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

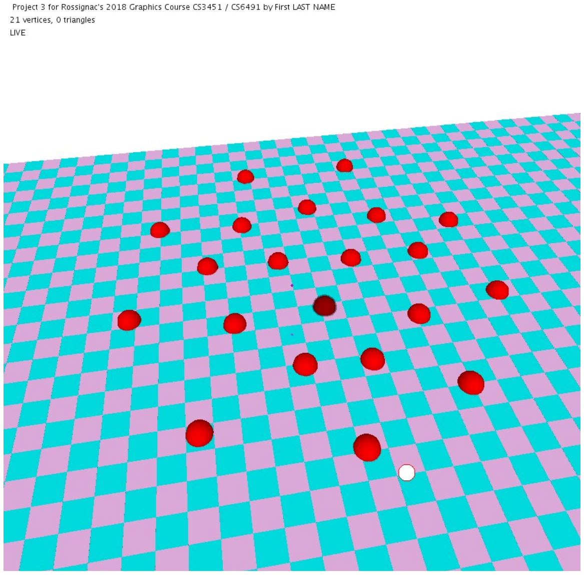
Individual

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.

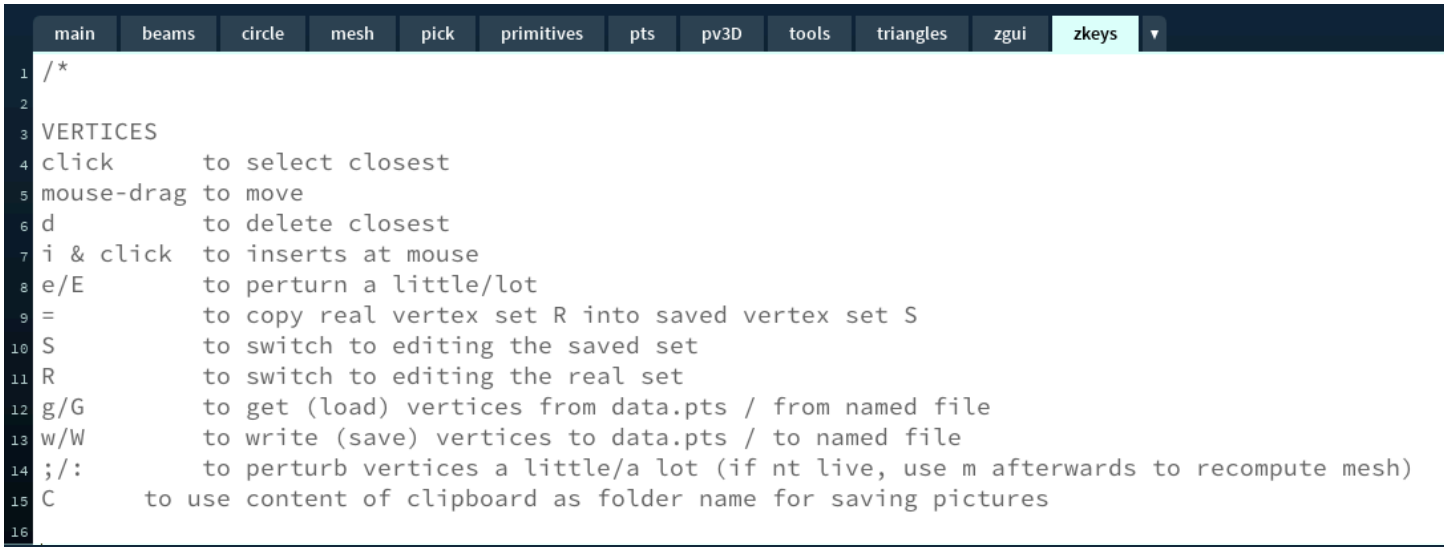
1 Provided code and functionalities

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.

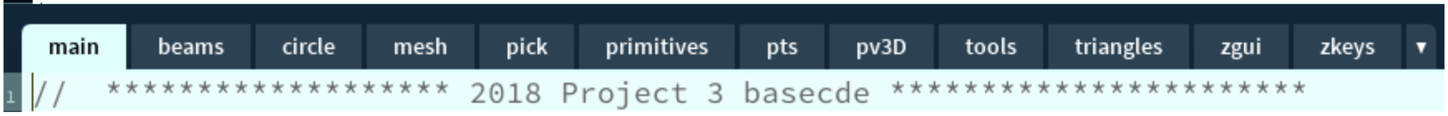


```
Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
21 vertices, 0 triangles
LIVE
```

The zkeys tab contains list of keys and corresponding actions

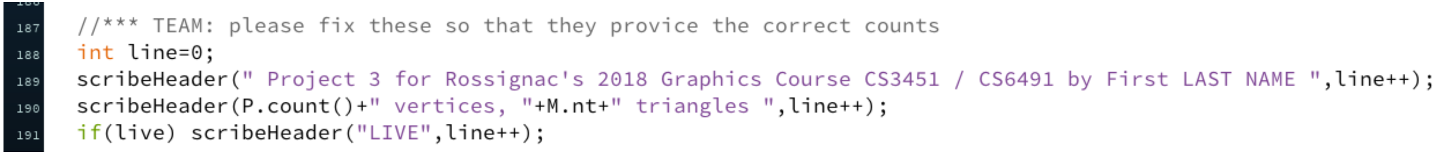


```
/*

VERTICES
click      to select closest
mouse-drag to move
d          to delete closest
i & click  to inserts at mouse
e/E        to perturn a little/lot
=          to copy real vertex set R into saved vertex set S
S          to switch to editing the saved set
R          to switch to editing the real set
g/G        to get (load) vertices from data.pts / from named file
w/W        to write (save) vertices to data.pts / to named file
;/:        to perturb vertices a little/a lot (if nt live, use m afterwards to recompute mesh)
C      to use content of clipboard as folder name for saving pictures
```



```
//  ****************** 2018 Project 3 basecde ***********************
```

