Polygon Filling

Polygon Fill Areas

- Polygon Classifications
- Identifying Concave Polygons
- Splitting Concave Polygons
- Inside-Outside Tests
- Plane Equations
- Front and Back Polygon Faces

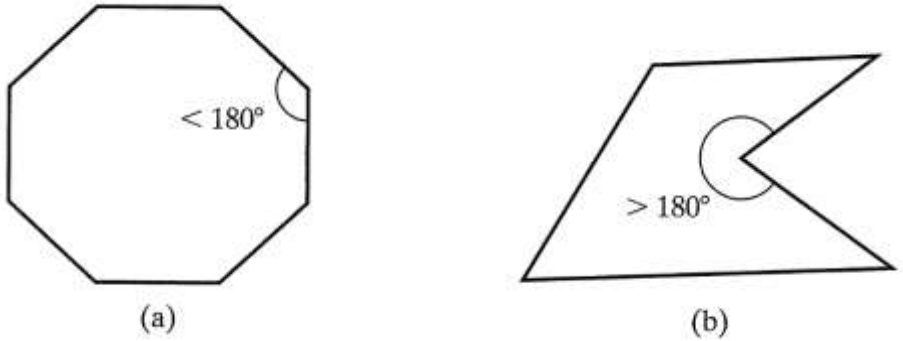
Definition

Mathematically defined, a **polygon** is a plane figure specified by a set of three or more coordinate positions, called *vertices*, that are connected in sequence by straight-line segments, called the *edges* or *sides* of the polygon. Further, in basic geometry, it is required that the polygon edges have no common point other than their endpoints. Thus, by definition, a polygon must have all its vertices within a single plane and there can be no edge crossings. Examples of polygons include triangles, rectangles, octagons, and decagons. Sometimes, any plane figure with a closed-polyline boundary is alluded to as a polygon, and one with no crossing edges is referred to as a *standard polygon* or a *simple polygon*. In an effort to avoid ambiguous object references, we will use the term "polygon" to refer only to those planar shapes that have a closed-polyline boundary and no edge crossings.

Polygon Classifications

An **interior angle** of a polygon is an angle inside the polygon boundary that is formed by two adjacent edges. If all interior angles of a polygon are less than or equal to 180°, the polygon **is convex.** An equivalent definition of a convex polygon is that its interior lies completely on one side of the infinite extension line of any one of its edges. Also, if we select any two points in the interior of a convex polygon, the line segment joining the two points is also in the interior. A polygon that is not convex is called a **concave** polygon. Figure 3-42 gives examples of convex and concave polygons.

FIGURE 3-42 A convex polygon (a), and a concave polygon (b).



Identifying Concave polygons

Interior angel test & boundary intersection

A concave polygon has at least one interior angle greater than 180°. Also, the extension of some edges of a concave polygon will intersect other edges, and some pair of interior points will produce a line segment that intersects the polygon boundary. Therefore, we can use any one of these characteristics of a concave polygon as a basis for constructing an identification algorithm.

Edge extension rule

Another way to identify a concave polygon is to take a look at the polygon vertex positions relative to the extension line of any edge. If some vertices are on one side of the extension line and some vertices are on the other side, the polygon is concave.

Vector Cross products

If we set up a vector for each polygon edge, then we can use the cross product of adjacent edges to test for concavity. All such vector products will be of the same sign (positive or negative) for a convex polygon. Therefore, if some cross products yield a positive value and some a negative value, we have a concave polygon. Figure 3-43 illustrates the edge-vector, cross-product method for identifying concave polygons.

