Computational Geometry

Chapter 3

Polygons and Triangulation

13.

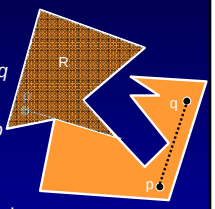
On the Agenda

- The Art Gallery Problem
- Polygon Triangulation

23.

Art Gallery Problem

- Given a simple polygon P, say that two points p and q can *see* each other if the open segment pq lies entirely within P.
- A point *guards* a region $R \subseteq P$ if p sees all $q \in R$.
- Given a polygon P, what is the minimal number of guards required to guard P, and what are their locations ?

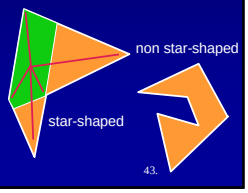


33.

Observations

- The *entire* interior of a convex polygon is visible from *any* interior point.

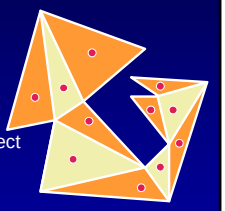
- A *star-shaped* polygon requires only one guard located in its *kernel*. (actually this is the definition of a star-shape polygon)



43.

Art Gallery Problem – Easy Upper Bound

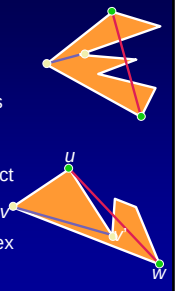
- n-2 guards suffice:
  - Subdivide the polygon into n-2 triangles (triangulation)
  - (we will show that this is the correct number of triangles)
  - Place one guard in each triangle.
- **Theorem**: Any simple planar polygon with n vertices has a triangulation of size n-2.



53.

Diagonals in Polygons

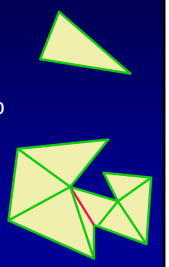
- A *diagonal* of a polygon P is a line segment connecting two non-adjacent vertices, which lies entirely within P.
- **Theorem:** Any polygon with $n>3$ vertices has a diagonal, which may be found in $O(n)$ time.
- **Proof:** Find the leftmost vertex v. Connect its two neighbors u and w. If this is not a diagonal there are other vertices inside the triangle uvw. Connect v to the vertex v' furthest from the (line containing the) segment uw.



63.

$O(n^2)$ Polygon Triangulation

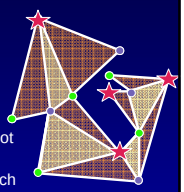
- **Theorem:** Every simple polygon with n vertices has a triangulation consisting of n-3 diagonals and n-2 triangles.
- **Proof:** By induction on n: Basis: A triangle (n=3) has a triangulation (itself) with no diagonals and one triangle.
- Induction: for a n+1 vertex polygon, construct a diagonal dividing the polygon into two polygons with n_1 and n_2 vertices such that $n_1+n_2-2=n$. Triangulate the two parts of the polygon. There are now $n_1-3+n_2-3+1=n-3$ diagonals and $n_1-2+n_2-2=n-2$ triangles.



73.

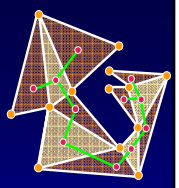
Art Gallery Problem – Upper Bound

- Claim: $\lfloor n/3 \rfloor$ guards are enough
- Color the vertices of the triangulated polygon with three colors, such that there is no edge between two vertices with the same color. (will show that this doable)
- Pick a color which is least used. This color cannot appear more than $\lfloor n/3 \rfloor$ times.
- Place a guard on each vertex with this color. Each triangle has only one guarded vertex, but this guard covers all triangles incident on the vertex.
- $\Rightarrow$ New upper bound: $\lfloor n/3 \rfloor$



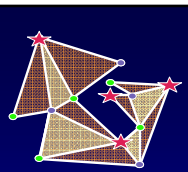
83.

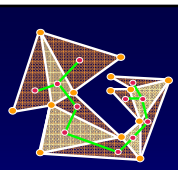
3-Coloring



- **Theorem:** Every triangulated polygon has an *ear* *Ear:* a triangle containing two boundary edges).
- **Proof:** Consider the *dual* to the triangulation.
  - Every triangle -> vertex.
  - Two neighboring triangles -> an edge.
  - Since any diagonal disconnects the polygon, the dual is a tree.
  - The vertex degrees are bounded by 3.
  - Ears correspond to leaves in the dual.

93.





