[Ba] Bykat, A., “Convex hull of a finite set of points in two dimensions,” *Info. Proc. Lett.* **7** (1978), 296-298.

[Ch] Chazelle, B., “A theorem on polygon cutting with applications,” *Proc. 23rd IEEE Symposium on Foundations of Computer Science* (1982), 339-349.

[CI] Chazelle, B. and Incerpi, J., “Triangulation and shape complexity,” *ACM Transactions on Graphics* **3** (1984), 135-152.

[EET] ElGindy, H., Everett, H. and Toussaint, G. T., “Slicing an Ear in Linear Time,” internal memorandum, School of Computer Science, McGill University, (1989).

[ET] ElGindy, H. and Toussaint, G., “On geodesic properties of polygons relevant to linear time triangulation,” *The Visual Computer* **5**(1989), 68-74.

[FM] Fournier, A. and Montuno, D., “Triangulating simple polygons and equivalent problems,” *ACM Transactions on Graphics* **3** (1984), 153-174.

[GJPT] Garey, M. R., Johnson, D. S., Preparata, F. P., and Tarjan, R. E., “Triangulating a simple polygon,” *Info. Proc. Lett.* **7** (1978), 175-180.

[Gr] Graham, R. L., “An efficient algorithm for determining the convex hull of a finite planar set,” *Info. Proc. Lett.* **1** (1972), 132-133.

[HM] Hertel, S. and Mehlhorn, K., “Fast triangulation of simple polygons,” Proc. FCT, LNCS 158 (1983), 207-215.

[KKT] Kirkpatrick, D. G., Klawe, M. M., and Tarjan, R. E., “O(n log log n) polygon triangulation with simple data structures,” to appear in *Sixth Annual ACM Symposium on Computational Geometry* (1990).

[Me] Meisters, G. H., “Polygons have ears,” *American Mathematical Monthly* (1975), 648-651.

[Sk] Sklansky, J., “Measuring concavity on a rectangular mosaic,” *IEEE Trans. Comput.* **C-21** (1972), 1355-1364.

[To] Toussaint, G. T., “An Output-Complexity-Sensitive Polygon Triangulation Algorithm”, to appear in *Proc. Computer Graphics International’90*, Singapore (1990).

[TV] Tarjan, R. E. and Van Wyk, C. J., “An O(n log log n)-time algorithm for triangulating a simple polygon,” *SIAM Journal on Computing* (1988), 143-178.

Proof: First we prove, by induction, that each time the algorithm reaches Step 2 there are no ears lying strictly between p_0 and PRED(p_i). This is true the first time that Step 2 is reached since there are no vertices lying strictly between p_0 and PRED(p_i). Consider the kth execution of the body of the loop. By the induction hypothesis there are no ears strictly between p_0 and PRED(p_i). If PRED(p_i) is not an ear then p_i is advanced to p_i' = SUCC(p_i) in Step 12 and the algorithm returns to Step 2 with no ears strictly between p_0 and PRED(p_i'). If PRED(p_i) is an ear then it is cut in Step 5 forming a smaller polygon P'. Let PRED'(p_i) be the predecessor of p_i in P'. The only vertex between p_0 and p_i in P' which may have become an ear as a result of this cut is PRED'(p_i). In the special case that PRED'(p_i) = p_0, p_i is advanced to p_i' = SUCC(p_i) in Step 11; otherwise, p_i remains unchanged so p_i' = p_i. In either case, the algorithm then returns to Step 2 with no vertices between p_0 and PRED'(p_i') being ears.

To complete the proof we show that when $p_i = p_0$, that is, when the algorithm terminates, the polygon is a triangle. This implies that n-3 ears have been cut. As soon as the polygon becomes a triangle, the test in Step 3 ensures that p_i is advanced to p_0. Consider the point in the algorithm when $p_i = p_{n-1}$ and suppose that the polygon at this stage, P', is not a triangle. We show that p_i is not advanced to p_0. By the previous argument there are no ears lying strictly between p_0 and PRED'(p_{n-1}). By the Two-Ears Theorem, P must have two non-overlapping ears. Thus two of p_0, p_{n-1} and PRED'(p_{n-1}) must be non-overlapping ears and the only possibility is that at least p_0 and PRED'(p_{n-1}) are ears. In this case, PRED'(p_{n-1}) will be cut in Step 5 and p_i is not advanced. Q.E.D.

4. Time Analysis

Theorem: Algorithm Triangulate runs in O(kn) time where n is the number of vertices in P and k-1 is the number of concave vertices in P.

Proof: In each execution of the loop either an ear is removed (Step 5) or p_i is advanced (Step 12). Since there are only n vertices, at most n ears can be removed. Step 10 ensures that p_0 is never cut as an ear, so p_i must reach p_0 after advancing all the way around P at which point the algorithm halts. Since p_i can be advanced at most n times, the loop is executed at most 2n times. All steps inside the loop can be done in constant time with the exception of Step 3 which may require O(k) time. Thus the time for the entire algorithm is O(kn). Q.E.D.

