Polygon Triangulation

Simple Polygons

Definition

1. A polygon is the region of a plane bounded by a finite collection of line segments forming a **simple <u>closed</u> curve**.

2. “Simple closed curve” means a certain deformation of a circle.

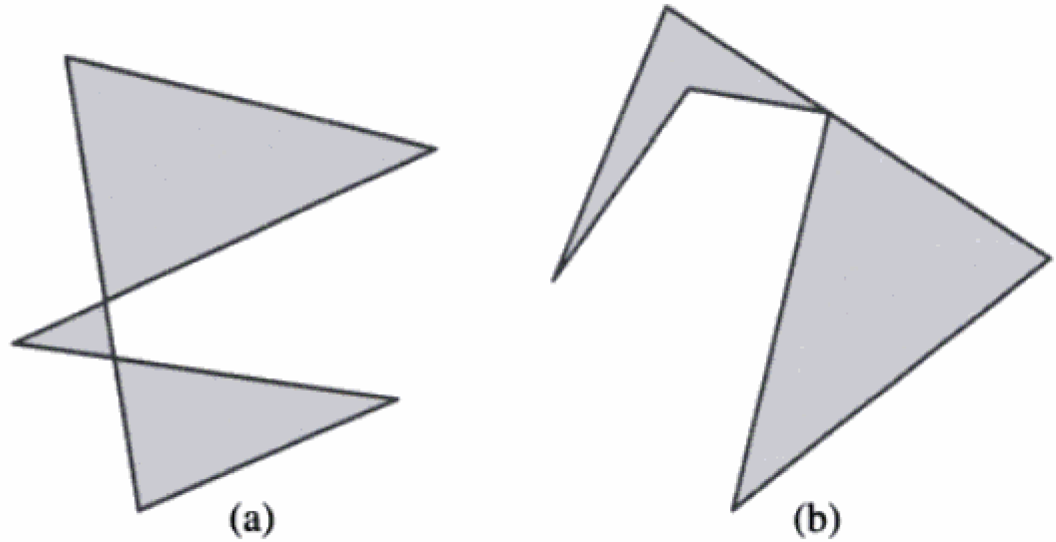


FIGURE 1.1 Nonsimple polygons.

Nonsimple polygons

1. For both objects in the figure, the segments satisfy condition (1) (adjacent segments share a common point)
2. but not condition (2): nonadjacent segments intersect.

Visibility

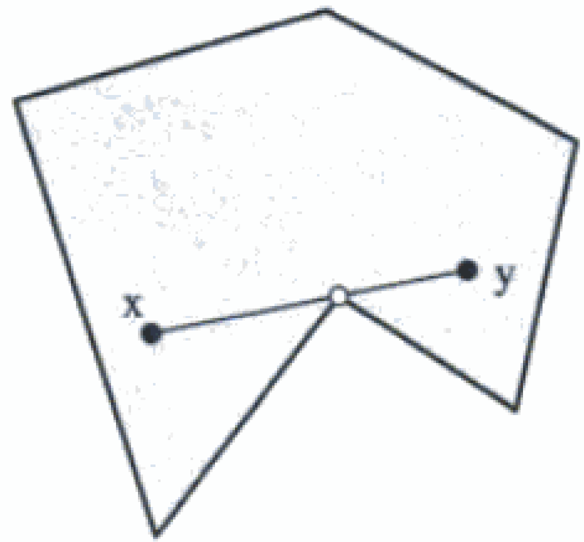


FIGURE 1.2 Grazing contact of line of sight.

Visibility

1. Point x can see point y (or y is visible to x) *iff* the closed segment xy is nowhere exterior to the polygon P ($xy \subseteq P$)
2. A vertex can block vision

Visibility

Visibility

1. x has clear visibility to y if $xy \subseteq P$ and $xy \cap \partial P \subseteq \{x,y\}$
2. ∂P means the boundary of a polygon P
3. By definition, $\partial P \subseteq P$
4. A guard is a point.
5. A set of guards is said to cover a polygon if every point in the polygon is visible to some guard.

Triangulation

Diagonals and Triangulation

1. A diagonal of a polygon P is a line segment between two of its vertices a and b that are clearly visible to one another
2. The open segment from a to b does not intersect ∂P; thus a diagonal cannot make grazing contact with the boundary.
3. Two diagonals are noncrossing if their intersection is a subset of their endpoints. They share no interior points.
4. A triangulation of a polygon P
   - Add as many noncrossing diagonals to a polygon as possible so that the interior can be partitioned into triangles.

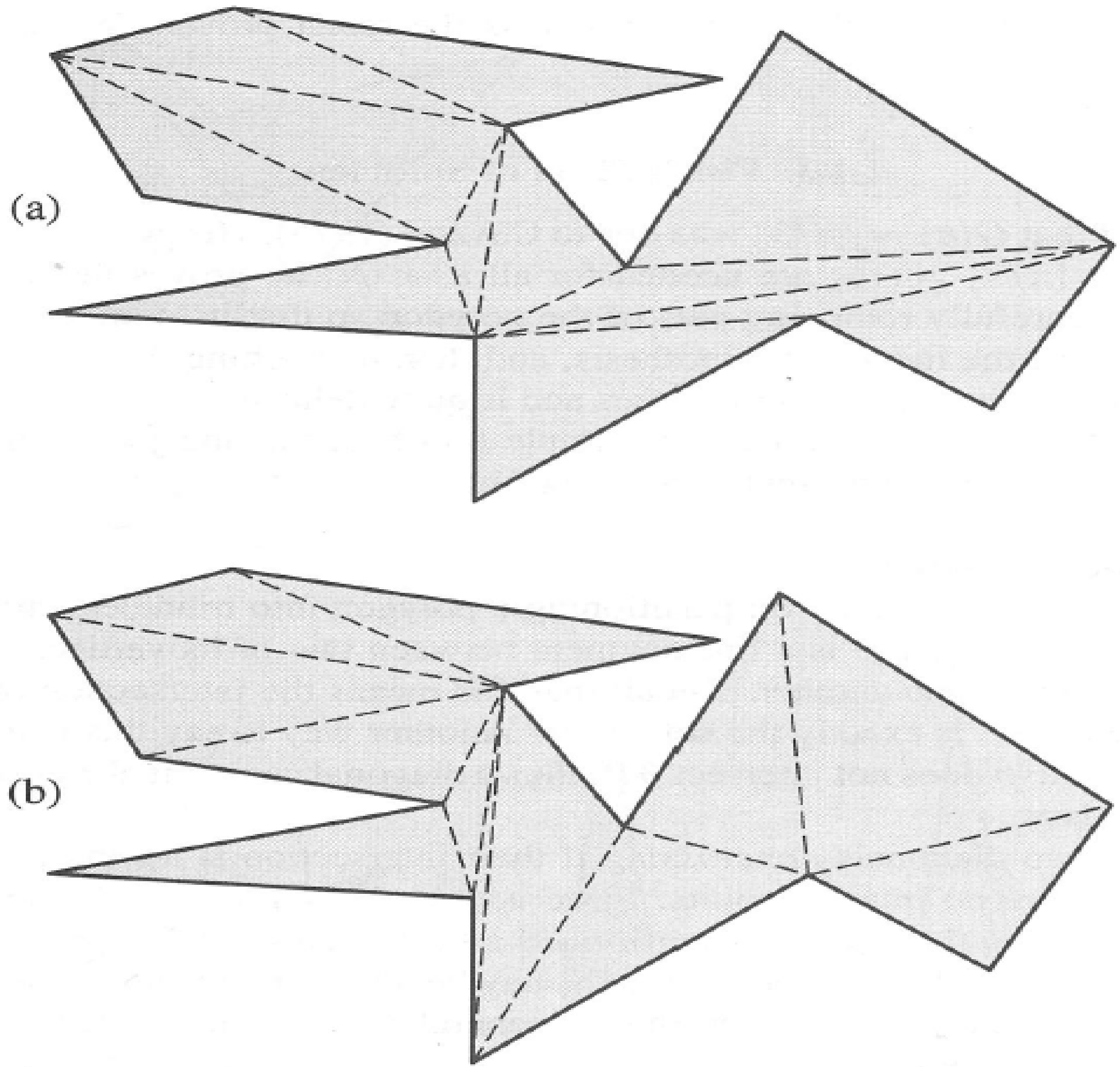


FIGURE 1.6 Two triangulations of a polygon of $n = 14$ vertices.

Polygon Triangulation

Triangulation Theory

Existence of a Diagonal

Lemma 1.2.1

Every polygon must have at least one strictly convex vertex.