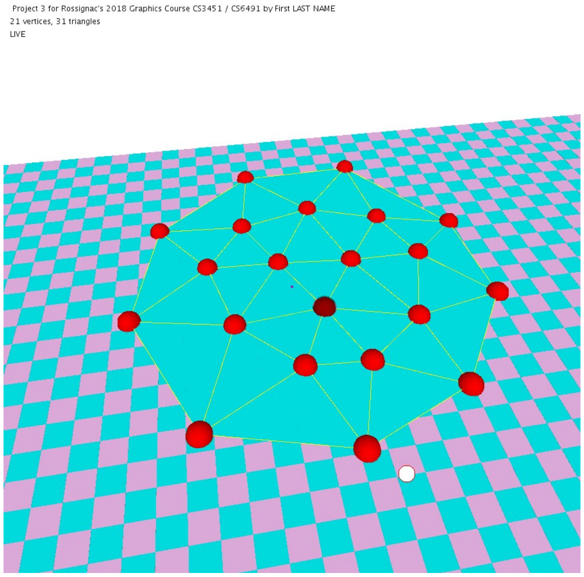
Enter your name in the main tab and remove the other course number



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

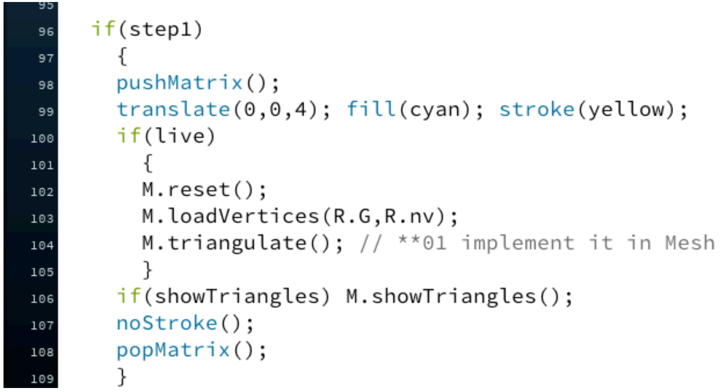
2 Step 1: Delaunay triangulation of disk centers

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



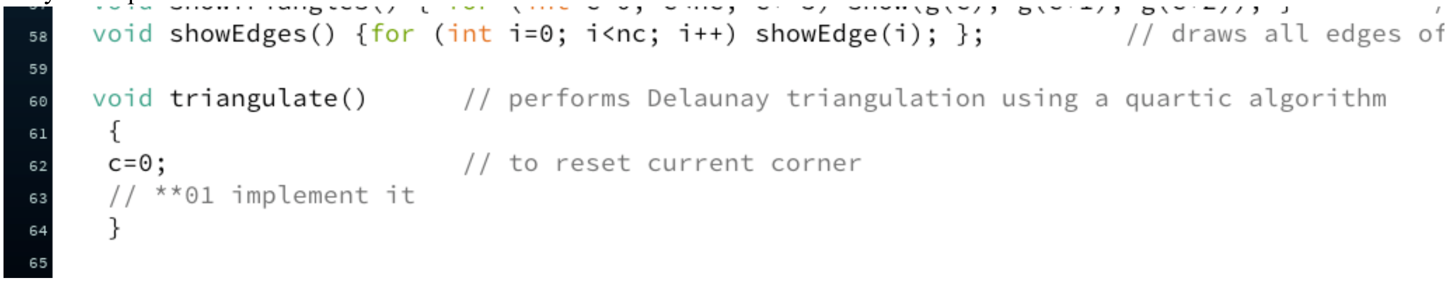
Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
21 vertices, 31 triangles
LIVE

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



```
if(step1)
  {
  pushMatrix();
  translate(0,0,4); fill(cyan); stroke(yellow);
  if(live)
    {
    M.reset();
    M.loadVertices(R.G,R.nv);
    M.triangulate(); // **01 implement it in Mesh
    }
  if(showTriangles) M.showTriangles();
  noStroke();
  popMatrix();
  }
```

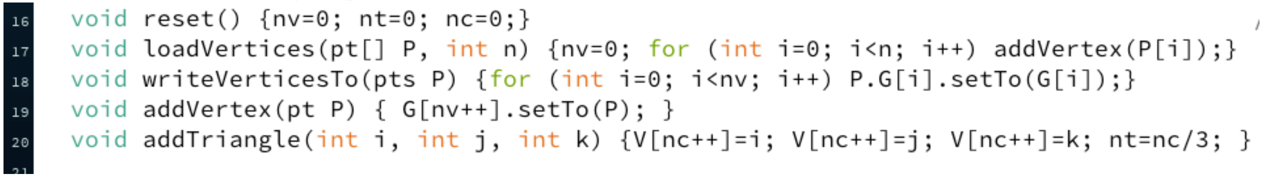
Put your implementation as a method in the tab mesh:



```
void showEdges() {for (int i=0; i<nc; i++) showEdge(i); };          // draws all edges of

void triangulate()      // performs Delaunay triangulation using a quartic algorithm
 {
 c=0;                   // to reset current corner
 // **01 implement it
 }
```

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

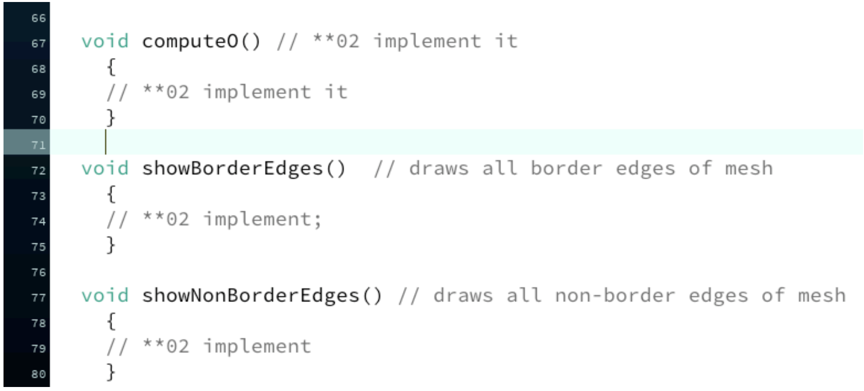
These methods of mesh may help:



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

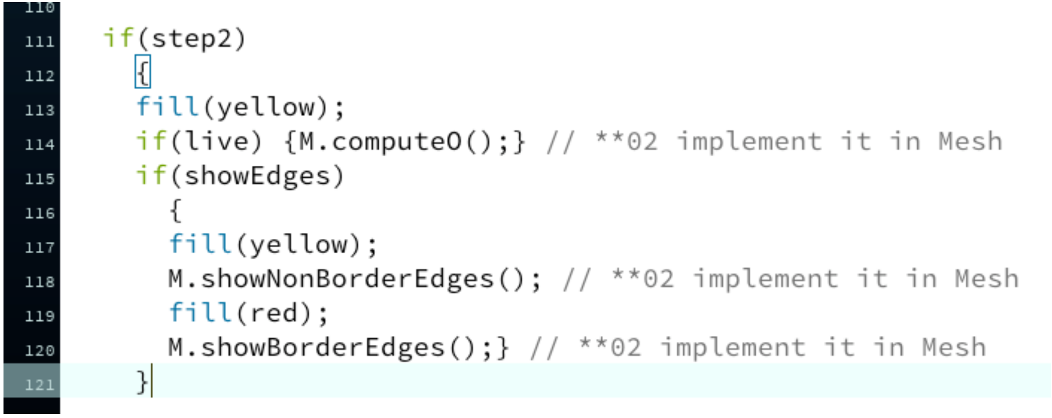
3 Step 2: Compute the O table and show border/interior edges

