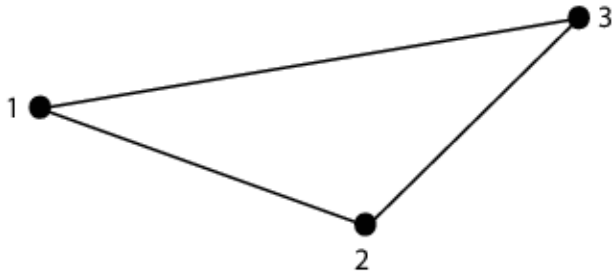
Problem Description:

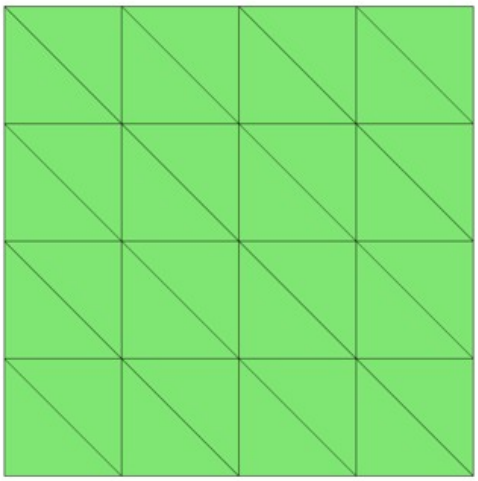
A triangle (or *element*) can be described as a series of *nodes* and nodal connections. For example, given three points in 3D space, labeled points 1, 2 and 3 below:



we can say that a triangular “element” is created by connecting “nodes” #1, #2, and #3. The data structures that you are provided implement this basic concept.

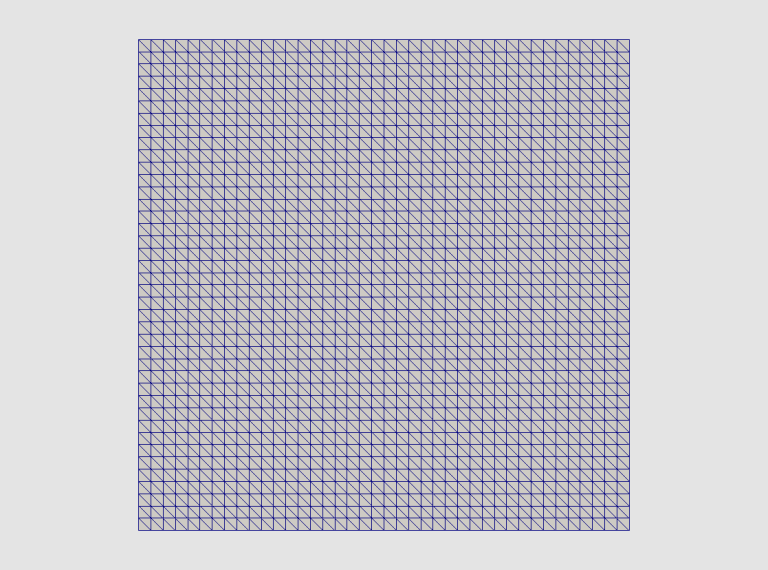
- Element.* – implementation of the (triangular) element class;
- Node.* – implementation of the node (3D point) class;
- Vector.*;Kpoint.* - supporting classes;

Now that we can represent triangles and nodes in 3D space, we want to create a grid / mesh as below:



This grid contains 5 x 5 = 25 nodes and 4 x 4 x 2 = 32 triangles (elements). Let's assume the length of the above square domain is 200 ft x 200 ft. Then, we can control the number of triangles using a uniform grid discretization. For example, in the above image, we say that we have a uniform grid discretization of 50 ft. This means that we divide the horizontal distance by 4 (=200/50) and the vertical distance by 4 (=200/50).

Then, with the same 200 ft x 200 ft domain, if I specify a uniform grid discretization of 5 ft, I get the grid below:



This grid contains 41 (200/5+1) x 41 = 1681 nodes and 40 (200/5) x 40 x 2 = 3200 elements.

We will be using this concept of uniform grid discretization to create grids of various sizes and then perform calculations based upon these grids. Upon arrival, you will be provided with a main.cpp file which uses these classes to build a mesh for different grid discretizations and perform some basic linear algebra calculations.