1. Introduction to Putting Lines Together: Polylines and Polygons
2. Polylines
3. Polygons
4. Convexity
5. Types of Polygons
6. Unusual Polygons
7. Turning Angles and Winding Numbers
8. Area
9. Application: Planarity Test
10. Application: Inside or Outside?
11. WYSK
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.
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

2D polyline examples
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

$$p_1, p_2, \ldots, p_n$$

Edge vectors

$$v_i = p_{i+1} - p_i \quad i = 1, \ldots, n$$

where $p_{n+1} = p_1$ — 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 $\alpha_i$ — turning angle or exterior angle

Interior angle: $\pi - \alpha_i$
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 \( \pi \)

Sum of the exterior angles:

\[ E = n\pi - (n - 2)\pi = 2\pi \]

Each interior and exterior angle sums to \( \pi \)
More convexity tests for an \( n \)-sided polygon

The *barycenter* of the vertices

\[
b = \frac{1}{n}(p_1 + \ldots + p_n) \quad \text{center of gravity}
\]

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
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
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 Angles and Winding Numbers

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

\[
\mathbf{p}_i = \begin{bmatrix} p_{1,i} \\ p_{2,i} \\ 0 \end{bmatrix}
\]

then

\[
\mathbf{u}_i = (\mathbf{p}_{i+1} - \mathbf{p}_i) \wedge (\mathbf{p}_{i+2} - \mathbf{p}_{i+1}) = \begin{bmatrix} 0 \\ 0 \\ u_{3,i} \end{bmatrix}
\]

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
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:

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

Extract a signed scalar value $\mathbf{n} \cdot \mathbf{u}_i$.
Polygons are convex if all scalar values are of the same sign.
Turning Angles and Winding Numbers

Total turning angle:
sum of all turning angles

Convex polygon: $2\pi$

$E = \text{Sum of signed turning angles}$

Winding number of the polygon

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

— Convex polygon: $W = 1$
— Decremented for each clockwise loop
— Incremented for each counterclockwise loop
Signed area of a 2D polygon

Polygon defined by vertices $p_i$:
- **Triangulate** the polygon (consistent orientation)
- Sum of the signed triangle areas

Form $v_i = p_i - p_1$

$$A = \frac{1}{2} (\det [v_2, v_3] + \det [v_3, v_4] + \det [v_4, v_5])$$
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
Determinants representing edges of triangles within the polygon cancel

\[ A = \frac{1}{2} (\det[p_1, p_2] + \ldots + \det[p_{n-1}, p_n] + \det[p_n, p_1]) \]

Geometric meaning? Yes: consider each point to be \( p_i - o \)

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 \( p_i \) and \( p_{i+1} \) will be close to parallel
— Advantage of vector-based: intermediate computations meaningful
⇒ 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: \( p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \), \( p_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \), \( p_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \), \( p_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) (square)

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

Example: \( p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \), \( p_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \), \( p_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \), \( p_4 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) (nonsimple)

\[
A = \frac{1}{2} \begin{vmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{vmatrix} = \frac{1}{2} [0 + 1 + 0 + 0 - 0 - 0 - 0 - 1 - 0] = 0
\]
Area of a planar polygon specified by 3D points $p_i$

Recall: cross product $\Rightarrow$ parallelogram area

Let $v_i = p_i - p_1$ and $u_i = v_i \wedge v_{i+1}$ for $i = 2, n-1$

Unit normal to the polygon is $n$ — shares same direction as $u_i$

$$A = \frac{1}{2} n \cdot (u_2 + \ldots + u_{n-1})$$ (sum of scalar triple products)
Example: Four coplanar 3D points

\[ p_1 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \quad p_2 = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \quad p_3 = \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} \quad p_4 = \begin{bmatrix} 0 \\ 2 \\ 3 \end{bmatrix} \]

\[ n = \begin{bmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \\ 0 \end{bmatrix} \quad v_2 = \begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix} \quad v_3 = \begin{bmatrix} 2 \\ -2 \\ 3 \end{bmatrix} \quad v_4 = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \]

\[ u_2 = v_2 \wedge v_3 = \begin{bmatrix} -6 \\ -6 \\ 0 \end{bmatrix} \quad \text{and} \quad u_3 = v_3 \wedge v_4 = \begin{bmatrix} -6 \\ -6 \\ 0 \end{bmatrix} \]

\[ A = \frac{1}{2} n \cdot (u_2 + u_3) = 6\sqrt{2} \]
Normal estimation method:

Good average normal to a non-planar polygon:

\[ n = \frac{(u_2 + u_3 + \ldots + u_{n-2})}{\|u_2 + u_3 + \ldots + u_{n-2}\|} \]

This method is a weighted average based on the areas of the triangles — To eliminate this weighting normalize each \( u_i \) before summing

**Example:** Estimate a normal to the non-planar polygon

\[ p_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad p_2 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad p_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad p_4 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \]

\[ u_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \quad u_3 = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \]

\[ n = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \]
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 $\mathbf{n}_i = [\mathbf{p}_i - \mathbf{c}] \land [\mathbf{p}_{i+1} - \mathbf{c}]$
  - If all angles formed by two subsequent normals less than tolerance then points coplanar

Tolerance types: $\Diamond$ volume $\Diamond$ distance $\Diamond$ angle
Inside/outside test or Visibility test
Given: a polygon in the \([e_1, e_2]-\)plane and a point \(p\)
Determine if \(p\) lies inside the polygon

This problem important for
— Polygon fill  — CAD \textit{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 \textit{minmax box} around the polygon
\(\Rightarrow\) Simple comparison of \(e_1\)- and \(e_2\)-coordinates
Application: Inside or Outside?

Even-Odd Rule

From point \( p \) construct a parametric line in any direction \( r \)

\[ l(t) = p + tr \]

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 \( l(t) \) passing through vertex or coincident with edge
Nonzero Winding Number Rule

Point \( p \) and any direction \( r \)

\[
l(t) = p + tr
\]

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

Convex polygons: allow for a simple visibility test

— Inside if \( p \) is on the same side of all oriented edges

Even-Odd Rule

Nonzero Winding Rule

Three examples to highlight differences in the algorithms
- 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