Mathematical Review

$$\begin{aligned}\mathbf{V} &= \mathbf{P}_2 - \mathbf{P}_1 \\ &= (x_2 - x_1,\ y_2 - y_1) \\ &= (V_x,\ V_y)\end{aligned} \qquad (A\text{-}8)$$

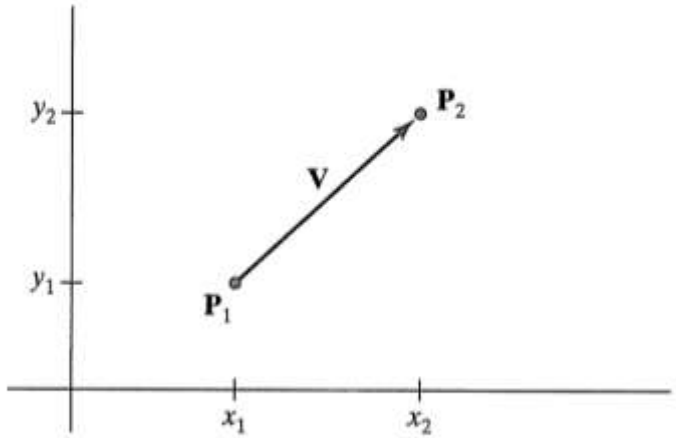


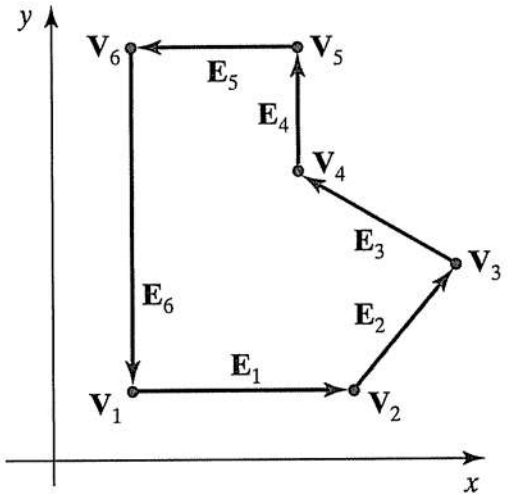
FIGURE A–13 A two-dimensional vector **V** defined in a Cartesian reference frame as the difference of two point positions.

cross product of two vectors is a vector that is perpendicular to the plane of the two vectors, and the magnitude of the cross-product vector is equal to the area of the parallelogram formed by the two vectors.

Vectors Cross product

We can also express the cross product in terms of vector components in a specific reference frame. In a Cartesian-coordinate system, we calculate the components of the cross product as

$$\mathbf{V}_1 \times \mathbf{V}_2 = (V_{1y}V_{2z} - V_{1z}V_{2y},\ V_{1z}V_{2x} - V_{1x}V_{2z},\ V_{1x}V_{2y} - V_{1y}V_{2x}) \qquad (A\text{-}21)$$

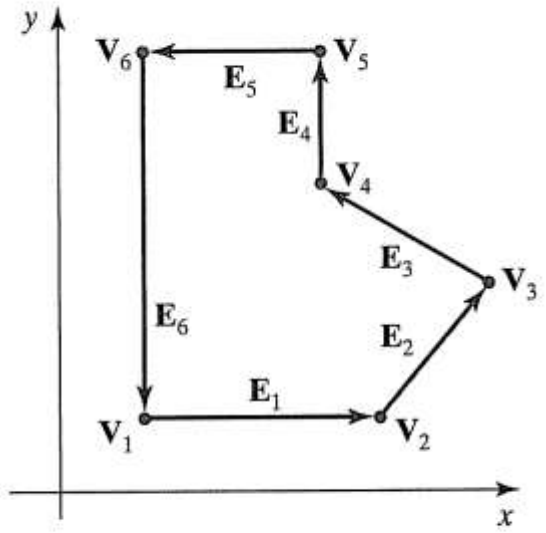


$$(\mathbf{E}_1 \times \mathbf{E}_2)_z > 0$$
$$(\mathbf{E}_2 \times \mathbf{E}_3)_z > 0$$
$$(\mathbf{E}_3 \times \mathbf{E}_4)_z < 0$$
$$(\mathbf{E}_4 \times \mathbf{E}_5)_z > 0$$
$$(\mathbf{E}_5 \times \mathbf{E}_6)_z > 0$$
$$(\mathbf{E}_6 \times \mathbf{E}_1)_z > 0$$

FIGURE 3-43 Identifying a concave polygon by calculating cross products of successive pairs of edge vectors.

