Outline

1. **Triangulation**
   - Polygon Terms and Definitions
   - The Art Gallery
   - Regularization and Triangulation
   - Literature
**Definition (Polygon)**

A *polygon* is a region of the plane bounded by a finite collection of line segments forming a simple close curve.

**Theorem (Jordan Curve Theorem)**

*If C is a simple closed curve in \( \mathbb{R}^2 \), then \( \mathbb{R}^2 \setminus C \) has two components (an "inside" and "outside"), with C the boundary of each.*

**Definition (Simple Polygon)**

A polygon is said to be *simple* (or *Jordan*) if it is enclosed by a single closed polygonal chain that does not cross itself. In particular, the polygon edges are pairwise disjoint in their interior and the degree of all vertices is two.
Polygon Terms & Definitions (Cont.)

The chain $v_1, v_2, \ldots, v_n$ defines a simple polygon iff

1. The segments $s_1 = v_1v_2$, $s_2 = v_2v_3$, \ldots, $s_{n-1} = v_{n-1}v_n$, $s_n = v_nv_1$ are disjoint in their interior.

2. Consecutive segments intersect only in their endpoints. Namely $s_i \cap s_{i+1} = v_{i+1}$, $i = 1, 2, \ldots, n - 1$ and $s_n \cap s_1 = v_1$

3. Non adjacent segments do not intersect $s_i \cap s_j = \emptyset$, $j > i + 1$.

- $P$ — a simple polygon.
- $\partial P$ — the boundary of $P$.
- $\partial P \subseteq P$, $P$ is closed and contains its boundary.
- By convention the vertices of a polygon are ordered counterclockwise around the interior of the polygon.
  - Interior of polygon is to the left of the boundary.
Outline

1. Triangulation
   • Polygon Terms and Definitions
   • The Art Gallery
   • Regularization and Triangulation
   • Literature
Application: Art Gallery

Application (Art Gallery)

Given the floor plan of an art gallery modeled as a simple polygon with \( n \) vertices. Find out how many (and where) guards are needed to see the entire gallery, where each guard is stationed at a fixed point, has 360° vision, and cannot see through the walls.

Problem posed to Vasek Chvatal by Victor Klee at a math conference in 1973. Chvatal solved it quickly with a complicated proof, which has since been simplified significantly using triangulation.
Art Gallery: Lower Bound

Definition (Seeing)
A (guard) point $p$ sees points $q \in P$ if $pq \subseteq P$.

Definition (Covering)
A set of guards $G$ covers a polygon $P$ if for any point $p \in P$ there is a guard $g \in G$ that sees $p$.

- $g(P)$ — minimum number of guards guarding $P$.
  - Cardinality of smallest set that covers $P$.
- $\mathcal{P}_n$ — set of all simple polygons with $n$ vertices.
- $G(n) = \max_{P \in \mathcal{P}_n} g(P)$ — maximum number of guards needed to guard a simple polygon with $n$ vertices.

- $G(n) \geq \lceil n/3 \rceil$
A diagonal of a polygon $P$ is a segment connecting two vertices of $P$ that strictly see each other.

A triangulation is a partition of $P$ into triangles formed by repeatedly inserting diagonals into $P$.

A vertex is strictly convex if its interior angle $\alpha < \pi$.

The interior angle of a reflex convex is $\alpha > \pi$.

Every polygon has at least one strictly convex vertex, e.g., the lexicographically smallest vertex $v$.

Every polygon with $n > 3$ vertices has a diagonal.

Every polygon may be partitioned into triangles by the addition of (0 or more) diagonals.

Proof by induction.

$T$ — a triangulation of a polygon $P$ of $n$ vertices.

$T$ uses $n - 3$ diagonals and consists of $n - 2$ triangles.
Art Gallery: Upper Bound (Cont.)

