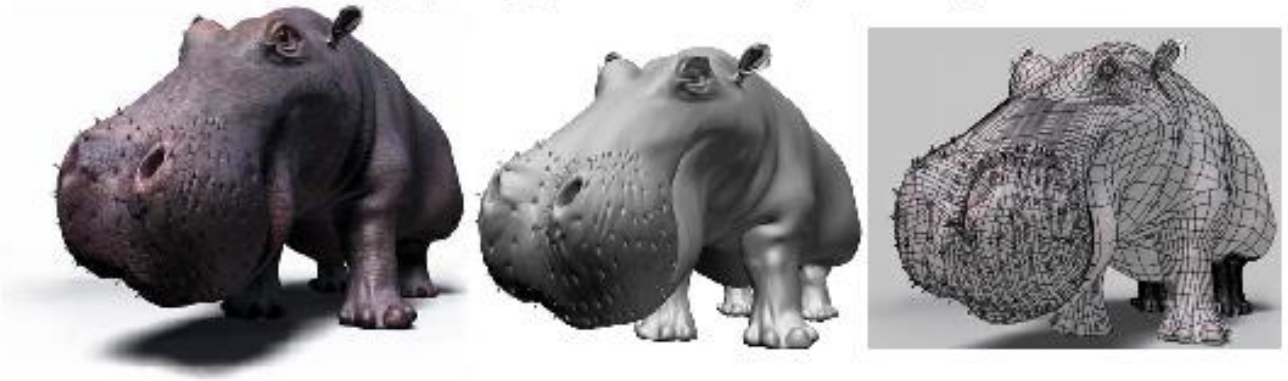
POLYGON CLIPPING

JAGJIT BHATIA
Dept. of computer Science & IT

Fill area : an area that is filled with solid colour or pattern

Polygon Fill Areas

- Most library routines require that a fill area be specified as a polygon
  - OpenGL only allows ***convex*** polygons
- Non-polygon (curved) objects can be approximated by polygons
  - Surface tessellation, polygon mesh, triangular mesh



Polygon types

- Simple polygons are either ***convex*** or ***concave***:
  - Convex polygon: All interior angles < 180°, or any line segment combining two points in the interior is also in the interior

convex polygon

concave polygon

can be split into a number of convex polygons

Identifying a concave polygon

- has at least one interior angle >180 degrees
- extensions of some edges will intersect other edges

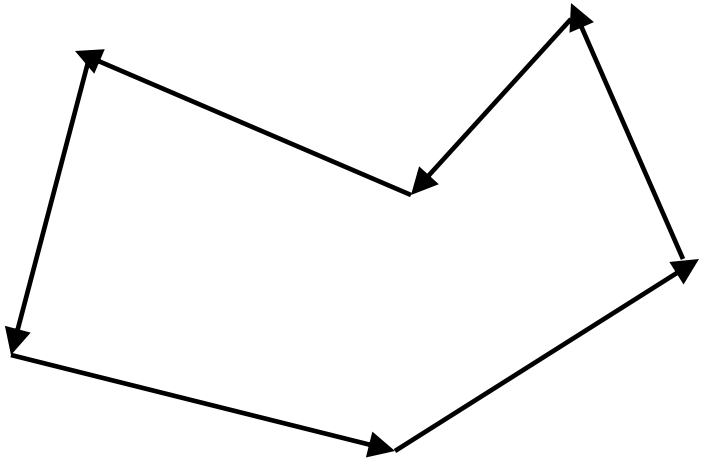
One test:

Express each polygon edge as a vector, with a consistent orientation.