Exercise

- **Check if the polygon is convex or not using vector cross product rule. The vertices coordinates are:**



V1(1,1); V2(3,1) ; V3(4,2);
V4(2.5,5), V5(2.5,6); V6(1,6)

E1=V2-V1=[(x2-x1),(y2-y1)]=(2,0)
E2=V3-V2=(1,1)
E3=V4-V3=(-1.5,3)
E4=V5-V4=(0,1)
E5=V6-V5=(-1.5,0)
E6=V1-V6=(0,-5)

E1×E2=E1xE2y-E2xE1y=2-0=2

E2×E3=E2xE3y-E3xE2y= 3+1.5=4.5

E3×E4=E3xE4y-E4xE3y=-1.5

E4×E5=E4xE5y-E5xE4y=1.5

E5×E6=E5xE6y-E6xE5y=7.5

E6×E1=E6xE1y-E1xE6y=10

Splitting Concave Polygons

Once we have identified a concave polygon, we can split it into a set of convex polygons. This can be accomplished using edge vectors and edge cross products. Or, we can use vertex positions relative to an edge extension line to determine which vertices are on one side of this line and which are on the other. For the following algorithms, we assume that all polygons are in the xy plane.

Exercise :

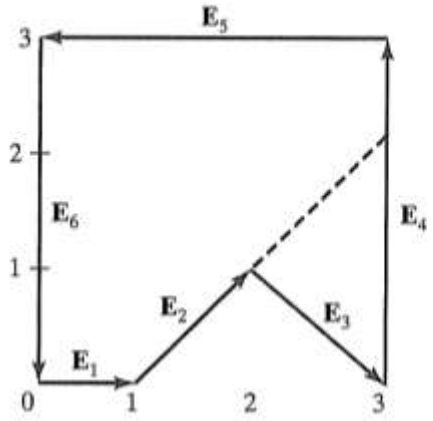


FIGURE 3-44 Splitting a concave polygon using the vector method.

EXAMPLE 3-4 Vector Method for Splitting Concave Polygons

Figure 3-44 shows a concave polygon with six edges. Edge vectors for this polygon can be expressed as

$$\mathbf{E}_1 = (1, 0, 0) \qquad \mathbf{E}_2 = (1, 1, 0)$$
$$\mathbf{E}_3 = (1, -1, 0) \qquad \mathbf{E}_4 = (0, 2, 0)$$
$$\mathbf{E}_5 = (-3, 0, 0) \qquad \mathbf{E}_6 = (0, -2, 0)$$

where the z component is 0, since all edges are in the xy plane. The cross product $\mathbf{E}_j \times \mathbf{E}_k$ for two successive edge vectors is a vector perpendicular to the xy plane with z component equal to $E_{jx}E_{ky} - E_{kx}E_{jy}$:

$$\mathbf{E}_1 \times \mathbf{E}_2 = (0, 0, 1) \qquad \mathbf{E}_2 \times \mathbf{E}_3 = (0, 0, -2)$$
$$\mathbf{E}_3 \times \mathbf{E}_4 = (0, 0, 2) \qquad \mathbf{E}_4 \times \mathbf{E}_5 = (0, 0, 6)$$
$$\mathbf{E}_5 \times \mathbf{E}_6 = (0, 0, 6) \qquad \mathbf{E}_6 \times \mathbf{E}_1 = (0, 0, 2)$$

