

2018 Project 3—Phases 1 (6 steps, 10 points each)

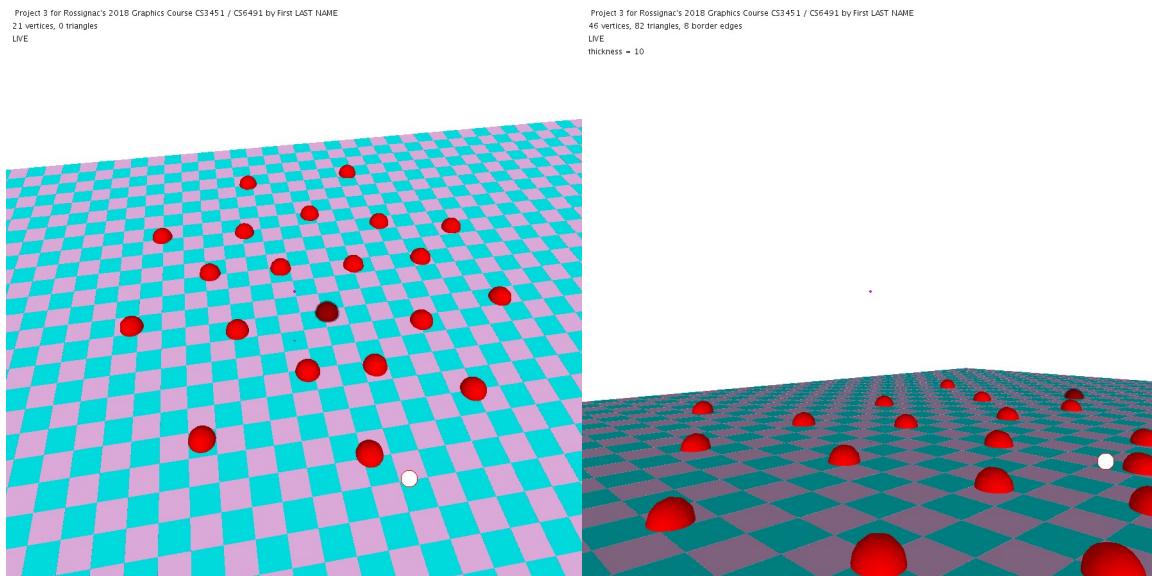
Individual

ABSTRACT

This project is designed to teach students about Triangle meshes, their representation, operators for traversing and processing them, about their construction through Delaunay Triangulation, about path planning amongst obstacles, about Voronoi regions, and about ray tracing.

1 Provided code and functionalities

The basecode is provided in Dropbox > P3... > 2018 Project 3 basecode.zip
It supports the editing of disks in the plane, shown as red balls.



The `zkeys` tab contains list of keys and corresponding actions

```
main beams circle mesh pick primitives pts pv3D tools triangles zgui zkeys ▾
1 /*
2
3 VERTICES
4 click      to select closest
5 mouse-drag to move
6 d          to delete closest
7 i & click  to inserts at mouse
8 e/E        to perturb a little/lot
9 =          to copy real vertex set R into saved vertex set S
10 S         to switch to editing the saved set
11 R         to switch to editing the real set
12 g/G       to get (load) vertices from data.pts / from named file
13 w/W       to write (save) vertices to data.pts / to named file
14 ;/:       to perturb vertices a little/a lot (if nt live, use m afterwards to recompute mesh)
15 C         to use content of clipboard as folder name for saving pictures
16
1 // ***** 2018 Project 3 basecde *****

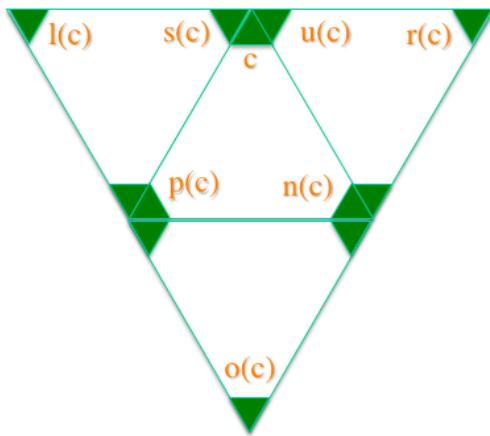
```

Enter your name in the main tab and remove the other course number

```
187 //*** TEAM: please fix these so that they provide the correct counts
188 int line=0;
189 scribeHeader(" Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME ",line++);
190 scribeHeader(P.count()+" vertices, "+M.nt+" triangles ",line++);
191 if(live) scribeHeader("LIVE",line++);
```

The goal of phase 1 is to teach you:

