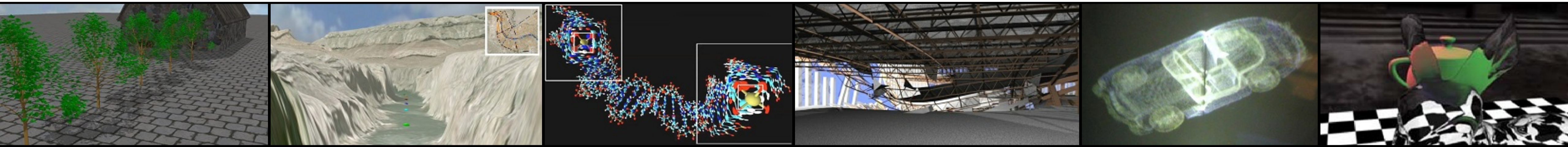
COT 4521: Introduction to Computational Geometry

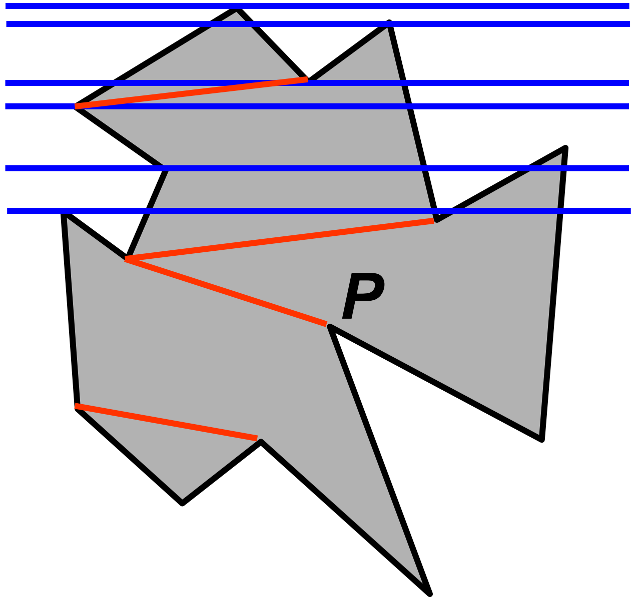
Monotone Triangulation

Paul Rosen
Assistant Professor
University of South Florida



POLYGON TRIANGULATION

- ALGORITHM: POLYGON TRIANGULATION: MONOTONE PARTITION
  - Partition into monotone polygons
  - **Triangulate each monotone polygon**



Description of the processing triangulation of Y-monotone polygons

- The algorithm processes one vertex at a time in order of decreasing Y coordinate, creating diagonals of polygon P.
  - The sweep line moves from top to down and stops at each vertex of polygon P

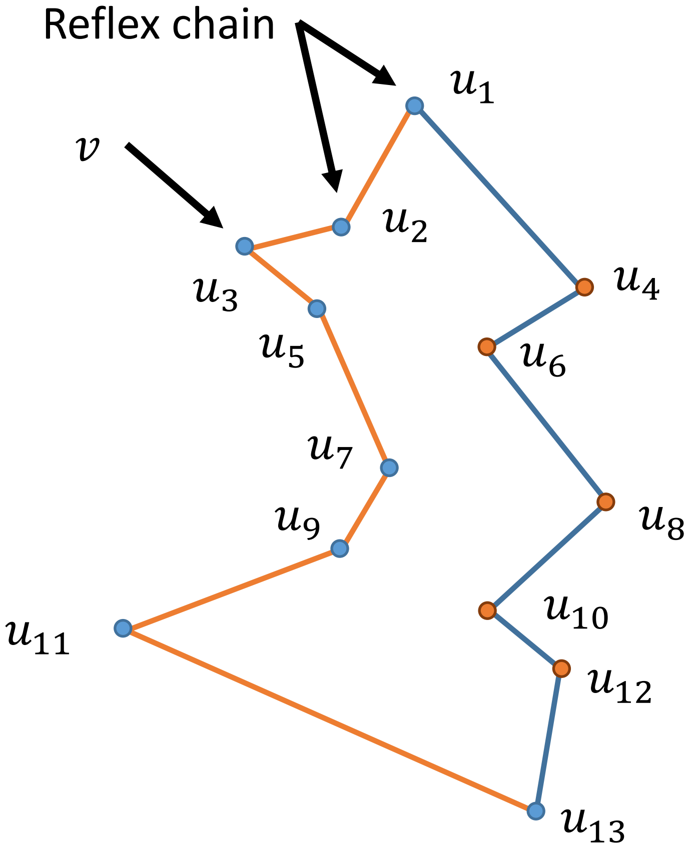
- Each diagonal bounds a triangle, and leaves a polygon with one less side still to be triangulated

