



Polygon Rasterization

Young J. Kim

What To Learn

Definition of a Polygon

Polygon Inside/Outside Test

- Odd/even rule
- Winding number

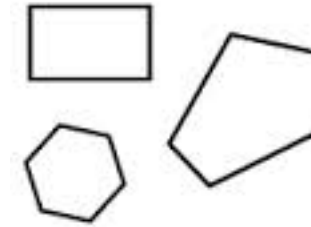
Polygon Rasterization

- Scanline algorithm
- Triangle rasterization

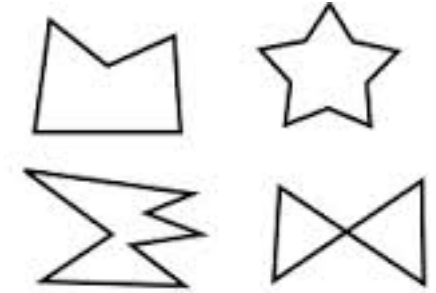


Polygon

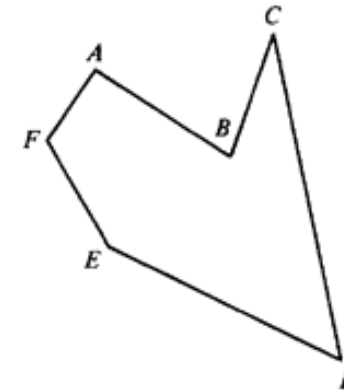
- ❑ Specified by a set of three or more *vertices* connected by *edges*
- ❑ Convex VS non-convex
- ❑ Simple VS non-simple
- ❑ Degenerate
 - Collinear or duplicated vertices



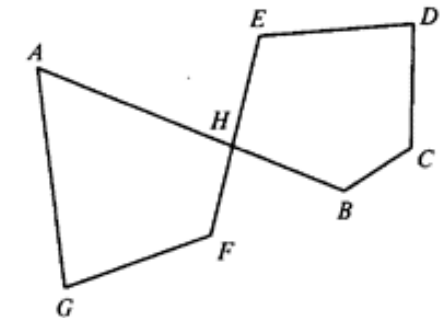
Convex Polygons



Non-convex Polygons



Simple

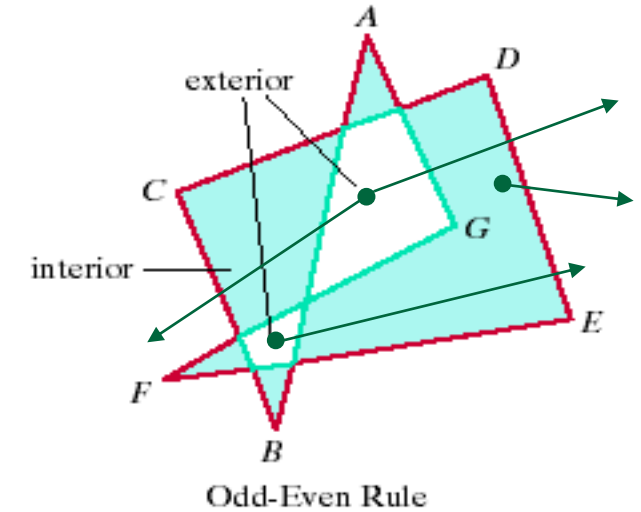


Non-simple

Inside-Outside Tests

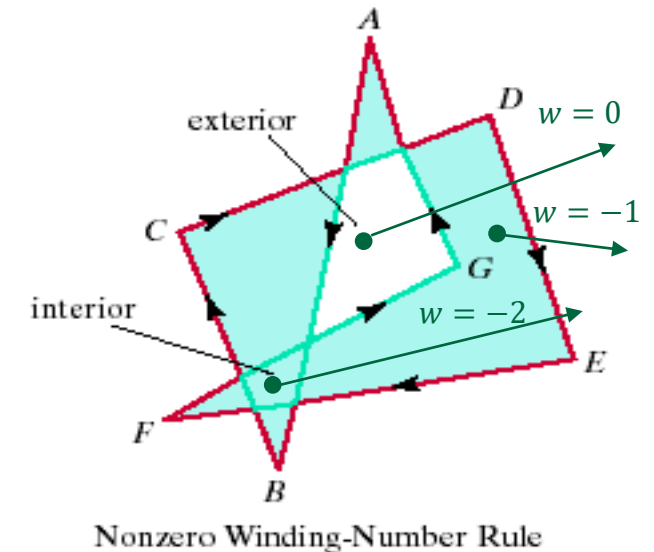
□ Odd-even rule

- Shoot a ray from the point and count edge crossings:
inside if the count is an odd number