Existence of a Diagonal

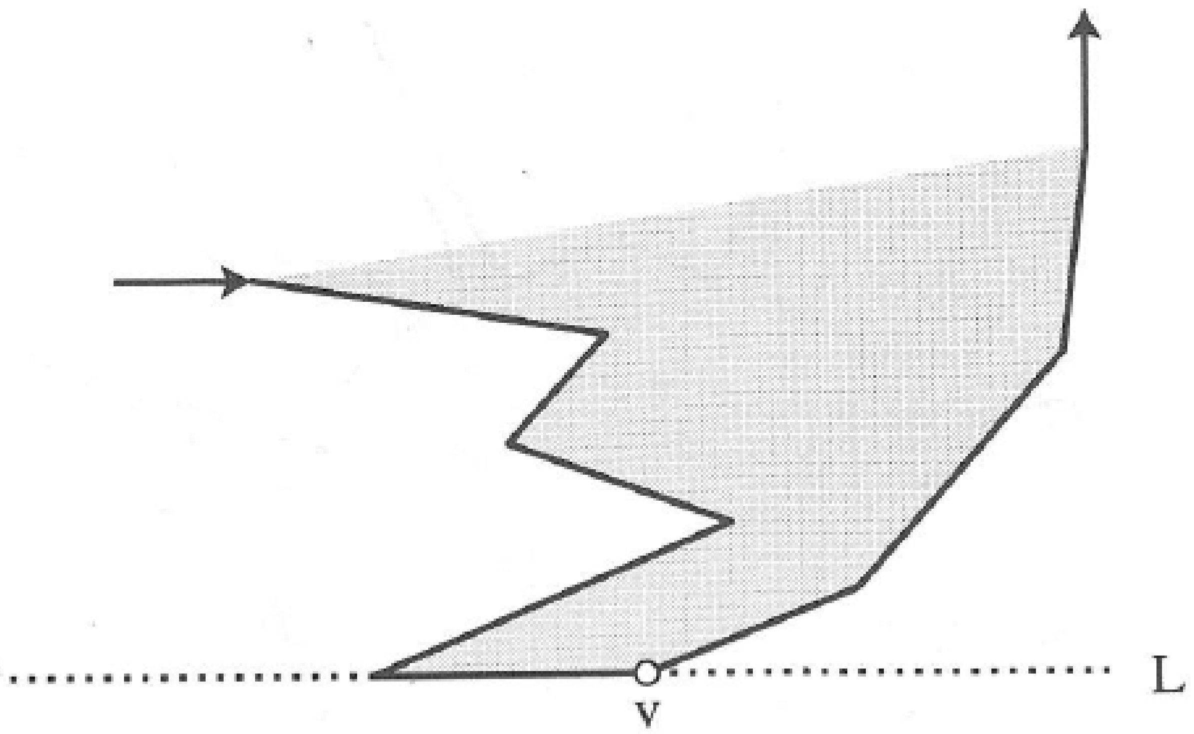


FIGURE 1.11 The rightmost lowest vertex must be strictly convex.

Existence of a Diagonal

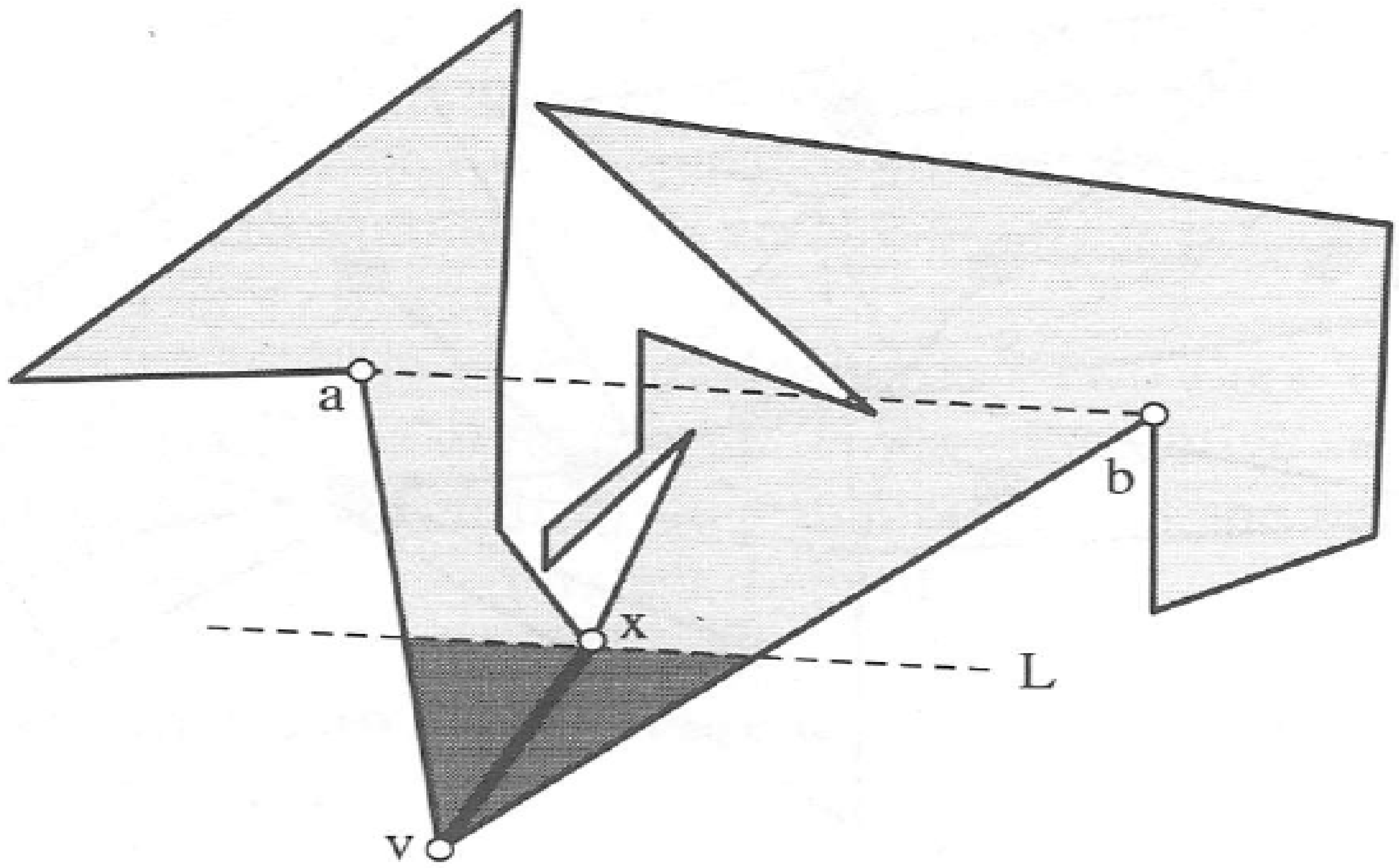


FIGURE 1.12 vx must be a diagonal.

Properties of Triangulations

Lemma 1.2.4 (Number of Diagonals)

Every triangulation of a polygon P of n vertices uses $n - 3$ diagonals and consists of $n - 2$ triangles

Triangulations Dual

1. The *dual* T of triangulation of a polygon is a graph with a node associated with each triangle and an arc between two nodes iff their triangles share a diagonal.