Can then calculate cross-products of adjacent edges

Identifying a concave polygon

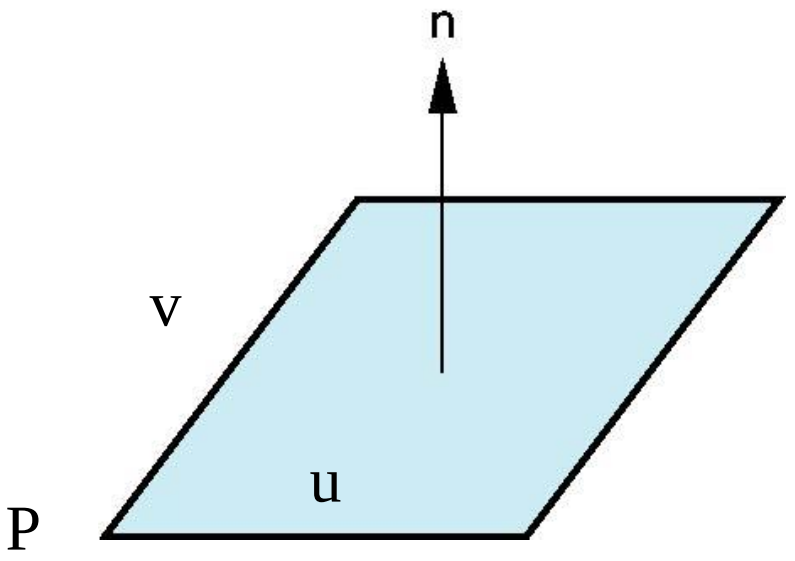
- When polygon edges are oriented with an anti-clockwise sense
  - cross product at convex vertex has positive sign
  - concave vertex gives negative sign



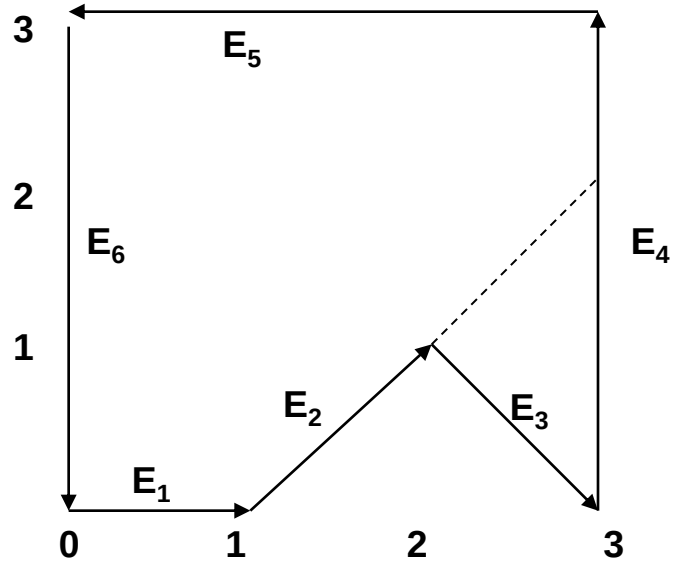
Normals

- Every plane has a vector **n** normal (perpendicular, orthogonal) to it
- **n** = **u** x **v** (vector cross product)

$$\bar{u} \times \bar{v} = \begin{pmatrix} u_y v_z - u_z v_y \\ u_z v_x - u_x v_z \\ u_x v_y - u_y v_x \end{pmatrix}$$



Vector method for splitting concave polygons

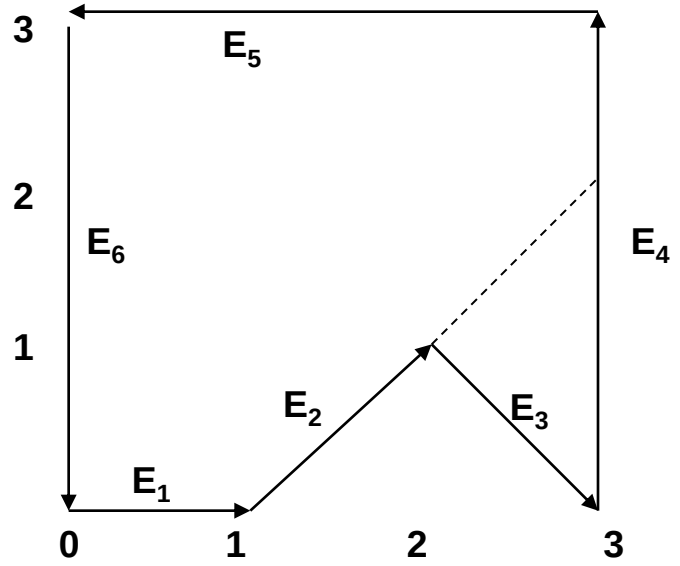


- E_1=(1,0,0) E_2=(1,1,0)
- E_3=(1,-1,0) E_4=(0,3,0)
- E_5=(-3,0,0) E_6=(0,-3,0)

