Triangulating a Polygon

Recall: PSLG = Planar Straightline graph.

**Def** (Simple) **Polygonal chain** is a PSLG consisting of a simple cycle $P$.

**Claim** A Polygonal chain has a unique interior.

**Def** Polygon is Polygonal chain + interior

**Triangulation**: Addition of seg so that

1) Still PSLG
2) Interior is decomposed into triangles
Theorem: Every simple polygon can be triangulated.

Proof: Induct on # of edges $n$.

Base case: $n = 3$

Assume no $180^\circ$ in $n < 3$ true for $m < n$.

Let $v$ be left most point with neither $w$ nor $u$.

1) Case 1: $\deg = [w, u]$ is interior
   
   We get two Poly $|P_1| = 3$ $|P_2| = n - 1$

2) Case 2: $3$ point $v'$ interior to $Tri = [v, v', u]$
   
   Let $v'$ be left most such point.
   
   $\deg = [v, v']$ is interior
   
   $|P_1| < n$ and $|P_2| < n$.
Thm: Not every simple polygonal surface in 3D can be decomposed into tetrahedra.

Example: Prism $xyz$ with twist top.

By CONTRA:
Consider Tet with faces $B$.

Missing vertex in $x$ or $y$ not $z$.

Not $x$ since seg $[a,x]$ is outside.

Not $y$ since seg $[g,y]$ is outside.

In general: Test if polygonal surface is decomposable in NP-Hard.
Guarding A Polygon

Input: Polygon $P$

Output: Locations $p_1, \ldots, p_k \in P$ (guards)

1) Guards cover $P$
2) $k$ small.

Thm. A polygon $P$ with $n$ vertices

$\frac{n}{3}$ guards suffice and maybe necessary.

Necessary.

$|P| = 12$ needs a guard per prong.

$\frac{n}{3}$ prongs
$\frac{3}{2}$-guards Alg ($\mathcal{P}$)

1) Tri $\mathcal{P}$ $\overline{\mathcal{P}}$
2) 3-color $\overline{\mathcal{P}}$
   a) Construct geometric dual $T$ (on degree 3 tree)
   b) 3-color $\overline{\mathcal{P}}$ by traversing tris in an
       inorder fashion.
3) Pick least used color.

Only non-linear time step is 1)
2D-Algorithm

Proof \Rightarrow O(n^2)

Known: O(n) Chazelle

today: O(n \log n) \text{ (sweep line)}

Thin Class: O(n \log^* n) Seidel (incremental randomized)

Def \log^* n = \min_k \log \log \ldots \log n \leq 1

(Prob. Give a O(n) time alg to determine which side of
an edge is interior/exterior)

(Prob. test P \in \text{Int}(P) \text{ in } O(n) \text{ time})

3.14 O(n \log n) O.K

O(n) ?

\text{Trap} \Rightarrow \text{Tr}^1
Step 1: Partition into Monotone Polygons

Definition: Y-monotone if
Every horizontal line l
\( l \cap P \) is connected or empty

Alg Type: Line Sweep

\( O(n \log n) \) time

- \( \square \): Start vertex
- \( \blacksquare \): End
- \( \circ \): Seq
- \( \triangle \): Split
- \( \triangledown \): Merge
Claim \( P \) is \( y \)-monotone iff no split or merge vertices.

\[ (\Rightarrow) \text{ (easy)} \]

\[ (\Leftarrow) \text{ (not mono } \Rightarrow \text{ split or merge)} \]

Assume not mono

\[ \quad \quad \quad \quad \quad \]

\[ \text{Case 1} \]

\[ \quad \quad \quad \quad \quad \]

\[ \text{Case 2} \]

\[ \quad \quad \quad \quad \quad \]
Alg: Sweep Line (top-to-bottom)

Events: endpoints

Dictionary: Intervals (sorted)

Interval: (left-seg, right-seg, helper vertex)

vertex: 2 edges before l after

Def: helper(l, e') = lowest vertex above l and between e & e'

no horizontal seg.

Procedure: add(l, e) = add seg from p to e if [l, e] not already an edge.
Make Monotone (G, event)

Case (Start Vertex)
   1) Add new interval
   2) set helper ∈ G

Case (End Vertex)
   if helper is a merge vertex then add (G, helper)
   2) remove interval

Case (Regular)
   \[ e \quad e' \]
   1) add (G, helper)
   2) replace e with e''
   3) helper ∈ G

Case (Split)
   1) add (G, helper)
   2) "split" interval say I_1, I_2
   3) helper(I_1) = helper(I_2) ∈ G
Case (Merge)

1) \text{add}(\text{help}_L, g); \text{add}(\text{help}_R, g)

2) "Merge" intervals

3) help = g
Another View

1) Make Trapezoidal Decom (sweep line)
2) For each trap add a diagonal if possible
3) Types of Traps