Polygon Triangulation (chapter 3)

Elisha Sacks
Polygon Triangulation

- Decompose a polygon into triangles.
- Applications: calculation, drawing, camera coverage.
Theorem 3.1 Every \( n \)-vertex polygon has a triangulation and every triangulation has \( n - 2 \) triangles.

**Proof**

- A triangle \((n = 3)\) is trivial, so consider \( n > 3 \).
- Lemma: there are vertices \( u \) and \( w \) with \( uw \) in the interior.
- This *diagonal* splits the polygon into polygons with \( m \) and \( n - m + 2 \) vertices.
- Triangulate them with \( m - 2 \) and \( n - m \) triangles.
- The union is a triangulation with \( n - 2 \) triangles.
- The proof yields an \( O(n^2) \) algorithm.
Proof of Lemma

Let $v$ be the leftmost vertex with neighbors $u$ and $w$.

If $uw$ is in the interior, we are done.

Otherwise, an edge intersects $uw$.

Consequently, a vertex is inside the triangle $uvw$.

Let $v'$ be the vertex in $uvw$ farthest from $uw$.

For $vv'$ to intersect an edge, one endpoint must be farther from $uw$ than $v'$, which contradicts its definition.

Hence, $vv'$ is a diagonal.
Convex Decomposition

- Decompose the polygon into convex polygons.
- Triangulate each convex polygon with diagonals from one vertex to the others.
- Problem: convex decomposition is hard.
Monotone Decomposition

A polygon is $y$-monotone if a horizontal line intersects it in a closed interval or in the empty set.

The polygon consists of two $y$-monotone vertex chains that share a top and a bottom vertex.

We decompose the input polygon into $y$-monotone polygons in $O(n \log n)$ time then triangulate them in $O(n)$ time.
Types of Vertices

- **Start**: maximum left turn (interior below).
- **End**: minimum left turn (interior above).
- **Split**: maximum right turn (the interior above).
- **Merge**: minimum right turn (interior below).
- **Regular**: increasing or decreasing (interior left or right).
**Lemma 3.4** A polygon $P$ is $y$-monotone iff it has no split or merge vertices.

*Proof* Only if is easy: a horizontal just above a merge or just below a split intersects the polygon in at least two segments. To prove if, suppose $P$ has no splits or merges yet is not $y$-monotone.
Proof of If

There is a line $\ell$ that intersects $P$ in multiple segments.

Let $pq$ be the leftmost segment.

Follow the boundary from $q$ with the interior on the left.

Let $r$ be the next intersection point with $\ell$.

(a) If $p \neq r$, the highest vertex between $q$ and $r$ is a split.

(b) if $p = r$, follow the boundary in the other direction to $r'$.

Since there are multiple components, $p \neq r'$ and the lowest vertex between $q$ and $r'$ is a merge.
Decomposition Strategy

Insert a diagonal from each split/merge to a vertex above/below. The merges and splits are monotone in the sub-polygons because the diagonals go in the opposite direction to their edges.
Algorithm

▶ Sweep the polygon from top to bottom with a horizontal line.
▶ The events are the vertices.
▶ Add an upward diagonal from each split vertex.
▶ Add a downward diagonal from each merge vertex.
The helper of an edge $e$ is the lowest vertex $v$ above the sweep line such that the horizontal segment from $v$ to $e$ is in the polygon.

- A split vertex $v_i$ connects to the helper of its left edge $e_j$.
- This occurs at the $v_i$ event.
Merge Vertices

A merge vertex $v_i$ connects to the highest vertex below $v_i$ that is a helper of its left edge $e_j$.

The $v_i$ event sets the helper of $e_j$ to $v_i$.

$v_m$ is the next helper of $e_j$ or its lower endpoint.
Every diagonal goes up from a vertex $v_i$ to the helper of its left edge $e_j$.

Either $v_i$ is a split vertex or the helper is a merge vertex.

Both cases can occur together.
The edges with the interior on the right that intersect the sweep line are stored in a tree $T$ in left to right order.

The tree and the helpers of its edges are updated at events.
1. Insert $e_i$ in $T$ and set $helper(e_i)$ to $v_i$.  
Example $v_5$: insert $e_5$.  

1. If \( \text{helper}(e_{i-1}) \) is a merge, add a diagonal from \( v_i \) to it.
2. Delete \( e_{i-1} \) from \( T \).

Example \( v_{15} \): \( \text{helper}(e_{14}) = v_{14} \), which is not a merge.
1. Use $T$ to find the edge $e_j$ left of $v_i$.
2. Add a diagonal from $v_i$ to $\text{helper}(e_j)$.
3. Set $\text{helper}(e_j)$ to $v_i$.
4. Insert $e_i$ in $T$ and set $\text{helper}(e_i)$ to $v_i$.

Example $v_{14}$: $e_9$ is the left edge with helper $v_8$. Add a diagonal from $v_{14}$ to $v_8$. 
1. If $\text{helper}(e_{i-1})$ is a merge, add a diagonal from $v_i$ to it.
2. Delete $e_{i-1}$ from $T$.
3. Use $T$ to find the edge $e_j$ left of $v_i$.
4. If $\text{helper}(e_j)$ is a merge vertex, insert a diagonal from $v_i$ to it.
5. Set $\text{helper}(e_j)$ to $v_i$.