DESCRIPTION OF THE PROCESSING TRIANGULATION OF Y-MONOTONE POLYGONS

- Sort the vertices top-to-down by a merge of the two chains
- Initialize a stack. Push the first two vertices in the stack
- Take the next vertex u, and triangulate as much as possible, top-down, while popping the stack
- Push u onto the stack

Triangulation of monotone polygon

- Sort vertices by y-coordinate by a merge of the two chains.
- Let $u_1, u_2, \dots, u_N$ be the sorted sequence of vertices, so $y(u_1) > y(u_2) > \cdots > y(u_N)$.
- The algorithm preforms operations on a reflex chain, which is stored as a stack
- Initialization
  - Reflex chain pushes two top vertices
  - Let v be the third highest vertex
- processes one vertex at a time in decreasing y
  - At each step process 1 of 3 cases

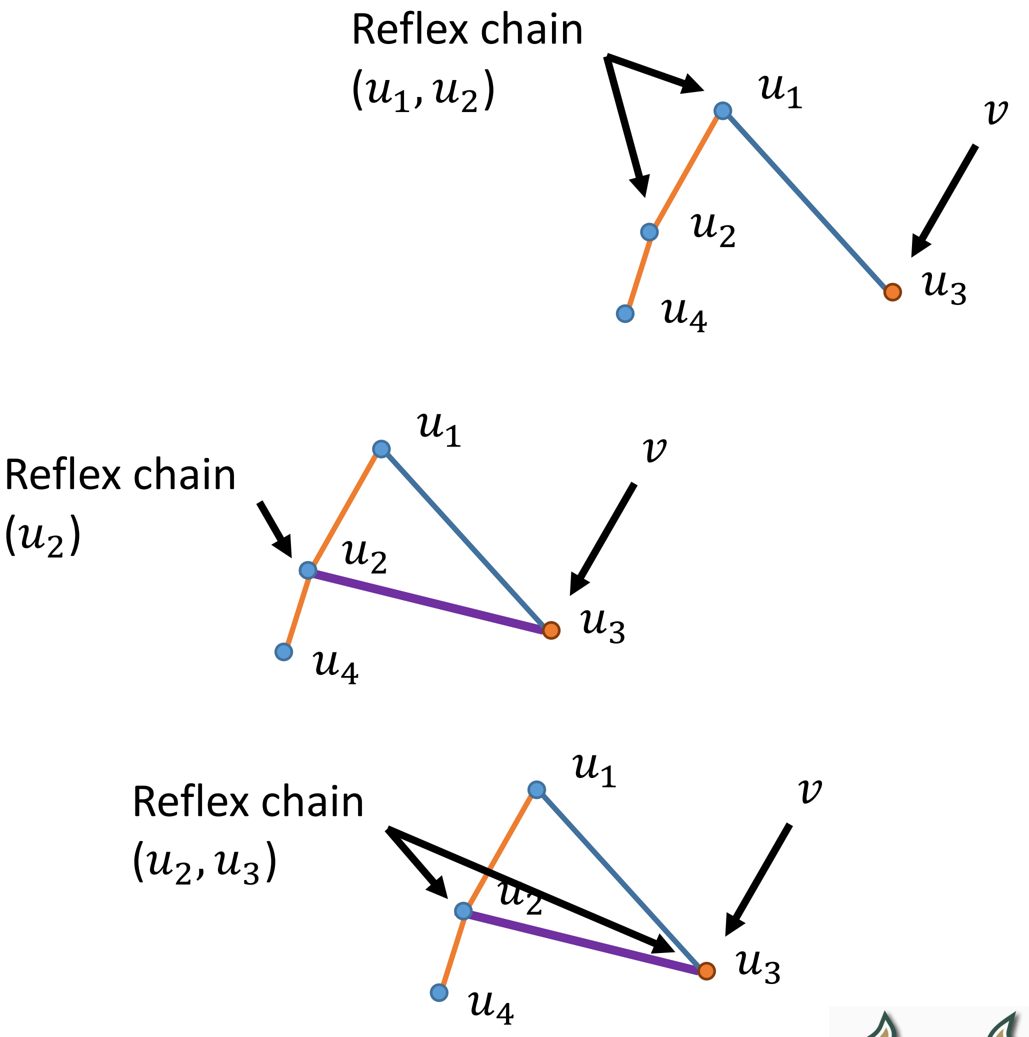


TRIANGULATION OF MONOTONE POLYGON

- WHILE V != LOWEST VERTEX DO:
  - Case 1: v is on chain opposite reflex chain
  - Case 2: v is adjacent to bottom of reflex chain and v+ is strictly convex
  - Case 3: v is adjacent to bottom of reflex chain and v+ is reflex or flat