- All z components have 0 value.
- Cross product of two vectors $E_j x E_k$ is perpendicular to them with z component

$E_{jx}E_{ky}-E_{kx}E_{jy}$

Example continued

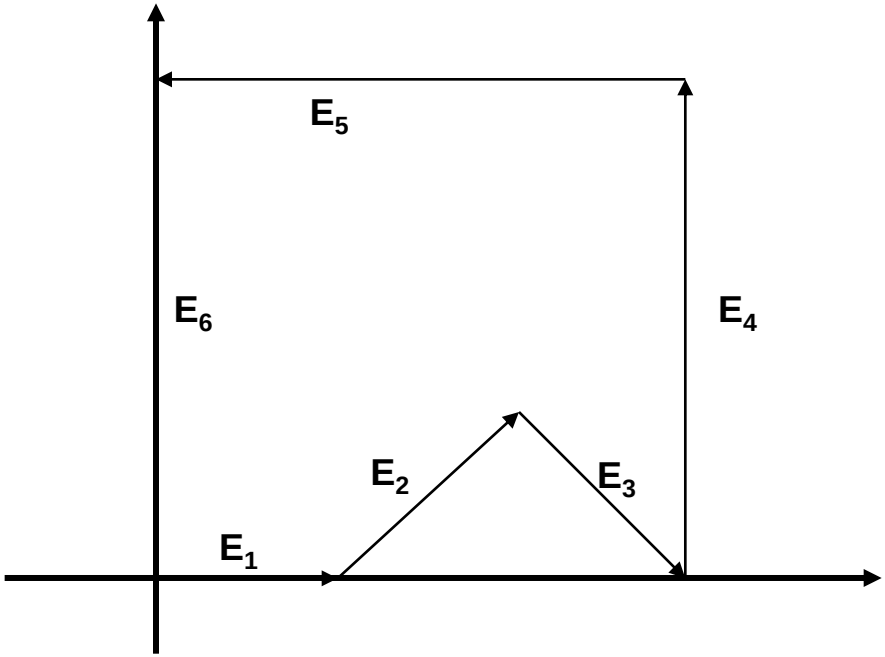


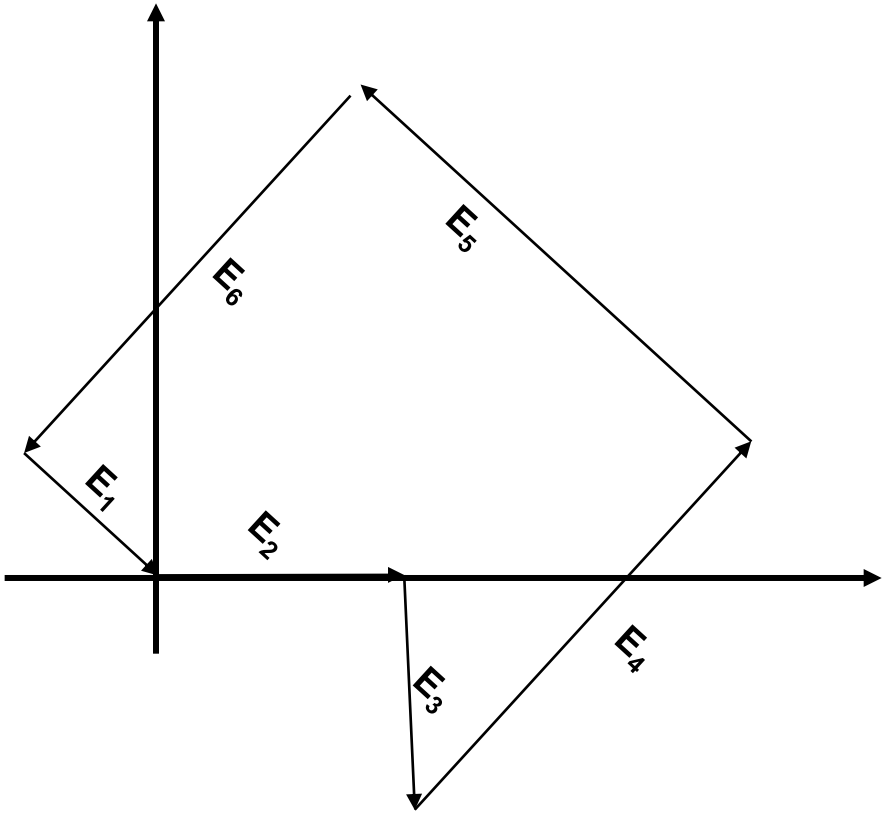
- $E_1 x E_2 = (0,0,1)$
  $E_2 x E_3 = (0,0,-2)$
  $E_3 x E_4 = \ldots$
  $E_4 x E_5 = \ldots$
- $E_5 x E_6 = \ldots$
  $E_6 x E_1 = \ldots$

- Since $E_2 x E_3$ has negative sign, split the polygon along the line of vector E_2

Rotational method

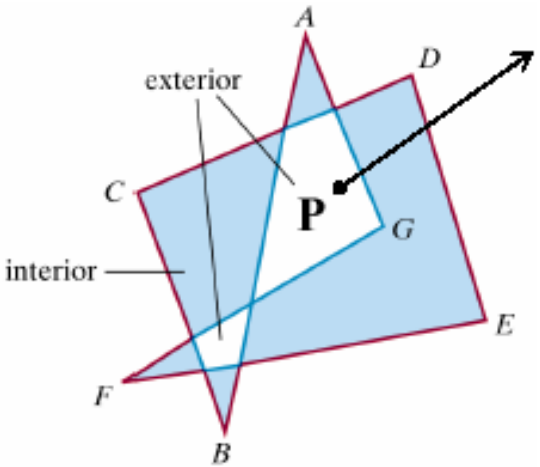
- Rotate the polygon so that each vertex in turn is at coordinate origin.