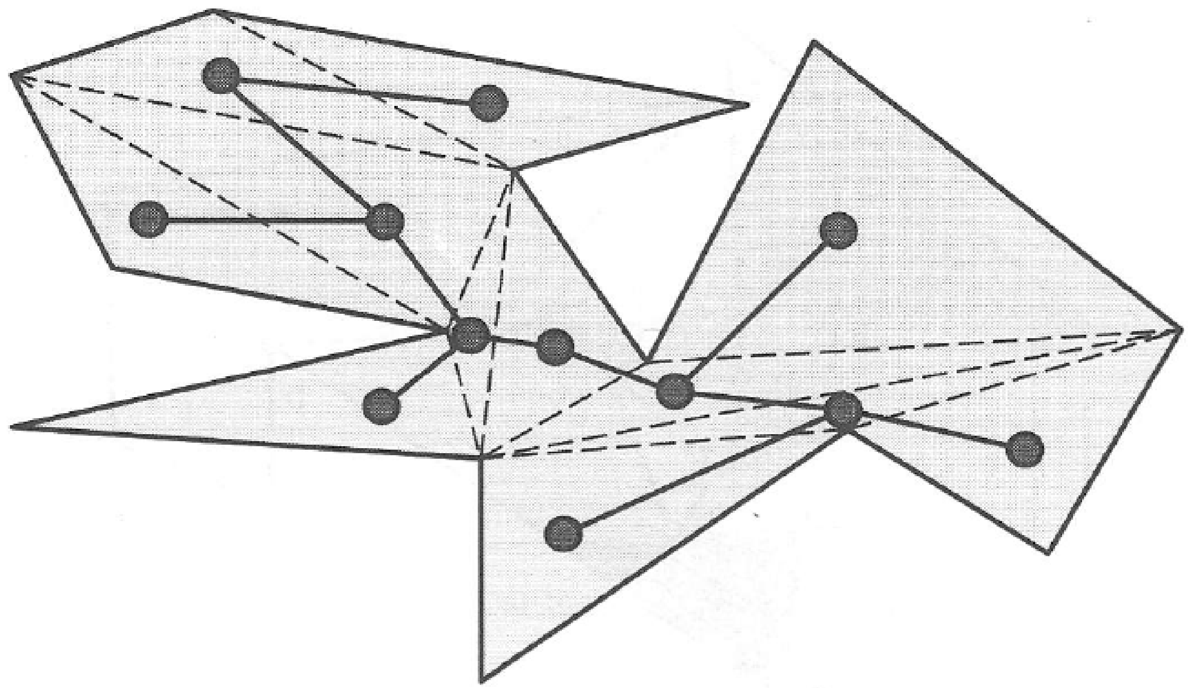


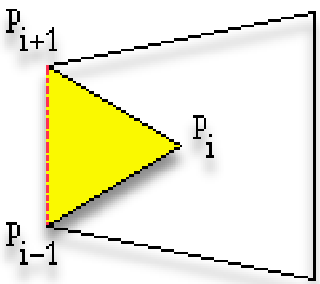
FIGURE 1.14 Triangulation dual.

Triangulation Dual

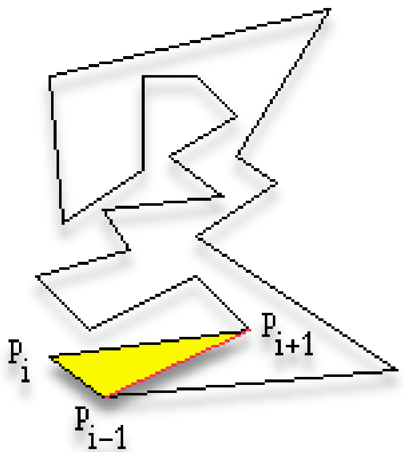
- Lemma 1.2.6
  1. The dual T of a triangulation is a **tree**, with each node of degree at most **three**.
  2. Proof

Triangulation Dual

1. Three consecutive vertices of a polygon *a*, *b*, *c* form an ear of the polygon if *ac* is a diagonal; *b* is the ear tip.



P_i is not an ear

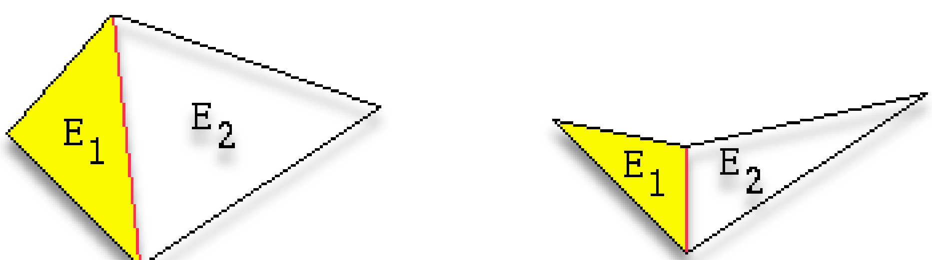


P_i is an ear

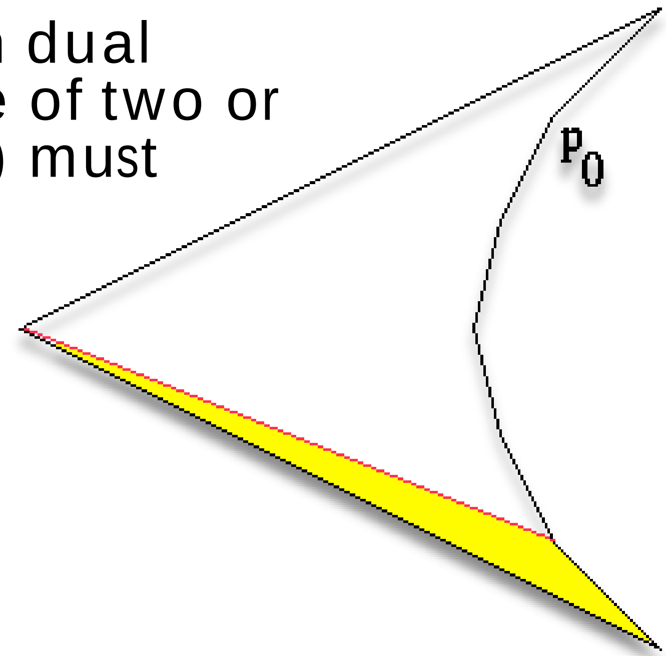
Triangulation Dual

Meisters's Two Ears Theorem [1975]

1. Every polygon of $n \geq 4$ vertices has at least two non-overlapping ears.
2. Proof
   - A leaf node in a triangulation dual corresponds to an ear. A tree of two or more nodes(by Lemma 1.2.4) must have at least two leaves.



Base Case: A quadrilateral has only 2 ears.



A simple polygon with only 2 ears.

Polygon Triangulation

Area of polygon

Area of a Triangle

- Let us denote this area as $A(T)$
- The area is one half the base times the altitude.
- The base is easy: $|a-b|$ (the length of the vector $a–b$)
- What about the altitude?
  - Not so easy.

Cross Product

- Recall the magnitude of the cross product of two vectors is the area of the parallelogram.
- A triangle is half of a parallelogram.
- Thus, the area of triangle whose three vertices are arbitrary points a, b, c is half the length of A X B
- A = $b - a$ and B = $c - a$

Area of a Convex Polygon

- Find the area of any polygon by using an expression for the area of a triangle.
- First triangulation
- Every convex polygon may be triangulated as a "fan," with all diagonals incident to a common vertex.
- The area of a polygon with vertices v_0, $v_1, \ldots, v_{n-1}$ labeled counterclockwise (Figure 1.16) can be calculated as

$$\mathcal{A}(Q) = \mathcal{A}(a, b, c) + \mathcal{A}(a, c, d) = \mathcal{A}(d, a, b) + \mathcal{A}(d, b, c).$$

Area of a Convex Polygon

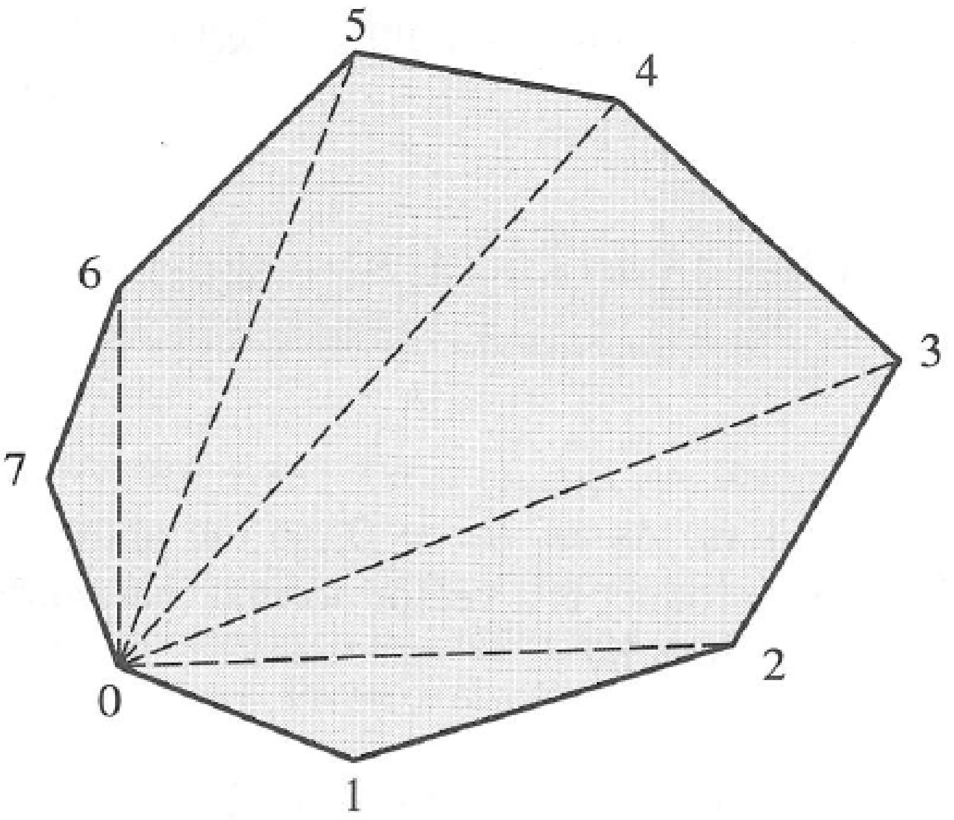


FIGURE 1.16 Triangulation of a convex polygon. The fan center is at 0.

