CMCS427 Notes
Example of creating polygonal mesh for parametric curve: cylinder

Given a parametric surface for a cylinder $p(u,v) = < Rcos(u), h * v, Rsin(u) >$
R is the radius of the cylinder, h the height.

Here is a systematic way to generate a polygon mesh from the parametric surface by sampling at regular intervals.

Step 1: Define two dimensional arrays of 3D points and vectors (note that for this document points and vertices are the same.) The constants n and m are the number of divisions of the mesh in the directions of u and v, respectively.

Point3D [n][m] vertices
Vector3D [n][m] normals

Step 2: Generate the vertices and normals for the mesh with nested loops.

```
for (i = 0; i < n; i++)
  for (j = 0; j < m; j++) {

     float u = i * 1/n * 2π     // u in the range [0,2π]
     float v = j * 1/m          // v in the range [0,1]

     float x = R*cos(u)     // Change this lines
     float y = h*v          // for a different surface
     float z = R*sin(u)

     float nx = cos(u)      // Also change these
     float ny = 0
     float nz = sin(u)

     vertices[i,j] = new Point3d(x,y,z)
     normals[i,j]  = new Vector3D(nx,ny,nz)

     }
```

Step 3: Generate the vertex, edge and face lists for the polygonal mesh from the vertices and normal arrays.

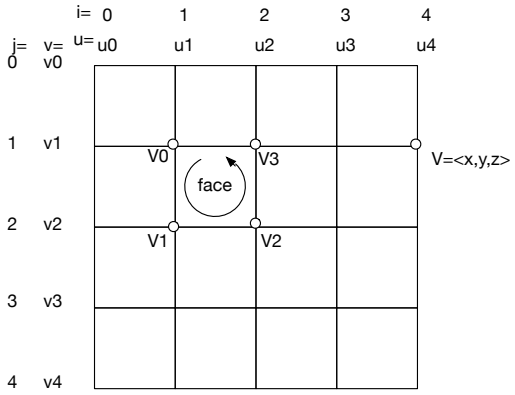
We could keep the vertices and normals in the 2D arrays if we're defining our own polygonal mesh data structure, but OpenGL and our indexed polygonal mesh structure from lecture both want a 1D array of vertices. We create these arrays and store the 2D data in row major order.

Point3D [n*m] vertexList
Vector3D [n*m] normalList

Now to generate the face list we look at four adjacent points in the 2D array and take them in counterclockwise order. These four points are a quad (4 pt face) on the surface. The face below would be given by

V0 = vertices[1,1]
V1 = vertices[1,2]
V2 = vertices[2,2]
V3 = vertices[2,1]

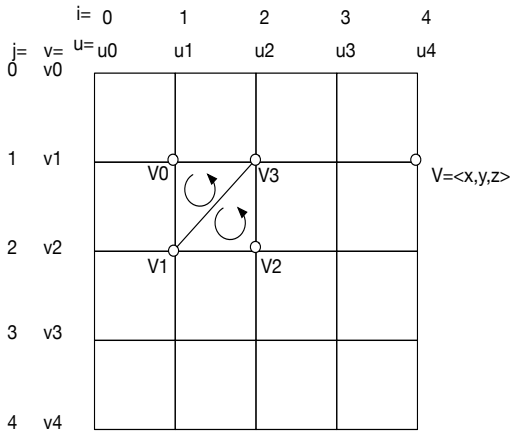
The new face would include V0, V1, V2, V3 and the corresponding normals



If the vertices are in a 1D array, then the indices for this face would be i, i+1, i+m, i+m+1

An edge list, if wanted, are the pairs of adjacent vertices in horizontal and vertical directions.

If you want to generate a triangular mesh you can create two triangles from each quad.



What happens at the edges of the arrays depends on the nature of the surface. A cylinder joins the left and right edges, and leaves the top and bottom open. A cone is similar, but joins the vertices at the top in the apex. A sphere joins the top and bottom at the north and south poles. A torus joins the left and right edges, and the top and bottom. A bilinear patch leaves all four sides as a separate boundaries.