References

[AT] Avis, D., and Toussaint, G., "An optimal algorithm for determining the visibility of a polygon from an edge," *IEEE Trans. Comp.* **C-30** (1981), 910-914.

FUNCTION *IsAnEar(P,R,p_j)*
1. if R = ∅ then return true {P is a convex polygon}
2. else if p_j is a convex vertex then
3. if triangle (PRED(p_j),p_j,SUCC(p_j)) contains no vertex of R then
4. return true
5. else return false
6. else return false

End *IsAnEar*

3. Proof of Correctness

In this section we prove that the algorithm correctly triangulates a simple polygon P. To show that the only vertices that are cut are ears it suffices to prove the correctness of the function IsAnEar and the following lemma.

Lemma 1: Each time that IsAnEar is called R consists of exactly those vertices in P which are concave.

Proof: First we show that cutting an ear from a polygon creates no new concave vertices. Suppose vertex p_j is cut. The only vertices which are affected are p_{j-1} and p_{j+1}. Neither of these will become concave since $\angle p_{j-2},p_{j-1},p_j > p_{j-2},p_{j-1},p_{j+1}$ and $\angle p_j,p_{j+1},p_{j+2} > p_{j-1}, p_{j+1},p_{j+2}$. Thus no vertex ever needs to be added to R. Next note that by cutting an ear, say p_j, the only concave vertices which may become convex are p_{j-1} or p_{j+1}. In the case that p_{j-1} (or p_{j+1}) does become convex, it is removed from R in Step 9 (or 7). Q.E.D.

The correctness of IsAnEar follows from Lemmas 1 and 2.

Lemma 2: If a convex vertex p_j is not an ear then triangle (p_{j-1},p_j,p_{j+1}) contains a concave vertex.

Proof: If p_j is not an ear then triangle (p_{j-1},p_j,p_{j+1}) contains some vertex. Let p_k be the vertex in triangle (p_{j-1},p_j,p_{j+1}), $j \neq k$, whose distance to line (p_{j-1},p_{j+1}) is maximized. Let r and s be the intersection points of the line going through p_k parallel to line (p_{j-1},p_{j+1}) with segments p_{j-1},p_j and p_j,p_{j+1} respectively. By choice of p_k triangle (r,p_j,s) is empty and lies entirely inside P. Thus, p_{k-1} and p_{k+1} must lie on the opposite side of line (r,s) as p_j making p_k a concave vertex. Q.E.D.

Next we will show that the algorithm cuts n-3 ears and the correctness of the entire algorithm follows.

Lemma 3: Algorithm Triangulate cuts n-3 ears.

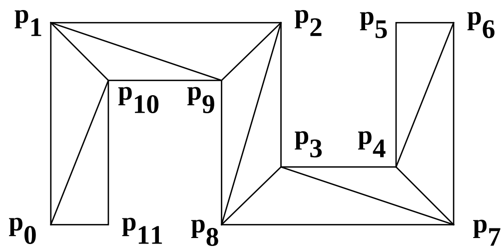

Figure 1.

ALGORITHM *Triangulate(P)*

The algorithm takes as input a simple polygon $P = (p_0, p_1, ..., p_{n-1})$, stored as a doubly linked circular list. $SUCC(p_i)$ and $PRED(p_i)$ indicate the successor and predecessor of p_i respectively. The algorithm produces a set D of diagonals comprising a triangulation of P. R is a set containing all the concave vertices of P. $IsAnEar(P,R,p_i)$ is a function which returns true if p_i is an ear in polygon P and false otherwise.

$p_i \leftarrow p_2$;
while $(p_i != p_0)$ do
    if $(IsAnEar(P,R,PRED(p_i))$ and P is not a triangle then {$PRED(p_i)$ is an ear.}
        $D \leftarrow D \cup (PRED(PRED(p_i)),p_i)$ {Store a diagonal.}
        $P \leftarrow P - PRED(p_i)$ {Cut the ear.}
        if $p_i \in R$ and p_i is a convex vertex then {p_i has become convex.}
            $R \leftarrow R - p_i$
        if $PRED(p_i) \in R$ and $PRED(p_i)$ is a convex vertex then {$PRED(p_i)$ has become
            $R \leftarrow R - PRED(p_i)$ convex.}
        if $(PRED(p_i) = p_0)$ then {$SUCC(p_0)$ was cut.}
            $p_i \leftarrow SUCC(p_i)$ {Advance the scan.}
    else $p_i \leftarrow SUCC(p_i)$ {$PRED(p_i)$ is not an ear or P is a triangle. Advance the scan.}
end while

END *Triangulate*

Two-Ears Theorem: Except for triangles every simple polygon has at least two non-overlapping ears.

This theorem forms the basis of the ear-cutting algorithm for triangulation. The algorithm finds an ear of the polygon, cuts it off and then recursively triangulates the rest of the polygon. A brute-force implementation of this approach yields an $O(n^3)$-time algorithm. This can be improved to $O(n^2)$ if the prune-and-search algorithm in [EET] is used to find an ear in linear time.