Area of a Convex Quadrilateral

- Two different triangulations of a convex quadrilateral $Q = (a, b, c, d)$

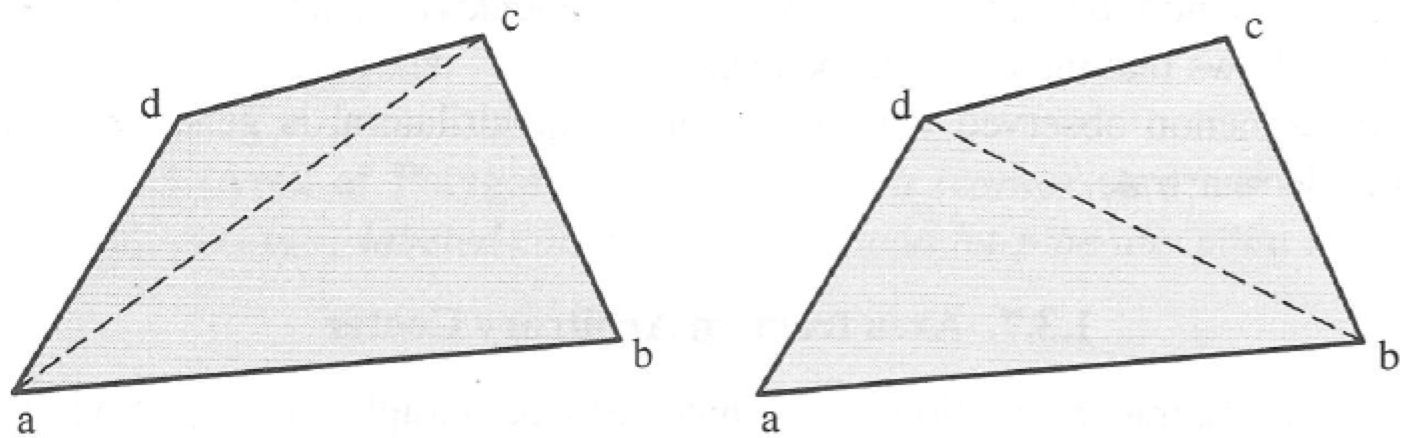


FIGURE 1.17 The two triangulations of a convex quadrilateral.

- The area may be written in two ways.

$$\mathcal{A}(Q) = \mathcal{A}(a, b, c) + \mathcal{A}(a, c, d) = \mathcal{A}(d, a, b) + \mathcal{A}(d, b, c).$$

Area of a Convex Quadrilateral

- Applying Equation (1.2) to

$$\mathcal{A}(Q) = \mathcal{A}(a, b, c) + \mathcal{A}(a, c, d)$$

- We get

$$\begin{aligned} 2\mathcal{A}(Q) = {} & a_0 b_1 - a_1 b_0 + a_1 c_0 - a_0 c_1 + b_0 c_1 - c_0 b_1 \\ & + a_0 c_1 - a_1 c_0 + a_1 d_0 - a_0 d_1 + c_0 d_1 - d_0 c_1 . \end{aligned}$$

- Notice ac or db cancel

Area of a Convex Quadrilateral

- Generalizing, we get two terms per polygon edge
- None for internal diagonals.
- So if the coordinates of vertex v_i are x_i and y_i, twice the area of a convex polygon is given by

$$2A(P) = \sum_{i=0}^{n-1} (x_i y_{i+1} - y_i x_{i+1}).$$

Area of a Nonconvex Quadrilateral

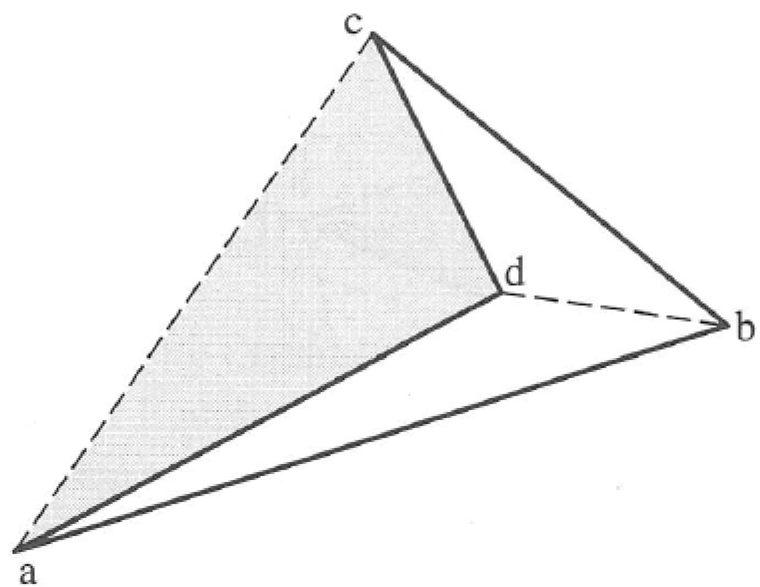


FIGURE 1.18 Triangulation of a nonconvex quadrilateral. The shaded area $\mathcal{A}(a, d, c)$ is negative.

- Can we use the same equation to calculate the area of a nonconvex quadrilateral?

Area of a Nonconvex Quadrilateral

- Suppose we have a nonconvex quadrilateral $Q = (a, b, c, d)$ (Figure 1.18)
- Only one triangulation, using the diagonal db
- But the algebraic expression obtained is independent of the diagonal chosen
- The equation is still true, even though the diagonal ac is external to Q

$$\mathcal{A}(Q) = \mathcal{A}(a, b, c) + \mathcal{A}(a, c, d)$$

Area of a Nonconvex Quadrilateral

- Notice $A(a, c, d)$ is negative and
- the area of a nonconvex quadrilateral is $\triangle abc$ minus $A(a, c, d)$.
- Indeed, $A(a, c, d)$ is a clockwise path, so the cross product formulation shows that the area will be negative.
- The phenomenon observe with a nonconvex quadrilateral is general

Area from an Arbitrary Center

- Let $T = \triangle abc$ be a triangle, with the vertices oriented counterclockwise,
- and let p be any external point in the plane.
- Then, we claim

$$\mathcal{A}(T) = \mathcal{A}(p, a, b) + \mathcal{A}(p, b, c) + \mathcal{A}(p, c, a).$$