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:



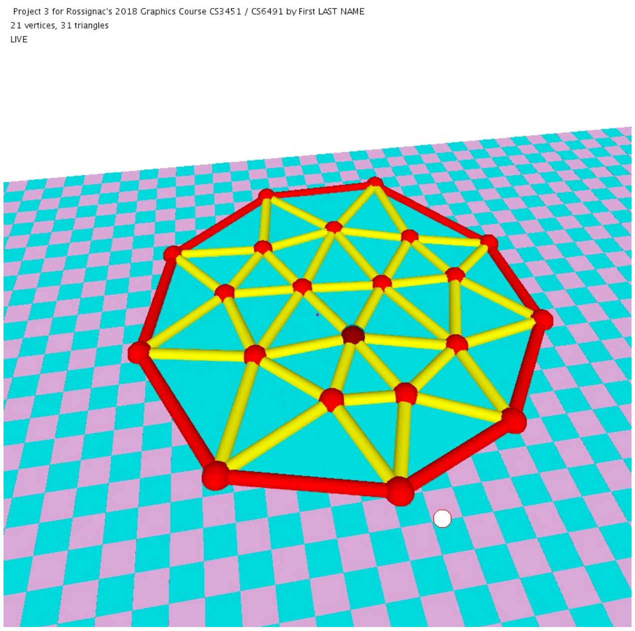
```
  void computeO() // **02 implement it
    {
    // **02 implement it
    }
    
  void showBorderEdges()  // draws all border edges of mesh
    {
    // **02 implement;
    }

  void showNonBorderEdges() // draws all non-border edges of mesh
    {
    // **02 implement
    }
```

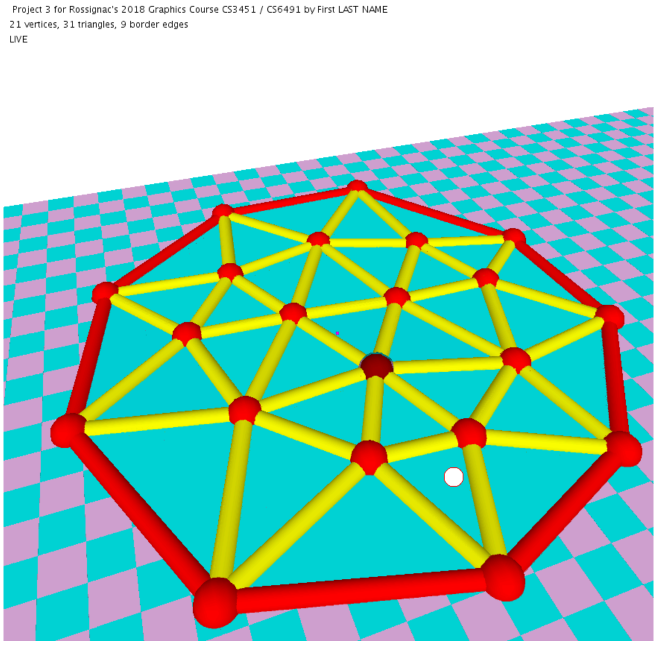
It is called in main for step 2 (when you press '2').



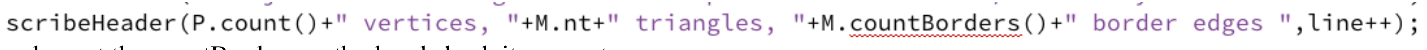
```
  if(step2)
    {
    fill(yellow);
    if(live) {M.computeO();} // **02 implement it in Mesh
    if(showEdges)
      {
      fill(yellow);
      M.showNonBorderEdges(); // **02 implement it in Mesh
      fill(red);
      M.showBorderEdges();} // **02 implement it in Mesh
    }
```

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
21 vertices, 31 triangles, 9 border edges
LIVE

Replace the displayed text by:

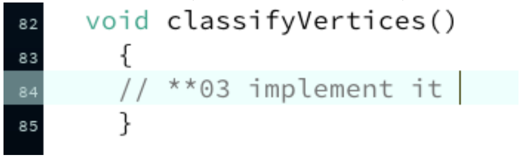


```
 scribeHeader(P.count()+" vertices, "+M.nt+" triangles, "+M.countBorders()+" border edges ",line++);
```

Implement the countBorders method and check its correctness.

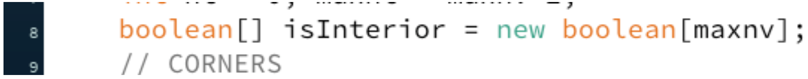
4 Step 3: Identify interior/border vertices

IN tab mesh, add your code to identify interior vertices



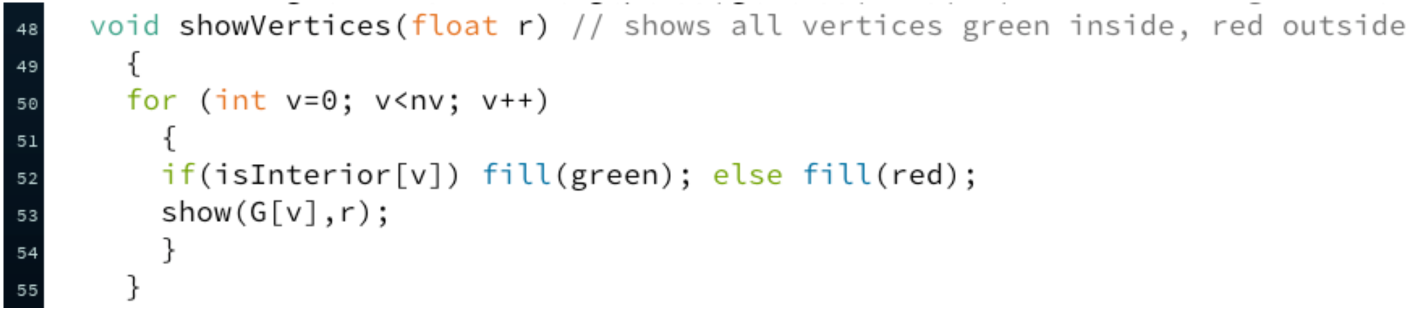
```
void classifyVertices()
  {
  // **03 implement it |
  }
```

Using the isInterior[] array.