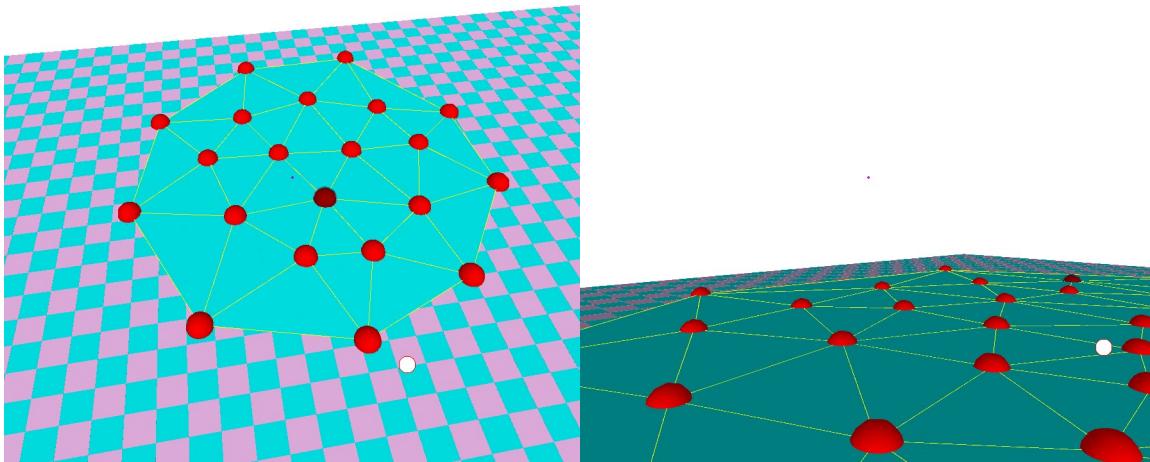
- 1-How to build the Delaunay triangulation of these points using a naïve $O(n^4)$ algorithm
- 2-How to use corners (3 per triangle) that each have a unique ID c
- 3-How to construct the V table of the Corner Table representation, where $V[c]$ is the index of the vertex that corner c is at
- 4-How to construct the O table where $O[c]$ is the corner $o(c)$ opposite to c as shown below
- 5-How to validate your implementation by showing curves the join pairs of opposite corners
- 7-How to identify interior and border vertices and to show border edges in a different color
- 8-How to implement and use interactively all of the corner operators shown below
- 9-How to beautify the mesh by adjusting the location of the interior vertices, even as you are editing the border vertices
- 10-How to trace the border loops of Voronoi faces
- 11-How to trace a smooth B-spline version of these loops



For clarity, in your code, you may want to use letters a, b, c, d , for corner IDs (for example in loops over corners) and i, k, j, m for vertex IDs. Also, I suggest that you use uppercase letters for points (such as vertex locations). This will avoid some confusion between corners (such as c), vertex IDs (such as $V[c]$, $v(c)$, or v), and points where the vertices are located (such as $G[V[c]]$ for which I provide the $g(c)$ method, which I should have written as $Pg[c]$ for consistency with the uppercase). Also note that I provide method $cg(c)$, which returns the point (a bit towards the centroid of the triangle) where I draw the corner.

2 Step 1: Delaunay triangulation of disk centers

Compute and display the Delaunay triangles for the centers of these disks.



This is called in main at each frame in step1 (if you pressed '1'):

```
95
96 if(step1)
97 {
98     pushMatrix();
99     translate(0,0,4); fill(cyan); stroke(yellow);
100    if(live)
101    {
102        M.reset();
103        M.loadVertices(R.G,R.nv);
104        M.triangulate(); // **01 implement it in Mesh
105    }
106    if(showTriangles) M.showTriangles();
107    noStroke();
108    popMatrix();
109 }
```

Put your implementation as a method in the tab mesh:

```
58 void showEdges() {for (int i=0; i<nc; i++) showEdge(i); } // draws all edges of
59
60 void triangulate() // performs Delaunay triangulation using a quartic algorithm
61 {
62     c=0; // to reset current corner
63 // **01 implement it
64 }
65
```

Make sure that the triangles are clockwise (use cw() to check).

These methods of mesh may help:

```
16 void reset() {nv=0; nt=0; nc=0;}
17 void loadVertices(pt[] P, int n) {nv=0; for (int i=0; i<n; i++) addVertex(P[i]);}
18 void writeVerticesTo(pts P) {for (int i=0; i<nv; i++) P.G[i].setTo(G[i]);}
19 void addVertex(pt P) {G[nv++].setTo(P);}
20 void addTriangle(int i, int j, int k) {V[nc++]=i; V[nc++]=j; V[nc++]=k; nt=nc/3; }
```

The code that I provided for computing circumcenters of candidate triplets of vertices does not deal well with the special-case configurations, where the three points are colinear.

I use the triThichnedd function proided in the tab triangles to implement isFlatterThan(...r)

```
boolean isFlatterThan(pt A, pt B, pt C, float r) {return triThickness(A,B,C)<r; }
```

and, in side my triple loop for computing the Delaunay triangles, I reject triangles flatter than thickness=10, using

```
if(!isFlatterThan(G[i],G[j],G[k], thickness))
```

3 Step 2: Compute the O table and show border/interior edges

The O tables identifies opposite corners. Set $O[c]=c$ for corners that do not have opposites.

Implement it as a method in tab mesh:

```
66  
67 void computeO() // **02 implement it  
68 {  
69 // **02 implement it  
70 }  
71  
72 void showBorderEdges() // draws all border edges of mesh  
73 {  
74 // **02 implement;  
75 }  
76  
77 void showNonBorderEdges() // draws all non-border edges of mesh  
78 {  
79 // **02 implement  
80 }
```

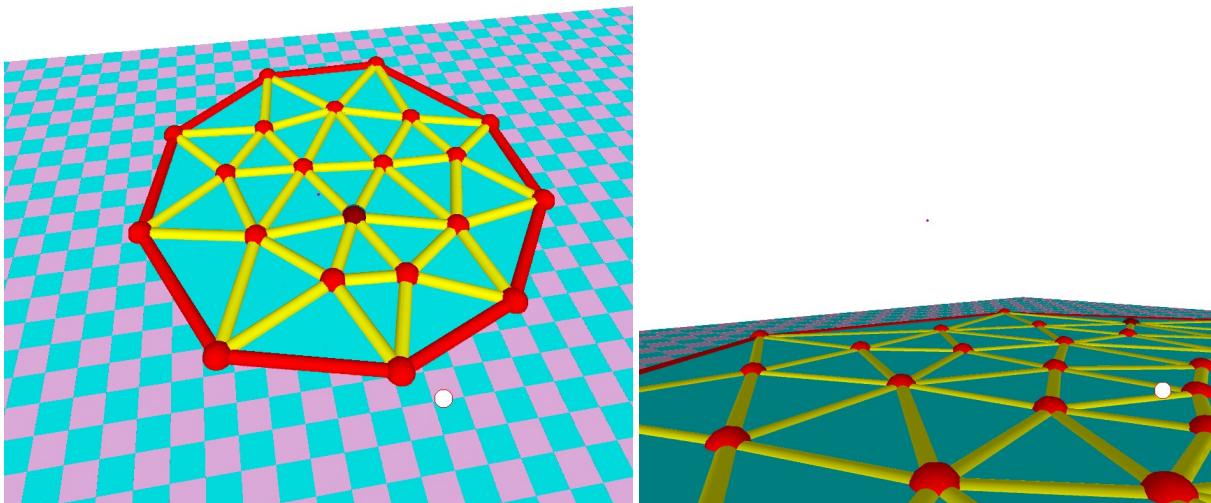
It is called in main for step 2 (when you press '2').

```
110 if(step2)  
111 {  
112 fill(yellow);  
113 if(live) {M.computeO();} // **02 implement it in Mesh  
114 if(showEdges)  
115 {  
116 fill(yellow);  
117 M.showNonBorderEdges(); // **02 implement it in Mesh  
118 fill(red);  
119 M.showBorderEdges();} // **02 implement it in Mesh  
120 }  
121 }
```

Also implement the display of border and non-border edges:

Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
21 vertices, 31 triangles
LIVE

Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
46 vertices, 82 triangles, 8 border edges
LIVE
thickness = 10



Replace the displayed text by:

```
scribeHeader(P.count()+" vertices, "+M.nt+" triangles, "+M.countBorders()+" border edges ",line++);  
Implement the countBorders method and check its correctness.
```

4 Step 3: Identify interior/border vertices

In tab mesh, add your code to identify interior vertices

```
82 void classifyVertices()
83 {
84     // **03 implement it |
85 }
```

Using the `isInterior[]` array,

```
8 boolean[] isInterior = new boolean[maxnv];
9 // CORNERS
```