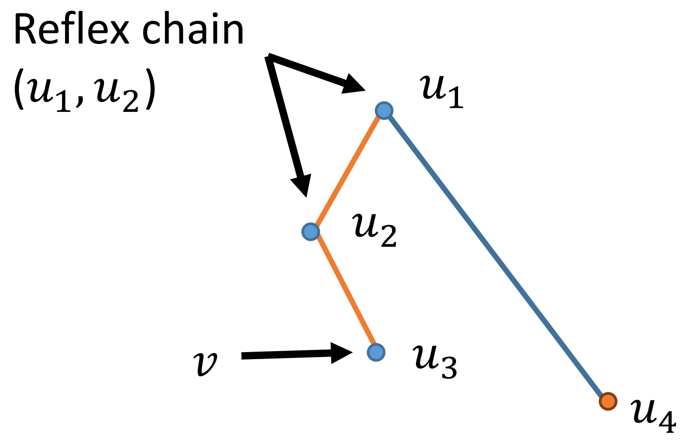
Triangulation of monotone polygon

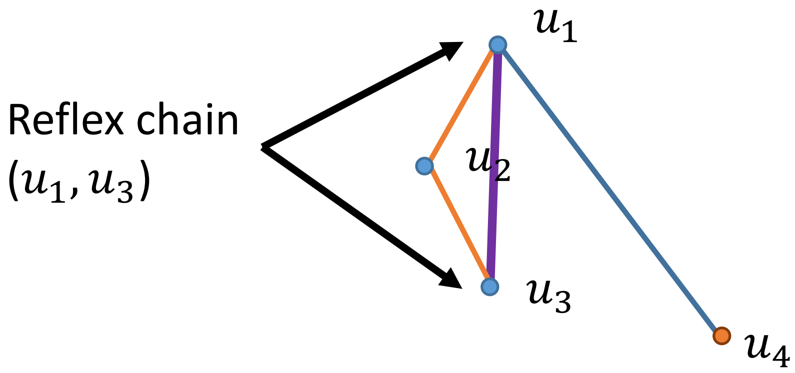
- Case 1: v is on chain opposite reflex chain
  - Draw diagonal from v to second vertex from top of chain
  - Remove top of chain
  - If chain has one element then add v, advance v



TRIANGULATION OF MONOTONE POLYGON

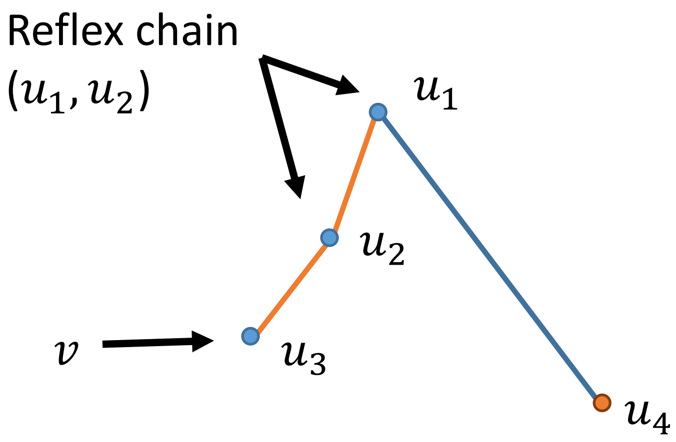
- CASE 2: V IS ADJACENT TO BOTTOM OF REFLEX CHAIN AND V+ IS STRICTLY CONVEX
    - Draw diagonal from v to second vertex from bottom of chain
    - Remove bottom of chain
    - Add v to chain, advance v

