# Triangle文件数据格式

Triangle输入文件（2个）：

.poly文件：

* First line: <# of vertices> <dimension (must be 2)> <# of attributes> <# of boundary markers (0 or 1)>
* Following lines: <vertex #> <x> <y> [attributes] [boundary marker]
* One line: <# of segments> <# of boundary markers (0 or 1)>
* Following lines: <segment #> <endpoint> <endpoint> [boundary marker]
* One line: <# of holes>
* Following lines: <hole #> <x> <y>
* Optional line: <# of regional attributes and/or area constraints>
* Optional following lines: <region #> <x> <y> <attribute> <maximum area>

A .poly file represents a PSLG, as well as some additional information. PSLG stands for [Planar Straight Line Graph](https://www.cs.cmu.edu/~quake/triangle.defs.html#pslg), a term familiar to computational geometers. By definition, a PSLG is just a list of vertices and segments. A .poly file can also contain information about holes and concavities, as well as regional attributes and constraints on the areas of triangles.

The first section lists all the vertices, and is identical to the format of [.node](https://www.cs.cmu.edu/~quake/triangle.node.html) files. <# of vertices> may be set to zero to indicate that the vertices are listed in a separate [.node](https://www.cs.cmu.edu/~quake/triangle.node.html) file; .poly files produced by Triangle always have this format. A vertex set represented this way has the advantage that it may easily be triangulated with or without segments (depending on whether the .poly or .node file is read).

The second section lists the [segments](https://www.cs.cmu.edu/~quake/triangle.defs.html#segment). Segments are edges whose presence in the triangulation is enforced (although each segment may be subdivided into smaller edges). Each segment is specified by listing the indices of its two endpoints. This means that you must include its endpoints in the vertex list. Each segment, like each vertex, may have a [boundary marker](https://www.cs.cmu.edu/~quake/triangle.markers.html).

The third section lists holes (and concavities, if [-c](https://www.cs.cmu.edu/~quake/triangle.c.html) is selected) in the triangulation. Holes are specified by identifying a point inside each hole. After the triangulation is formed, Triangle creates holes by eating triangles, spreading out from each hole point until its progress is blocked by PSLG segments; you must be careful to enclose each hole in segments, or your whole triangulation might be eaten away. If the two triangles abutting a segment are eaten, the segment itself is also eaten. Do not place a hole directly on a segment; if you do, Triangle will choose one side of the segment arbitrarily.

.area files

* First line: <# of triangles>
* Following lines: <triangle #> <maximum area>

An .area file associates with each triangle a maximum area that is used for [mesh refinement](https://www.cs.cmu.edu/~quake/triangle.refine.html). As with other file formats, every triangle must be represented, and they must be numbered consecutively (from one or zero). Blank lines and comments prefixed by `#' may be placed anywhere. A triangle may be left unconstrained by assigning it a negative maximum area.

Triangle输出文件（4个）：

.node files

* First line: <# of vertices> <dimension (must be 2)> <# of attributes> <# of boundary markers (0 or 1)>
* Remaining lines: <vertex #> <x> <y> [attributes] [boundary marker]

Blank lines and comments prefixed by `#' may be placed anywhere. Vertices must be numbered consecutively, starting from one or zero.

The attributes, which are typically floating-point values of physical quantities (such as mass or conductivity) associated with the nodes of a finite element mesh, are copied unchanged to the output mesh. If [-q](https://www.cs.cmu.edu/~quake/triangle.q.html), [-a](https://www.cs.cmu.edu/~quake/triangle.a.html), -u, or -s is selected, each new [Steiner point](https://www.cs.cmu.edu/~quake/triangle.defs.html#steiner) added to the mesh will have quantities assigned to it by linear interpolation.

If the fourth entry of the first line is `1', the last column of the remainder of the file is assumed to contain boundary markers. [Boundary markers](https://www.cs.cmu.edu/~quake/triangle.markers.html) are used to identify boundary vertices and vertices resting on [PSLG](https://www.cs.cmu.edu/~quake/triangle.defs.html#pslg) [segments](https://www.cs.cmu.edu/~quake/triangle.defs.html#segment). The .node files produced by Triangle contain boundary markers in the last column unless they are suppressed by the -B switch.

.ele files

* First line: <# of triangles> <nodes per triangle> <# of attributes>
* Remaining lines: <triangle #> <node> <node> <node> ... [attributes]

Blank lines and comments prefixed by `#' may be placed anywhere. Triangles must be numbered consecutively, starting from one or zero. Nodes are indices into the corresponding [.node](https://www.cs.cmu.edu/~quake/triangle.node.html) file. The first three nodes are the corner vertices, and are listed in counterclockwise order around each triangle. (The remaining nodes, if any, depend on the type of finite element used.)

As in [node](https://www.cs.cmu.edu/~quake/triangle.node.html) files, the attributes are typically floating-point values of physical quantities (such as mass or conductivity) associated with the elements (triangles) of a finite element mesh. Because there is no simple mapping from input to output triangles, an attempt is made to interpolate attributes, which may result in a good deal of diffusion of attributes among nearby triangles as the triangulation is refined. Attributes do not diffuse across segments, so attributes used to identify segment-bounded regions remain intact.

In output .ele files, all triangles have three nodes each unless the -o2 switch is used, in which case [subparametric quadratic elements](https://www.cs.cmu.edu/~quake/triangle.highorder.html) with six nodes are generated. The fourth, fifth, and sixth nodes lie on the midpoints of the edges opposite the first, second, and third vertices.