Area from an Arbitrary Center

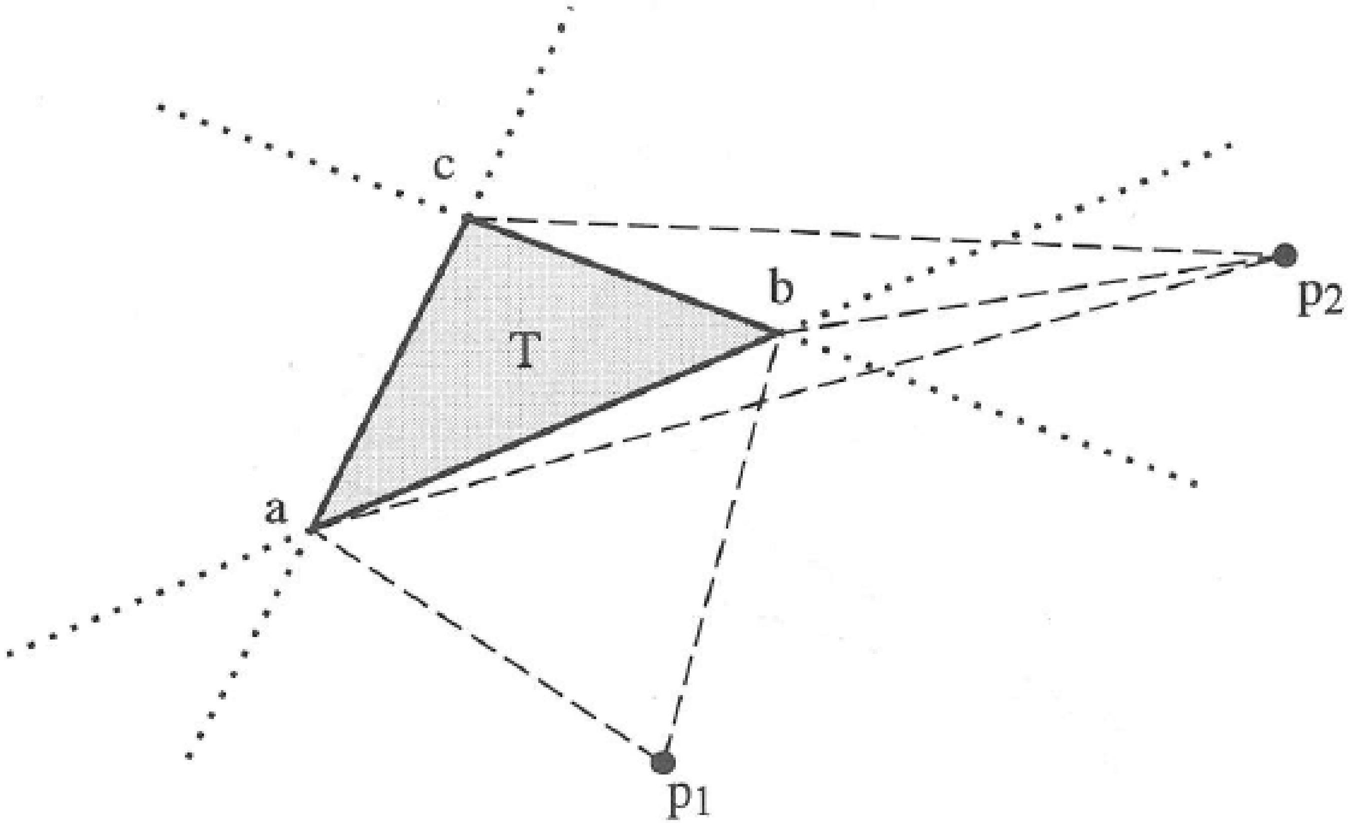


FIGURE 1.19 Area of T based on various external points p_1, p_2.

Area from an Arbitrary Center

- Case 1: $p = p_1$
  1. $A(p_1, a, b)$ is negative (clockwise)
  2. $A(p_1, b, c)$ is positive (counterclockwise)
  3. $A(p_1, c, a)$ is positive (counterclockwise)
- Case 2: $p = p_2$
  1. $A(p_2, a, b)$ is negative (clockwise)
  2. $A(p_2, b, c)$ is negative (clockwise)
  3. $A(p_2, c, a)$ is positive (counterclockwise)
- All other positions for external p in the plane are equivalent to either p_1 or p_2 by symmetry

Area from an Arbitrary Center

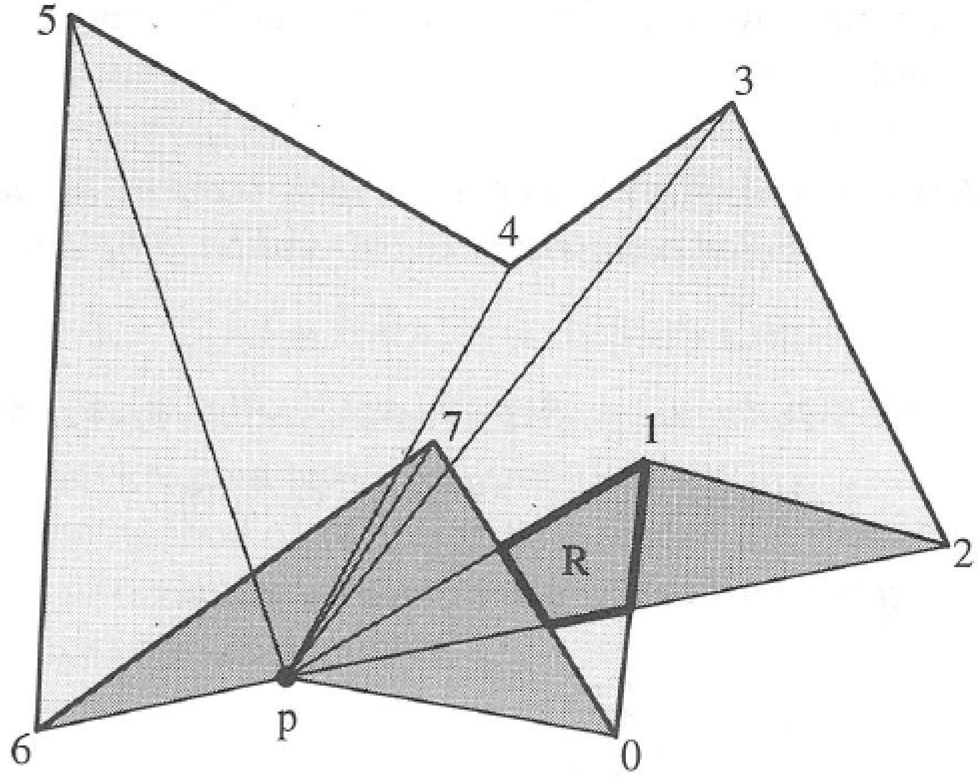


FIGURE 1.20 Computation of the area of a nonconvex polygon from point p. The darker triangles are oriented clockwise, and thus they have negative area.

Polygon Triangulation

Segment Intersection

Left turn

- The point of intersection
  1. Can be decided by using a Left predicate
- A directed line is determined by two points given (a, b)
- If c is to the left of the line, then the triple (a,b,c) forms counterclockwise circuit. (see figure 1.22)
- c is to the left of (a,b) iff $A(a,b,c)$ is positive
  - This can be implemented by a single call to Area2 (See code 1.6)
- Straightforward but subject to the special case objections raised earlier.

Left turn

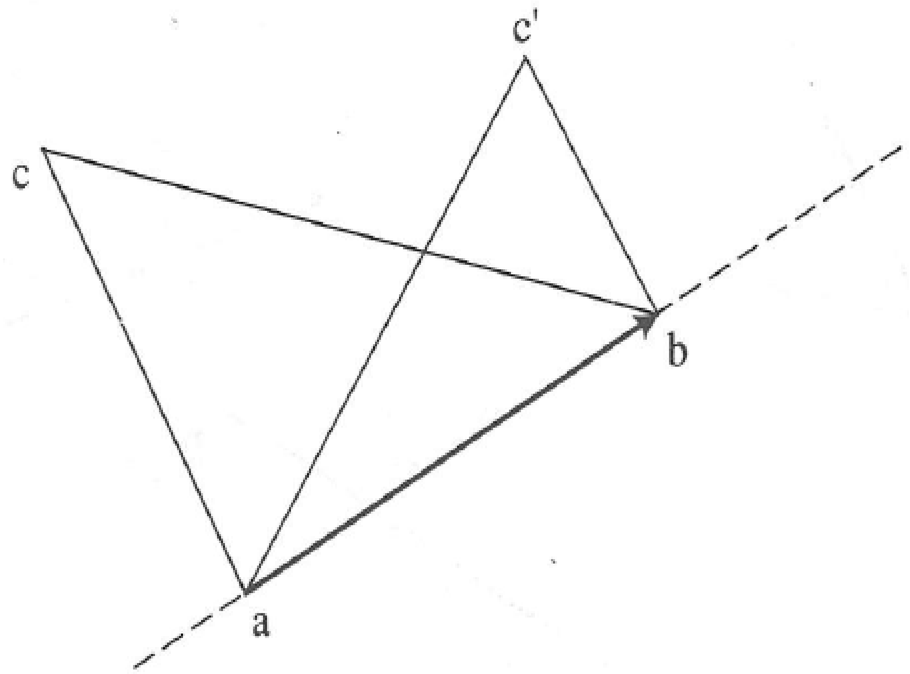


FIGURE 1.22 c is left of ab iff Δabc has positive area; $\Delta abc'$ also has positive area.

Left turn

```
bool  Left( tPointi a, tPointi b, tPointi c )
{
  return  Area2( a, b, c ) > 0;
}

bool  LeftOn( tPointi a, tPointi b, tPointi c )
{
  return  Area2( a, b, c ) >= 0;
}

bool  Collinear( tPointi a, tPointi b, tPointi c )
{
  return  Area2( a, b, c ) == 0;
}
```

Code 1.6 Left

- If c is collinear with ab, then the determined triangle has zero area.