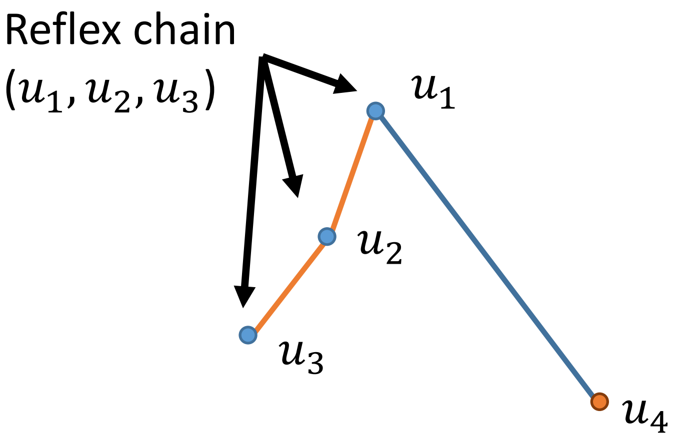




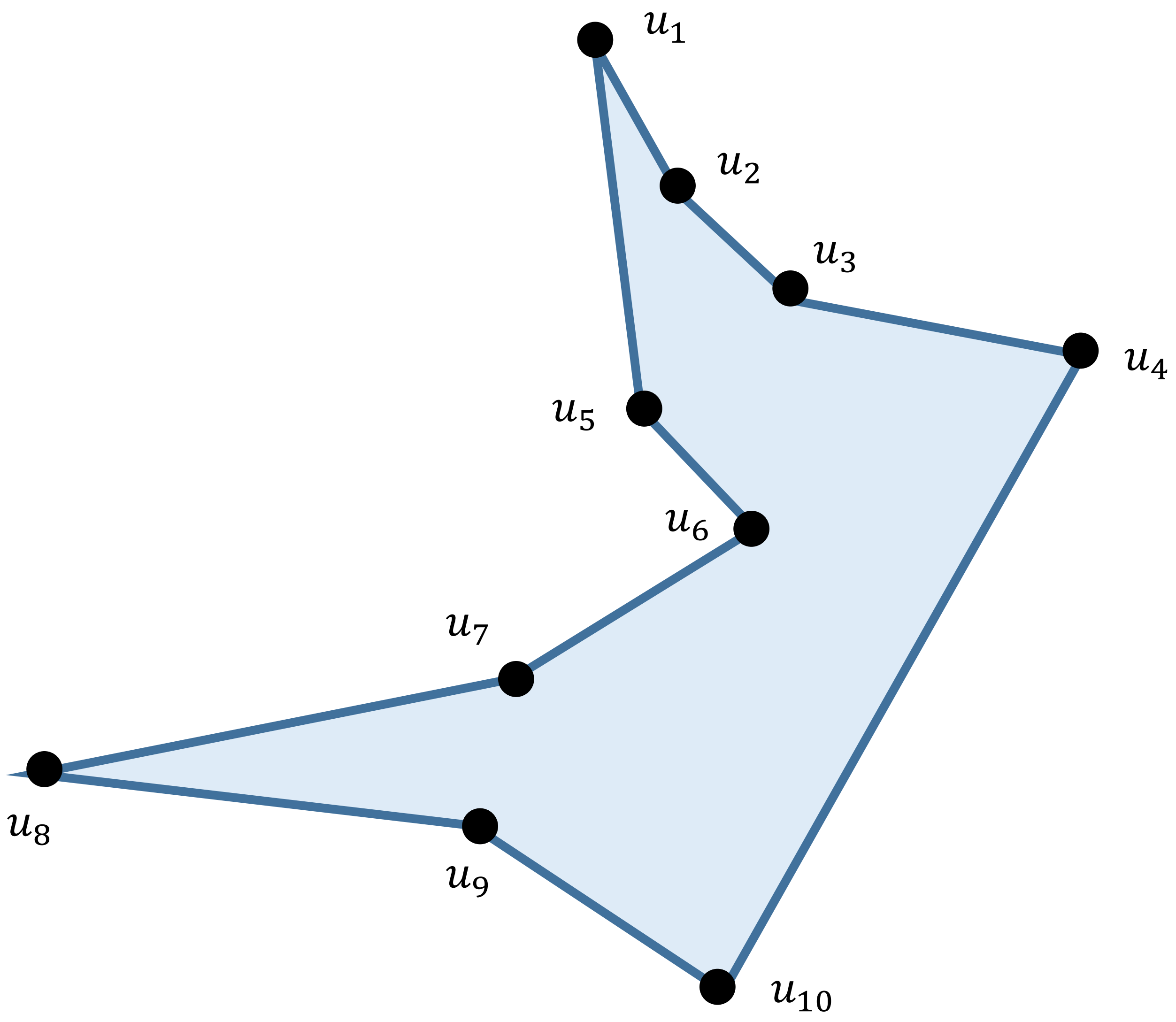
Triangulation of monotone polygon

- Case 3: v is adjacent to bottom of reflex chain and v+ is reflex or flat
  - Add v to bottom of reflex chain, advance v





TRIANGULATION OF MONOTONE POLYGON EXAMPLE



Triangulation of monotone polygon Example

<u>Case 1: v is on chain opposite reflex chain</u>

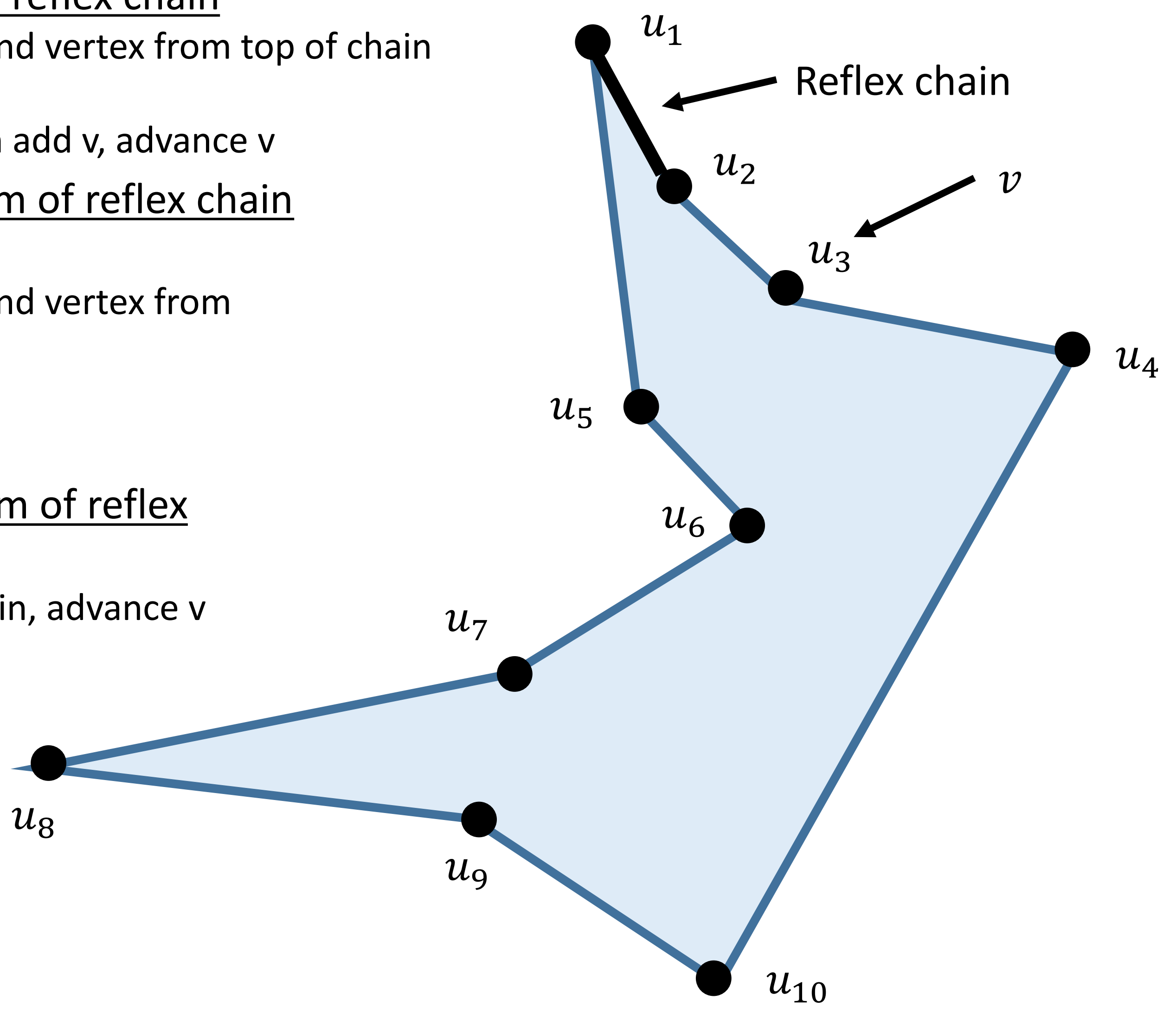
- Draw diagonal from v to second vertex from top of chain
- Remove top of chain
- If chain has one element then add v, advance v

<u>Case 2: v is adjacent to bottom of reflex chain AND v+ is strictly convex</u>

- Draw diagonal from v to second vertex from bottom of chain
- Remove bottom of chain
- Add v to chain, advance v