- If following vertex is below the x axis, polygon is concave.
- Split the polygon by x axis.


E5
E6
E4
E2
E3
E1


E6
E5
E1
E2
E3
E4

Inside-Outside Tests

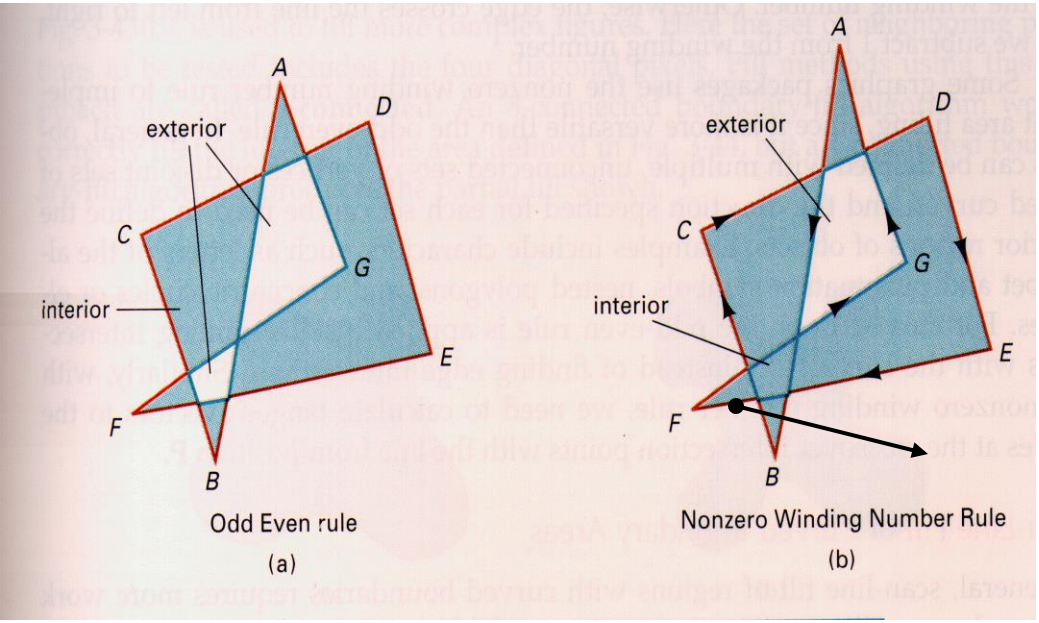
- Identifying the interior of a polygon (simple or complex) is important to identify the region to be filled
- Odd-even rule: To determine whether point **P** is inside or not. Draw a line starting from P to a distant position. Count the number of edges that cross this line. If the count is **odd** then the point is **inside**, otherwise it is outside.