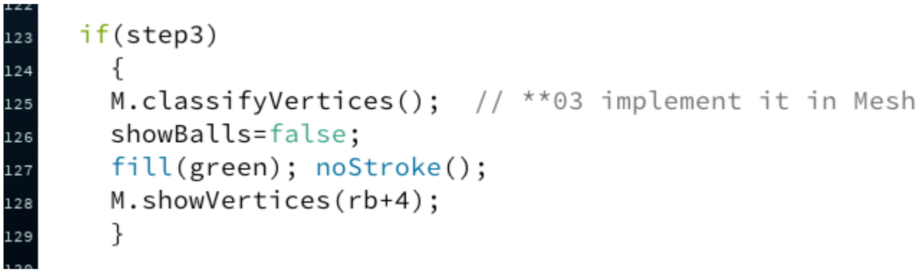
```
  boolean[] isInterior = new boolean[maxnv];
  // CORNERS
```

It is used in mesh by the showVertices method



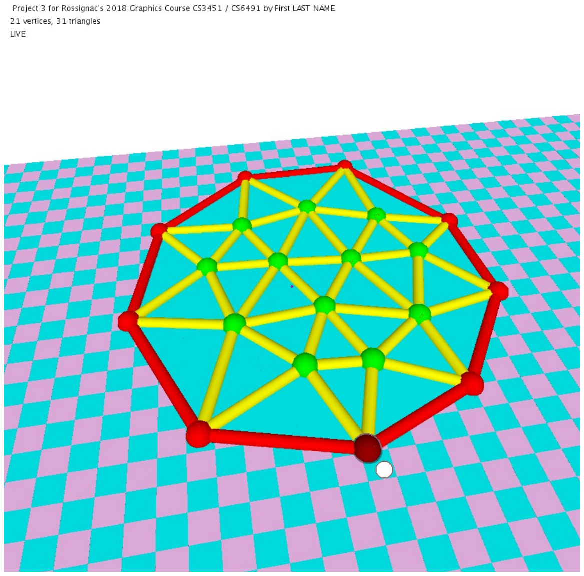
```
void showVertices(float r) // shows all vertices green inside, red outside
  {
  for (int v=0; v<nv; v++)
    {
    if(isInterior[v]) fill(green); else fill(red);
    show(G[v],r);
    }
  }
```

Which is invoked from main



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

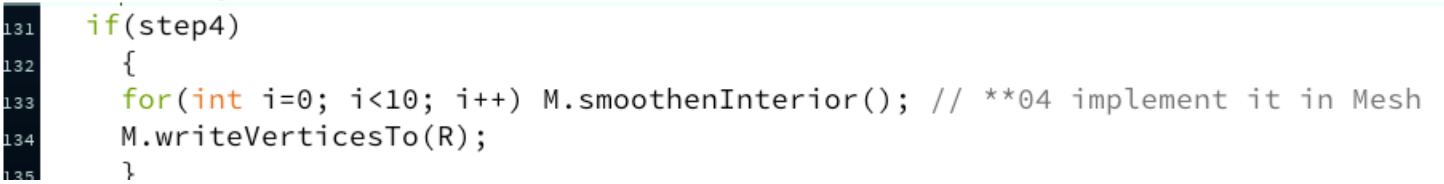
To display border edges in red and interior edges in yellow.



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

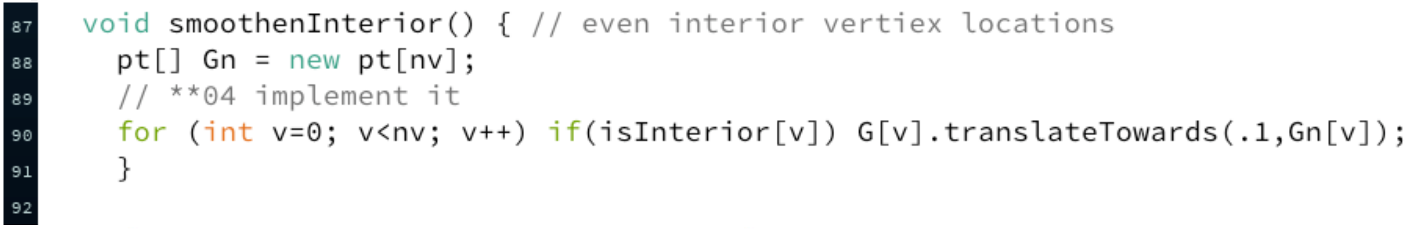
5 Step 4: Smoothen the interior vertices

This is called for step4 in main

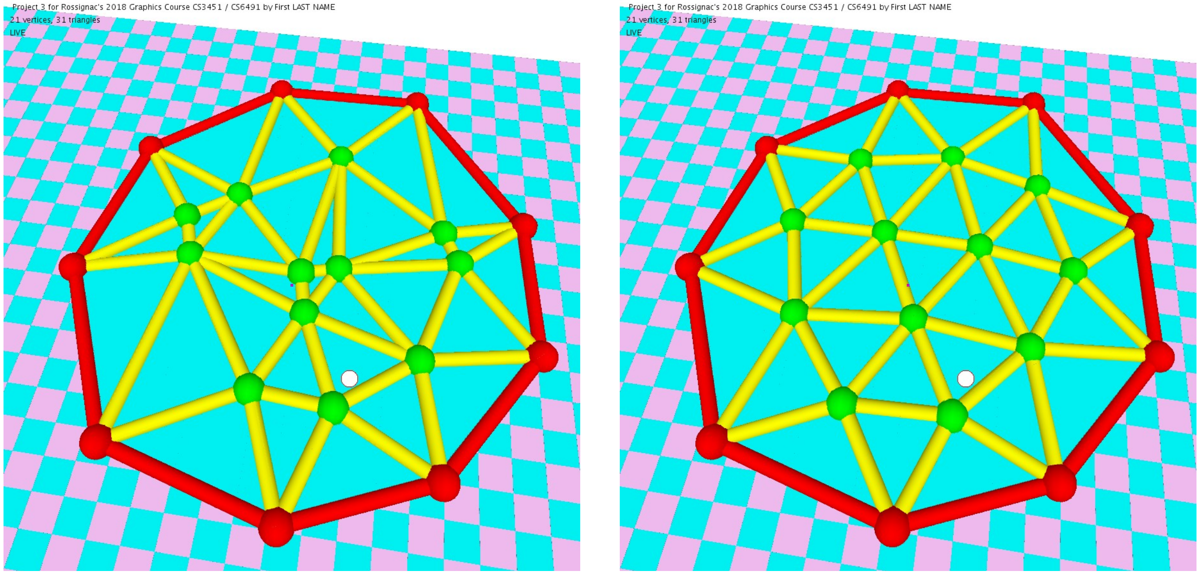


```
  if(step4)
    {
    for(int i=0; i<10; i++) M.smoothenInterior(); // **04 implement it in Mesh
    M.writeVerticesTo(R);
    }
```