Example $v_8$: the helper $v_2$ of $e_7$ is a merge, so add a diagonal from $v_8$ to $v_2$; the edge to the left is $e_9$ and its helper $v_9$ is a start vertex.
HANDLE REGULAR VERTEX ($v_i$)
1. if the interior of $\mathcal{P}$ lies to the right of $v_i$
2. then if $\text{helper}(e_{i-1})$ is a merge vertex
3. then Insert the diagonal connecting $v_i$ to $\text{helper}(e_{i-1})$ in $\mathcal{D}$.
4. Delete $e_{i-1}$ from $\mathcal{T}$.
5. Insert $e_i$ in $\mathcal{T}$ and set $\text{helper}(e_i)$ to $v_i$.
6. else Search in $\mathcal{T}$ to find the edge $e_j$ directly left of $v_i$.
7. if $\text{helper}(e_j)$ is a merge vertex
8. then Insert the diagonal connecting $v_i$ to $\text{helper}(e_j)$ in $\mathcal{D}$.
9. $\text{helper}(e_j) \leftarrow v_i$

Example: $v_6$ has the interior to the right: add a diagonal to $v_4$. 
Correctness

Lemma 3.5 The diagonals do not intersect each other or $P$.

Proof We will discuss a split vertex $v_i$; the other cases are similar.

- $e_j$ and $e_k$ are the closest edges left and right of $v_i$.
- $v_m$ is the helper of $e_j$, so the diagonal is $v_i v_m$.
- The rectangle $Q$ contains no vertices.
- An edge of $P$ that intersects $v_i v_m$ must intersect the horizontal that connects $v_i$ to $e_j$ or that connects $v_m$ to $e_j$.
- This is impossible because $e_j$ is directly left of $v_i$ and $v_m$.
- Prior diagonals cannot intersect $v_i v_m$ because they are above $v_i$ and cannot have an endpoint in $Q$. 
Complexity

- The running time is $O(n \log n)$ for $y$ sorting and $T$ insertion.
- The space complexity is $O(n)$. 
A monotone polygon is triangulated with a greedy algorithm that processes the vertices in decreasing $y$ order.

The algorithm employs a stack of vertices in decreasing $y$-order with the lowest vertex at the top. Example: $abcde$.

The bottom vertex is from one side of the polygon and the rest of the stack is a portion of the other side that is concave upward.
The vertices of the stack form a polygonal chain, comprised of polygon edges and of diagonals, above which the polygon has already been triangulated.

The chain and the downward edge from the bottom vertex form a funnel.

The triangulation is extended by inserting diagonals from the vertex below the stack to the stack vertices wherever possible.
1. Place the vertices in decreasing $y$ order $v_1, \ldots, v_n$.
2. Initialize the stack to $(v_1, v_2)$.
3. For $j = 3$ to $n - 1$
   - Case 1: $v_j$ on the opposite chain from the stack top.
   - Case 2: $v_j$ on the same chain as the stack top.
4. Insert a diagonal from $v_n$ to each stack vertex, except for the first and last.

Why is the last vertex a special case?
Case 1

- The vertex $v_j$ is on the opposite chain from the stack top.
- Generate triangles from $v_j$ and the stack edges.
- The new stack is the old top then $v_j$. 
Case 2

- The vertex $v_j$ is on the same side as the stack top.
- Generate triangles from $v_j$ and the stack edges that it sees.
- The new stack is the other stack edges then $v_j$. 
Case 2

▶ The vertex $v_j$ is on the same side as the stack top.
▶ Generate triangles from $v_j$ and the stack edges that it sees.
▶ The new stack is the other stack edges then $v_j$.

When does $v_j$ see stack edge $ab$ with $a_y < b_y$?
Case 2

- The vertex $v_j$ is on the same side as the stack top.
- Generate triangles from $v_j$ and the stack edges that it sees.
- The new stack is the other stack edges then $v_j$.

When does $v_j$ see stack edge $ab$ with $a_y < b_y$?

$abv_j$ is a left/right turn for $ab$ a left/right edge.
Degeneracy

- Vertices $a_1 = (x_1, y)$ and $a_2 = (x_2, y)$ are degenerate.
- The book specifies that $a_1$ is below $a_2$ if $x_1 > x_2$.
- This handles the degeneracies in the sweep algorithm.
- The greedy algorithm is degenerate when three consecutive vertices are collinear.
- The lowest vertex does not see the edge formed by the upper two vertices.
Generalizations

polygon with holes

subdivision
Art Gallery Problem

Compute a minimal set of points inside a polygon from which every point in the polygon is visible.
Solution Using Triangulation

1. Triangulate the polygon.
2. Compute a vertex 3-coloring of the triangulation.
3. Select the smallest color class.

▶ Conclusion: ⌊n/3⌋ of the n vertices suffice.
▶ The book shows a polygon for which this bound is tight.
Computing the 3-coloring

- The dual graph has a node for each triangle and an edge between triangles that share a diagonal.
- It is acyclic because every diagonal splits the polygon.
- Traverse the dual graph in depth-first order.
- Color the vertices of the first triangle.
- When crossing an edge into a triangle, color its third vertex differently from the edge vertices.