Computational Geometry

Chapter 3

Polygons and Triangulation

On the Agenda

- The Art Gallery Problem
- Polygon Triangulation
Art Gallery Problem

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

Observations

- The entire interior of a convex polygon is visible from any interior point. (Why?)
- A star-shaped polygon requires only one guard located in its kernel.
Art Gallery Problem: Easy Upper Bound

- **Theorem** (to be proven later): Every simple planar polygon with \( n \) vertices has a triangulation into \( n-2 \) triangles.
- \( n-2 \) guards suffice for an \( n \)-gon:
  - Subdivide the polygon into \( n-2 \) triangles (triangulation).
  - Place one guard in each triangle.

Diagonals in Polygons

- A diagonal of a polygon \( P \) is a line segment connecting two vertices, which lies entirely within \( P \).
- **Theorem**: Every polygon with \( n>3 \) vertices has a diagonal.
- **Proof**: Find the leftmost vertex \( v \). Connect its two neighbors \( u \) and \( w \). If this is not a diagonal there must be other vertices inside the triangle \( uvw \). Connect \( v \) with the vertex \( v' \) farthest from the segment \( uw \). This must be a diagonal.

- **Questions**:
  1. Why is \( v'v \) a diagonal?
  2. Why not connect \( v \) with the leftmost vertex inside \( uvw \)?
Diagonals in Polygons

- A diagonal of a polygon $P$ is a line segment connecting two vertices, which lies entirely within $P$.

- **Theorem:** Every polygon with $n>3$ vertices has a diagonal.

- **Proof:** Find the leftmost vertex $v$. Connect its two neighbors $u$ and $w$. If this is not a diagonal there must be other vertices inside the triangle $uvw$. Connect $v$ with the vertex $v'$ farthest from the segment $uw$. This must be a diagonal.

- **Questions:**
  1. Why is $v'v$ a diagonal?
  2. Why not connect $v$ with the leftmost vertex inside $uvw$?

Complexity of Triangulations

- **Theorem:** Any triangulation of a simple polygon with $n$ vertices consists 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.
  - Inductive step:
    1. For an $n$-vertex polygon, construct a diagonal dividing the polygon into two polygons with $n_1$ and $n_2$ vertices such that $n_1+n_2-2=n$. (Why ”$-2$“?)
    2. Triangulate the two parts of the polygon.
    3. Diagonals: $(n_1-3)+(n_2-3)+1=(n_1+n_2-2)-3=n-3$;
       Triangles: $(n_1-2)+(n_2-2)=(n_1+n_2-2)-2=n-2$. 

Center for Graphics and Geometric Computing, Technion
**$\Theta(n^2)$-Time Polygon Triangulation**

- **Algorithm:**
  1. Input: A simple $n$-gon.
  2. Find a diagonal.
  3. Call the algorithm recursively for the two subpolygons.

- **Analysis:**
  \[ T(n) = O(n) + \max_{n_1 + n_2 = n+2} \left( T(n_1) + T(n_2) \right) \]

- **Solution:**
  \[ T(n) = \Theta(n^2) \]

- **Space:** $\Theta(n)$

---

**Art Gallery Problem: Upper Bound**

- **Color the vertices of the (triangulated) polygon with three colors such that there is no edge between two vertices with the same color.**

- **Question:** Why is this possible?
  (Hint: The dual of any triangulation is a tree with vertex degree at most 3. Full proof later.)

- **Corollary:** All triangles are 3-colored.

- **Pick the color that is the least used. This color is used in at most $\lfloor n/3 \rfloor$ vertices.**

- **Place a guard on each vertex with this color. Due to the corollary all the triangles are guarded!**

- $\Rightarrow$ **New upper bound:** $\lfloor n/3 \rfloor$
3-Coloring

- **Theorem**: Every triangulated polygon can be 3-colored.
- **Proof**: Consider the dual graph.
  - Since every diagonal disconnects the polygon, the dual graph is a tree.
  - Since every node in the graph is the dual of a triangle, its degree is ≤ 3.
  - Since any tree has a leaf, any triangulation has an ear (a triangle containing two polygon edges).
  - Finally, by induction on $n$:
    - Basis: Trivial if $n=3$.
    - Induction: Cut off an ear. 3-color the remaining $(n-1)$-gon.
      Color the $n$th vertex with the third color different from the two on its supporting edge.
A Matching Lower Bound

- Fact: There exists a polygon with \( n \) vertices, for which \( n/3 \) guards are necessary.

- Therefore, \( \lceil n/3 \rceil \) guards are needed in the worst case.

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

- A simple polygon is called monotone with respect to a direction \( v \) if for any line \( \ell \) perpendicular to \( v \), the intersection of the polygon with \( \ell \) is connected.
- A polygon is called monotone if there exists any such direction \( v \).
- A polygon that is monotone with respect to the x- (or y-) axis is called x- (or y-) monotone.

