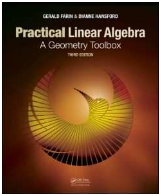
Practical Linear Algebra: A Geometry Toolbox

Third edition

Chapter 18: Putting Lines Together: Polylines and Polygons

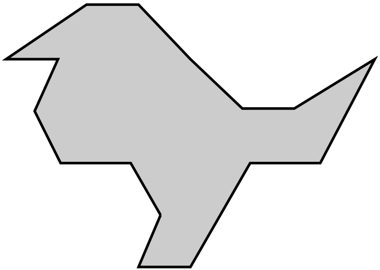
Gerald Farin & Dianne Hansford

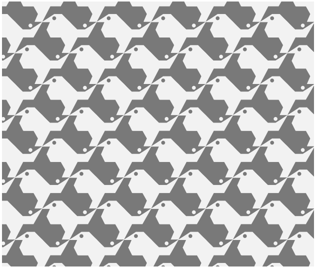
CRC Press, Taylor & Francis Group, An A K Peters Book
www.farinhansford.com/books/pla



Outline

Introduction to Putting Lines Together: Polylines and Polygons



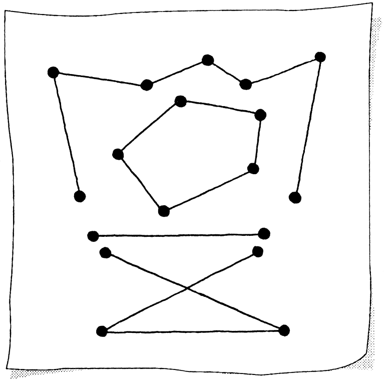


Left Figure shows a polygon
— just about every computer-generated drawing consists of polygons

Add an "eye" and apply a sequence of rotations and translations
⇒ Right Figure: copies of the bird polygon can cover the whole plane
Technique is present in many illustrations by *M. C. Escher*

Polylines

2D polyline examples



Polyline: edges connecting an ordered set of vertices

First and last vertices not necessarily connected

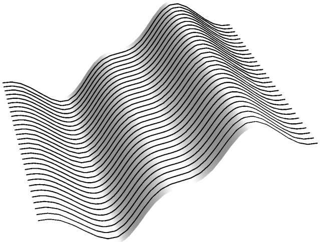
Vertices are ordered and edges are oriented
⇒ edge vectors

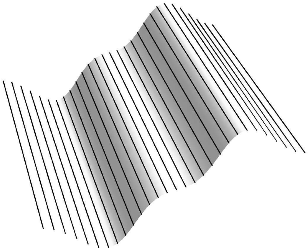
Polyline can be 3D

Polylines

Polylines are a primary *output primitive* in graphics standards
— Example: GKS (*Graphical Kernel System*)
— Postscript (printer language) based on GKS

Polylines used to outline a shape 2D or 3D – see Figure
— Surface evaluated in an organized fashion ⇒ polylines
— Gives feeling of the "flow" of the surface
Modeling application: polylines approximate a complex curve or data
⇒ analysis easier and less costly





Polygons

Polygon: polyline with first and last vertices connected

Here: Polygon encloses an area $\Rightarrow$ planar polygons only

Polygon with n edges is given by an ordered set of 2D points

$$\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$$

Edge vectors

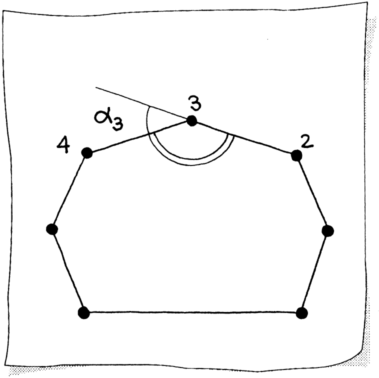
$$\mathbf{v}_i = \mathbf{p}_{i+1} - \mathbf{p}_i \quad i = 1, \ldots, n$$

where $\mathbf{p}_{n+1} = \mathbf{p}_1$ — cyclic numbering convention

— Edge vectors sum to the *zero vector*
— Number of vertices equals the number of edges

