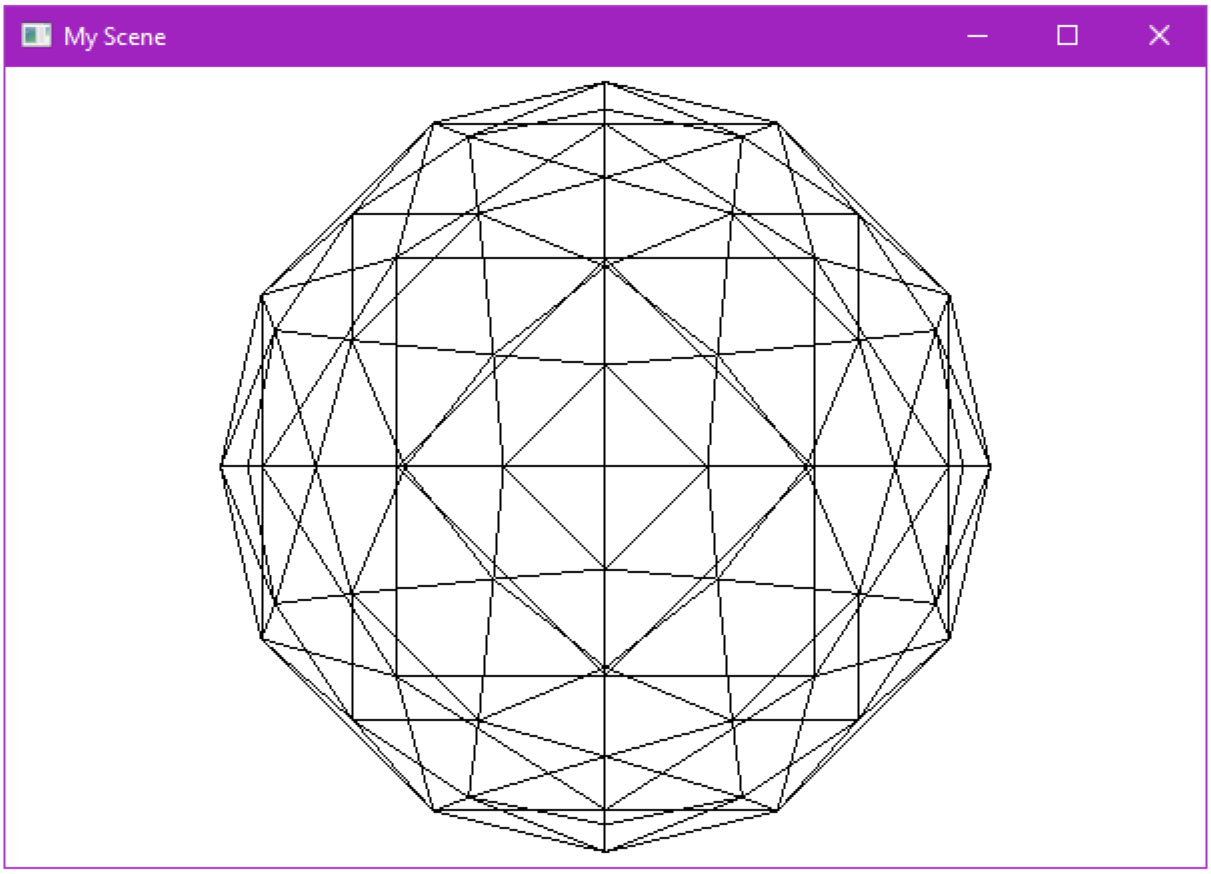
$n = 0$ (Octahedron)