**Question 1:** How can we check in O(\( n \)) time whether a polygon is y-monotone?

**Question 2:** What is a polygon that is monotone with respect to all directions?
Triangulation Algorithm – cont.

1) Partition the polygon into y-monotone pieces (“חתיכות מונוטוניות”).

2) Triangulate each y-monotone piece separately.

---

y-Monotone Polygons

- Classifying polygon vertices:
  - A start (resp., end) vertex is a vertex whose interior angle is less than \( \pi \) and its two neighboring vertices both lie below (resp., above) it.
  - A split (resp., merge) vertex is a vertex whose interior angle is greater than \( \pi \) and its two neighboring vertices both lie below (resp., above) it.
  - All other vertices are regular.
**y-Monotone Polygons (cont.)**

- **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.

- 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!

---

**Monotone Partitioning**

- Classify all vertices.
- Sweep the polygon from top to bottom.
- Maintain the edges intersected by the sweep line in a **sweep line status** (SLS sorted by x coordinates).
- Maintain vertex events in an event queue (EQ sorted by y coordinates). All events are known in advance!
- Eliminate split/merge vertices by connecting them to other vertices (to be explained later).
- For each edge \( e \), define \( helper(e) \) as the lowest vertex (seen so far) above the sweep line **visible** to the right of the edge.
- \( helper(e) \) is initialized by the upper endpoint of \( e \).
Monotone Partitioning (cont.)

- A split vertex may be connected to the helper vertex of the edge immediately to its left.
- However, a merge vertex should be connected to a vertex which has not been processed yet!
- Clever idea: Every merge vertex $v$ is the helper of some edge $e$, so that $v$ will be resolved either
  - when $e$ disappears; or
  - when $v$ ceases to be the helper of $e$.
  It will be the last time $v$ can be resolved!

Monotone Partitioning Algorithm

- Input: A polygon $P$, given as a list of vertices ordered counterclockwise. The edge $e_i$ immediately follows the vertex $v_i$.
- Construct EQ containing the vertices of $P$ sorted by their $y$-coordinates. (In case two or more vertices have the same $y$-coordinate, the vertex with the smaller $x$-coordinate has a higher priority.)
- Initialize SLS 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!
Monotone Partitioning

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

- Implementation detail:
  Only “left” edges (for which the polygon is on the right) need a helper and are thus kept in the status.

Monotone Partitioning

- Handling an end vertex ($v_i$):
  - If $helper(e_{i-1})$ is a merge vertex, then connect $v_i$ to $helper(e_{i-1})$ (Why?!)  
  - 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 $\text{helper}(e_j)$
  - $\text{helper}(e_j) := v_i$
  - Insert $e_i$ into SLS
  - $\text{helper}(e_i) := v_i$

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

Proof of Correctness: Split Vertices

- Assume that the split vertex \( v_5 \) was connected to \( v_2 \).
- Assume that \( s = v_5v_2 \) intersects another original edge \( e \).
- Draw horizontal lines through \( v_5 \) and \( v_2 \).
- Where can the endpoint of \( e \), that is to the left of \( s \), be?
  - Below \( t_1 \): Impossible. (Why?)
  - Between \( t_1 \) and \( t_2 \): Ditto. (Why?)
  - Above \( t_2 \): Ditto. (Why?)

- Now assume that \( s \) intersects another diagonal. Why can’t that be?
- Conclusion:
  Split events are resolved correctly.
Proof of Correctness (cont.)

- Merge vertices: Exercise.

- Complete the details of the proof as an exercise.

Triangulating a $y$-monotone Polygon

In Theory

- Sweep the polygon from top to bottom.
- Greedily triangulate anything possible above the sweep line, and then forget about this region.
  - 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.
Triangulating a Y-monotone Polygon

**In Practice**

- Continue sweeping while one chain contains only one edge, while the other edge is concave.
- When a “convex edge” appears in the concave chain, triangulate as much as possible by connecting the new vertices to all visible vertices of the concave chain.
- When the edge in the other chain terminates, connect it to all the vertices of the concave chain using a “fan”.
- Time complexity: $O(k)$, where $k$ is the complexity of the polygon.

**Question:** Why?!
Historical Perspective

- $O(n^2)$: Diagonal insertion
- $O(n \log n)$: Lee and Preparata
  - (Monotone decomposition, 1977)
  - Avis and Toussaint (1981)
  - Chazelle (1982)
- Optimal??
- $O(n \log \log n)$: Tarjan and Van Wyk (1988)
- $O(n \log^* n)$: Randomized:
  - Clarkson, Tarjan, and Van Wyk (1989)
  - Seidel (Trapezoidal decomposition, 1991)
  - Devillers (1992)
- $\Theta(n)$: Optimal (yet deterministic):
  - Chazelle (1991)