Implement it in mesh

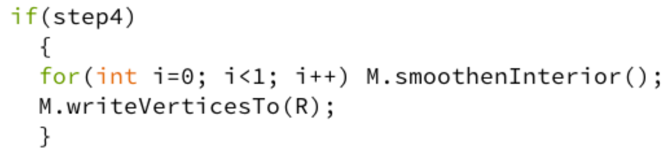


```
  void smoothenInterior() { // even interior vertiex locations
    pt[] Gn = new pt[nv];
    // **04 implement it
    for (int v=0; v<nv; v++) if(isInterior[v]) G[v].translateTowards(.1,Gn[v]);
    }
```


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
21 vertices, 31 triangles
LIVE

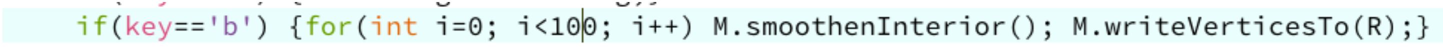
Make it work in Live mode and in your video, show first its action on a static mesh and then show what happens when you edit some of the border vertices, so that the connectivity of M changes.

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.smoothenInterior();
    M.writeVerticesTo(R);
    }
```

Change the 'b' action in zgui to:

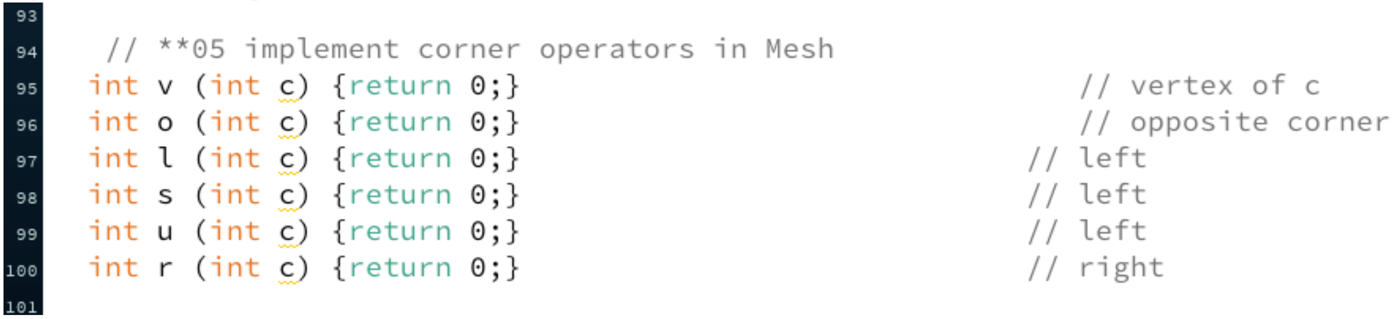