Odd-Even Rule

Inside-Outside? nonzero winding-number rule

A winding number is an attribute of a point with respect to a polygon that tells us how many times the polygon encloses (or wraps around) the point. It is an integer, greater than or equal to 0. Regions of winding number 0 (unenclosed) are obviously outside the polygon, and regions of winding number 1 (simply enclosed) are obviously inside the polygon.



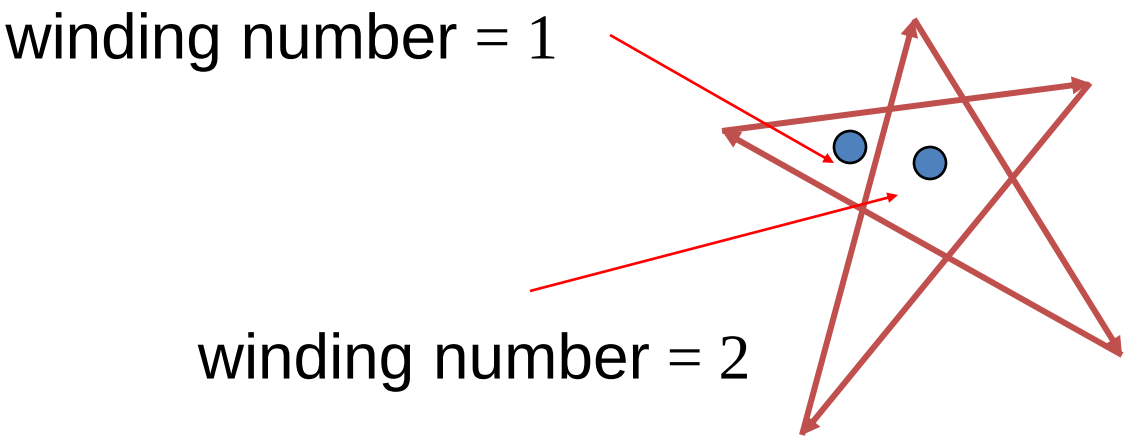
Initially 0

+1: edge crossing the line from right to left

-1: left to right

Winding Number

- Count clockwise encirclements of point



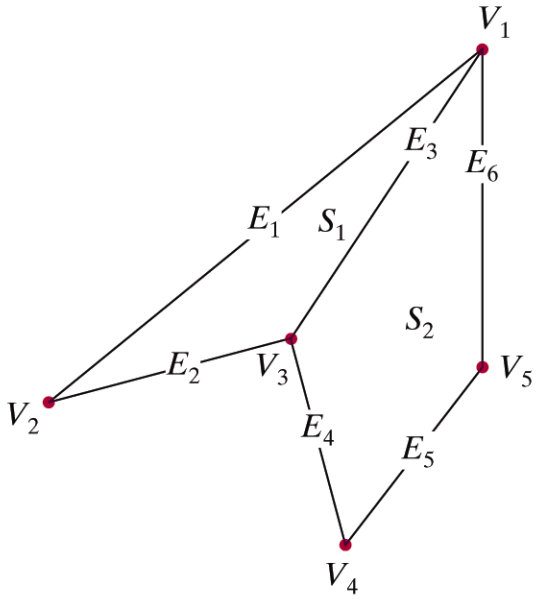
- Alternate definition of inside: inside if winding number $\neq 0$