Boolean Intersection

- If ab and cd intersect in their interiors,
  1. c and d are split by the line L_1 containing ab
  2. Likewise a and b are split by line L_2 containing cd
- Neither one of these conditions is alone sufficient to guarantee intersection
- When two segments intersect at a point interior to both, if it is known that no three of the four endpoints are collinear

Boolean Intersection

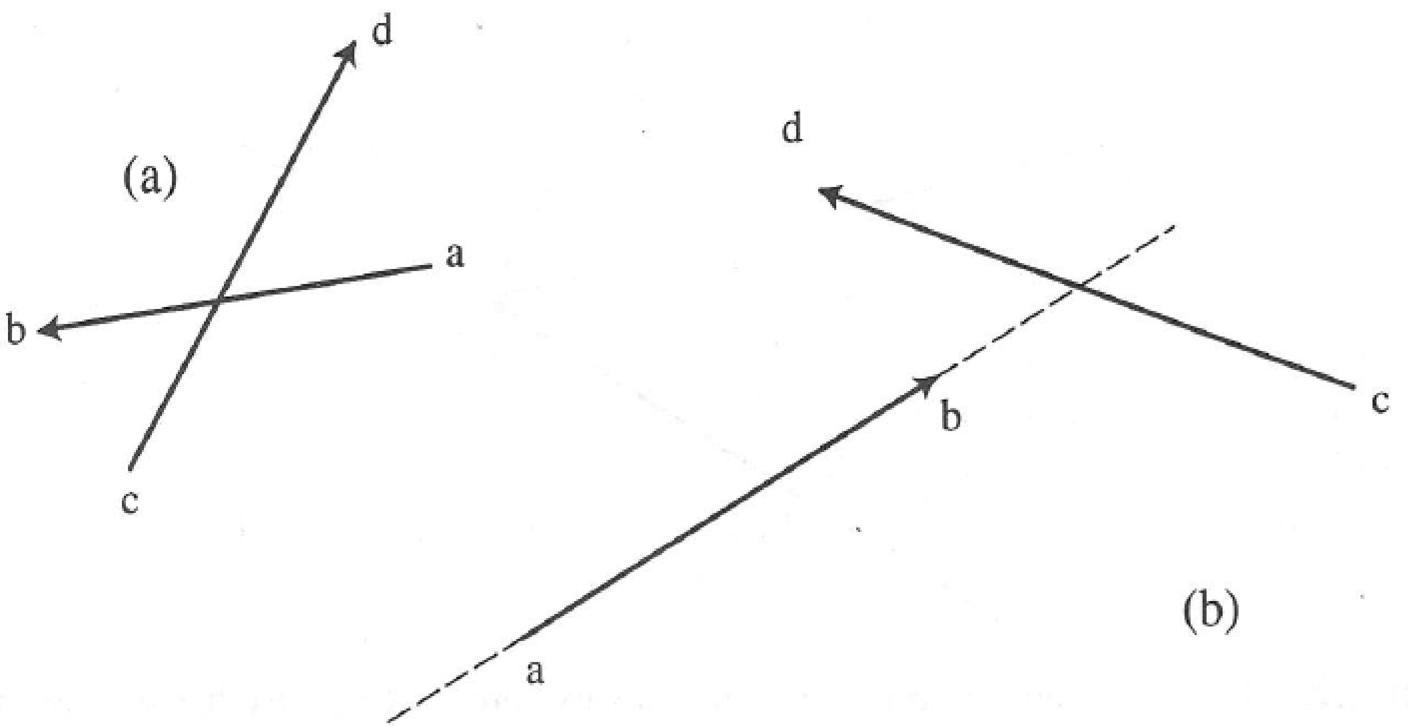


FIGURE 1.23 Two segments intersect (a) iff their endpoints are split by their determined lines; both pair of endpoints must be split (b).

```
bool  IntersectProp( tPointi a, tPointi b, tPointi c, tPointi d )
{
  /* Eliminate improper cases. */
  if (
    Collinear(a,b,c)  ||
    Collinear(a,b,d)  ||
    Collinear(c,d,a)  ||
    Collinear(c,d,b)
    )
    return FALSE;

  return
         Xor( Left(a,b,c), Left(a,b,d) )
    && Xor( Left(c,d,a), Left(c,d,b) );
}
/*Exclusive or: T iff exactly one argument is true. */
bool  Xor( bool x, bool y )
{
  /* The arguments are negated to ensure that they are 0/1 values. */
  return  !x ^ !y;
}
```

Code 1.7 IntersectProp

Boolean Intersection

- Redundancy in this code
  1. Four relevant triangle areas are being computed twice each.
- Two ways to remove redundancy
  1. Storing computed areas in local variables
  2. Designing other primitives that fit the problem better.
- The first if-statement may be removed entirely for the purposes of triangulation
- But sacrifice efficiency for clarity and leave IntersectProp as is

Boolean Intersection

- It might be tempting to implement the exclusive-or by

```
Area2(a, b, c) * Area2(a, b, d) < 0
&& Area2(c, d, a) * Area2(c, d, b) < 0;
```

- But the product of the areas might cause integer word overflow!

Improper Intersection

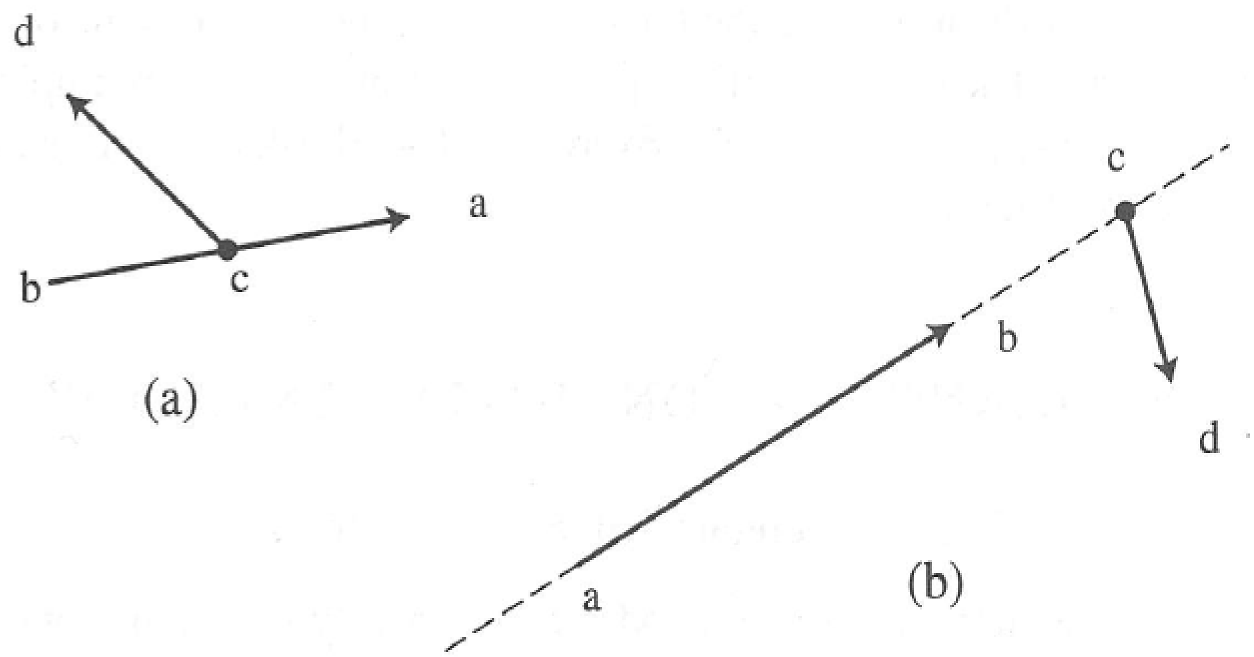


FIGURE 1.24 Improper intersection between two segments (a); collinearity is not sufficient (b).

Improper Intersection

- Special case of improper intersection
  1. An endpoint of one segment (say c) lies somewhere on the other segment ab (Figure 1.24(a))
  2. This only happens if a, b, c are collinear
  3. But collinearity is not a sufficient condition for intersection (Figure 1.24(b))
- Need to decide betweenness

Betweenness

- Only check betweenness of c when we know it lies on the line containing ab
- If ab is not vertical
  1. c lies on ab iff the x *coordinate* of c falls in the interval of the x coordinates of a and b
- If ab is vertical
  1. Similarly check on y coordinates

Betweenness

```
bool   Between( tPointi a, tPointi b, tPointi c )
{
   tPointi   ba, ca;

   if ( ! Collinear( a, b, c ) )
      return  FALSE;

   /* If ab not vertical, check betweenness on x; else on y. */
   if ( a[X] != b[X] )
      return ((a[X] <= c[X]) && (c[X] <= b[X]))  ||
             ((a[X] >= c[X]) && (c[X] >= b[X]));
   else
      return ((a[Y] <= c[Y]) && (c[Y] <= b[Y]))  ||
             ((a[Y] >= c[Y]) && (c[Y] >= b[Y]));
}
```

Code 1.8 Between

Segment Intersection Code

```
bool   Intersect( tPointi a, tPointi b, tPointi c, tPointi d )
{
   if       ( IntersectProp( a, b, c, d ) )
      return  TRUE;
   else if (   Between( a, b, c )
               || Between( a, b, d )
               || Between( c, d, a )
               || Between( c, d, b )
             )
      return  TRUE;
   else    return  FALSE;
}
```

