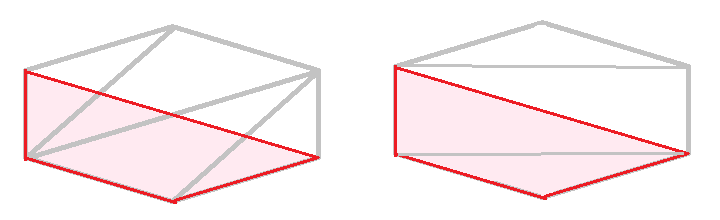
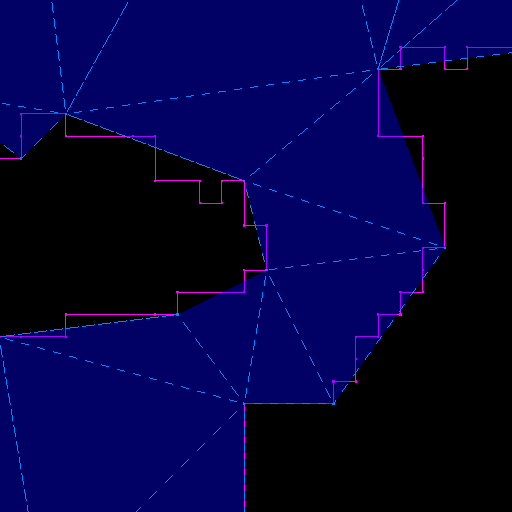
Here's one of the methods I came up with when doing navmesh for an RTS game. Note that it is homebrew, no third-party tools were used, it took me about 3 weeks to implement and bugfix:

1. **Use**[**Marching Squares**](http://en.wikipedia.org/wiki/Marching_squares)**algorithm to convert obstacle tiles into outlines**. Note that map edges is an outline too and need to be included as well.
2. **Reduce number of points in outlines using**[**Douglas-Peucker**](http://en.wikipedia.org/wiki/Ramer%E2%80%93Douglas%E2%80%93Peucker_algorithm)**algorithm** (purple lines on the bottom picture)
3. **Feed all points into**[**Delaunay**](http://en.wikipedia.org/wiki/Delaunay_triangulation)**triangulation** (to get most uniform triangles)
4. **Add additional points in empty areas and along the map edges** (to get more even navmesh)
5. **Check along obstacle outlines and flip polygons produced by Delaunay to match outlines.** - Often Delaunay could place triangles (grey) mismatching your outlines (red), then you need to detect and flip them. Adjoining them back into a polygon, split it along outline(s) and triangulate it manually [](https://i.stack.imgur.com/DWXL4.png)
6. **Clip obstacles innards** - remove polygons that are within obstacles (pink on the picture above)
7. **Fill in connectivity data between remaining triangles and vertices as you need** - that's your navmesh.

Result:



Źródło:

<https://gamedev.stackexchange.com/questions/38721/how-can-i-generate-a-navigation-mesh-for-a-tile-grid>