- **Rembrandt Theorem:** The vertices of the (triangulated) polygon may be 3-colored, so the colors of the vertices of each edge are different.
- **Proof:** By induction on n:
  - Basis: Trivial if n=3.
  - Induction: Cut off an ear.
    - 3-color the n-1 vertex remainder.
    - Color the n'th vertex with the third color different from the two on its supporting edge.

103.

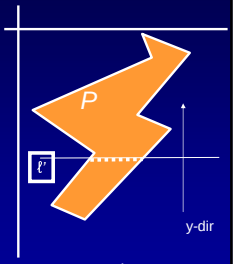
$\lfloor n/3 \rfloor$ - Tight Bound

- There exists a polygon with n vertices, for which $n/3$ guards are necessary.

113.

$O(n \log n)$-Time Polygon Triangulation

- A simple (no holes) polygon P is called *monotone* with respect to a direction ℓ if for any line ℓ' perpendicular to ℓ the intersection of P with ℓ' is connected.
- A polygon is called *monotone* if there exists any such direction ℓ.
- A polygon that is monotone with respect to both the x and y-axis is called *x/y-monotone.*



Question: How can we check in $O(n)$ time whether a polygon is y-monotone?

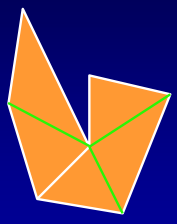
y-monotone but not x-monotone polygon

123.

Triangulation Algorithm – cont.

1) Partition the polygon into *y*-monotone pieces.

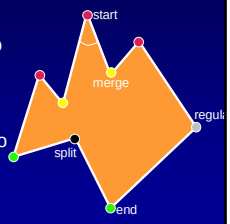
2) Triangulate each *y*-monotone piece separately.



133.

Classifying the vertices of *Y*-Monotone Polygons

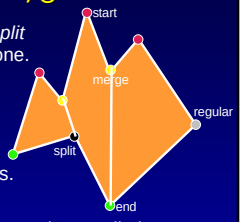
- A ***start*** (resp., ***end***) vertex is one whose interior angle is $\leq \pi$ and its two neighboring vertices both lie below (resp., above) it.
- A ***split*** (resp., ***merge***) vertex is one whose interior angle is $\geq \pi$ and its two neighboring vertices both lie below (resp., above) it.
- All other vertices are ***regular***.



143.

Y-Monotone Polygons

- **Theorem:** A polygon without *split* and *merge* vertices is *y*-monotone.
- Proof: Since there are only start/end/regular vertices, the polygon must consist of two *y*-monotone chains. Alternatively, do a case analysis.

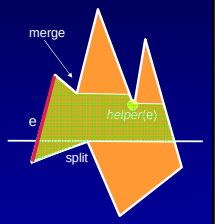


- To partition a polygon to monotone pieces, eliminate split (merge) vertices by adding diagonals upward (downward) from the vertex. Naturally, the diagonals must not intersect!

153.

Monotone Partitioning

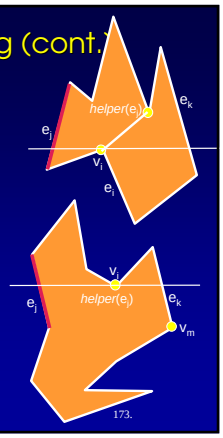
- Classify all vertices.
- Sweep *P* (with a line) from top to bottom.
- Maintain the edges intersected by the sweep line in a *sweep line status* (SLS sorted by *x* coordinate).
- Maintain vertex events in an event queue (EQ sorted by *y* coordinate).
- Eliminate split/merge vertices by connecting them to other vertices, as follow:
- For each edge *e*, define *helper*(*e*) as the lowest vertex (seen so far) above the sweep line **vertically visible** to the right of the edge.
- *helper*(*e*) is initialized to the upper endpoint of *e*.



163.

Monotone Partitioning (cont.)

- A split vertex may be connected to the helper vertex of the edge immediately to its left (along the sweep line).
  - Question: Why this connection does not intersect previously-made connections ?
- However, a merge vertex should be connected to a vertex which has not yet been processed !
- Clever idea: Every merge vertex is the helper of some edge, and will be handled when this edge "terminates".



173.

Monotone Partitioning Algorithm

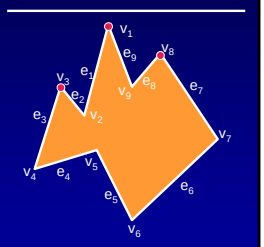
- Input: A counterclockwise ordered list of vertices of *P*.
  - /* Notation: The edge e_i immediately follows the vertex v_i CCK*/
- Construct an EQ (events queue) on the vertices of *P* using *y*-coordinates.
- Initialize SLS (*sweep line status*) to be empty.
- While EQ is not empty:
  - Pop vertex v;
  - Handle v.