Polygons

Interior and exterior angles



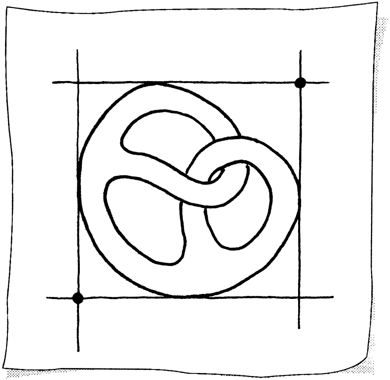
Polygon is closed $\Rightarrow$ divides the plane into two parts:
1) a finite part: *interior*
2) an infinite part: *exterior*

Traverse the boundary of a polygon:
Move along the edges and
at each vertex rotate angle α_i
— turning angle or exterior angle

Interior angle: $\pi - \alpha_i$

Polygons

A minmax box is a polygon

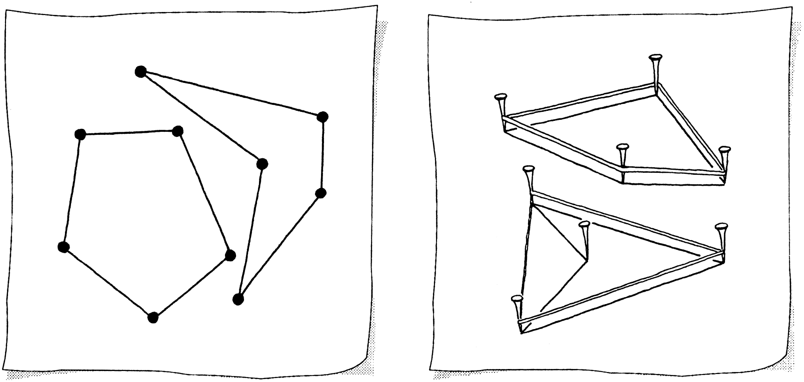


Polygons are used a lot!

Fundamental Examples:

- Extents of geometry: the *minmax box*
- Triangles forming a *polygonal mesh* of a 3D model

Convexity



Left Sketch: Polygon classification

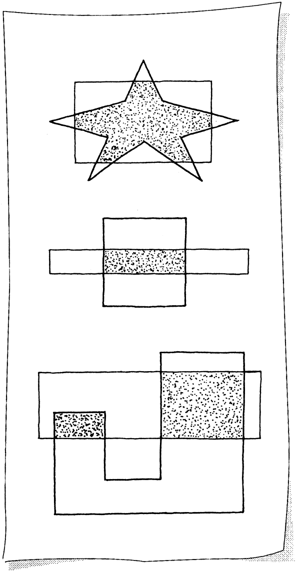
— Left polygon: convex   Right polygon: nonconvex

Convexity tests:

1) Rubberband test described in right Sketch

2) Line connecting any two points in/on polygon never leaves polygon

Convexity



Some algorithms simplified or specifically designed for convex polygons

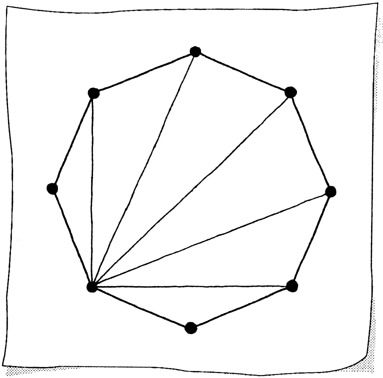
Example: polygon clipping

Given: two polygons
Find: intersection of polygon areas

If both polygons are convex results in one convex polygon

Nonconvex polygons need more record keeping

Convexity



n-sided convex polygon

Sum of interior angles:

$$I = (n - 2)\pi$$

Triangulate $\Rightarrow$ $n - 2$ triangles
Triangle: sum of interior angles is π