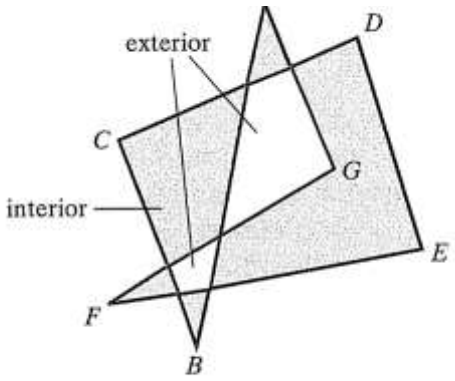
Since the cross product $\mathbf{E}_2 \times \mathbf{E}_3$ has a negative z component, we split the polygon along the line of vector $\mathbf{E}_2$. The line equation for this edge has a slope of 1 and a y intercept of -1. We then determine the intersection of this line with the other polygon edges to split the polygon into two pieces. No other edge cross products are negative, so the two new polygons are both convex. ■

Inside Outside Tests

Various graphics processes often need to identify interior regions of objects. Identifying the interior of a simple object, such as a convex polygon, a circle, or a sphere, is generally a straightforward process. But sometimes we must deal with more complex objects. For example, we may want to specify a complex fill region with intersecting edges, as in Fig. 3-46. For such shapes, it is not always clear which regions of the xy plane we should call "interior" and which regions we should designate as "exterior" to the object boundaries.

Odd-Even rule

We apply the **odd-even rule,** also called the *odd-parity rule* or the *even-odd rule,* by first conceptually drawing a line from any position **P** to a distant point outside the coordinate extents of the closed polyline. Then we count the number of line-segment crossings along this line. If the number of segments crossed by this line is odd, then **P** is considered to be an *interior* point. Otherwise, **P** is an *exterior* point. To obtain an accurate count of the segment crossings, we must be sure that the line path we choose does not intersect any line-segment endpoints. Figure 3-46(a) shows the interior and exterior regions obtained using the odd-even rule for a self-intersecting closed polyline.



Odd-Even Rule
(a)

General Scan-Line Polygon Fill Algorithm

Figure 4-20 illustrates the basic scan-line procedure for a solid-color fill of a polygon. For each scan line that crosses the polygon, the edge intersections are sorted from left to right, and then the pixel positions between, and including, each intersection pair are set to the specified fill color. In the example of Fig. 4-20, the four pixel intersection positions with the polygon boundaries define two stretches of interior pixels. Thus, the fill color is applied to the five pixels from $x = 10$ to $x = 14$ and to the seven pixels from $x = 18$ to $x = 24$. If a pattern fill is to be applied to the polygon, then the color for each pixel along a scan line is determined from its overlap position with the fill pattern.

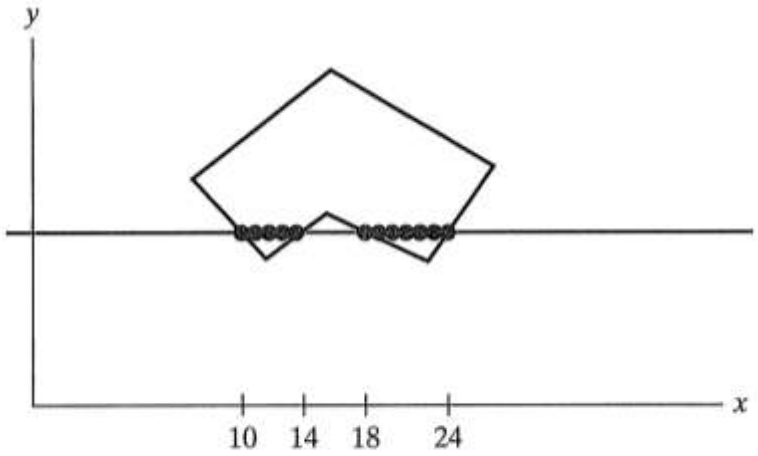


FIGURE 4-20 Interior pixels along a scan line passing through a polygon fill area.