<u>Case 3: v is adjacent to bottom of reflex chain AND v+ is reflex or flat</u>

- Add v to bottom of reflex chain, advance v



Case 3

Triangulation of monotone polygon Example

<u>Case 1: v is on chain opposite reflex chain</u>

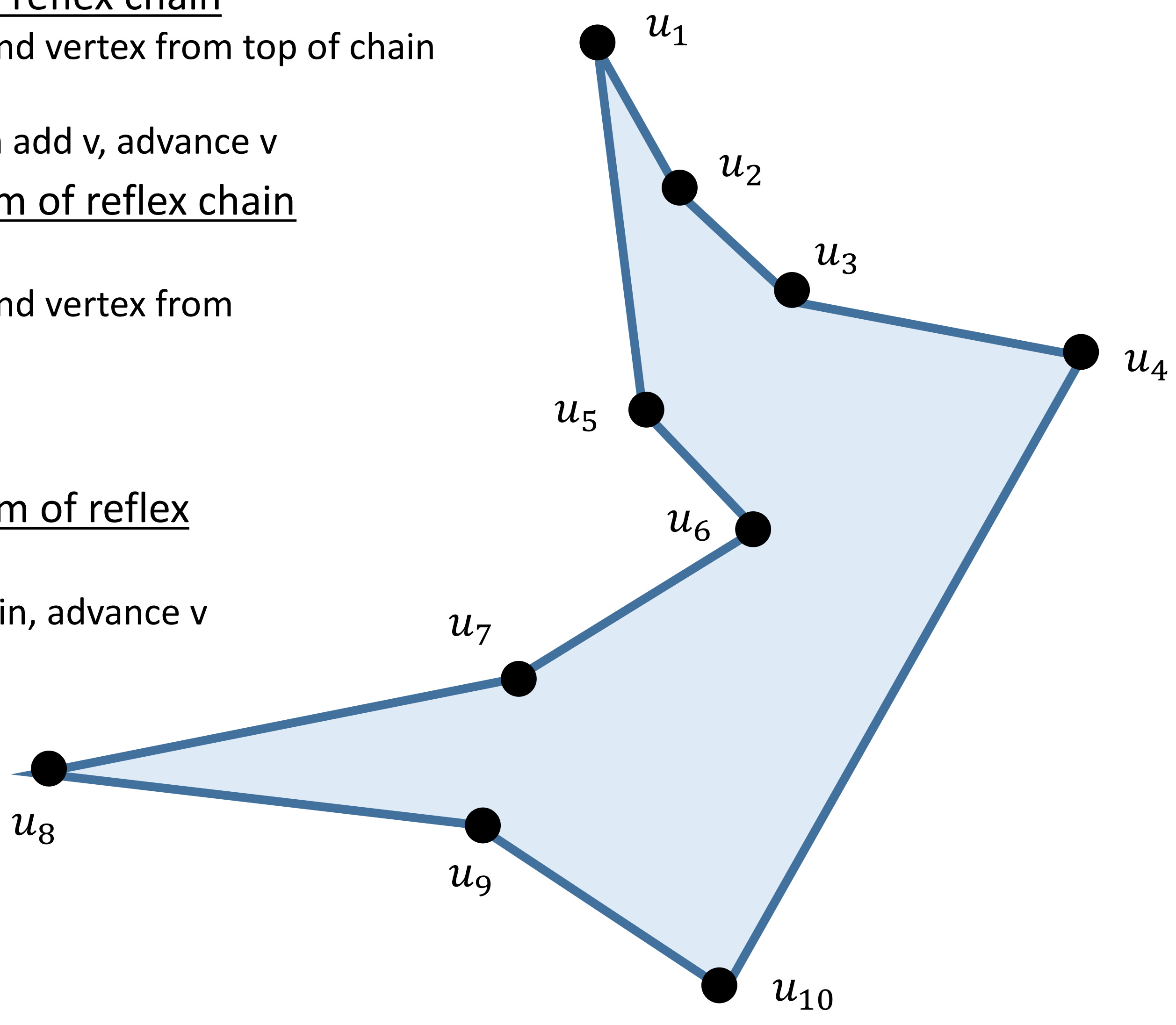
- Draw diagonal from v to second vertex from top of chain
- Remove top of chain
- If chain has one element then add v, advance v

