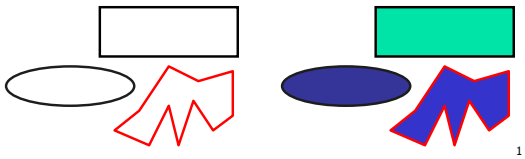
Filling Graphical Shapes

- Know how to draw outlines
  - Polygons, circles, ellipses, etc.
- What if we want to fill them?



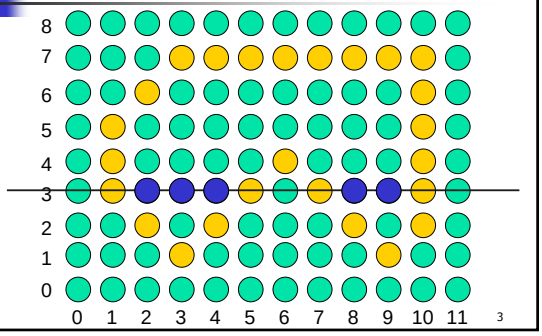
1

Calculating Fill Areas

- Fill one scan line at a time
  - Odd/even rule
- Need intersection points
  - Use pixels directly?
    - In a buffer? On screen (GetPixel)?
  - Calculate intersections from polygon edge equations?

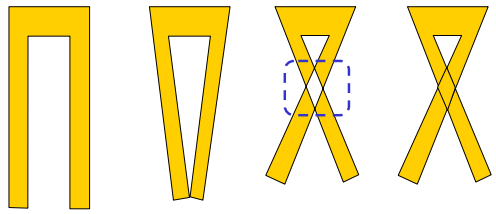
2

Scan-Line Polygon Fill



3

A Filling Anomaly?



4

Inside-Outside Tests

- Odd-even rule
  - Generalized from scan-line fill
  - May produce unusual results if edges intersect
- Nonzero winding number rule
  - Alternate way of determining interior

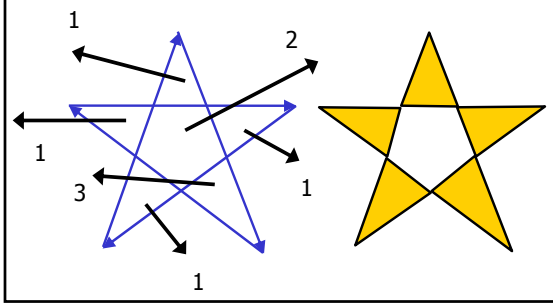
5

Odd-Even Rule

- Choose a point
- Draw ray to a distant point
  - Don't intersect any vertices
- Count edges crossed
  - Odd count means interior
  - Even count means exterior
- Same idea as scan-line even/odd

6

Odd-Even Example



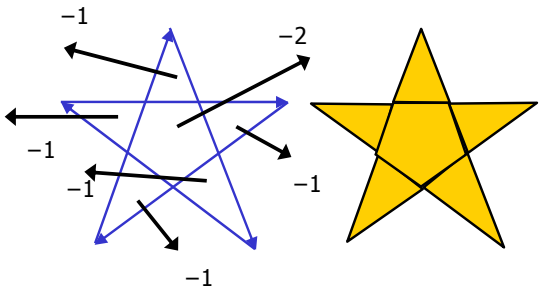
7

Nonzero Winding Rule

- Choose a point
- Draw ray to a distant point
  - Don't intersect any vertices
  - (or shorten an edge when the edges lie on opposite sides of the ray)
- Consider edges crossed (right hand rule)
  - Subtract 1 when ray to edge is clockwise
    - "Left-to-right" as seen along ray
  - Add 1 when ray to edge is counter-clockwise
- Nonzero count means interior

8

Nonzero Winding Example



9

Computing Winding Number

- Edge crossing direction
- Compute cross product
  - Between ray and edge
  - Sign of z value determines direction
- Compute dot product
  - Use perpendicular to ray vs. edge
  - Sign of product determines direction

See text, page 127

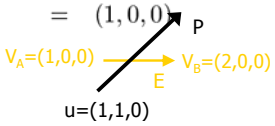
10

Cross Product Example

$$\begin{aligned} u \times E &= (u_y E_z - u_z E_y, u_z E_x - u_x E_z, u_x E_y - u_y E_x) \\ &= (1 \cdot 0 - 0 \cdot 0, 0 \cdot 1 - 1 \cdot 0, 1 \cdot 0 - 1 \cdot 1) \\ &= (0, 0, -1) \end{aligned}$$

"−1" means E crosses P from left to right

$$\begin{aligned} E &= V_B - V_A \\ &= (1, 0, 0) \end{aligned}$$



u=(1,1,0)

11

Cross Product Simplification

$$\begin{aligned} u \times E &= (u_y E_z - u_z E_y, u_z E_x - u_x E_z, u_x E_y - u_y E_x) \\ &= (u_y \cdot 0 - 0 \cdot E_y, 0 \cdot E_x - u_x \cdot 0, u_x E_y - u_y E_x) \\ &= (0, 0, u_x E_y - u_y E_x) \end{aligned}$$

negative means E crosses P from left to right

12

Dot Product Example

$$\begin{aligned} w \cdot E &= w_x E_x + w_y E_y + w_z E_z \\ &= -1 \cdot 1 + 1 \cdot 0 + 0 \cdot 0 \\ &= -1 \end{aligned}$$

Again, "−1" means E crosses P from left to right

u=(1,1,0), w = (−1,1,0) ["right-to-left" perpendicular to u]

13

Dot Product Simplification

$$w = u_\perp$$
$$m_\perp = -\frac{\Delta x}{\Delta y}$$

$$\begin{aligned} w \cdot E &= u_\perp \cdot E \\ &= (-u_y, u_x, 0) \cdot (E_x, E_y, E_z) \\ &= u_x E_y - u_y E_x \end{aligned}$$

Same result as from cross product!

14

What Does Fill Mode Mean?

- Mathematical definition
  - Ray / edge crossings
- Odd/even (alternating)
  - Overlapping parts not filled
- Nonzero winding
  - Overlapping parts are filled

15

Filling a Polygon (Qt) (1)

QCanvasPolygon
- inherits QCanvasPolygonalItem
- inherits QCanvasItem
- inherits Qt (namespace for enums)

```
QCanvasPolygon::setPoints(
    QPointArray pa)
QCanvasPolygonalItem::setBrush(
    Qbrush b)
QCanvasPolygonalItem::setWinding(
    bool enable) // protected
```

16

Filling a Polygon (Qt) (2)

```
class myPolygon : public QCanvasPolygon
   {
     myPolygon();
     ...
   };

myPolygon::myPolygon()
   {
     setBrush(QBrush(green,
                   DiagCrossPattern));
     setWinding(true);
     ...
   }
```

17

Filling a Polygon (MS Windows)

```
BOOL CDC::Polygon
  (LPPOINT points,
   int count);

int CDC::SetPolyFillMode
  (int mode);
```

ALTERNATE
WINDING

18