Code 1.9 Intersect

Segment Intersection Code

```
bool   Diagonalie( tVertex a, tVertex b )
{
  tVertex c, c1;

  /* For each edge (c,c1) of P */
  c = vertices;
  do {
    c1 = c->next;
    /* Skip edges incident to a or b */
    if (      ( c != a ) && ( c1 != a )
         && ( c != b ) && ( c1 != b )
         && Intersect( a->v, b->v, c->v, c1->v )
       )
       return FALSE;
    c = c->next;
  } while ( c != vertices );
  return TRUE;
}
```

Code 1.10 Diagonalie

Polygon Triangulation

Internal or External

InCone

- Goal: distinguish the internal from the external diagonals
- One vector B (along the diagonal) lies strictly in the open cone counterclockwise between two other vectors A and C (along two consecutive edges)
- Need to consider convex and reflex angles

InCone

- Convex case (Figure 1.25 a)
  1. s is internal to P iff it is internal to the cone whose apex is a, and whose sides pass through a_- and a_+
  2. Easily determined by our Left function
  3. a_- must be left of ab and a_+ must be left of ba
- Reflex case (Figure 1.25 b)
  1. reverse of the convex case

InCone

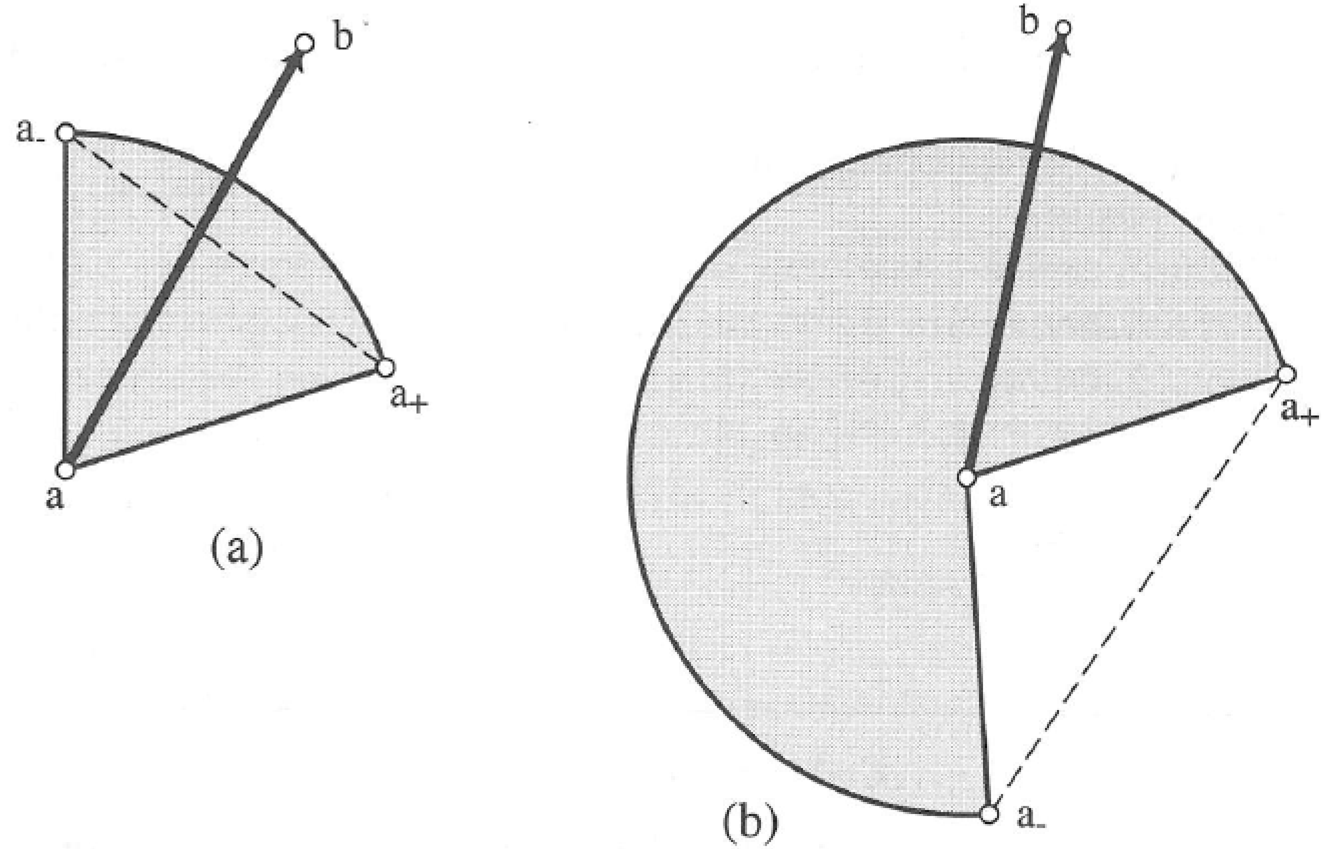


FIGURE 1.25 Diagonal $s = ab$ is in the cone determined by a_-, a, a_+: (a) convex; (b) reflex. In (b), both a_- and a_+ are right of ab.

InCone

- Distinguishing between the convex and reflex cases is accomplished with one invocation of Left
- a is convex *iff* a_- is left or on aa_+
- Note that if (a_-, a, a_+) are collinear, the internal angle at a is π, which we define as convex

InCone

```
bool   InCone( tVertex a, tVertex b )
{
   tVertex a0,a1;       /* a0,a,a1 are consecutive vertices. */

   a1 = a->next;
   a0 = a->prev;

   /* If a is a convex vertex ... */
   if( LeftOn( a->v, a1->v, a0->v ) )
       return     Left( a->v, b->v, a0->v )
              && Left( b->v, a->v, a1->v );

   /* Else a is reflex: */
       return !(    LeftOn( a->v, b->v, a1->v )
                 && LeftOn( b->v, a->v, a0->v ) );
}
```

Code 1.11 InCone

Diagonal

- *ab* is a diagonal *iff* Diagonalie(*a*, *b*), InCone(*a*, *b*), and InCone(*b*, *a*) are **ture**
- How to order function calls
  1. InCones should be first
  2. They are each constant-time calculation
  3. Each performs in the neighborhood *a* and *b* without regard to the remainder of the polygon, whereas Diagonalie includes a loop over all *n* polygon edges.

Diagonal

```
bool   Diagonal( tVertex a, tVertex b )
{
   return InCone( a, b ) && InCone( b, a ) && Diagonalie( a, b );
}
```

Code 1.12 Diagonal

Triangulation

Diagonal-Based Algorithm

- It is an $O(n^4)$ algorithm
  1. (n choose 2) diagonal candidates = $O(n^2)$
  2. Testing each for diagonalhood = $O(n)$
  3. Reapting this $O(n^3)$ computation for each of the n-3 diagonals = $O(n^4)$
- Use the two ears theorem to speed up
  - Only $O(n)$ “ear diagonal” candidates
  - We can achieve a worst-case complexity of $O(n^3)$ this way

Ear Removal

Ear Removal

- Improve the above algorithm to $O(n^2)$
  1. Because one call to Diagonal costs $O(n)$, Diagonal may only be called $O(n)$ times
- Key idea
  1. Removal of one ear does not change the polygon very much
  2. Not change whether or not many of its vertices are potential ear tips
- Determination for potential ear tip of each vertex already uses $O(n^2)$, but is not repeated

Ear Removal

Ear Removal

- Let $(v_0, v_1, v_2, v_3, v_4)$ be five consecutive vertices of P
- Suppose v_2 is an ear tip and the ear $E_2 = \triangle(v_1, v_2, v_3)$ is deleted (see Figure 1.26)
- Only v_1 and v_3 change
- Neighbor vertices remain unchanged