Polygon tables

- store descriptions of polygon geometry and topology, and surface parameters: colour, transparency, light-reflection
- organise in 2 groups
  - geometric data
  - attribute data

Polygon Tables:

Geometric data



VERTEX TABLE	
V_1:	x_1, y_1, z_1
V_2:	x_2, y_2, z_2
V_3:	x_3, y_3, z_3
V_4:	x_4, y_4, z_4
V_5:	x_5, y_5, z_5

EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

SURFACE-FACET TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

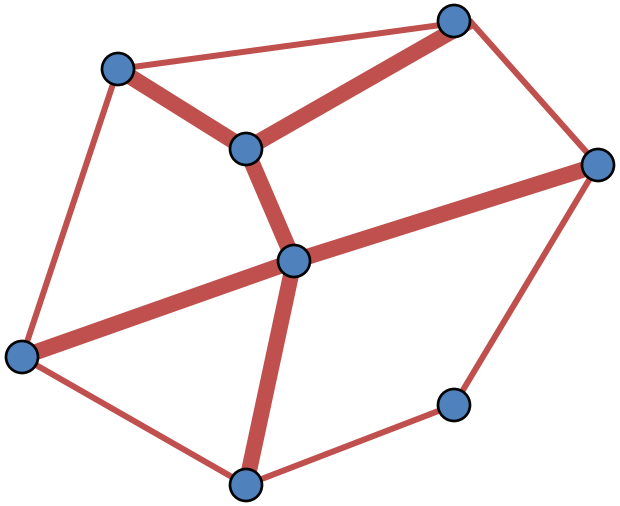
Figure 3-50

Geometric data-table representation for two adjacent polygon surface facets, formed with six edges and five vertices.

- Data can be used for consistency checking
- Additional geometric data stored: slopes, bounding boxes

Shared Edges

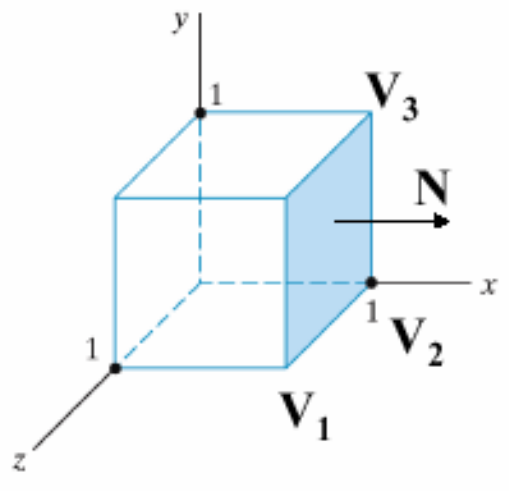
- Vertex lists will draw filled polygons correctly but if we draw the polygon by its edges, shared edges are drawn twice



- Can store mesh by *edge list*

Front and Back Face of a Polygon

- The normal vector points in a direction from the back face of the polygon to the front face
- Normal vector is the cross product of the two edges of the polygon in counter-clockwise direction



$$\mathbf{N} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_2)$$

Inward and Outward Facing Polygons

- The order $\{v_1, v_6, v_7\}$ and $\{v_6, v_7, v_1\}$ are equivalent in that the same polygon will be rendered by OpenGL but the order $\{v_1, v_7, v_6\}$ is different
- The first two describe *outwardly facing* polygons
- Use the *right-hand rule* = counter-clockwise encirclement of outward-pointing normal
- OpenGL can treat inward and outward facing polygons differently