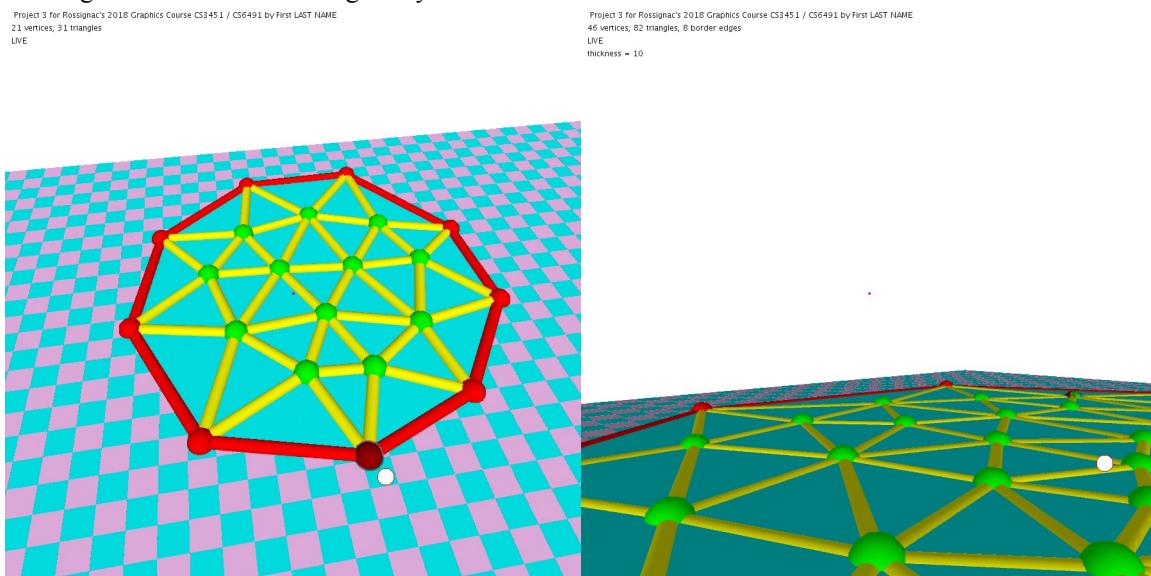
It is used in mesh by the `showVertices` method

```
48 void showVertices(float r) // shows all vertices green inside, red outside
49 {
50     for (int v=0; v<nv; v++)
51     {
52         if(isInterior[v]) fill(green); else fill(red);
53         show(G[v],r);
54     }
55 }
```

Which is invoked from main

```
123 if(step3)
124 {
125     M.classifyVertices(); // **03 implement it in Mesh
126     showBalls=false;
127     fill(green); noStroke();
128     M.showVertices(rb+4);
129 }
```

To display border edges in red and interior edges in yellow.



What is the formula that relates the number T of triangles to the number V of vertices and to the number E of border edges?

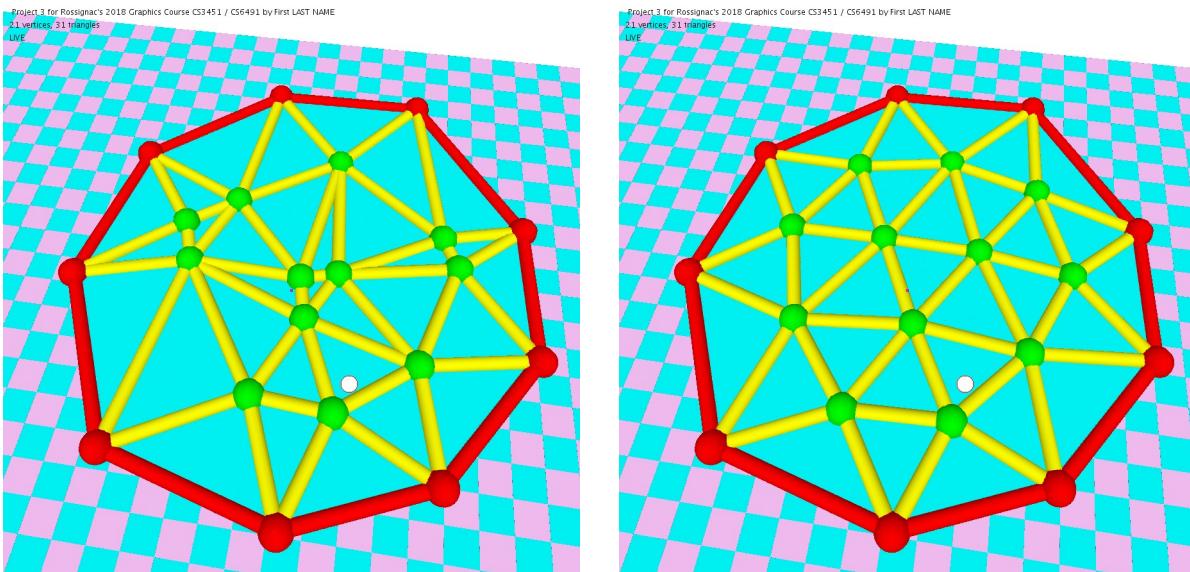
5 Step 4: Smoothen the interior vertices

This is called for step4 in main

```
131 if(step4)
132 {
133     for(int i=0; i<10; i++) M.smoothInterior(); // **04 implement it in Mesh
134     M.writeVerticesTo(R);
135 }
```

Implement it in mesh

```
87 void smoothInterior() { // even interior vertex locations
88     pt[] Gn = new pt[nv];
89     // **04 implement it
90     for (int v=0; v<nv; v++) if(isInterior[v]) G[v].translateTowards(.1,Gn[v]);
91 }
92
```



Make it work in Live mode and in your video, show first its action on a static mesh and then show what happens when you edit some of the border vertices, so that the connectivity of M changes.

As an option, you may want to reduce the number of iterations of the smoothing or the step size to give this interactive mode a more fluid look:

```
if(step4)
{
    for(int i=0; i<1; i++) M.smoothInterior();
    M.writeVerticesTo(R);
}
```

Change the 'b' action in zgui to:

```
if(key=='b') {for(int i=0; i<100; i++) M.smoothInterior(); M.writeVerticesTo(R);}
```

Execute it in non-Live mode. It will not change the connectivity. Then, turn on the Live mode and see whether it changes the mesh connectivity. Show this effect in the video.