```
    if(key=='b') {for(int i=0; i<100; i++) M.smoothenInterior(); M.writeVerticesTo(R);}
```

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.

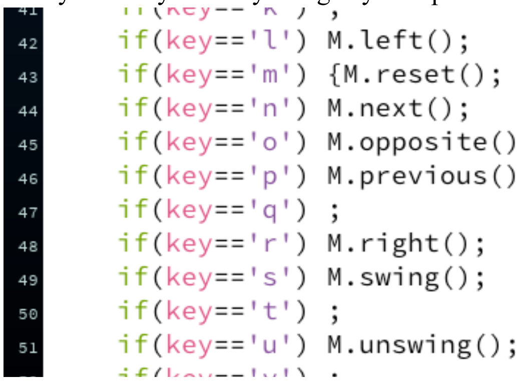
6 Step 5: Corner operators

Implement the corner operators in mesh:



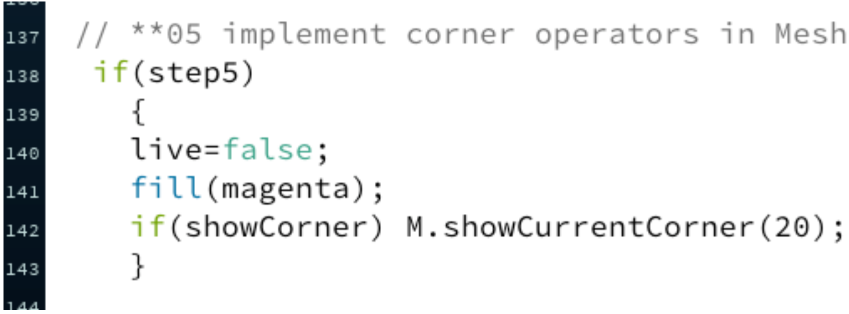
```
  // **05 implement corner operators in Mesh
 int v (int c) {return 0;}                           // vertex of c
 int o (int c) {return 0;}                           // opposite corner
 int l (int c) {return 0;}                        // left
 int s (int c) {return 0;}                        // left
 int u (int c) {return 0;}                        // left
 int r (int c) {return 0;}                        // right
```

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



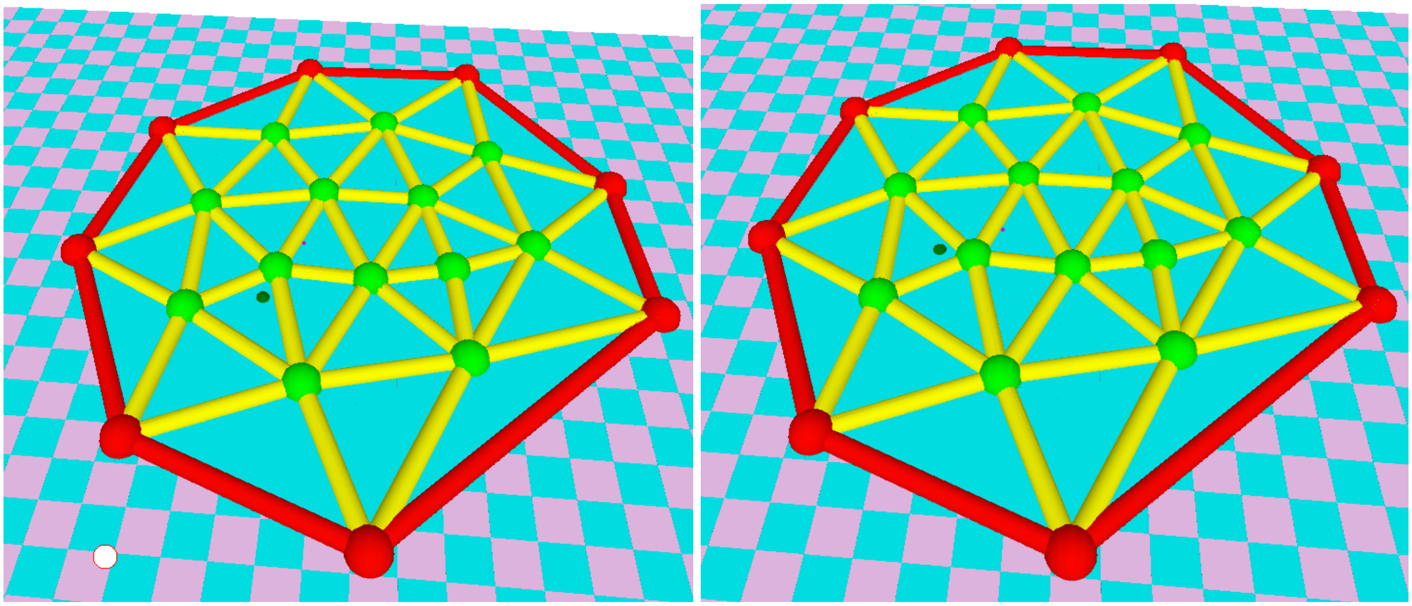
```
   if(key=='l') M.left();
   if(key=='m') {M.reset();
   if(key=='n') M.next();
   if(key=='o') M.opposite()
   if(key=='p') M.previous()
   if(key=='q') ;
   if(key=='r') M.right();
   if(key=='s') M.swing();
   if(key=='t') ;
   if(key=='u') M.unswing();
```

The current corner is displayed as a small dot

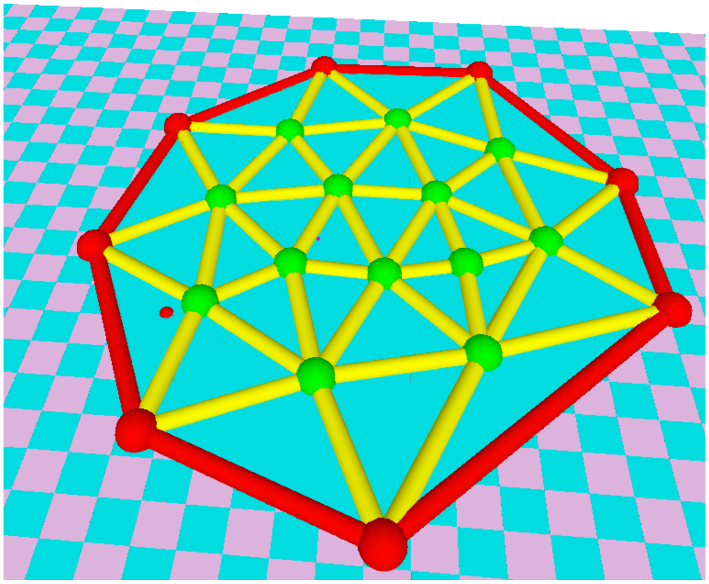


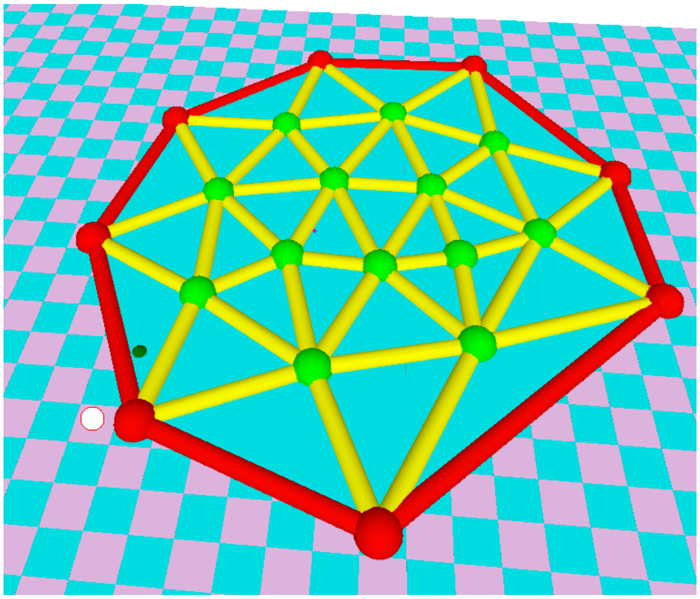
```
 // **05 implement corner operators in Mesh