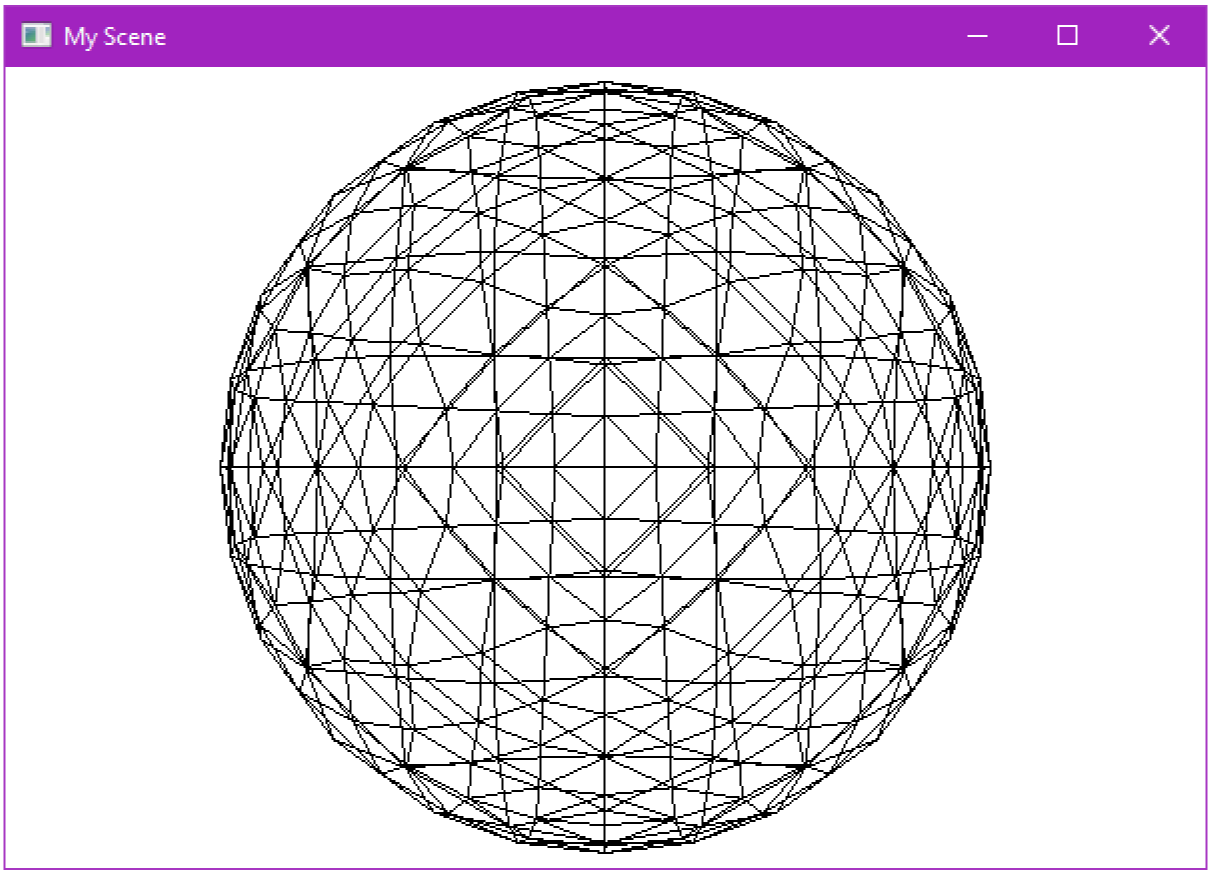
$n = 1$



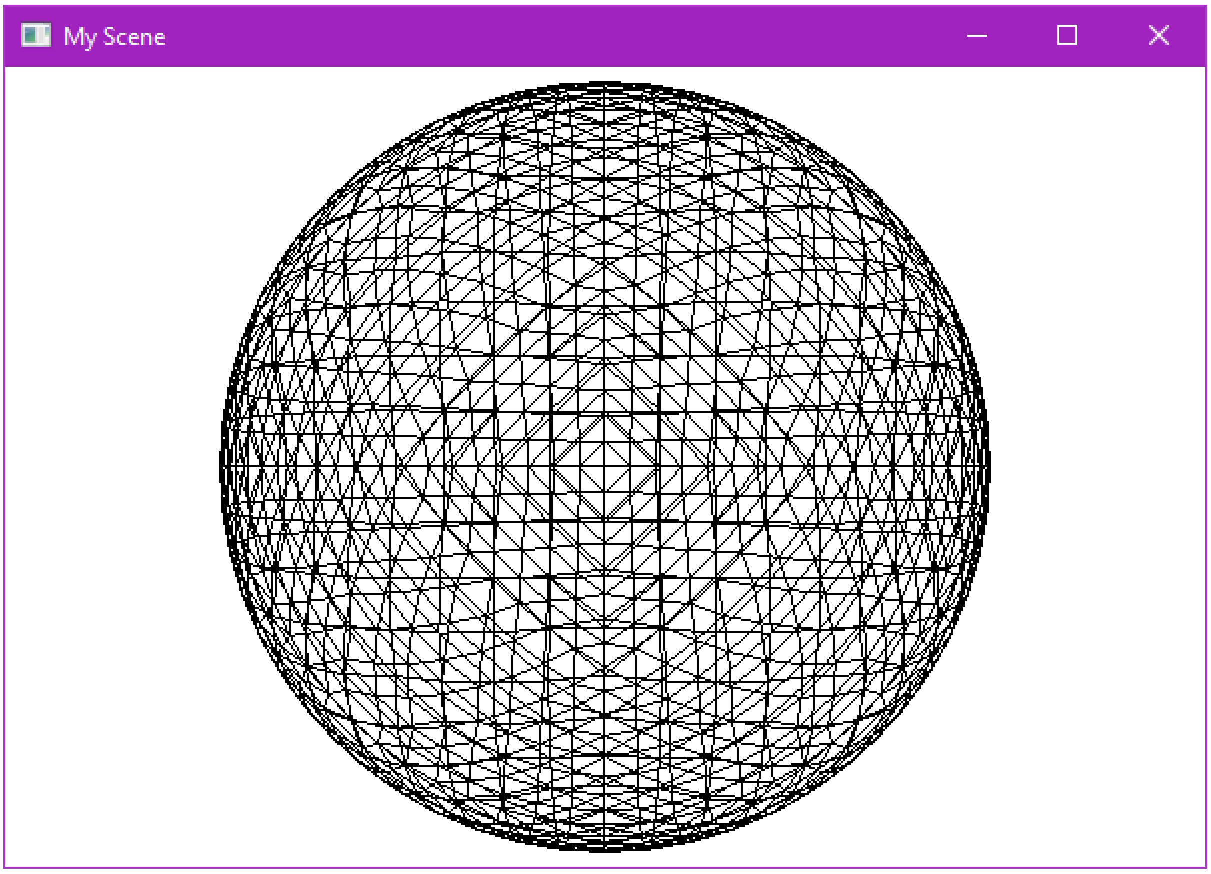
$n = 2$



$n = 3$



$n = 4$



$n = 5$