You may also experiment with a “tuck/untuck” smoothing:

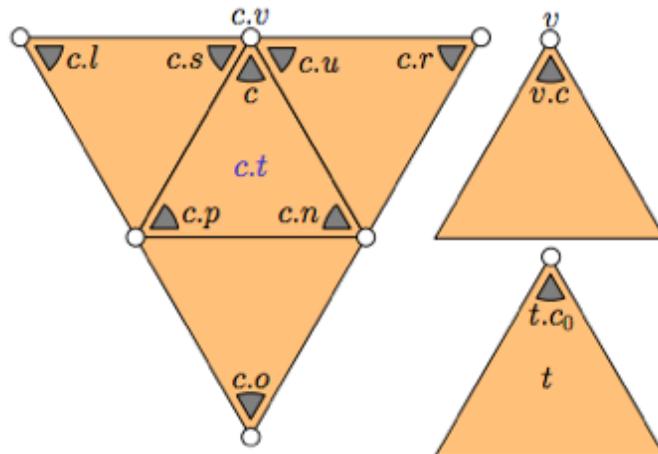
```
if(step4)
{
    for(int i=0; i<10; i++)
    {
        M.smoothInterior(0.5);
        M.smoothInterior(-0.5);
    }
}
```

6 Step 5: Corner operators

Implement the corner operators in mesh:

```
93 // **05 implement corner operators in Mesh
94 int v (int c) {return 0;}                                // vertex of c
95 int o (int c) {return 0;}                                // opposite corner
96 int l (int c) {return 0;}                                // left
97 int s (int c) {return 0;}                                // left
98 int u (int c) {return 0;}                                // left
99 int r (int c) {return 0;}                                // right
100
101
```

Using the naming below. Note that in some papers, to simplify cascading operations, I use the $c.s$ notation for $s(c)$, so that $(o(n(c)))$ is written $c.n.o$. Note that I have provided you with the code for $p(c)$ and $n(c)$.



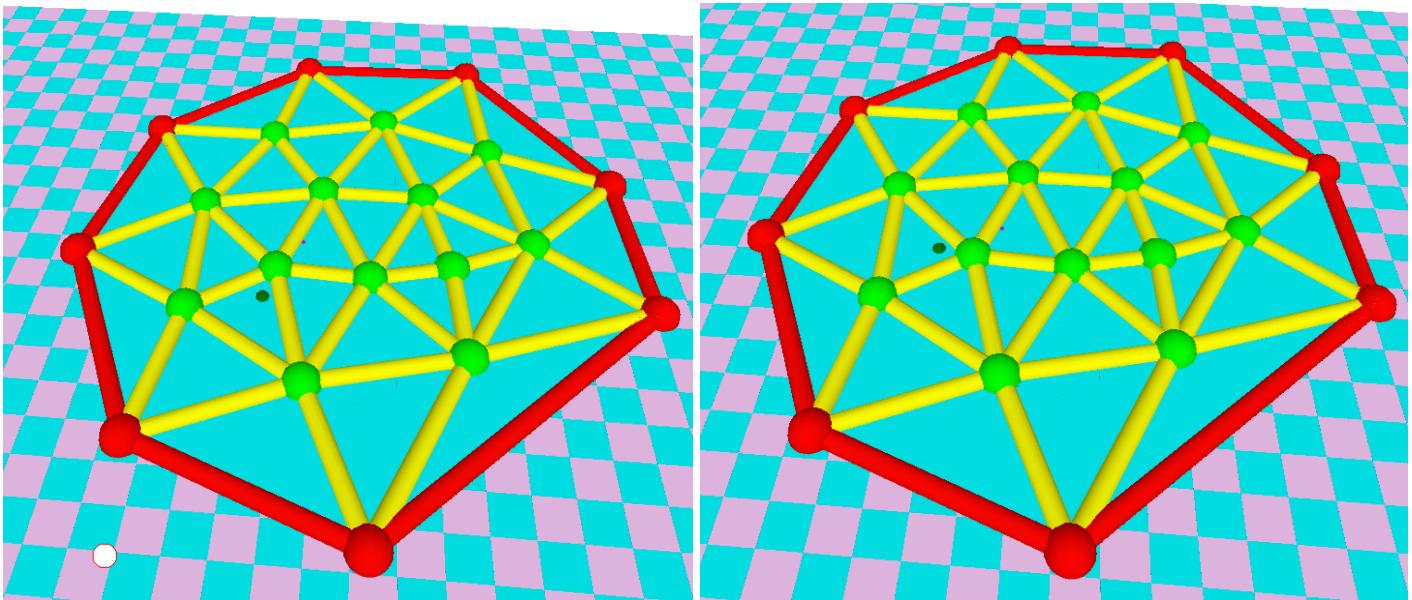
Verify that they work by using keys as specified in the zgui tab

```
41
42 if(key=='l') M.left();
43 if(key=='m') {M.reset();}
44 if(key=='n') M.next();
45 if(key=='o') M.opposite();
46 if(key=='p') M.previous();
47 if(key=='q') ;
48 if(key=='r') M.right();
49 if(key=='s') M.swing();
50 if(key=='t') ;
51 if(key=='u') M.unswing();
52 //
```