  (No new events are generated during execution.)
- Idea: No split/merge vertex remains unhandled!

183.

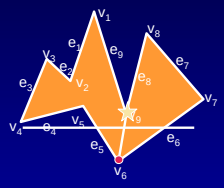
Monotone Partitioning

- Handling a *start* vertex (v_i):
  - Add e_i to SLS
  - *helper*(e_i) := v_i



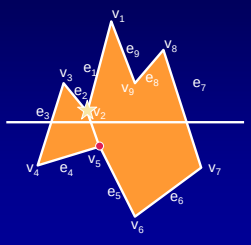
Monotone Partitioning

- Handling an *end* vertex (v_i):
  - If *helper*(e_{i-1}) is a merge vertex, then connect v_i to *helper*(e_{i-1})
    - Else don't connect v_i
  - Remove e_{i-1} from SLS



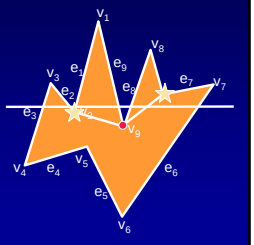
Monotone Partitioning

- Handling a *split* vertex (v_i):
  - Find in SLS the edge e_j directly to the left of v_i
  - Connect v_i to *helper*(e_j)
  - *helper*(e_i) := v_i
  - Insert e_i into SLS
  - *helper*(e_i) := v_i



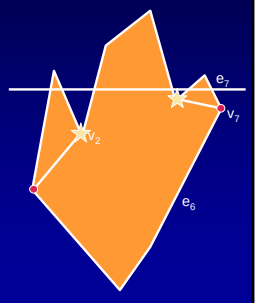
Monotone Partitioning

- Handling a *merge* vertex (v_i):
  - If *helper*(e_{i-1}) is a merge vertex, then connect v_i to *helper*(e_{i-1})
  - Remove e_{i-1} from SLS
  - Find in SLS the edge e_j directly to the left of v_i
  - If *helper*(e_j) is a merge vertex, then connect v_i to *helper*(e_j)
  - *helper*(e_j) := v_i



Monotone Partitioning

- Handling a *regular* vertex (v_i):
  - If the polygon's interior lies to the left of v_i then:
    - Find in SLS the edge e_j directly to the left of v_i
    - If *helper*(e_j) is a merge vertex, then connect v_i to *helper*(e_j)
    - *helper*(e_j) := v_i
  - Else:
    - If *helper*(e_{i-1}) is a merge vertex, then connect v_i to *helper*(e_{i-1})
    - Remove e_{i-1} from SLS
    - Insert e_i into SLS
    - *helper*(e_i) := v_i



Triangulating a Y-monotone Polygon-warm-ups

Recall – must consist of a left chain and right chain

Def: Convex vertex – interior angle $\leq \pi$. Otherwise called reflex vertex.

Convex chain – consecutive part of the chain consisting of convex vertices.

Very easy to triangulate a chain of convex vertices greedily connect them to a vertex on the other side, or between themselves

Preference to connect left chain to right, but sometimes must connect within the same chain.

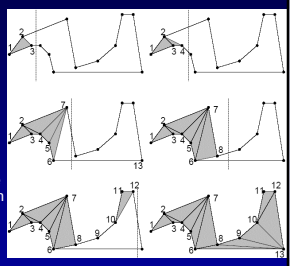
Triangulating a Y-monotone Polygon

Sweep the polygon from top to bottom.

Greedily triangulate anything possible above the sweep line, and then forget about this region. (order irrelevant)

- When we process a vertex *v*, the unhandled region above it always has a simple structure:
  Two *y*-monotone (left and right) chains, each containing at least one edge. If a chain consists of two or more edges, it is *reflex*, and the other chain consists of a single edge whose bottom endpoint has not been handled yet.
- Each diagonal is added in O(1) time – only constant number of checks

A chain consists of both polygon edges and diagonals (connecting left-left or right-right)



253.

Triangulating a Y-monotone Polygon

- Continue sweeping while one chain contains only one edge, while the other edges are reflex.
- When a "convex edge" appears in the concave chain (or a second edge appears in the other one), triangulate as much as possible using a "fan".
- Time complexity: O(*n*), where *n* is the complexity of the polygon.

263.