However, the scan-line fill algorithm for a polygon is not quite as simple as Fig. 4-20 might suggest. Whenever a scan line passes through a vertex, it intersects two polygon edges at that point. In some cases, this can result in an odd number of boundary intersections for a scan line. Figure 4-21 shows two scan lines that cross a polygon fill area and intersect a vertex. Scan line y' intersects an even number of edges, and the two pairs of intersection points along this scan line correctly identify the interior pixel spans. But scan line y intersects five polygon edges. To identify the interior pixels for scan line y, we must count the vertex intersection as only one point. Thus, as we process scan lines, we need to distinguish between these cases.

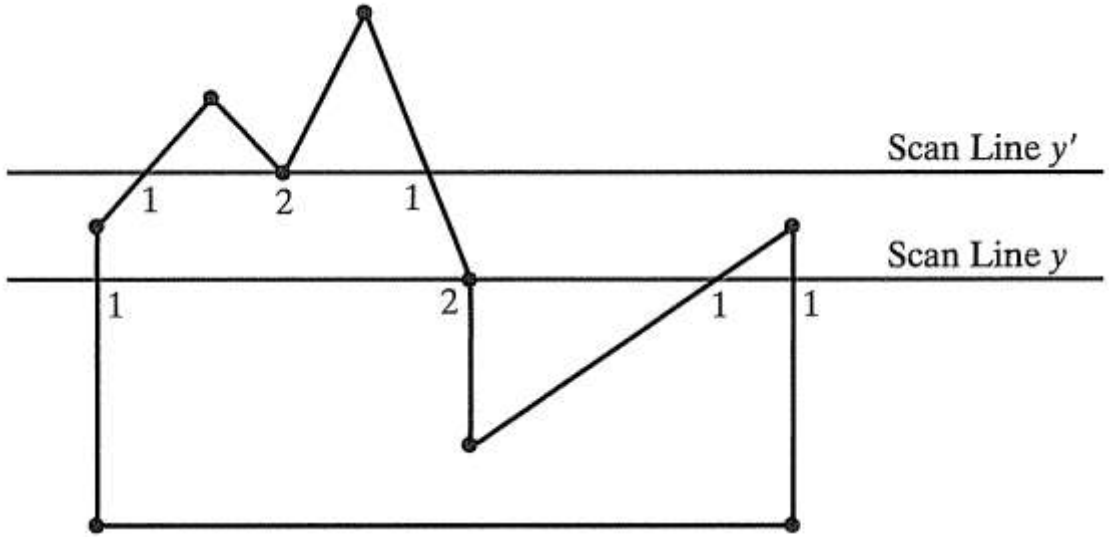


FIGURE 4-21 Intersection points along scan lines that intersect polygon vertices. Scan line y generates an odd number of intersections, but scan line y' generates an even number of intersections that can be paired to identify correctly the interior pixel spans.

We can identify these vertices by tracing around the polygon boundary in either clockwise or counterclockwise order and observing the relative changes in vertex y coordinates as we move from one edge to the next. If the three endpoint y values of two consecutive edges monotonically increase or decrease, we need to count the shared (middle) vertex as a single intersection point for the scan line passing through that vertex. Otherwise, the shared vertex represents a local extremum (minimum or maximum) on the polygon boundary, and the two edge intersections with the scan line passing through that vertex can be added to the intersection list.

Plane Equation

Each polygon in a scene is contained within a plane of infinite extent. The general equation of a plane is

$$A x + B y + C z + D = 0 \qquad (3\text{-}59)$$

where (x, y, z) is any point on the plane, and the coefficients A, B, C, and D (called *plane parameters*) are constants describing the spatial properties of the plane. We can obtain the values of A, B, C, and D by solving a set of three plane equations using the coordinate values for three noncollinear points in the plane. For this purpose, we can select three successive convex-polygon vertices, (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3), in a counterclockwise order and solve the following set of simultaneous linear plane equations for the ratios A/D, B/D, and C/D:

$$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1, \qquad k = 1, 2, 3 \qquad (3\text{-}60)$$

How to find constants A,B,C and D for the plane equation containing a polygon ?