<u>Case 2: v is adjacent to bottom of reflex chain AND v+ is strictly convex</u>

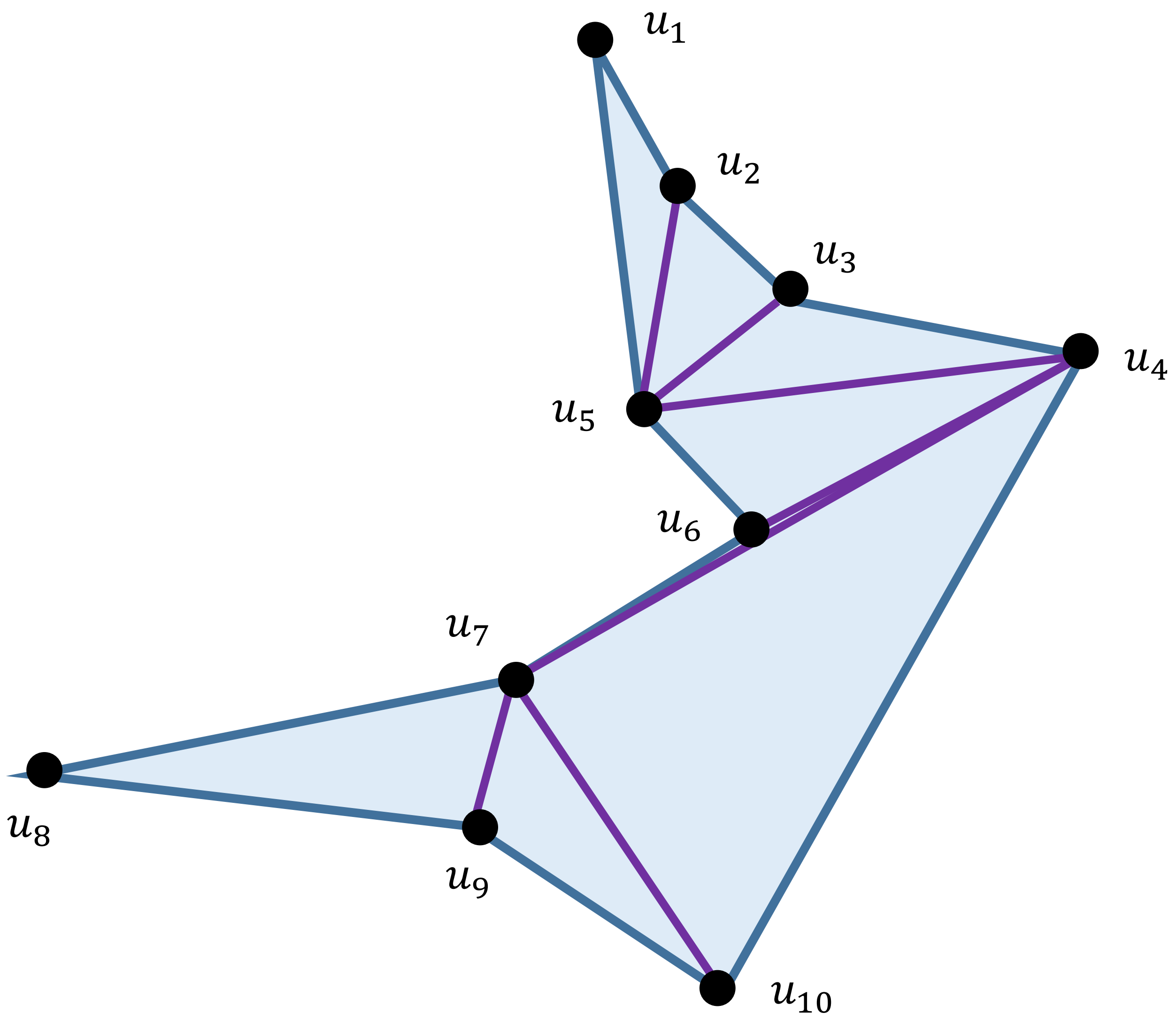
- Draw diagonal from v to second vertex from bottom of chain
- Remove bottom of chain
- Add v to chain, advance v

<u>Case 3: v is adjacent to bottom of reflex chain AND v+ is reflex or flat</u>

- Add v to bottom of reflex chain, advance v



Triangulation of monotone polygon Example



ANALYSIS OF TRIANGULATING A MONOTONE POLYGON

- THE INITIAL SORT (MERGE) REQUIRES O(N) TIME.
- TRIANGULATION VISITS EACH OF THE N VERTICES AND PLACES ON THE STACK EXACTLY ONCE, EXCEPT WHEN THE WHILE FAILS IN CASE (II).
  - This happens at most once per vertex, so that time can be charged to the current vertex.

- $\Rightarrow$ THE ALGORITHM REQUIRES O(N) TIME TO TRIANGULATE A MONOTONE POLYGON, WHERE N IS THE NUMBER OF VERTICES OF THE POLYGON.

Putting it all together

- Analysis of Triangulation of Simple Polygon
  - Sort vertices by y coordinates: O(N log N)
  - Perform plane sweep to partition polygons: O(N log N)
  - Triangulate each monotone polygon: O(N)

- Overall: O(N log N)
  - N is the number of vertices of the polygon

SUMMARY

- TRIANGULATION BY EAR REMOVAL IMPROVES THE DIAGONAL-BASED TRIANGULATION FROM $O(n^4)$ TO $O(n^2)$
- PARTITION A SIMPLE POLYGON INTO MONOTONE PARTS AND TRIANGULATION OF MONOTONE POLYGONS REQUIRES $O(n \log n)$ TIME COMPLEXITY

Towards Linear-Time Triangulation

Year	Complexity	Authors
1911	$O(n^2)$	Lennes
1978	$O(n \log n)$	Garey et al.
1983	$O(n \log r)$, r reflex	Hertel & Melhorn
1984	$O(n \log s)$, s sinuosity	Chazelle & Incerpi
1986	$O(n \log \log n)$	Tarjan & Van Wyk
1988	$O(n + nt_0)$, t_0 int. triangles	Toussaint
1989	$O(n \log^* n)$, randomized	Clarkson, Tarjan & Van Wyk
1990	$O(n \log^* n)$ bounded integer coordinates	Kirkpatrick, Klawe, Tarjan
1990	$O(n)$	Chazelle
1991	$O(n \log^* n)$, randomized	Seidel