.edge files

* First line: <# of edges> <# of boundary markers (0 or 1)>
* Following lines: <edge #> <endpoint> <endpoint> [boundary marker]

Blank lines and comments prefixed by `#' may be placed anywhere. Edges are numbered consecutively, starting from one or zero. Endpoints are indices into the corresponding [.node](https://www.cs.cmu.edu/~quake/triangle.node.html) file.

Triangle can produce .edge files (use the -e switch), but cannot read them. The optional column of [boundary markers](https://www.cs.cmu.edu/~quake/triangle.markers.html) is suppressed by the -B switch.

In [Voronoi diagrams](https://www.cs.cmu.edu/~quake/triangle.defs.html#voronoi), one also finds a special kind of edge that is an infinite ray with only one endpoint. For these edges, a different format is used:

<edge #> <endpoint> -1 <direction x> <direction y>

The `direction' is a floating-point vector that indicates the direction of the infinite ray.

.neigh files

* First line: <# of triangles> <# of neighbors per triangle (always 3)>
* Following lines: <triangle #> <neighbor> <neighbor> <neighbor>

Blank lines and comments prefixed by `#' may be placed anywhere. Triangles are numbered consecutively, starting from one or zero. Neighbors are indices into the corresponding [.ele](https://www.cs.cmu.edu/~quake/triangle.ele.html) file. An index of -1 indicates no neighbor (because the triangle is on an exterior boundary). The first neighbor of triangle i is opposite the first corner of triangle i, and so on.

Triangle can produce .neigh files (use the -n switch), but cannot read them.

# POLY Files

**POLY** is a data directory which contains examples of POLY files, a format used by Jonathan Shewchuk to define PSLG's, planar straight line graphs, for use with his program [TRIANGLE](https://people.sc.fsu.edu/~jburkardt/c_src/triangle/triangle.html).

A Planar Straight Line Graph (PSLG) is a set of vertices and segments. Segments are simply edges, whose endpoints are vertices in the PSLG. Segments may intersect each other only at their endpoints.

## POLY File Characteristics:

* ASCII
* 2D
* vertices are specified by coordinates.
* line segments are specified by listing the indices of pairs of vertices.
* a hole may be specified by listing the coordinates of a point inside the hole.
* No compression
* 1 image

Comments are prefixed by the character '#'. Everything from the comment character to the end of the line is ignored.

Vertices, segments, holes, and regions must be numbered and listed consecutively, starting from either 1 or 0.

The first line lists

* The number of vertices (this is sometimes set to 0, to indicate that the vertices should be read from a NODE file);
* The spatial dimension, which must be 2;
* The number of vertex attributes;
* The number of vertex boundary markers, which must be 0 or 1.

The vertex records must follow, with the format:

* vertex index (these must be consecutive, starting either from 0 or 1);
* X and Y coordinates;
* The vertex attributes (if any);
* The vertex boundary marker (if any).

The next line lists

* The number of segments;
* The number of segment boundary markers (0 or 1).

Segments should not cross each other; vertices should only lie on the ends of segments, and are never contained inside a segment.

The segments records must follow, with the format:

* segment index;
* start vertex, end vertex;
* Boundary marker (if any).

The third section lists holes (and concavities, if -c is selected) in the triangulation. Holes are specified by identifying a point inside each hole. After the triangulation is formed, Triangle creates holes by eating triangles, spreading out from each hole point until its progress is blocked by PSLG segments; you must be careful to enclose each hole in segments, or your whole triangulation might be eaten away. If the two triangles abutting a segment are eaten, the segment itself is also eaten. Do not place a hole directly on a segment; if you do, Triangle chooses one side of the segment arbitrarily.

The next line lists

* The number of holes.

The hole records must follow, with the format:

* hole index;
* X coordinate, Y coordinate of some point within the hole.

The optional fourth section lists regional attributes (to be assigned to all triangles in a region) and regional constraints on the maximum triangle area. Triangle reads this section only if the -A switch is used or the -a switch is used without a number following it, and the -r switch is not used. Regional attributes and area constraints are propagated in the same manner as holes; you specify a point for each attribute and/or constraint, and the attribute and/or constraint affects the whole region (bounded by segments) containing the point. If two values are written on a line after the x and y coordinate, the first such value is assumed to be a regional attribute (but is only applied if the -A switch is selected), and the second value is assumed to be a regional area constraint (but is only applied if the -a switch is selected). You may specify just one value after the coordinates, which can serve as both an attribute and an area constraint, depending on the choice of switches. If you are using the -A and -a switches simultaneously and wish to assign an attribute to some region without imposing an area constraint, use a negative maximum area.

The next line is optional. If given, it lists

* The number of region attributes.

The optional regional attributes records must follow, with the format:

* region index;
* X coordinate, Y coordinate of a point in the region;
* Attributes (if any);
* Maximum area of triangles in the region;

**A Sample POLY file:**

Here is a sample file *box.poly* describing a square with a square hole.

# A box with eight vertices in 2D, no attributes, one boundary marker.

8 2 0 1

# Outer box has these vertices:

1 0 0 0

2 0 3 0

3 3 0 0

4 3 3 33 # A special marker for this vertex.

# Inner square has these vertices:

5 1 1 0

6 1 2 0

7 2 1 0

8 2 2 0

# Five segments with boundary markers.

5 1

1 1 2 5 # Left side of outer box.

# Square hole has these segments:

2 5 7 0

3 7 8 0

4 8 6 10

5 6 5 0

# One hole in the middle of the inner square.

1

1 1.5 1.5