- The dual of a triangulation $T$ is a graph $G(T)$ with a node associated with each triangle and an arc between two nodes iff their triangles share an edge.
- The dual graph $G(T)$ is a tree with a vertex degree at most 3.
- 3 consecutive vertices $u$, $v$, $w$, form an ear if $uw$ is a diagonal
  - $v$ is the ear tip.
- Every polygon of $n > 3$ has at least 2 non-overlapping ears.
- The graph of the triangulation $T(P)$ is three colorable.
- Every simple polygon $P$ with $n$ vertices can be guarded using $\leq \lfloor n/3 \rfloor$ guards; $G(n) \leq \lfloor n/3 \rfloor$.
- Compute the triangulation of $P$.
- Compute a 3 coloring for $T(P)$.
- Choose the smallest set of vertices with the same color.
  - Its cardinality must be $\leq \lfloor n/3 \rfloor$. 
Art Gallery: Minimum Number of Guards

- A 3-coloring of the vertices yields 3 guards.
- However, the polygon can be guarded by only 2 guards.
- Finding the minimum number of guards is NP-hard.

Problem (Art Gallery Decision)

Given both a polygon and a number $k$, determine whether the polygon can be guarded with $k$ or fewer guards.

- Even the decision problem and all of its standard variations (such as restricting the guard locations to vertices or edges of the polygon) is NP-hard.
Art Gallery in $\mathbb{R}^3$

- Even $n$-vertex guards do not suffice!
- Different triangulations can have different number of tetrahedra.
- Determining whether a polyhedron requires Steiner vertices for triangulation is NP-Complete.
  - Smallest example of a polyhedron that cannot be triangulated without adding new vertices. (Schoenhardt [1928]).
- Every 3D polyhedron with $n$ vertices can be triangulated with $O(n^2)$ tetrahedra. [Cha84]
Outline

1 Triangulation
   • Polygon Terms and Definitions
   • The Art Gallery
   • Regularization and Triangulation
   • Literature
Check all $n^2$ choices for a diagonal, each in $O(n)$ time. Repeat this $n - 1$ times, $O(n^4)$.

Find an ear in $O(n)$ time; then recurse, $O(n^2)$ time.

First non-trivial algorithm: $O(n \log n)$. \[\text{[GJP}^+78]\]

A long series of papers and algorithms in 80s until Chazelle produced an optimal $O(n)$ algorithm. \[\text{[Cha91]}\]

Linear time algorithm insanely complicated; there are randomized, expected linear time algorithm that are more accessible.
Regularization and Triangulation Algorithm Outline

Definition (Monotone Polygonal Chain)
A polygonal chain $C$ is **monotone** w.r.t. line $L$ if any line orthogonal to $L$ intersects $C$ in at most one point.

Definition (Monotone Polygon)
A polygon is monotone w.r.t. $L$ if it can be decomposed into two chains, each monotone w.r.t. $L$.

- Partition polygon into trapezoids.
- Use trapezoids to make a monotone subdivision.
- Triangulate each monotone piece.

[？]+78

An $x$-monotone polygon
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. $\bullet$ — merge vertex
2. $\bigcirc$ — split vertex
3. $\bullet$ — start vertex
4. $\bullet$ — end vertex
5. $\bigcirc$ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — merge vertex
2. ○ — split vertex
3. ● — start vertex
4. ● — end vertex
5. ○ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. $\bigcirc$ — merge vertex
2. $\circ$ — split vertex
3. $\bullet$ — start vertex
4. $\bullet$ — end vertex
5. $\bigcirc$ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

**Vertex Ontology**

1. *merge vertex*
2. *split vertex*
3. *start vertex*
4. *end vertex*
5. *regular vertex*
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. $\bullet$ — merge vertex
2. $\circ$ — split vertex
3. $\bullet$ — start vertex
4. $\bullet$ — end vertex
5. $\circ$ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. o — merge vertex
2. ○ — split vertex
3. ● — start vertex
4. ● — end vertex
5. ● — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — merge vertex
2. ○ — split vertex
3. ● — start vertex
4. ● — end vertex
5. ○ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

**Vertex Ontology**

1. ○ → merge vertex
2. ○ → split vertex
3. ● → start vertex
4. ● → end vertex
5. ○ → regular vertex

---

**Triangulation**
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — merge vertex
2. ● — split vertex
3. ● — start vertex
4. ● — end vertex
5. ○ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — merge vertex
2. ○ — split vertex
3. ● — start vertex
4. ● — end vertex
5. ● — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — *merge vertex*
2. ○ — *split vertex*
3. ● — *start vertex*
4. ● — *end vertex*
5. ○ — *regular vertex*
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. ○ — merge vertex
2. ○ — split vertex
3. ■ — start vertex
4. ● — end vertex
5. ♦ — regular vertex
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. — *merge vertex*
2. — *split vertex*
3. — *start vertex*
4. — *end vertex*
5. — *regular vertex*
Partitioning a Polygon into Monotone Pieces

- At each vertex, extend vertical line until it hits a polygon edge.
  - Each face of this decomposition is a pseudo trapezoid.
- Use plane sweep algorithm.
  - Time complexity is $O(n \log n)$.
- For each split (resp. merge) vertex $v$, add a diagonal that connects $v$ to the vertex of its left (resp. right) trapezoid.