Sum of the exterior angles:

$$E = n\pi - (n - 2)\pi = 2\pi$$

Each interior and exterior angle sums to π

Convexity

More convexity tests for an *n*-sided polygon

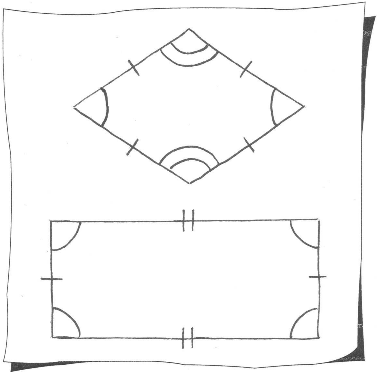
The *barycenter* of the vertices

$$\mathbf{b} = \frac{1}{n}(\mathbf{p}_1 + \ldots + \mathbf{p}_n) \qquad \text{center of gravity}$$

Construct the implicit line equation for each edge vector
— Needs to be done in a consistent manner
Polygon convex if **b** on "same" side of every line
— Implicit equation evaluations result in all positive or all negative values

Another test for convexity:
— Check if there is a re-entrant angle: an interior angle $> \pi$

Types of Polygons



Variety of special polygons

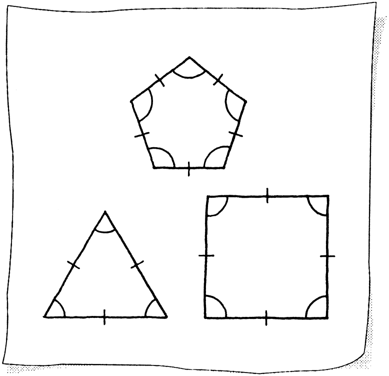
- equilateral: all sides equal length
- equiangular: all interior angles at vertices equal
- regular: equilateral and equiangular

Rhombus:
equilateral but not equiangular

Rectangle:
equiangular but not equilateral

Square: equilateral and equiangular

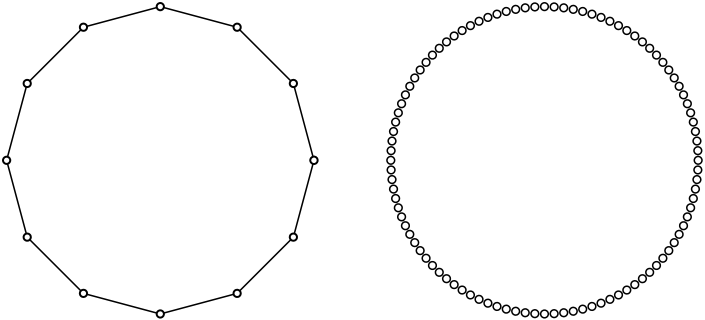
Types of Polygons



Regular polygon also referred to as an n-gon

- a 3-gon is an equilateral triangle
- a 4-gon is a square
- a 5-gon is a pentagon
- a 6-gon is a hexagon
- an 8-gon is an octagon

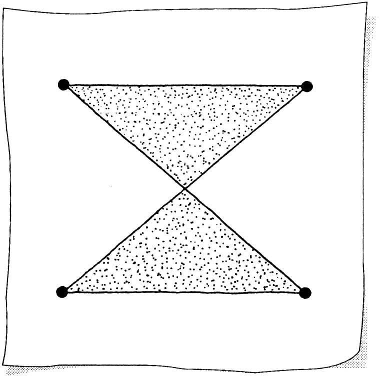
Types of Polygons



Circle approximation: using an n-gon to represent a circle

Unusual Polygons

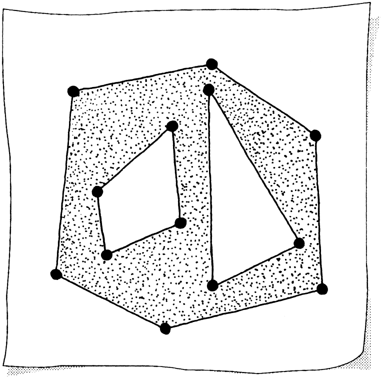
Nonsimple polygon: edges intersecting other than at the vertices
Can cause algorithms to fail