  if(step5)
    {
    live=false;
    fill(magenta);
    if(showCorner) M.showCurrentCorner(20);
    }
```

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

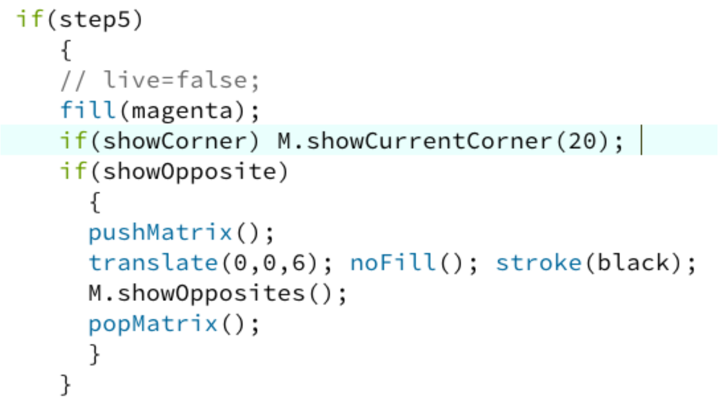


Make sure that the dot around c is colored red when c has no opposite.



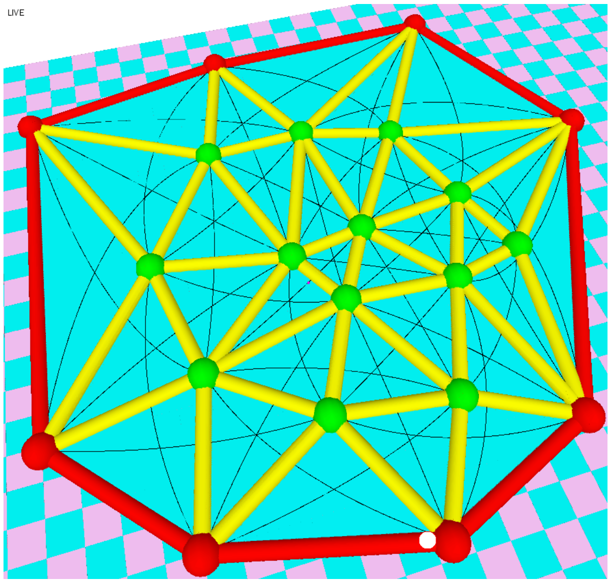


Modify step 5 as follows



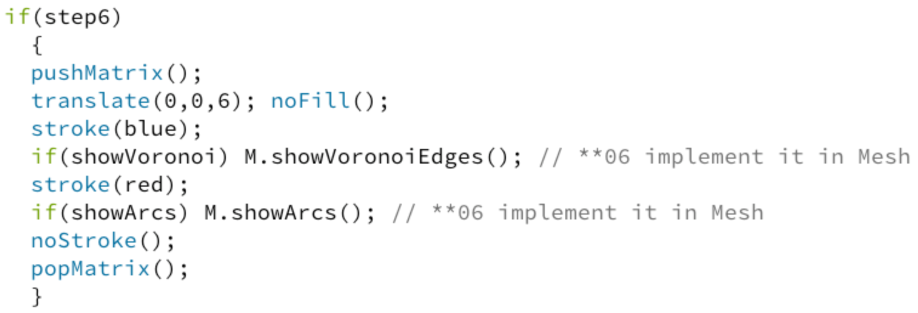
```
 if(step5)
    {
    // live=false;
    fill(magenta);
    if(showCorner) M.showCurrentCorner(20);
    if(showOpposite)
      {
      pushMatrix();
      translate(0,0,6); noFill(); stroke(black);
      M.showOpposites();
      popMatrix();
      }
    }
```

and implement the method showOpposites (template not provided) that draws parabola between the vertices of all pairs of opposite corners.



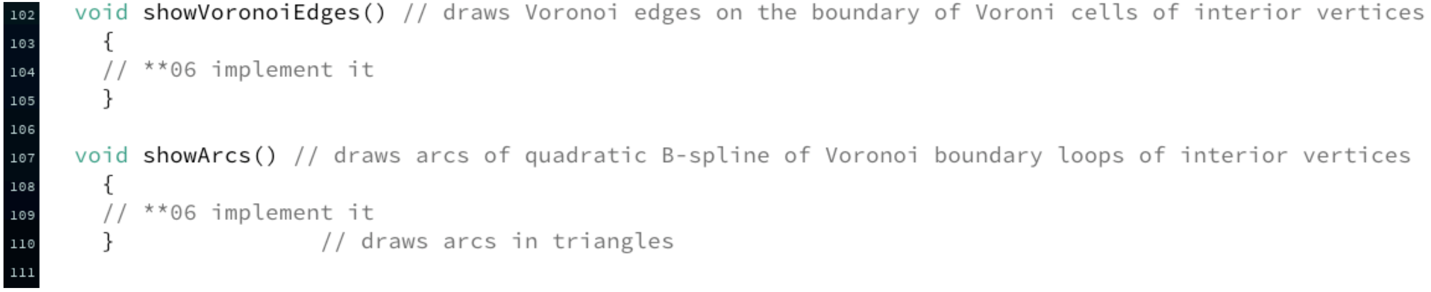
7 Step 6: Voronoi loops

To support step 6:



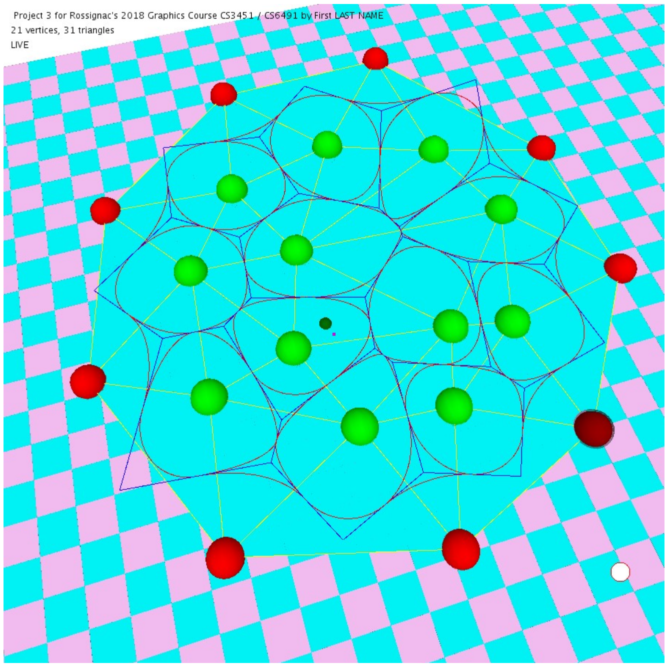
```
if(step6)
  {
  pushMatrix();
  translate(0,0,6); noFill();
  stroke(blue);
  if(showVoronoi) M.showVoronoiEdges(); // **06 implement it in Mesh
  stroke(red);
  if(showArcs) M.showArcs(); // **06 implement it in Mesh
  noStroke();
  popMatrix();
  }
```

Implement in mesh the methods

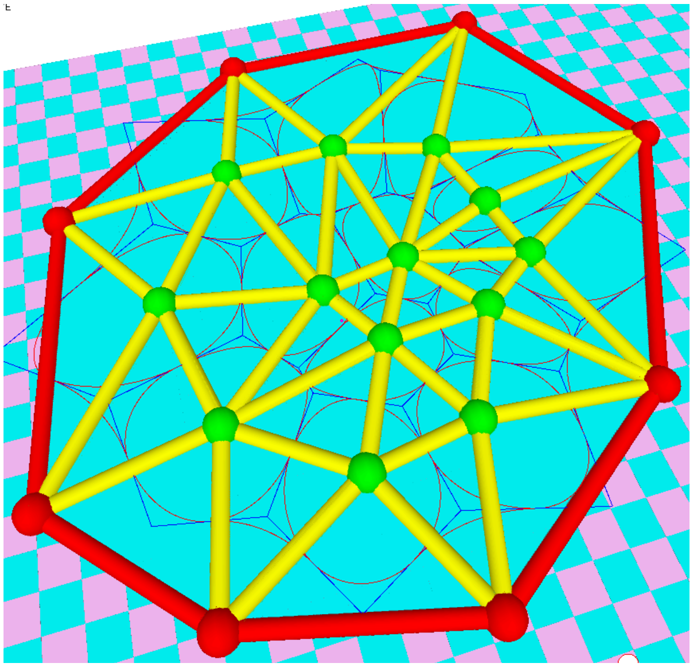