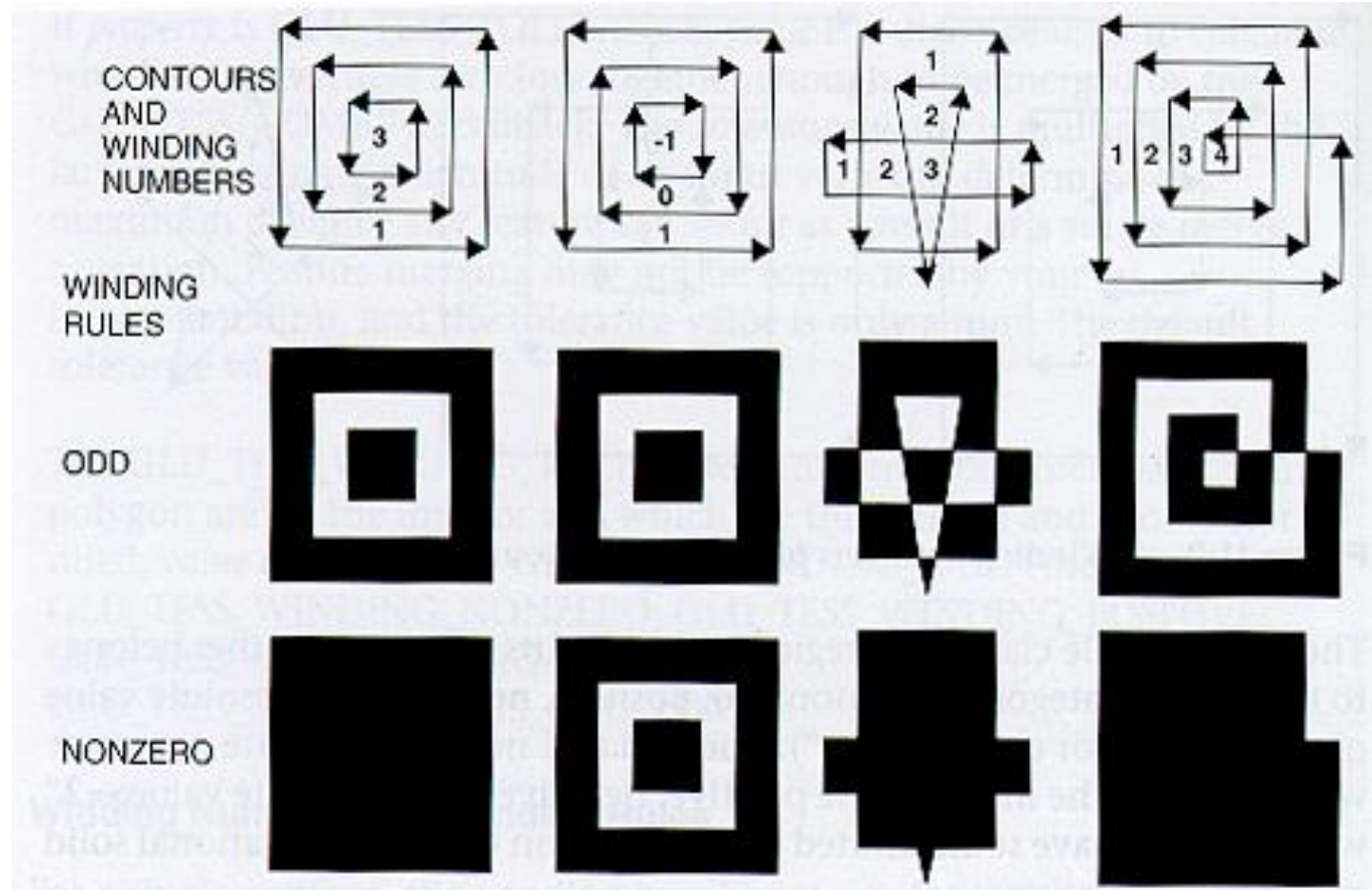


□ Winding number

- Shoot a ray from the point and +1 when crossing a counter-clockwise encircling line and -1 otherwise
- Inside if winding number $\neq 0$



Odd-Even VS Winding



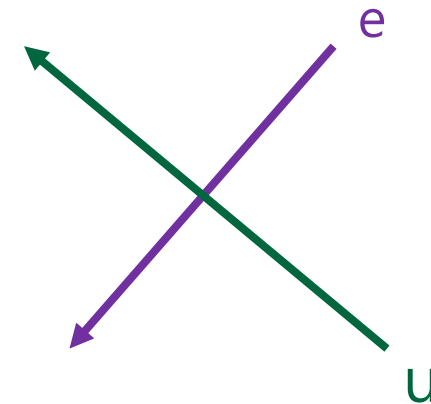
Implementation of Winding Rule

□Inputs

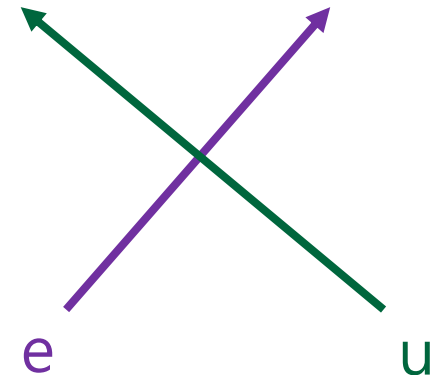
- Edge direction: $e = (e_x, e_y, 0)$
- Ray direction: $u = (u_x, u_y, 0)$

□Using cross product (\times)

- CCW: z component of $u \times e = u_x e_y - u_y e_x > 0$
- CW: otherwise



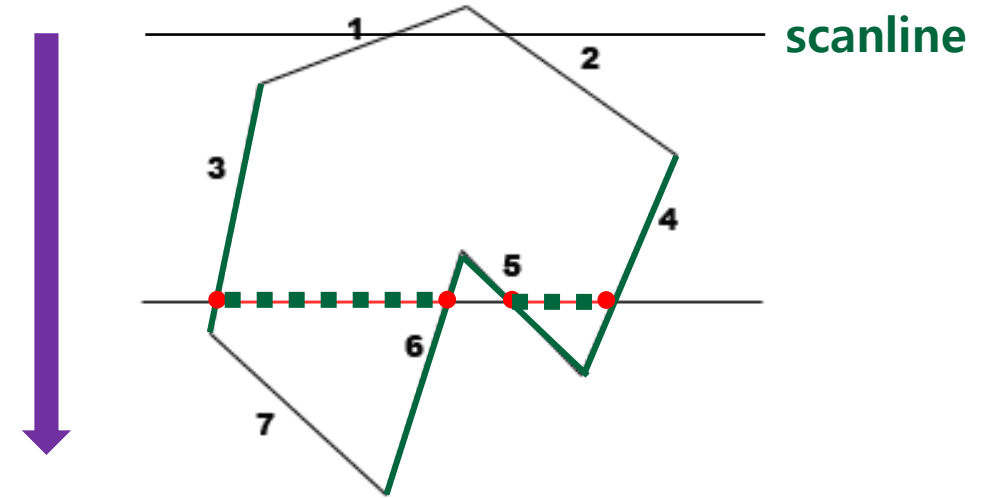
CCW



CW

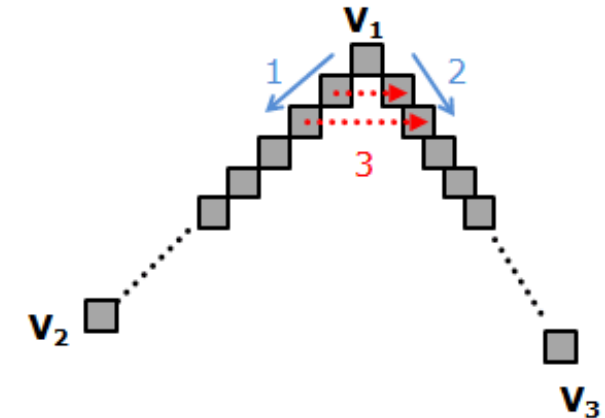
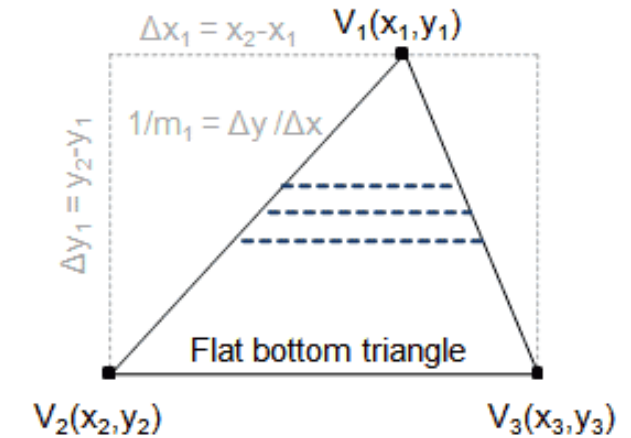
General Polygon Rasterization: Scanline Algorithm

1. Sorting the edges in y-coordinates
2. For each scan line
 - ① Get all edges intersected
 - ② Get all intersected points and sort them
 - ③ Fill the intersected segments using a polygon I/O test



Special Case: Triangle Rasterization

1. Draw v_1v_2 using Bresenham's algorithm until it proceeds by one pixel in y-direction
2. Do the same for v_1v_3
3. Fill the current line segments
4. Repeat 1-3 until arriving at v_2 or v_3



Special Case: Triangle Rasterization