Traverse along the boundary
At the mid-edge intersections
polygon's interior switches sides

Nonsimple polygons can arise due to an error
Example: polygon clipping algorithm involves sorting vertices to form polygon
— If sorting goes haywire result could be a nonsimple polygon

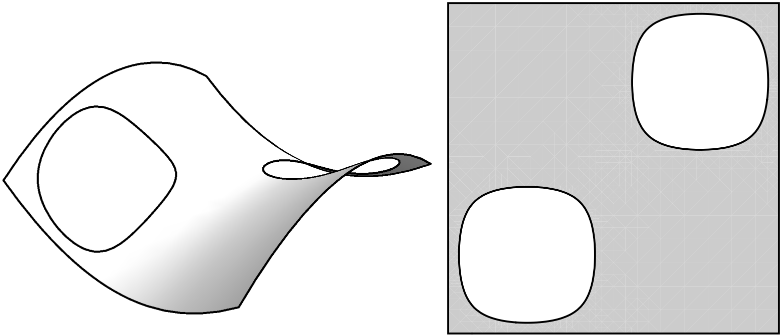
Unusual Polygons



Polygons with holes defined by
1) boundary polygon
2) interior polygons

Convention: The *visible region* or the region that is not cut out is to the "left"
⇒ Outer boundary oriented counterclockwise
⇒ Inner boundaries are oriented clockwise

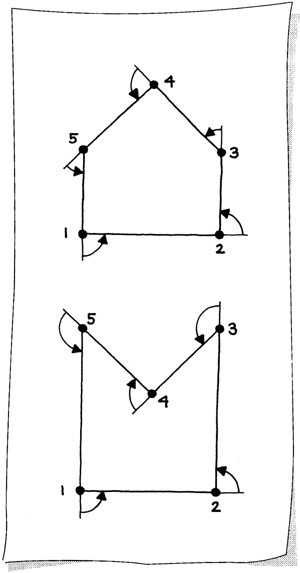
Unusual Polygons



Trimmed surface: an application of polygons with holes
Left: trimmed surface
Right: rectangular parametric domain with polygonal holes

A Computer-Aided Design and Manufacturing (CAD/CAM) application
Polygons define parts of the material to be cut or punched out
This allows other parts to fit to this one

Turning Angles and Winding Numbers



Turning angle: rotation at vertices as boundary traversed

For convex polygons:
— All turning angles have the same orientation
— Same as exterior angle

For nonconvex polygons:
— Turning angles do not have same orientation

Turning angle application
Given: 2D polygon living in the $[\mathbf{e}_1, \mathbf{e}_2]$-plane with vertices

$$\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_n$$

Find: Is the polygon is convex?
Solution: Embed the 2D vectors in 3D

$$\mathbf{p}_i = \begin{bmatrix} p_{1,i} \\ p_{2,i} \\ 0 \end{bmatrix} \quad \text{then} \quad \mathbf{u}_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \wedge (\mathbf{p}_{i+2} - \mathbf{p}_{i+1}) = \begin{bmatrix} 0 \\ 0 \\ u_{3,i} \end{bmatrix}$$

Or use the *scalar triple product*:

$$u_{3,i} = \mathbf{e}_3 \cdot ((\mathbf{p}_{i+1} - \mathbf{p}_i) \wedge (\mathbf{p}_{i+2} - \mathbf{p}_{i+1}))$$

Polygon is convex if the sign of $u_{3,i}$ is the *same* for all angles
$\Rightarrow$ Consistent orientation of the turning angles