Vertex Ontology

1. $\circ$ — merge vertex
2. $\circ$ — split vertex
3. $\bullet$ — start vertex
4. $\bullet$ — end vertex
5. $\bullet$ — regular vertex
Lemma (x-monotone)

A polygon is x-monotone if it has no split vertices and no merge vertices.

- Suppose $P$ is a non x-monotone polygon.
- We have to prove that $P$ contains a split or a merge vertex.

$q$ — the top endpoint of $\ell \cap P$.

$p$ — the bottom endpoint of $\ell \cap P$.

$r$ — the first intersection of $\ell$ with $\partial P$ starting at $q$ going counterclockwise.

There are two cases

(a) $p = r \Rightarrow \exists$ merge vertex

(b) $p \neq r \Rightarrow \exists$ split vertex
**Triangulating Monotone Polygons**

Triangulate a monotone polygon $P$ on $n$ vertices.

1. Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
2. Initialize a stack $\Gamma$, push($\Gamma, v_1, v_2$).
3. for $i = 3, \ldots, n$ do
4.   if $v_i$ and $v_j = \text{top}(\Gamma)$ on same chain
5.     Add diagonals $v_i v_j, \ldots, v_i v_k$, where $v_k$ is last to admit legal diagonal.
6.     $v_{k+1}, \ldots, v_j = \text{pop}(\Gamma)$.
7.     push($\Gamma, v_i$).
8. else
9.     Add diagonals from $v_i$ to all vertices on stack.
10. $v_j = \text{pop}(\Gamma)$.
11. clear($\Gamma$).
12. push($\Gamma, v_j, v_i$).

**Case I**

$\Gamma : v_{\text{bot}}, \ldots, v_{\text{top}}$

<table>
<thead>
<tr>
<th>$v_{\text{bot}}$</th>
<th>$v_k$</th>
<th>$v_{\text{top}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$v_i$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$v_{\text{bot}}$</th>
<th>$v_k$</th>
<th>$v_{\text{top}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$v_i$</td>
</tr>
</tbody>
</table>

**Case II**

$\Gamma : v_{\text{bot}}, \ldots, v_{\text{top}}$

<table>
<thead>
<tr>
<th>$v_{\text{bot}}$</th>
<th>$v_k$</th>
<th>$v_{\text{top}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$v_i$</td>
</tr>
</tbody>
</table>

Diagram of Case I and Case II showing the triangulation process.
Triangulating Monotone Polygons

Triangulate a monotone polygon $P$ on $n$ vertices.

1. Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$.
2. Initialize a stack $\Gamma$, push($\Gamma, v_1, v_2$).
3. for $i = 3, \ldots, n$ do
4. if $v_i$ and $v_j = \text{top}(\Gamma)$ on same chain
5. I $\{$
6. Add diagonals $v_i v_j, \ldots, v_i v_k$, where $v_k$ is last to admit legal diagonal.
7. $v_{k+1}, \ldots, v_j = \text{pop}(\Gamma)$.
8. push($\Gamma, v_i$).

9. else
10. II $\{$
11. Add diagonals from $v_i$ to all vertices on stack.
12. $v_j = \text{pop}(\Gamma)$.
13. clear($\Gamma$).
14. push($\Gamma, v_j, v_i$).

Case I

Case II

\(\Gamma : v_{\text{bot}}, \ldots, v_{\text{top}}\)}
Triangulate a monotone polygon $P$ on $n$ vertices.

1. Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
2. Initialize a stack $\Gamma$, push($\Gamma, v_1, v_2$).
3. for $i = 3, \ldots, n$ do
4.  if $v_i$ and $v_j = \text{top}(\Gamma)$ on same chain
5.  I Add diagonals $v_i v_j, \ldots, v_i v_k$, where $v_k$ is last to admit legal diagonal.
6.  $v_{k+1}, \ldots, v_j = \text{pop}(\Gamma)$.
7.  push($\Gamma, v_i$).
8. else
9.  II Add diagonals from $v_i$ to all vertices on stack.
10. $v_j = \text{pop}(\Gamma)$.
11. clear($\Gamma$).
12. push($\Gamma, v_j, v_i$).

**Case I**

$\Gamma$ : $v_{\text{bot}}, \ldots, v_k, v_i$

**Case II**

$\Gamma$ : $v_{\text{bot}}, \ldots, v_{\text{top}}$
Triangulating Monotone Polygons

Triangulate a monotone polygon $P$ on $n$ vertices.

1. Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$
2. Initialize a stack $\Gamma$, push($\Gamma, v_1, v_2$).
3. for $i = 3, \ldots, n$ do
   4. if $v_i$ and $v_j = \text{top}(\Gamma)$ on same chain
      5. I Add diagonals $v_i v_j, \ldots, v_i v_k$, where $v_k$ is last to admit legal diagonal.
         6. $v_{k+1}, \ldots, v_j = \text{pop}(\Gamma)$.
         7. push($\Gamma, v_i$).
      8. else
         9. II Add diagonals from $v_i$ to all vertices on stack.
            10. $v_j = \text{pop}(\Gamma)$.
            11. clear($\Gamma$).
            12. push($\Gamma, v_j, v_i$).

\[ \text{Case I: } \Gamma : v_{\text{bot}}, \ldots, v_k, v_i \]
\[ \text{Case II: } \Gamma : v_{\text{bot}}, \ldots, v_{\text{top}} \]
## Triangulating Monotone Polygons

Triangulate a monotone polygon $P$ on $n$ vertices.

1. Sort the vertices in lexicographically increasing order to yield $v_1, v_2, \ldots, v_n$.
2. Initialize a stack $\Gamma$, push($\Gamma, v_1, v_2$).
3. for $i = 3, \ldots, n$ do
   4. if $v_i$ and $v_j = \text{top}(\Gamma)$ on same chain
       5. I Add diagonals $v_i v_j, \ldots, v_i v_k$, where $v_k$ is last to admit legal diagonal.
       6. $v_{k+1}, \ldots, v_j = \text{pop}(\Gamma)$.
       7. $\text{push}(\Gamma, v_i)$.
6. else
      7. II $\text{Add diagonals from } v_i \text{ to all vertices on stack.}$
      8. $v_j = \text{pop}(\Gamma)$.
      9. $\text{clear}(\Gamma)$.
     10. $\text{push}(\Gamma, v_j, v_i)$.

### Case I

```
\[ \Gamma : v_{\text{bot}}, \ldots, v_k, v_i \]
```

### Case II

```
\[ \Gamma : v_j, v_i \]
```
Regularization and Triangulation Algorithm Complexity

- Regularization via plane sweep takes $O(n \log n)$ time.

- Triangulation
  - Sorting by merging the two monotone chains of $P$ takes $O(n)$ time.
  - A vertex is added to stack once. Once it’s visited during a scan, it’s removed from the stack.
  - In each step, at least one diagonal is added; or the reflex stack chain is extended by one vertex.
  - Triangulating a monotone polygon takes $O(n)$ time.

- Total time for polygon triangulation is therefore $O(n \log n)$. 
Outline

1. Triangulation
   - Polygon Terms and Definitions
   - The Art Gallery
   - Regularization and Triangulation
   - Literature
Triangulation Bibliography I

B. Chazelle.
Triangulating a Simple Polygon in Linear Time.

Triangulating a Simple Polygon.

An $O(n \log \log(n))$ Time Algorithm for Triangulating a Simple Polygon.

Mark de Berg, Mark van Kreveld, Mark H. Overmars, and Otfried Cheong.
*Computational Geometry: Algorithms and Applications*.

Alexey V. Skvortsov, Yuri L. Kostyuk
Efficient algorithms for Delaunay triangulation.

Bernard Chazelle.
Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm.

Michael J. Laszlo.
*Computational Geometry and Computer Graphics in C++*.
Prentice-Hall, 1996