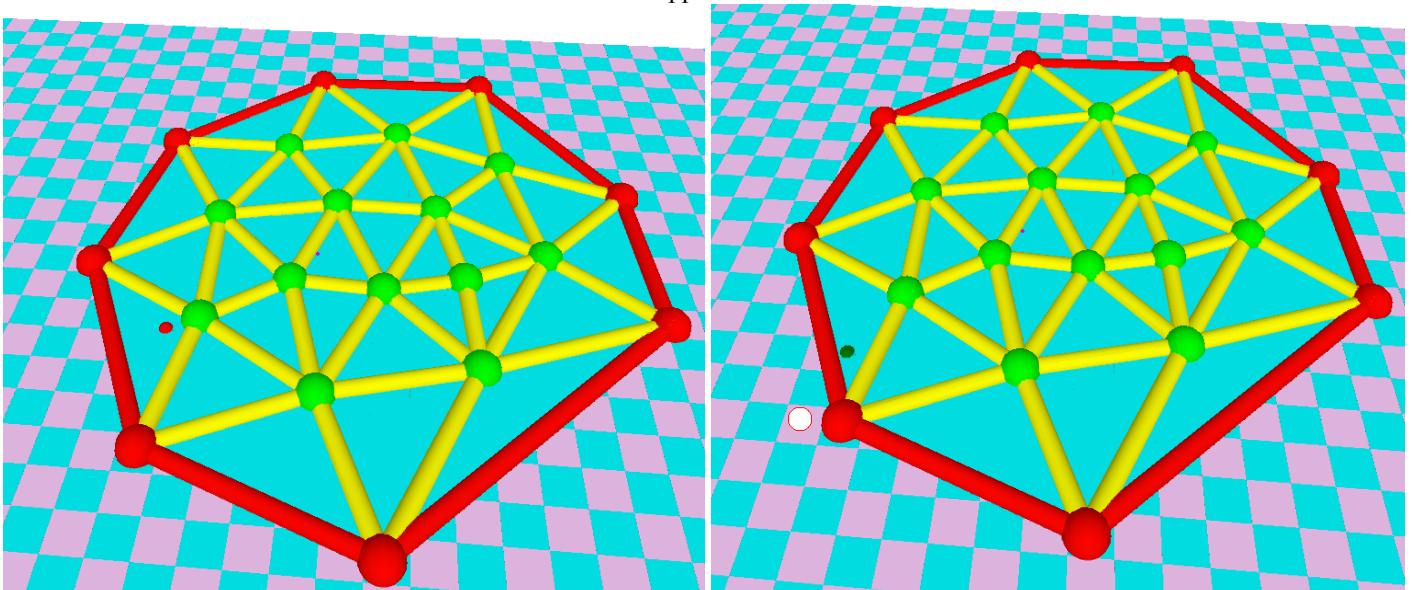
The current corner is displayed as a small dot

```
137 // **05 implement corner operators in Mesh
138 if(step5)
139 {
140     live=false;
141     fill(magenta);
142     if(showCorner) M.showCurrentCorner(20);
143 }
```

In your video show that these work keys ‘p’, ‘n’, ‘s’, ‘o’....



Make sure that the dot around c is colored red when c has no opposite.

