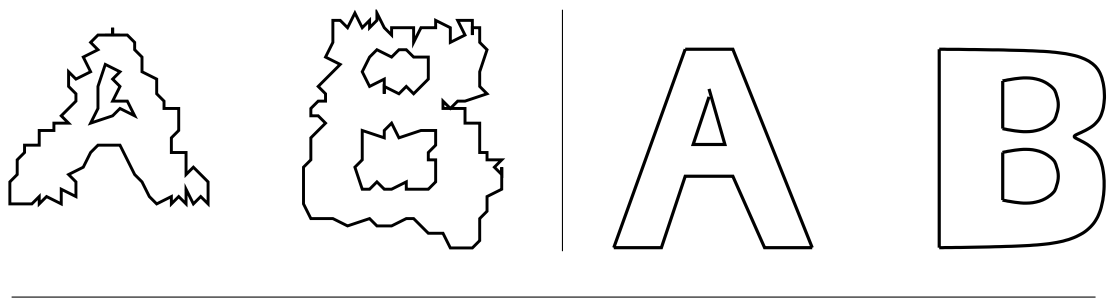

17.12 Simplifying Polygons

Input description: A polygon or polyhedron p, with n vertices.

Problem description: Find a polygon or polyhedron p' containing only n' vertices, such that the shape of p' is as close as possible to p.

Discussion: Polygon simplification has two primary applications. The first involves cleaning up a noisy representation of a shape, perhaps obtained by scanning a picture of an object. Simplifying it can remove the noise and reconstruct the original object. The second involves data compression, where we seek to reduce detail on a large and complicated object yet leave it looking essentially the same. This can be a big win in computer graphics, where the smaller model might be significantly faster to render.

Several issues arise in shape simplification:

- *Do you want the convex hull?* – The simplest simplification is the convex hull of the object's vertices (see Section 17.2 (page 568)). The convex hull removes all internal concavities from the polygon. If you were simplifying a robot model for motion planning, this is almost certainly a good thing. But using the convex hull in an OCR system would be disastrous, because the concavities of characters provide most of the interesting features. An 'X' would be identical to an 'I', since both hulls are boxes. Another problem is that taking the convex hull of a convex polygon does nothing to simplify it further.

- *Am I allowed to insert or just delete points?* – The typical goal of simplification is to to represent the object as well as possible using a given number of vertices. The simplest approaches do local modifications to the boundary in order to reduce the vertex count. For example, if three consecutive vertices form a small-area triangle or define an extremely large angle, the center vertex can be deleted and replaced with an edge without severely distorting the polygon.

Methods that only delete vertices quickly melt the shape into unrecognizability, however. More robust heuristics move vertices around to cover up the gaps that are created by deletions. Such "split-and-merge" heuristics can do a decent job, although nothing is guaranteed. Better results are likely by using the Douglas-Peucker algorithm, described next.

- *Must the resulting polygon be intersection-free?* – A serious drawback of incremental procedures is that they fail to ensure *simple* polygons, meaning they are without self-intersections. Thus "simplified" polygons may have ugly artifacts that may cause problems for subsequent routines. If simplicity is important, you should test all the line segments of your polygon for any pairwise intersections, as discussed in Section 17.8 (page 591).

  A polygon simplification approach that guarantees a simple approximation involves computing minimum-link paths. The *link distance* of a path between points s and t is the number of straight segments on the path. An as-the-crow-flies path has a link distance of one, while in general the link distance is one more than the number of turns on the path. The link distance between points s and t in a scene with obstacles is defined by the minimum link distance over all paths from s to t.

  The link distance approach "fattens" the boundary of the polygon by some acceptable error window ϵ (see Section 17.16 (page 617)) in order to construct a channel around the polygon. The minimum-link cycle in this channel represents the simplest polygon that never deviates from the original boundary by more than ϵ. An easy-to-compute approximation to link distance reduces it to breadth-first search, by placing a discrete set of possible turn points within the channel and connecting each pair of mutually visible points by an edge.

- *Are you given an image to clean up instead of a polygon to simplify?* – The conventional approach to cleaning up noise from a digital image is to take the Fourier transform of the image, filter out the high-frequency elements, and then take the inverse transform to recreate the image. See Section 13.11 (page 431) for details on the fast Fourier transform.

The Douglas-Puecker algorithm for shape simplification starts with a simple approximation and then refines it, instead of starting with a complicated polygon and trying to simplify it. Start by selecting two vertices v_1 and v_2 of polygon P, and propose the degenerate polygon v_1, v_2, v_1 as a simple approximation P'. Scan through each of the vertices of P, and select the one that is farthest from the corresponding edge of the polygon P'. Inserting this vertex adds the triangle to P' to minimize the maximum deviation from P. Points can be inserted until satisfactory results are achieved. This takes $O(kn)$ to insert k points when $|P| = n$.

Simplification becomes considerably more difficult in three dimensions. Indeed, it is NP-complete to find the minimum-size surface separating two polyhedra. Higher-dimensional analogies of the planar algorithms discussed here can be used to heuristically simplify polyhedra. See the Notes section.

Implementations: The Douglas-Peucker algorithm is readily implemented. Snoeyink provides a C implementation of his algorithm with efficient worst-case performance [HS94] at *http://www.cs.unc.edu/∼snoeyink/papers/DPsimp.arch.*

Simplification envelopes are a technique for automatically generating level-of-detail hierarchies for polygonal models. The user specifies a single error tolerance, and the maximum surface deviation of the simplified model from the original model, and a new, simplified model is generated. An implementation is available from *http://www.cs.unc.edu/∼geom/envelope.html.* This code preserves holes and prevents self-intersection.

QSlim is a quadric-based simplification algorithm that can produce high quality approximations of triangulated surfaces quite rapidly. It is available at *http://graphics.cs.uiuc.edu/∼garland/software.html.*

Yet another approach to polygonal simplification is based on simplifying and expanding the medial-axis transform of the polygon. The medial-axis transform (see Section 17.10 (page 598)) produces a skeleton of the polygon, which can be trimmed before inverting the transform to yield a simpler polygon. *Cocone* (*http://www.cse.ohio-state.edu/∼tamaldey/cocone.html*) constructs an approximate medial-axis transform of the polyhedral surface it interpolates from points in E^3. See [Dey06] for the theory behind *Cocone*. *Powercrust* [ACK01a, ACK01b] constructs a discrete approximation to the medial-axis transform, and then reconstructs the surface from this transform. When the point samples are sufficiently dense, the algorithm is guaranteed to produce a geometrically and topologically correct approximation to the surface. It is available at *http://www.cs.utexas.edu/users/amenta/powercrust/.*

CGAL (*www.cgal.org*) provides support for the most extreme polygon/polyhedral simplification, finding the smallest enclosing circle/sphere.

Notes: The Douglas-Peucker incremental refinement algorithm [DP73] is the basis for most shape simplification schemes, with faster implementations due to [HS94, HS98]. The link distance approach to polygon simplification is presented in [GHMS93]. Shape simplification problems become considerably more complex in three dimensions. Even finding the minimum-vertex convex polyhedron lying between two nested convex polyhedra is NP-complete [DJ92], although approximation algorithms are known [MS95b].

Heckbert and Garland [HG97] survey algorithms for shape simplification. Shape simplification using medial-axis transformations (see 17.10) are presented in [TH03].

Testing whether a polygon is simple can be performed in linear time, at least in theory, as a consequence of Chazelle's linear-time triangulation algorithm [Cha91].

Related Problems: Fourier transform (see page 431), convex hull (see page 568).