Polygon triangulation
(I)

Complex shapes are decomposed into simpler ones. In the plane the simplest shape is the triangle. Triangulation is the problem of subdividing/partitioning into triangles. The simplest version of the problem is polygon triangulation: given a polygon, subdivide it into triangles.

1 Notation and definitions

As usual, assume that a simple polygon is given as a list of n vertices $p_0, p_2, ...p_{n-1}$, starting at an arbitrary vertex p_0, in ccw order.

Two points on the boundary of the polygon are *visible* if the segment between them lies entirely within the polygon.

A *diagonal* of a polygon is a pair of vertices p_ip_j of the polygon that are visible (i.e. p_ip_j does not intersect the boundary of P except at p_i and p_j, and p_ip_j is inside P).

A *ear* $p_{i-1}p_ip_{i+1}$ with tip p_i is a set of 3 consecutive vertices of P such that $p_{i-1}p_{i+1}$ is a diagonal.

2 Properties

Lemma: Any simple polygon must have a convex vertex.
Proof:

Lemma: Any simple polygon with $n \geq 4$ must have a diagonal.
Proof:

Lemma: Any simple polygon can be triangulated by adding diagonals.
Proof:

Lemma: Every triangulation of a simple polygon uses $n-3$ diagonals and has $n-2$ triangles.
Proof:

Lemma: Any simple polygon $n \geq 4$ must have two ears.
Proof:

3 Algorithms

3.1 Algorithm: Is $p_i p_j$ a diagonal?

Come up with an algorithm to determine if two vertices of P form a diagonal.

`diagonalie(vertex` p_i`, vertex` p_j`, polygon` P`)`

Analysis:

3.2 Algorithm: Finding a diagonal of P

Come up with an algorithm to find a diagonal of P.

Idea 1: Try all diagonal candidates; for each one check if diagonal. Analysis:

Idea 2: Mimic the proof that any polygon has a diagonal: find convex vertex, check if it's an ear, if not find lowest vertex inside. Analysis:

3.3 Algorithm: triangulating a polygon by finding diagonals

Come up with an algorithm to find triangulate P by recursively finding diagonals.
Idea: find a diagonal, recurse.

Analysis:

3.4 Algorithm: triangulating a polygon by finding ears

Speed up finding diagonals by finding ears. Idea: Find a ear; recurse.

Analysis:

3.5 Algorithm: triangulating a polygon by finding ears (improved)

Idea: Compute ear status. Pick an ear, delete it and output its diagonal and recurse. Which vertices can change their ear status?

Analysis: