

Plygns

CSCI 4229/5229

Computer Graphics

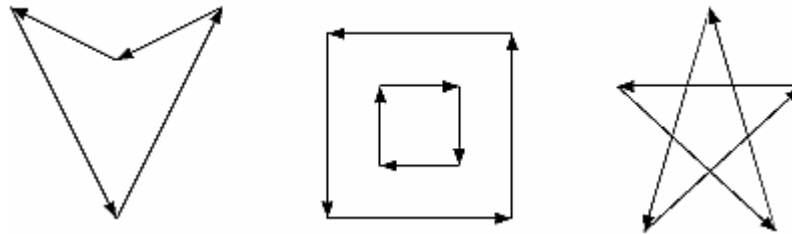
Fall 2006

Polygon Definitions

- A polygon is strictly a planar object
 - Plane defined as $ax + by + cz = 1$
 - Three distinct (x,y,z) points
 - One (x,y,z) point and a normal vector
 - Finite subset of plane defined by set of vertices
 - Vertexes **must** be in the plane
- In OpenGL you can specify 3D vertices
 - When vertexes are not in a plane, the results are implementation dependent

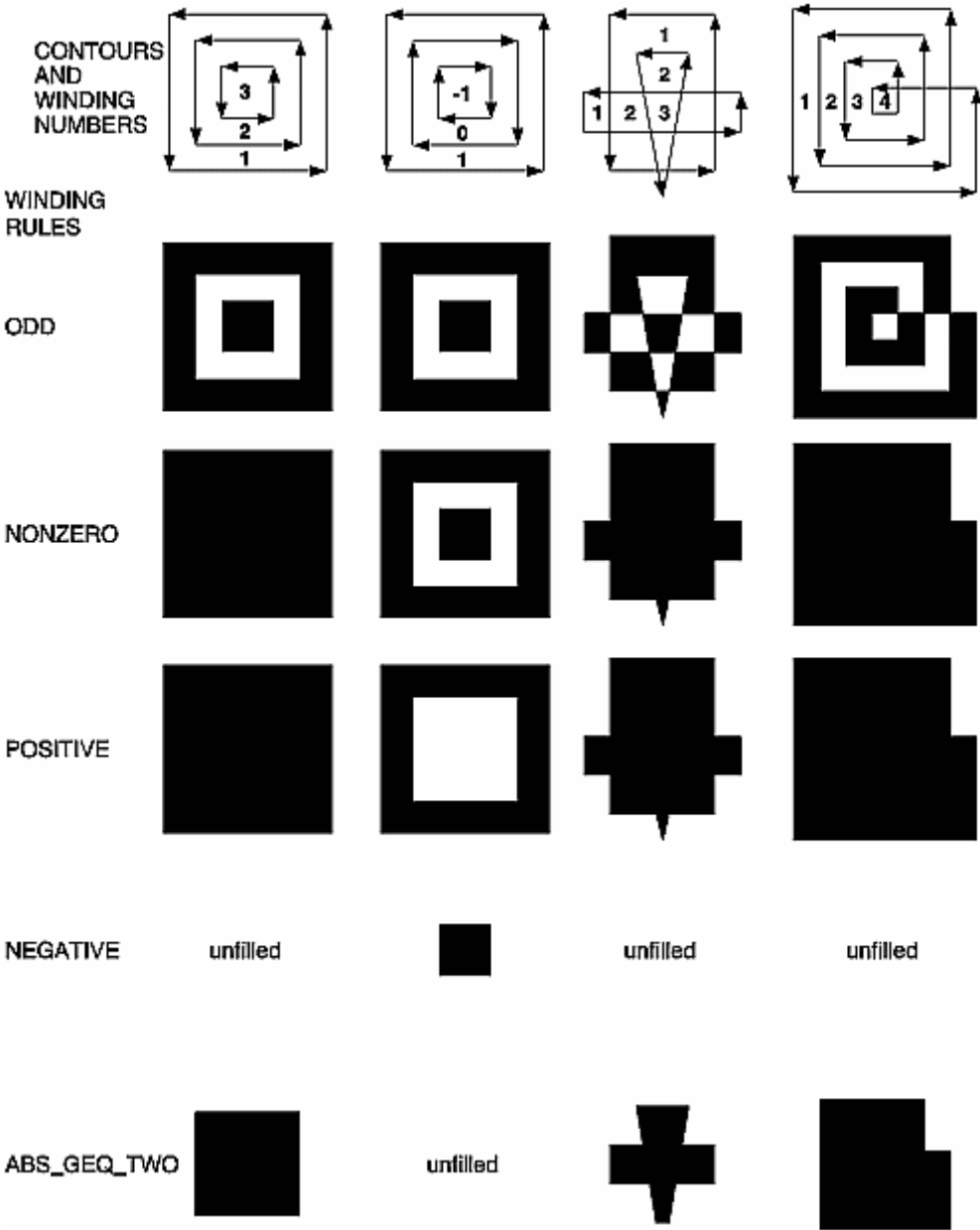
Convex vs. Concave

- Convex polygons: Given any two points \mathbf{a} and \mathbf{b} in the polygon $\mathbf{c} = f\mathbf{a} + (1-f)\mathbf{b}$ is also inside the polygon for any f in $(0,1)$
- Concave polygons: Some \mathbf{c} is outside the polygon
- Concave examples:



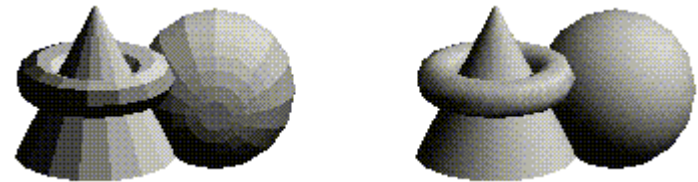
- OpenGL requires **convex** polygons

What is Inside?



Normals for Polygons

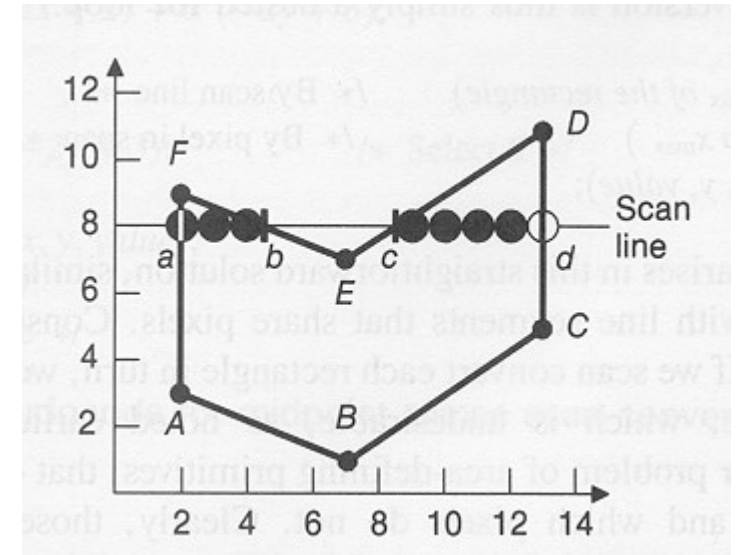
- Given 3 points in the plane P_1 , P_2 and P_3
 - Normalize $(P_2 - P_1) \times (P_3 - P_1)$
 - Use any three distinct vertexes of the polygon not on a line



- True Gouraud shading
 - Calculate normals for all polygons
 - At common vertexes, average all the normals
 - Interpolate across polygons
- OpenGL normals are set at vertexes

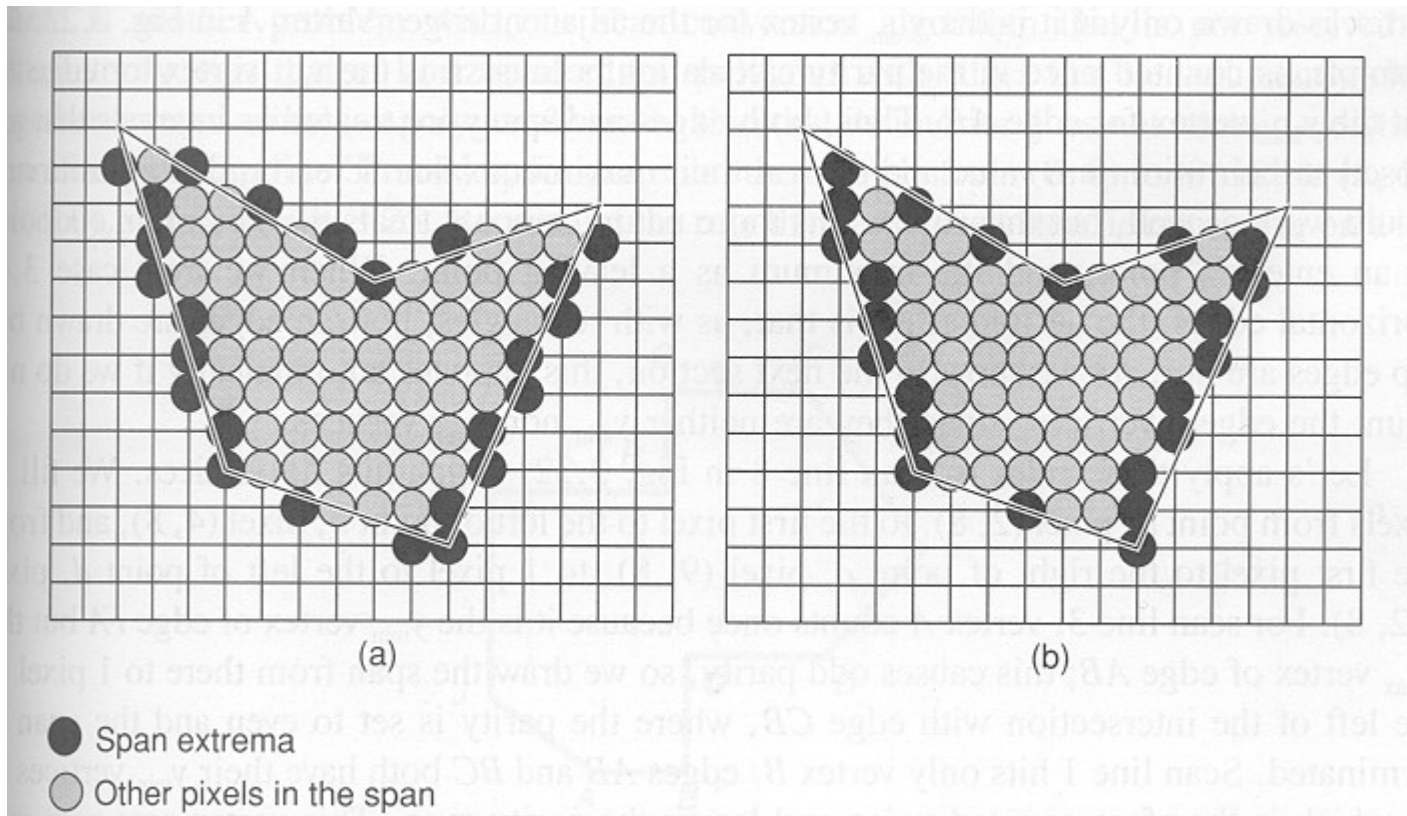
Scan Converting Polygons

- Draw horizontal lines to fill the polygon
- Pairs of points are interior
- Vertices on a scanline is a problem
- Convex polygons are easy



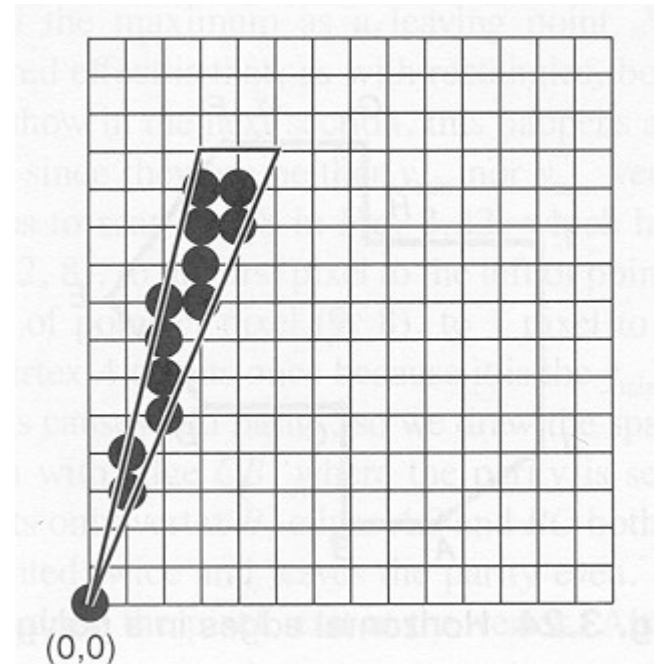
Deciding the Polygon Extent

- (a) Bresenham Outline
- (b) Strictly Interior Outline



Edge Coherence

- Scan lines intersects near the last scanline
- Slivers may be just a line



OpenGL Polygons

- `glPolygonMode(type)`
 - `GL_POINT` draws vertexes
 - `GL_LINE` draws outline
 - `GL_FILL` fills polygon
- `glPolygonStipple(mask)`
 - 32x32 pixel (byte) mask
 - `glEnable(GL_POLYGON_STIPPLE)`

glPolygonOffset(factor,units)

- Modifies the depth stored in the Z-buffer
- Offset = $m \cdot \text{factor} + r \cdot \text{units}$
 - m maximum slope relative to viewing direction
 - r minimum resolvable difference in Z-buffer
- Separate settings for POINT/LINE/FILL
 - glEnable(GL_POLYGON_OFFSET_POINT)
 - glEnable(GL_POLYGON_OFFSET_LINE)
 - glEnable(GL_POLYGON_OFFSET_FILL)
- Typically **glPolygonOffset(1,1)** works

Offset Uses

- Hidden line removal in wireframes
- Drawing lines on polygons
- Resolving z-fighting