The Graham scan is an important technique in computational geometry which was independently proposed by Graham [Gr] to compute the convex hull of a sorted set of points and by Sklansky [Sk] to compute the convex hull of a simple polygon. Whereas the Sklansky scan fails for simple polygons [By] it succeeds for star-shaped polygons, a fact upon which the correctness of the Graham scan relies. The idea of the Graham scan is to make a single scan through a sorted list of the points. At each step in the scan an appropriate constant time test is made. After each test either a point is deleted from the list or the scan is advanced. If there are n points in the list then only n points can be deleted and the scan can be advanced at most n times. Thus the algorithm takes O(n) time.

Since its introduction the Graham-Sklansky scan has found widespread application to other problems. For example, it has been used to determine in O(n) time whether a simple polygon is weakly visible from a specified edge [AT] and to triangulate in O(n) time a polygon known to be palm-shaped with respect to a point in the polygon [ET].

In this paper we show how to use the Graham scan to obtain an O(kn)-time implementation of the ear-cutting algorithm. Since k-1 is the number of concave vertices this algorithm can be as bad as $O(n^2)$. The elegance and familiarity of the Graham scan combined with the simplicity of the ear-cutting approach yields an algorithm which is both simple to state and straightforward to implement. If the polygon is represented as a doubly-linked circular list then no additional data structures are required.

2. The Algorithm

The algorithm adapts the Graham scan in the following manner. The vertices of the polygon are scanned in order starting with p_2. At each step the current vertex is tested to see if it is the top of an ear. If it is not the top of an ear then the current vertex is advanced. If it is the top of an ear then the ear is cut off; that is, a diagonal is added to the triangulation and a vertex is deleted from the polygon. The current vertex is not advanced in this case except in the special case that the ear is the vertex following p_0. This prevents p_0 from being cut as an ear.

To illustrate the execution of the algorithm consider the polygon in Figure 1. Initially, the algorithm tests p_1 and determines that it is not an ear (note that this is equivalent to testing whether p_2 is the top of an ear). The scan is advanced through p_2, p_3, p_4 and p_5 at which time p_5 is determined to be an ear. p_5 is cut and then p_4 is tested and found not to be an ear. p_6 is the next vertex tested. It is found to be an ear and cut. Again p_4 is tested and this time it is an ear so it is cut. The remaining vertices will be cut in the order p_7, p_3, p_8,p_2, p_9, p_1.

The Graham Scan Triangulates Simple Polygons

Xianshu Kong, Hazel Everett and Godfried Toussaint

ABSTRACT

The Graham scan is a fundamental backtracking technique in computational geometry which was originally designed to compute the convex hull of a set of points in the plane and has since found application in several different contexts. In this note we show how to use the Graham scan to triangulate a simple polygon. The resulting algorithm triangulates an n vertex polygon P in O(kn) time where k-1 is the number of concave vertices in P. Although the worst case running time of the algorithm is $O(n^2)$, it is easy to implement and is therefore of practical interest.

1. Introduction

A *polygon* P is a closed path of straight line segments. A polygon is represented by a sequence of vertices $P = (p_0,p_1,...,p_{n-1})$ where p_i has real-valued x,y-coordinates. We assume that no three vertices of P are collinear. The line segments (p_i,p_{i+1}), $0 \le i \le n-1$, (subscript arithmetic taken modulo n) are the *edges* of P. A polygon is *simple* if no two nonconsecutive edges intersect. A simple polygon partitions the plane into two open regions; one unbounded called the *exterior* of P and one bounded called the *interior* of P. We follow convention in including the interior of P when referring to P. We assume that the vertices are given in clockwise order so that the interior of the polygon lies to the right as the edges are traversed. The line segment joining two non-consecutive vertices p_i and p_j of P is called a *diagonal* of P if it lies entirely inside P. A *triangulation* of a simple polygon consists of n-3 non-intersecting diagonals.

Many algorithms exist for triangulating simple polygons [Ch] [CI] [FM] [GJPT] [HM] [KKT] [To] [TV]. These algorithms vary in their worst case time complexities, in the complexity of their descriptions and in the data structures they use. The fastest algorithm known is the O(n log log n)-time algorithm of Tarjan and Van Wyk [TV]. The O(n log n)-time algorithm of Fournier and Montuno [FM] and the O(tn)-time algorithm of Toussaint [T] (t is a measure of the “shape-complexity” of the triangulation) are among the simplest.

Perhaps the simplest polygon triangulation algorithm of all from a conceptual viewpoint is the classic ear-cutting algorithm. A vertex p_i of a simple polygon P is called an *ear* if the line segment (p_{i-1},p_{i+1}) is a diagonal. We call p_{i+1} the *top* of ear p_i. We say that two ears p_i and p_j are *non-overlapping* if the interior of triangle (p_{i-1},p_i,p_{i+1}) does not intersect the interior of triangle (p_{j-1},p_j,p_{j+1}). Meisters [Me] has given an elegant inductive proof of the following theorem.