Turning Angles and Winding Numbers

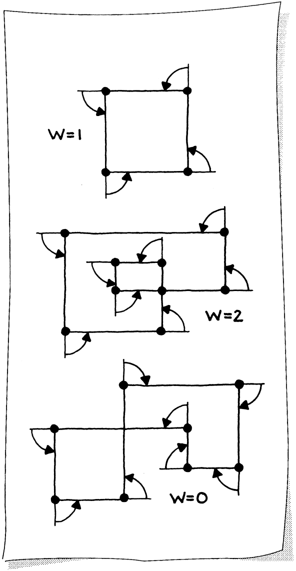
Consistent orientation of the turning angles can be determined from the *determinant* of the 2D vectors as well
3D approach needed if vertices lie in an *arbitrary* plane with normal $\mathbf{n}$

3D polygon convexity test:

$$\mathbf{u}_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \wedge (\mathbf{p}_{i+2} - \mathbf{p}_{i+1}) \qquad \text{(has direction } \pm\mathbf{n})$$

Extract a signed scalar value $\mathbf{n} \cdot \mathbf{u}_i$
Polygon is convex if all scalar values are the same sign

Turning Angles and Winding Numbers



Total turning angle:
sum of all turning angles

Convex polygon: 2π

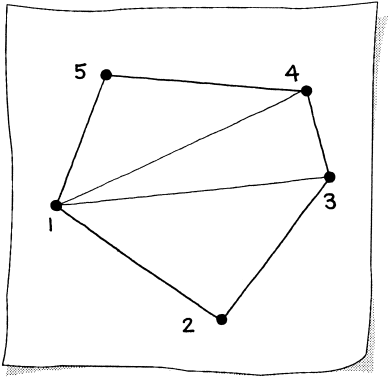
E = Sum of *signed* turning angles

Winding number of the polygon

$$W = \frac{E}{2\pi}$$

— Convex polygon: $W = 1$
— Decremented for each clockwise loop
— Incremented for each counterclockwise loop

Area



Signed area of a 2D polygon

Polygon defined by vertices $\mathbf{p}_i$

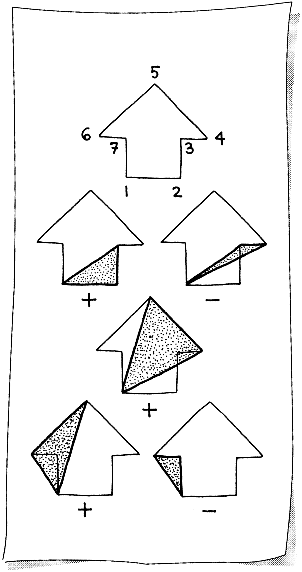
— *Triangulate* the polygon (consistent orientation)

— Sum of the signed triangle areas

Form $\mathbf{v}_i = \mathbf{p}_i - \mathbf{p}_1$

$$A = \frac{1}{2}(\det[\mathbf{v}_2, \mathbf{v}_3] + \det[\mathbf{v}_3, \mathbf{v}_4] + \det[\mathbf{v}_4, \mathbf{v}_5])$$

Area



Signed area of a nonconvex 2D polygon

Use of signed area
makes sum of triangle areas method work for non-convex polygons

Negative areas cancel duplicate and extraneous areas

Area

Determinants representing edges of triangles within the polygon cancel

$$A = \frac{1}{2}(\det[\mathbf{p}_1, \mathbf{p}_2] + \ldots + \det[\mathbf{p}_{n-1}, \mathbf{p}_n] + \det[\mathbf{p}_n, \mathbf{p}_1])$$

Geometric meaning? Yes: consider each point to be $\mathbf{p}_i - \mathbf{o}$

Which equation is better?

— Amount of computation for each is similar

— Drawback of point-based: If polygon is far from the origin
then numerical problems can occur
vectors $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ will be close to parallel

— Advantage of vector-based: intermediate computations meaningful

$\Rightarrow$ Reducing an equation to its "simplest" form not always "optimal"!