Linear-Time Triangulation

- Chazelle's Algorithm (High-Level Sketch)
  - Computes visibility map
  - Algorithm is like MergeSort (divide-and-conquer)
    - Partition polygon of n vertices into n/2 vertex chains
    - Merge visibility maps of subchains to get one for chain
  - Improve this by dividing process into 2 phases:
    1. Coarse approximations of visibility maps for linear-time merge
    2. Refine coarse map into detailed map in linear time

Convex Partitioning

- Polygon partitioning is an important preprocessing step for many geometric algorithms

- Partitioning a polygon means completely dividing the interior into nonoverlapping pieces.

- Covering a polygon means that our decomposition is permitted to contain mutually overlapping pieces.

- An issue associated with polygon decomposition is whether we are allowed to add Steiner vertices (either by splitting edges or adding interior points) or whether we are restricted to adding chords between two existing vertices.

Convex Partitioning

- Competing Goals:
  - minimize number of convex pieces
  - minimize partitioning time
- Add (Steiner) points or just use diagonals and not add points?

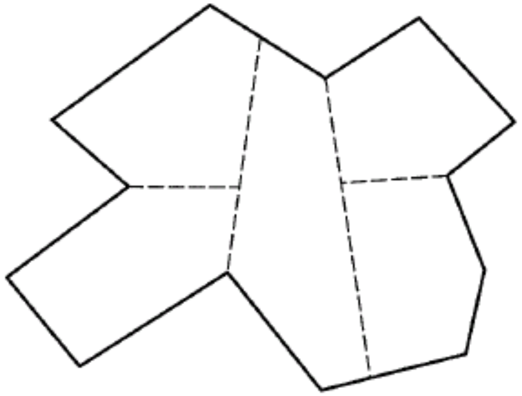
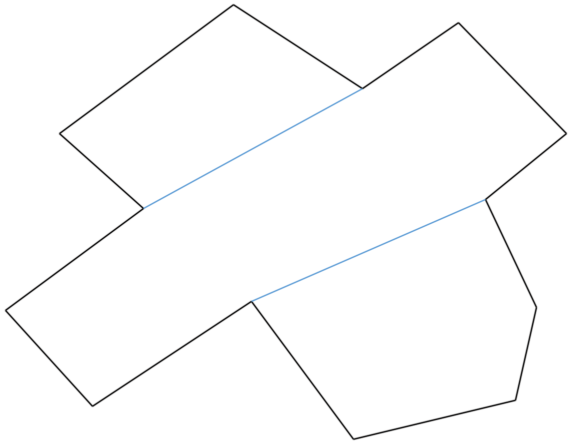


FIGURE 2.10 $r + 1$ convex pieces: $r = 4$; 5 pieces.

Adding segments with Steiner points.
r = number of reflex vertices



Adding only diagonals.

Convex Partitioning

- Theorem (Chazelle): Let F be the fewest number of convex pieces into which a polygon may be partitioned. For a polygon of r reflex vertices:

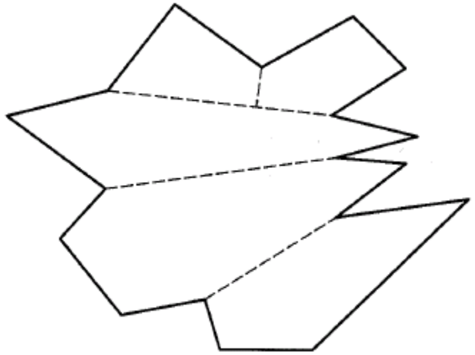


FIGURE 2.11 $\lceil r/2 \rceil + 1$ convex pieces: $r = 7$; 5 pieces.

$$\left\lceil \frac{r}{2} \right\rceil + 1 \le \Phi \le r + 1$$

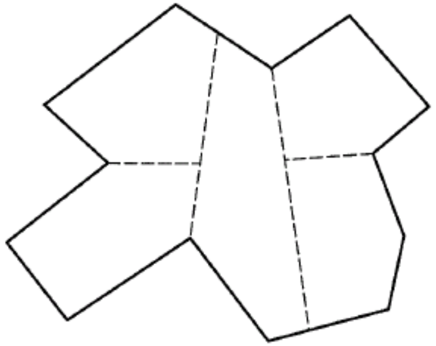


FIGURE 2.10 $r + 1$ convex pieces: $r = 4$; 5 pieces.

<u>Lower bound</u>:
Must eliminate all reflex vertices.
Single segment resolves at most 2 reflex angles.

<u>Upper bound</u>:
Bisect each reflex angle.