```
  void showVoronoiEdges() // draws Voronoi edges on the boundary of Voroni cells of interior vertices
    {
    // **06 implement it
    }

  void showArcs() // draws arcs of quadratic B-spline of Voronoi boundary loops of interior vertices
    {
    // **06 implement it
    }                 // draws arcs in triangles

```

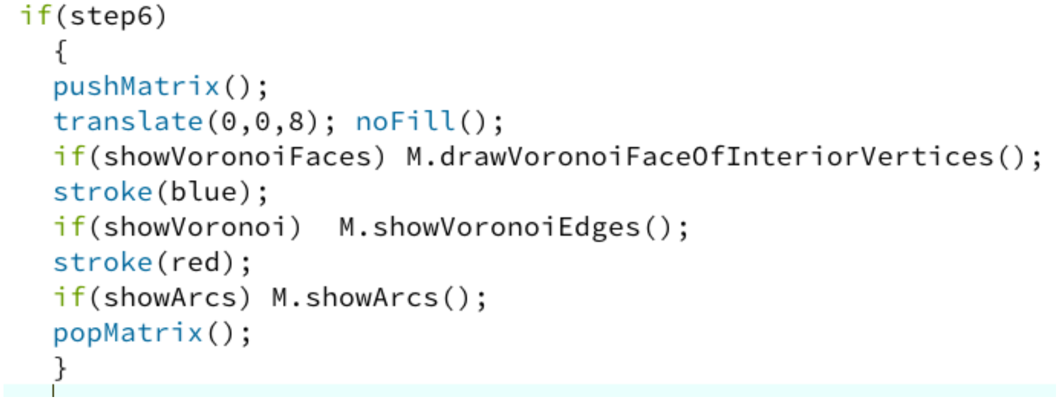
to display the Voronoi edges for Voronoi faces of interior vertices and to display their quadratic B-spline borders.



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

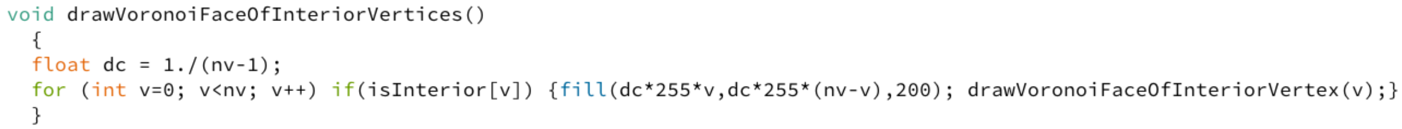


Modify step6 to fill each Voronoi face with a different color.



```
 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();
   }
```

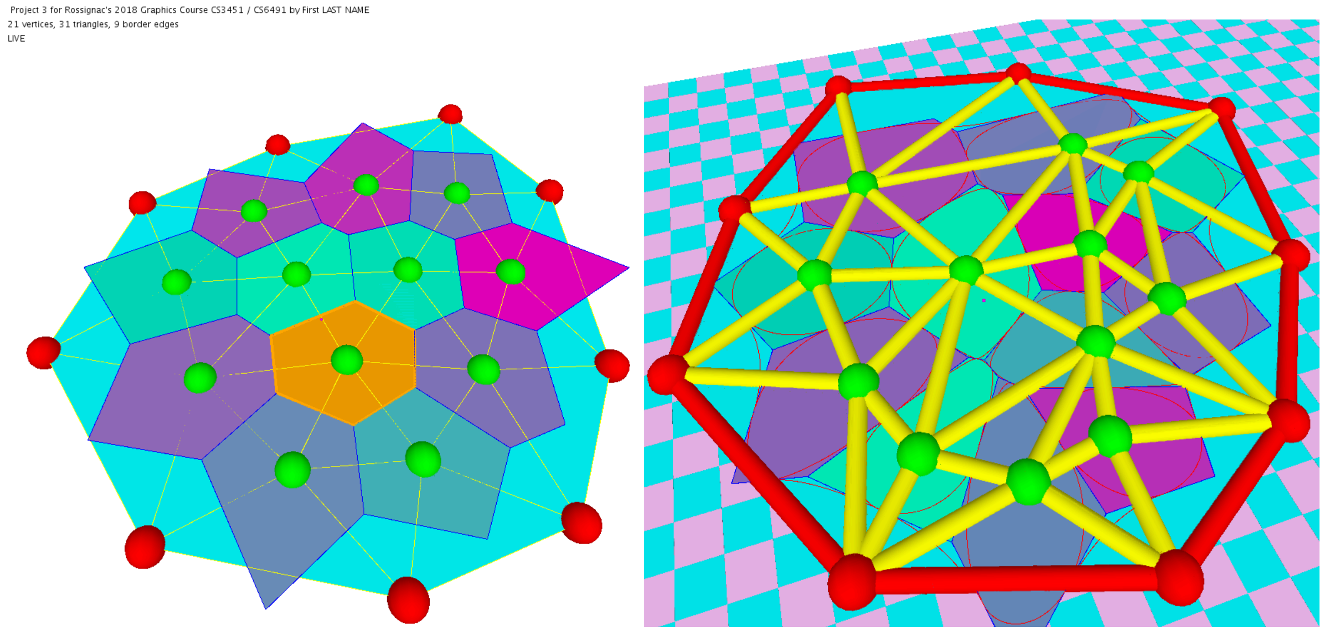
Implement the mesh method drawVoronoiFaceOfInteriorVertices. You may start from my implementation:



```
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.



Project 3 for Rossignac's 2018 Graphics Course CS3451 / CS6491 by First LAST NAME
21 vertices, 31 triangles, 9 border edges
LIVE

8 Submission

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