Area

Generalized determinant:

$$A = \frac{1}{2} \begin{vmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} & p_{1,1} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} & p_{2,1} \end{vmatrix}$$

Compute by adding products of "downward" diagonals and subtracting products of "upward" diagonals

Example: $\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ (square)

$$A = \frac{1}{2} \begin{vmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{vmatrix} = \frac{1}{2}[0 + 1 + 1 + 0 - 0 - 0 - 0 - 0] = 1$$

Example: $\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{p}_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ (nonsimple)

$$A = \frac{1}{2} \begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{vmatrix} = \frac{1}{2}[0 + 1 + 0 + 0 - 0 - 0 - 1 - 0] = 0$$

Area of a planar polygon specified by 3D points $\mathbf{p}_i$

Recall: cross product $\Rightarrow$ parallelogram area

$$\text{Let } \mathbf{v}_i = \mathbf{p}_i - \mathbf{p}_1 \quad \text{and} \quad \mathbf{u}_i = \mathbf{v}_i \wedge \mathbf{v}_{i+1} \quad \text{for} \quad i = 2, n-1$$

Unit normal to the polygon is $\mathbf{n}$ — shares same direction as $\mathbf{u}_i$

$$A = \frac{1}{2}\mathbf{n} \cdot (\mathbf{u}_2 + \ldots + \mathbf{u}_{n-1}) \qquad \text{(sum of scalar triple products)}$$

Example: Four coplanar 3D points

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix}$$

$$\mathbf{n} = \begin{bmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix} \quad \mathbf{v}_2 = \begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix} \quad \mathbf{v}_3 = \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix} \quad \mathbf{v}_4 = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

$$\mathbf{u}_2 = \mathbf{v}_2 \wedge \mathbf{v}_3 = \begin{bmatrix} -6 \\ -6 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{u}_3 = \mathbf{v}_3 \wedge \mathbf{v}_4 = \begin{bmatrix} -6 \\ -6 \\ 0 \end{bmatrix}$$

$$A = \frac{1}{2}\mathbf{n} \cdot (\mathbf{u}_2 + \mathbf{u}_3) = 6\sqrt{2}$$

Area

Normal estimation method:

Good average normal to a non-planar polygon:

$$\mathbf{n} = \frac{(\mathbf{u}_2 + \mathbf{u}_3 + \ldots + \mathbf{u}_{n-2})}{\|\mathbf{u}_2 + \mathbf{u}_3 + \ldots + \mathbf{u}_{n-2}\|}$$

This method is a weighted average based on the areas of the triangles — To eliminate this weighting normalize each $\mathbf{u}_i$ before summing

Example: Estimate a normal to the non-planar polygon

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{p}_4 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{u}_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \quad \mathbf{u}_3 = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{n} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Application: Planarity Test

CAD file exchange scenario:
— Given a polygon oriented arbitrarily in 3D
— For your application the polygon must be 2D
— How do you verify that the data points are *coplanar*?

Many ways to solve this problem

Considerations in comparing algorithms:

- numerical stability
- speed
- ability to define a meaningful tolerance
- size of data set
- maintainability of the algorithm

Order of importance?

Application: Planarity Test

Three methods to solve this planarity test

- **Volume test:**
  — Choose first polygon vertex as a base point
  — Form vectors to next three vertices
  — Calculate volume spanned by three vectors
  — If less than tolerance then four points are coplanar
  — Continue for all other sets
- **Plane test:**
  — Construct plane through first three vertices
  — If all vertices lie in this plane (within tolerance) then points coplanar
- **Average normal test:**
  — Find centroid $\mathbf{c}$ of all points
  — Compute all normals $\mathbf{n}_i = [\mathbf{p}_i - \mathbf{c}] \wedge [\mathbf{p}_{i+1} - \mathbf{c}]$
  — If all angles formed by two subsequent normals less than tolerance then points coplanar

Tolerance types: ◇ volume ◇ distance ◇ angle

Application: Inside or Outside?

