On the Triangulation of Convex Polygons Presenting T–Vertices

P. Cignoni†, C. Montani†, R. Scopigno⋆
†IEI–CNR, Pisa, Italy – {cignoni,montani}@iei.pi.cnr.it
⋆CNUCE–CNR, Pisa, Italy – r.scopigno@cnuce.cnr.it

Technical Report

Abstract

A technique to triangulate planar convex polygons presenting T–vertices is described. Simple strip or fan tessellation of a polygon with T–vertices can result in zero–area triangles and compromise the rendering process. Our technique splits such a polygon into one triangle strip and, at most, one triangle fan. The utility is particularly useful in multiresolution or adaptive representation of polygonal surfaces and surfaces' simplification.

Triangulating a planar convex polygon is a simple, but important task. Though modern graphic languages provide for the planar convex polygon as one of the elementary geometric primitives, better performances are obtained by breaking down polygons into compact sequences of triangles. Triangle–strip (Fig. 1.a) and triangle–fan (Fig. 1.b) are compact and efficient methods to represent triangles' sequences and are provided as output primitives in most graphics libraries (OpenGL [7] and, partially, PHIGS [2], for example).

The triangulation becomes a little more complicated when one or more vertices of the polygon are T–vertices, i.e. there exist three or more aligned vertices. This problem arise, for example, in the adaptive representation of polygonal surfaces ([6], [3]). One of the simplest algorithm to avoid cracks (small holes between polygons at different levels of resolution) is to move some of the vertices of the lower resolution polygons onto the boundary of the higher ones, so introducing T–vertices. In these cases, naively creating a triangle strip or fan, as per Fig. 1, can yield degenerate (zero-area) triangles.

In this note, we present a simple technique which permits to split convex polygons presenting T–vertices into one triangle–strip and, possibly, one triangle–fan. We assume that the user is acquainted with the existence of T–vertices and with their position on the polygon's boundary (this is not a limi-

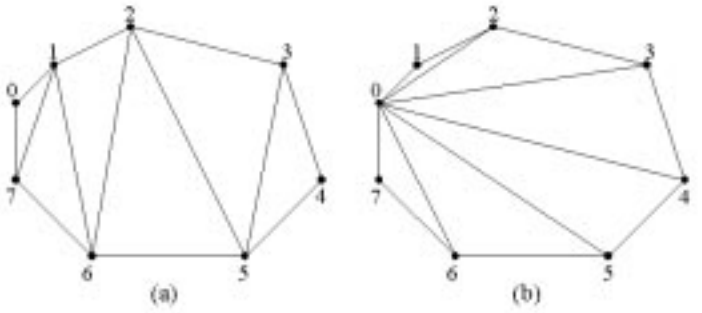

Figure 1: Triangulating a convex polygon: (a) a triangle–strip with vertices $v_0, v_7, v_1, v_6, v_2, v_5, v_3, v_4$ and (b) a triangle–fan with vertices $v_0, v_7, v_6, v_5, v_4, v_3, v_2, v_1$.

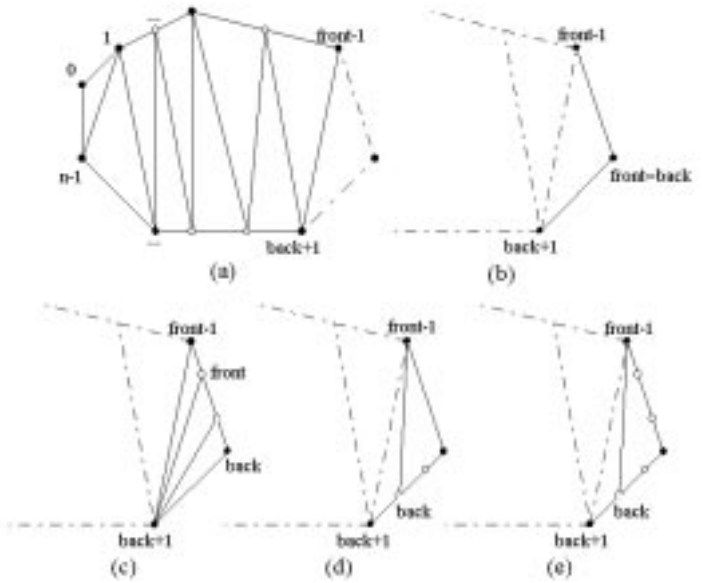

Figure 2: Triangulating a convex polygon presenting T–vertices (the vertices with a white circle): (a) the construction of the strip interrupts when just one unprocessed regular vertex (the black vertex on the right) is left; (b) if the regular vertex ($v_{front=back}$) is the only vertex to be processed, then it is added as last point of the strip; (c) the next vertex to be processed (v_{back}) is the regular one. A triangle–fan is built (in the example the fan $v_{back+1}, v_{back}, \cdots, v_{front}, v_{front-1}$); (d) and (e) the candidate vertex (v_{back}) is a T–vertex. It is added to the strip and the algorithm continues until either (b) or (c) occurs.

tation in the practical cases), and that the starting vertex of the chain is not a T–vertex (a simple circular shift of the vertices can solve this problem).

The algorithm is shown in Fig. 2. In it, the indices *front* and *back* assumes the values $0, 1, \cdots$, and $n-1, n-2, \cdots$, respectively. Starting from a *regular* (non T–vertex) vertex of the polygon the algorithm adds vertices to the triangle–strip in the usual zigzag way (Fig. 2.a) until just one unprocessed regular vertex is left. Then, three different cases can occur:

1. the regular vertex is the only vertex to be processed (Fig. 2.b). It can be added as last point of the strip. There is no need for a triangle–fan;

2. the vertex candidate for the insertion into the strip is the regular one (Fig. 2.c). A triangle–fan is built. The sequence of vertices of the fan assures the correct triangles' orientation;

3. the vertex candidate for the insertion into the strip is a T–vertex (Fig. 2.d or 2.e). The construction of the triangle–strip continues until one of the previous situations occurs.

Our simple triangulation is computed in linear time. A Delaunay triangulation of a convex polygon can also be computed in linear time [1]; our technique is not optimal with respect the *best* shape of the possible resulting triangles. However, we think the simplicity and effectiveness of our solution make it valuable in practical cases. The algorithm has been extensively used in the implementation of the DiscMC [5] [4], a public domain software for the extraction, simplification, and rendering of isosurfaces from high resolution regular datasets[1].

References

[1] Aggarwal, A., Guibas, L. J., Saxe, J., and Shor, P. W. A linear time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete and Computational Geometry 4* (1989), 591–604.

[2] Gaskins, T. *PHIGS Programming Manual*. O'Reilly & Associates, 1992.

[3] Haemer, M. D., and Zyda, M. Simplification of objects rendered by polygonal approximations. *Computers & Graphics 15*, 2 (1991), 175–184.

[4] Montani, C., Scateni, R., and Scopigno, R. Discretized Marching Cubes. In *Visualization '94 Proceedings* (1994), R. Bergeron and A. Kaufman, Eds., IEEE Computer Society Press, pp. 281–287.

[5] Montani, C., Scateni, R., and Scopigno, R. Decreasing iso-surface complexity via discretized fitting. Tech. Rep. B4-37-11-95, IEI - CNR, Pisa, Italy, November 1995.

[1] DiscMC is available on `http://vcg.iei.pi.cnr.it/swOnTheWeb.html`.

[6] Muller, H., and Stark, M. Adaptive generation of surfaces in volume data. *The Visual Computer 9*, 4 (1993), 182–199.

[7] Neider, J., Davis, T., and Woo, M. *OpenGL Programming Guide.* Addison Wesley, 1993.