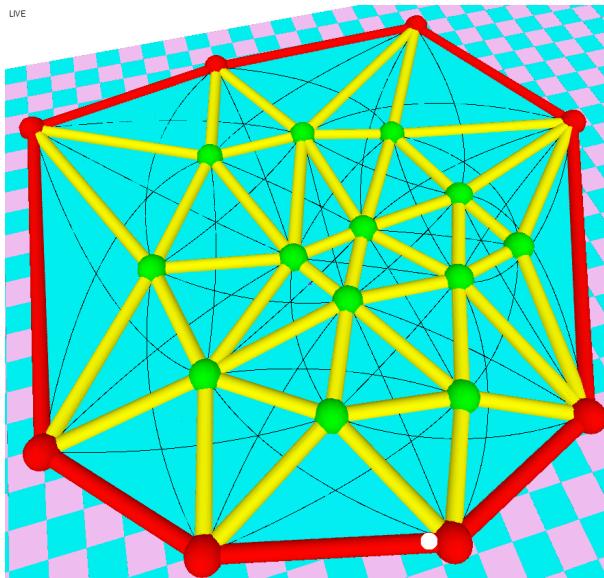


Modify step 5 as follows

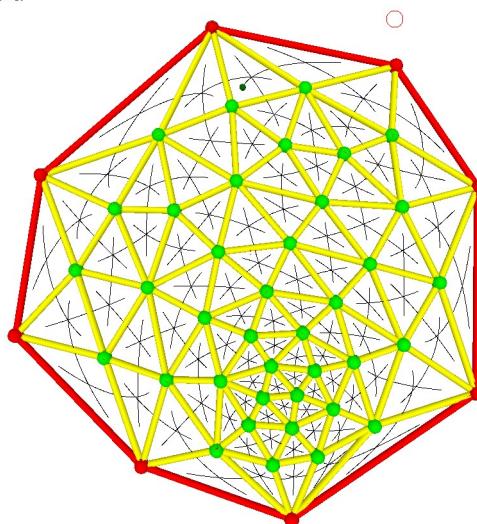
```
if(step5)
{
// live=false;
fill(magenta);
if(showCorner) M.showCurrentCorner(20); |
if(showOpposite)
{
pushMatrix();
translate(0,0,6); noFill(); stroke(black);
M.showOpposites();
popMatrix();
}
}
```

and implement the method showOpposites (template not provided) that draws parabola between the vertices of all pairs of opposite corners or between the locations of the corners (computed using $cg(c)$) See below-right. As tip of the hat, I use the midpoint of the edge that is shared by $t \odot$ and $t(o(c))$.

LIVE



Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
46 vertices, 82 triangles, 8 border edges
thickness = 10



7 Step 6: Voronoi loops

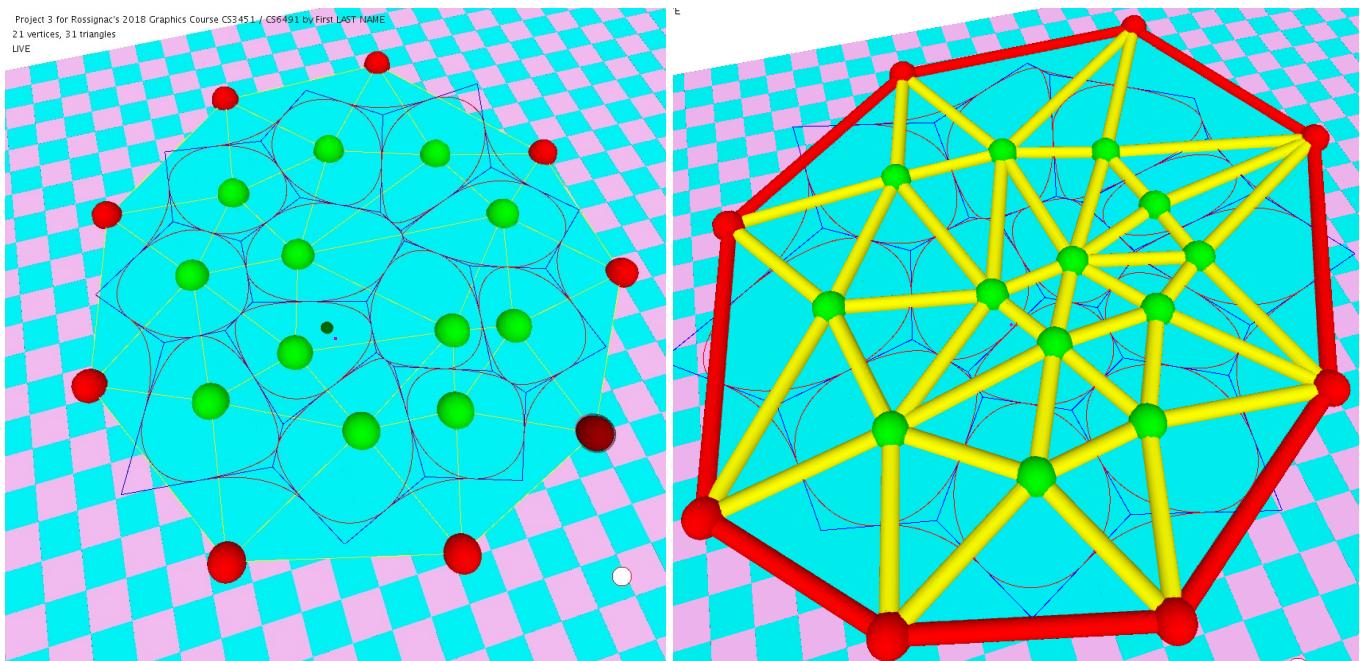
To support step 6:

```
if(step6)
{
pushMatrix();
translate(0,0,6); noFill();
stroke(blue);
if(showVoronoi) M.showVoronoiEdges(); // **06 implement it in Mesh
stroke(red);
if(showArcs) M.showArcs(); // **06 implement it in Mesh
noStroke();
popMatrix();
}
```

Implement in mesh the methods

```
102 void showVoronoiEdges() // draws Voronoi edges on the boundary of Voroni cells of interior vertices
103 {
104     // **06 implement it
105 }
106
107 void showArcs() // draws arcs of quadratic B-spline of Voronoi boundary loops of interior vertices
108 {
109     // **06 implement it
110 }           // draws arcs in triangles
111
```

to display the Voronoi edges for Voronoi faces of interior vertices and to display their quadratic B-spline borders.