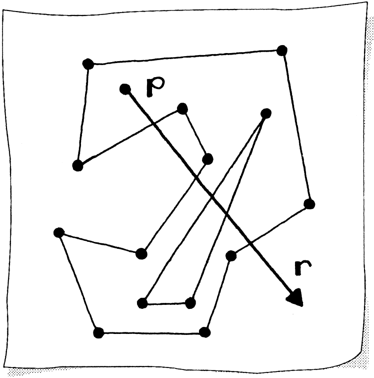
Inside/outside test or Visibility test
Given: a polygon in the $[\mathbf{e}_1, \mathbf{e}_2]$-plane and a point $\mathbf{p}$
Determine if $\mathbf{p}$ lies inside the polygon

This problem important for
— Polygon fill — CAD *trimmed surfaces*
Polygon can have one or more holes

Important element of visibility algorithms: trivial reject test
— If a point is "obviously" not in the polygon
then output result immediately with minimal calculation

Here: trivial reject based on *minmax box* around the polygon
$\Rightarrow$ Simple comparison of $\mathbf{e}_1$- and $\mathbf{e}_2$-coordinates

Even-Odd Rule



From point **p** construct a parametric line in any direction **r**

$$\mathbf{l}(t) = \mathbf{p} + t\mathbf{r}$$

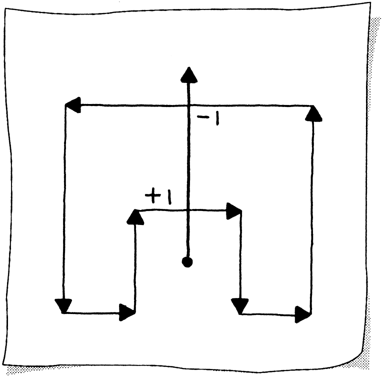
Count the number of intersections with the polygon edges for $t \geq 0$

Number of intersections
— Odd if **p** is inside
— Even if **p** is outside

Best to avoid $\mathbf{l}(t)$ passing through vertex or coincident with edge

Application: Inside or Outside?

Nonzero Winding Number Rule



Point **p** and any direction **r**

$$\mathbf{l}(t) = \mathbf{p} + t\mathbf{r}$$

Count intersections for $t \geq 0$

Counting method depends on the orientation of the polygon edges

— Start with a winding number $W = 0$

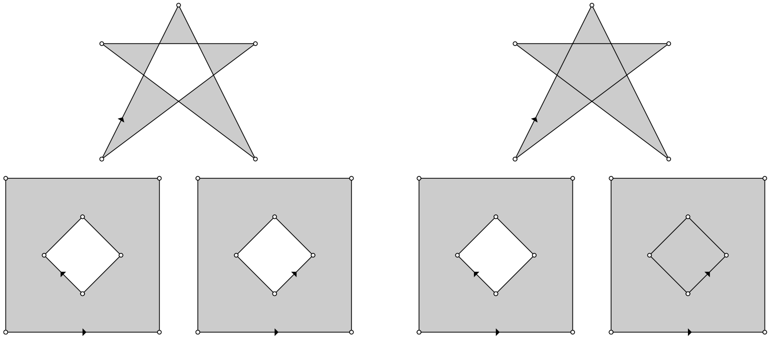
— "right to left" polygon edge $W = W + 1$

— "left to right" polygon edge $W = W - 1$

If final $W = 0$ then point outside

Application: Inside or Outside?

Visibility test applied to polygon fill



Even-Odd Rule   Nonzero Winding Rule

Three examples to highlight differences in the algorithms

Convex polygons: allow for a simple visibility test
— Inside if **p** is on the same side of all oriented edges

WYSK

- polygon
- polyline
- cyclic numbering
- turning angle
- exterior angle
- interior angle
- polygonal mesh
- convex
- concave
- polygon clipping
- sum of interior angles
- sum of exterior angles
- re-entrant angle
- equilateral polygon
- equiangular polygon
- regular polygon
- n-gon
- rhombus
- simple polygon
- trimmed surface
- visible region
- total turning angle
- winding number
- polygon area
- planarity test
- trivial reject
- inside/outside test
- scalar triple product