Cramer's rule, as

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \qquad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \qquad D = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{3-61}$$

Expanding the determinants, we can write the calculations for the plane coefficients in the form

$$\begin{aligned} A &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\ B &= z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2) \\ C &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\ D &= -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1) \end{aligned} \tag{3-62}$$

Front and back polygon faces

$$A x + B y + C z + D \neq 0$$

Thus we can identify the point as either behind or in front of a polygon surface contained within that plane according to the sign (negative or positive) of $Ax + By + Cz + D$:

if $A x + B y + C z + D < 0$, the point (x, y, z) is behind the plane

if $A x + B y + C z + D > 0$, the point (x, y, z) is in front of the plane

Normal Vector

Orientation of a polygon surface in space can be described with the **normal vector** for the plane containing that polygon, as shown in Fig. 3-53. This surface normal vector is perpendicular to the plane and has Cartesian components (A, B, C), where parameters A, B, and C are the plane coefficients calculated in Eqs. 3-62. The normal vector points in a direction from inside the plane to the outside; that is, from the back face of the polygon to the front face.

The elements of a normal vector can also be obtained using a vector cross-product calculation. Assuming we have a convex-polygon surface facet and a right-handed Cartesian system, we again select any three vertex positions, $\mathbf{V}_1$, $\mathbf{V}_2$, and $\mathbf{V}_3$, taken in counterclockwise order when viewing from outside the object toward the inside. Forming two vectors, one from $\mathbf{V}_1$ to $\mathbf{V}_2$ and the second from $\mathbf{V}_1$ to $\mathbf{V}_3$, we calculate $\mathbf{N}$ as the vector cross product:

$$\mathbf{N} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1) \qquad (3\text{-}63)$$

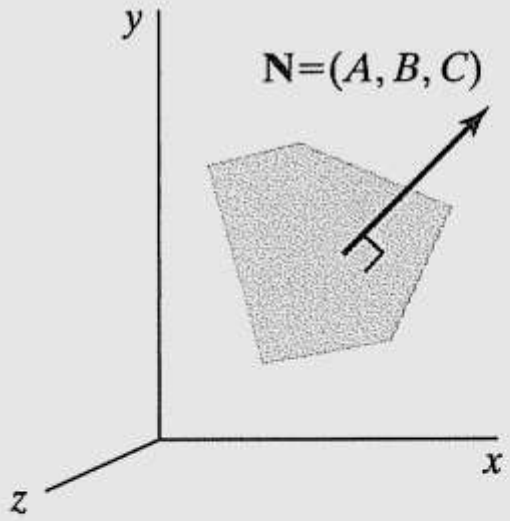


FIGURE 3-53 The normal vector **N** for a plane described with the equation $Ax + By + Cz + D = 0$ is perpendicular to the plane and has Cartesian components (A, B, C).

Exercise

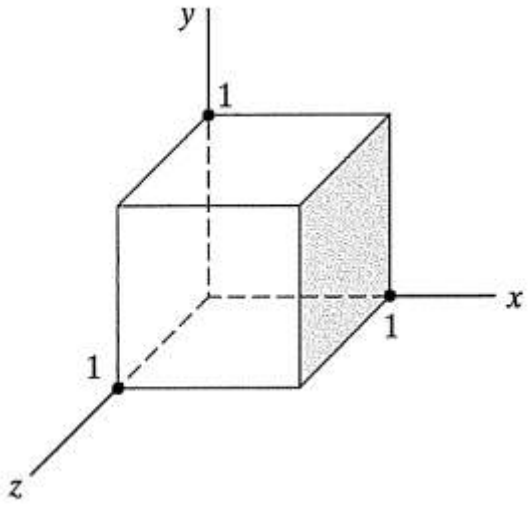
- What is The **normal** vector componenets for the shaded surface?

- Is the point (1,1,0) in the shaded plane? (1,3,8); (5,1,1)

- What is surface equation for the polygon ?(1,1,1); (1,0,0); (0,0,1) ?

✓ A=-1, B=1 , C=-1 , D=1

- What are the normal vector components ?



End Of Presentation