Ear Removal

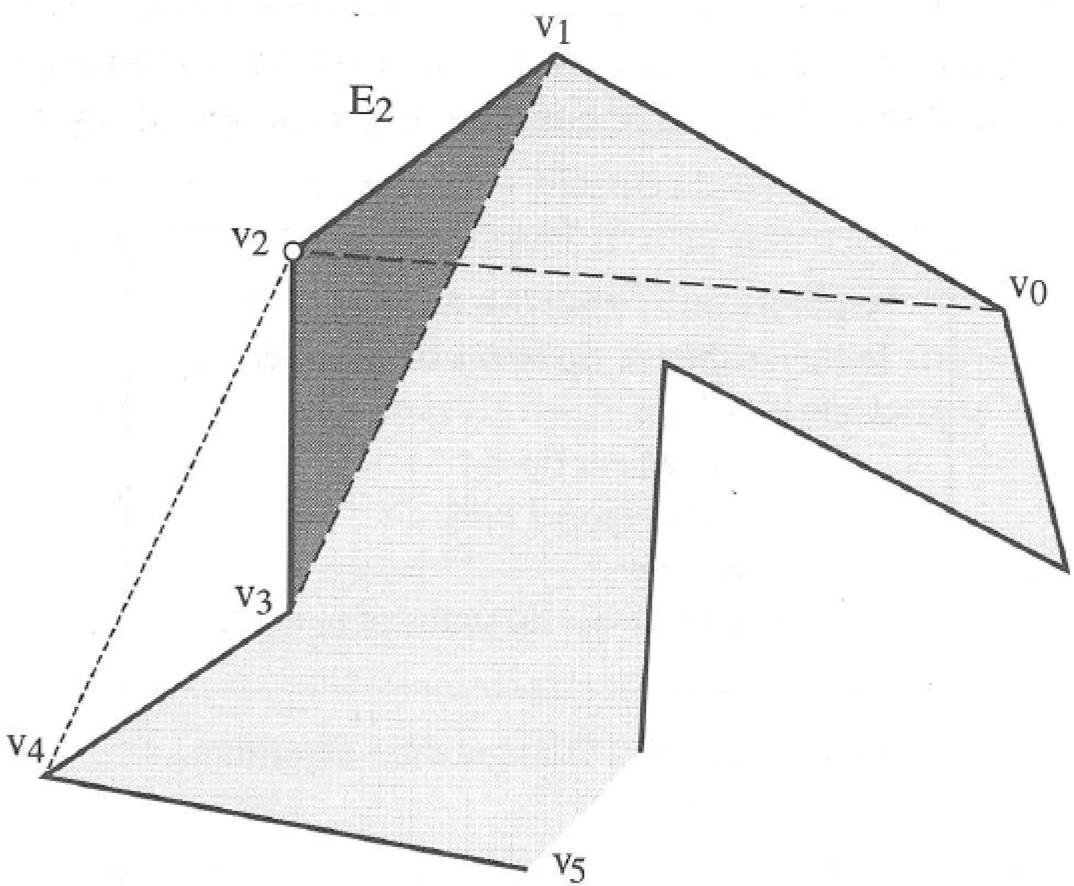


FIGURE 1.26 Clipping an ear $E_2 = \Delta(v_1, v_2, v_3)$. Here the ear status of v_1 changes from TRUE to FALSE.

Ear Removal

Ear Removal

- After the expensive initialization step, the ear tip status information can be updated with two calls to Diagonal per iteration

Algorithm: TRIANGULATION
Initialize the ear tip status of each vertex
while $n > 3$ do
    Locate an ear tip v_2.
    Output diagonal $v_1 v_3$.
    Delete v_2.
    Update the ear tip status of v_1 and v_3.

Example

- Figure 1.27 shows a polygon and the triangulation produced by the simple main program(Code 1.15)

```
main()
{
  ReadVertices();
  PrintVertices();
  Triangulate();
}
```

Code 1.15 main

Example

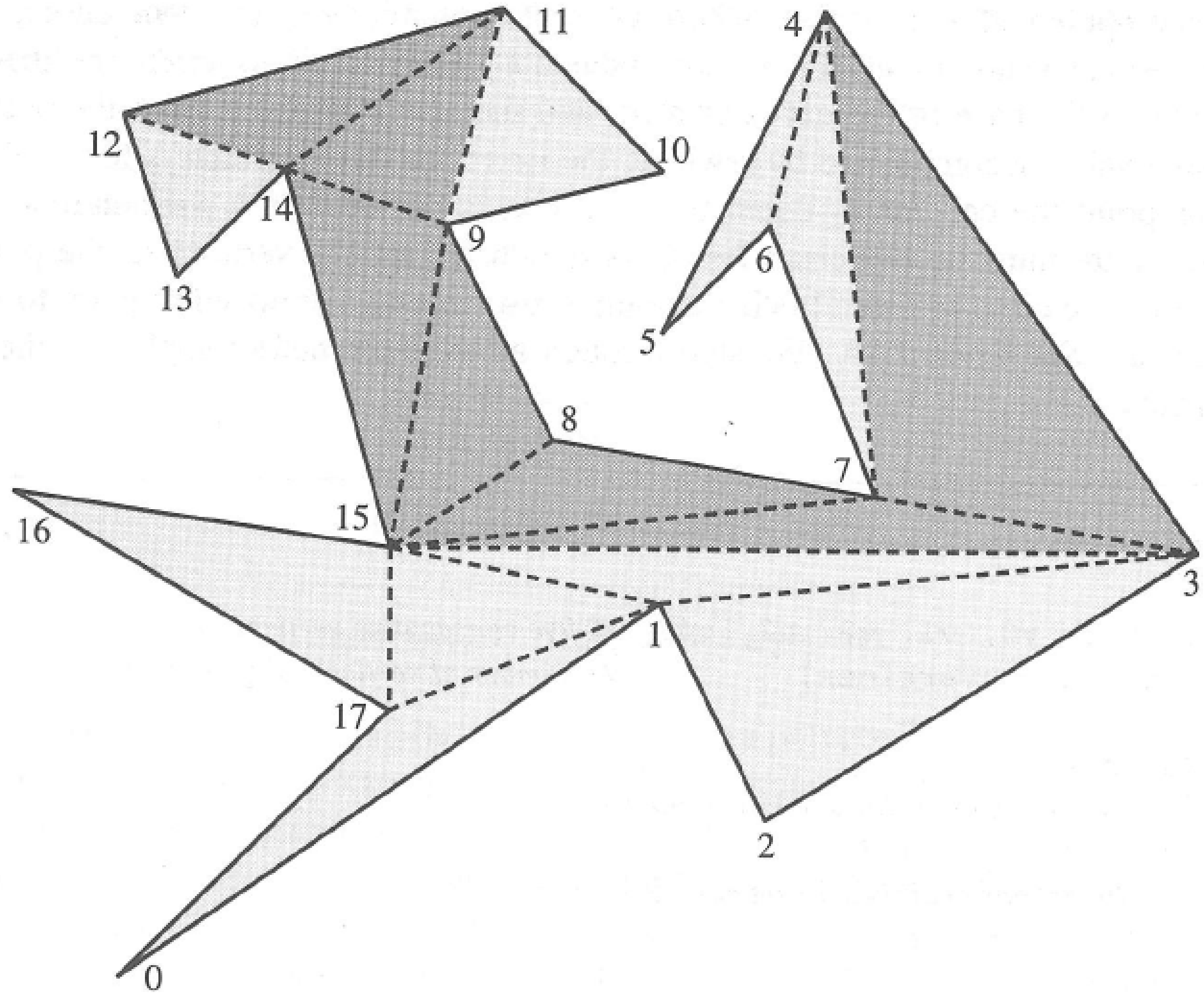


FIGURE 1.27 A polygon of 18 vertices and the triangulation produced by `Triangulate`. The dark subpolygon is the remainder after the 9th diagonal (15, 3) is output. Vertex coordinates are displayed in Table 1.1.

Example

- Now walk through the output of the diagonals
- v_0 is an ear tip, so the first diagonal output is (17,1)
- v_1 is not an ear tip, so v2 pointer moves to v_2
- v_2 is a tip, so print the diagonal (1,3)
- Neither v_3 nor v_4 is an ear tip
- At v_5, the next diagonal is (4,6)
- v_3 v_8 is collinear with v_7, so the next ear detected is not until v_{10}
- …
- Another collinearity, v_9 with (v_{11} v_{15}), prevents v_9 from being an ear
- …

Example

Table 1.2. The columns show the order in which the diagonals, specified as pairs of endpoint indices, are output

Order	Diagonal Indices	Order	Diagonal Indices
1	(17, 1)	10	(3, 7)
2	(1, 3)	11	(11, 14)
3	(4, 6)	12	(15, 7)
4	(4, 7)	13	(15, 8)
5	(9, 11)	14	(15, 9)
6	(12, 14)	15	(9, 14)
7	(15, 17)		
8	(15, 1)		
9	(15, 3)		

Conclusion

- We learned
  1. what is a polygon, diagonal, triangulation
  2. how to determine
     - the area of polygons
     - if 3 points, collinear, turn-left, in-between
     - if two segment intersect
     - if a segment is internal or external a polygon
     - ears in a polygon
- Homework assignment: 1.1.4-1, 1.3.9-4, 1.6.8-2, 1.6.8-3 (due 9/10 before the class)
- Programming assignment: Coming up next week