Modify step6 to fill each Voronoi face with a different color.

```
if(step6)
{
    pushMatrix();
    translate(0,0,8); noFill();
    if(showVoronoiFaces) M.drawVoronoiFaceOfInteriorVertices();
    stroke(blue);
    if(showVoronoi) M.showVoronoiEdges();
    stroke(red);
    if(showArcs) M.showArcs();
    popMatrix();
}
```

Implement the mesh method drawVoronoiFaceOfInteriorVertices. You may start from my implementation:

```
void drawVoronoiFaceOfInteriorVertices()
{
    float dc = 1./(nv-1);
    for (int v=0; v<nv; v++) if(isInterior[v]) {fill(dc*255*v,dc*255*(nv-v),200); drawVoronoiFaceOfInteriorVertex(v);}
}
```

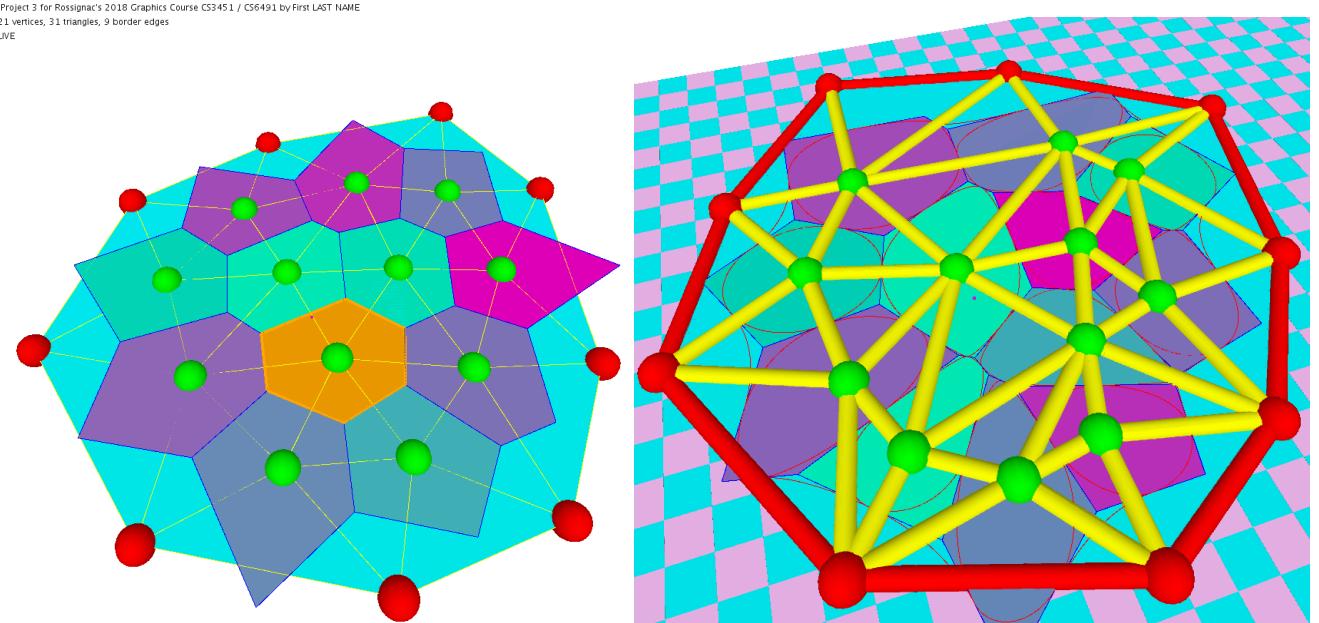
But have to implement drawVoronoiFaceOfInteriorVertex(v).

To do that, I implemented a helper method:

```
int cornerIndexFromVertexIndex(int v) {...}
```

that returns the Id of one of the corners on vertex v, since this information is not encoded explicitly in the Corner Table.

In your video, show how it behaves in Live mode as you edit the vertices.



In your video, you may run this live and show how the Voronoi faces are changed when you insert new vertices (pressing 'i') or moving them.

8 Submission of PHASE 1

Upload your source code (no images) and a **short** video (less than one mn) showing off your implementation of each step. Please use a modest size video file. Please upload your submission way in advance of the deadline and make sure that you did it correctly.

A **penalty of 20%** will be taken off your grade for this phase for submissions after the deadline for this phase. So, you may be better off submitting an incomplete phase 1.

Submissions after the deadline of the final project will not be graded.