Computer Graphics

Polygon Filling
Drawing Geometry on Raster Displays

Raster Display
- Discrete grid of elements (frame buffer pixels)
- Frame buffer is scanned, one line at a time, to refresh the image.
- So:
  - Difficult to draw smooth lines
  - Displays only a discrete approximation of any shape

Terminology
- Pixel: Picture element
  - Smallest accessible element in picture
  - Usually rectangular or circular
- Aspect Ratio: Ratio between physical dimensions of pixel (not necessarily 1:1)
- Dynamic Range: Ratio between minimal (not zero) and maximal light intensity emitted by displayed pixel. Measured in bits.

Terminology (cont')
- Resolution: number of distinguishable rows and columns on device. Measured in
  - Absolute values (1K x 1K)
  - Relative values (300 dots per inch)
- Screen Space: discrete 2D Cartesian coordinate system of screen pixels
- Object Space: 3D Cartesian coordinate system of the universe where the objects (to be displayed) are embedded

Problem
- Given a closed two dimensional polygon, fill its interior with specified color on graphics display
- Assumptions:
  - Polygon is simple, i.e. no self intersections
  - Polygon is simply connected (no holes)
- Solutions:
  - Flood fill
  - Scan conversion

Flood Fill Algorithm
- Let P be a polygon whose boundary is drawn
- Let C be the color to fill the polygon
- Let p = (x, y) ∈ P be a point inside P

Copyright  Gotsman, Elber, Barequet, Karni, Sheffer
**Flood Fill**

FloodFill(Polygon P, x, y, Color C)
if not (OnBoundary(x, y, P) or Colored(x, y, C))
begin
FloodFill(x, y, C);
FloodFill(x + 1, y, C);
FloodFill(x - 1, y, C);
FloodFill(x, y + 1, C);
end;

**Pros and Cons?**
- Correctness
- Simplicity
- Efficiency in time and space
- Limitations

**Basic Scan Conversion Algorithm**
- Let P be a polygon with n vertices \( v_0 \) to \( v_n \) (\( v_0 = v_n \))
- Let C be the color to paint this polygon
- Each intersection of straight line with boundary moves into/out-of the polygon
- Detect (and set) pixels inside the polygon boundary

**Optimized Scan Conversion**
- Maintain list of active edges \( A \)
- \( A \) contains edges that intersect current scan line

**Special Cases**
- Comment: Errors in both algorithms can have global effects or “leaks” ⇒ need very stable implementation
- Question: Any advantage to making the flood fill in eight instead of four directions? Any disadvantages?
- Question: What is the bound on stack size of flood fill, given a screen size \( x \) by \( y \)?
## Comparison

<table>
<thead>
<tr>
<th>Flood Fill</th>
<th>Scan Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very simple</td>
<td>More complex</td>
</tr>
<tr>
<td>Discrete algorithm in screen space</td>
<td>Discrete algorithm in object and/or screen space</td>
</tr>
<tr>
<td>Requires GetPixelVal system call</td>
<td>Device independent</td>
</tr>
<tr>
<td>Requires a seed point</td>
<td>No seed point required</td>
</tr>
<tr>
<td>Requires very large stack</td>
<td>Requires small stack</td>
</tr>
<tr>
<td>Common in paint packages</td>
<td>Used in image rendering</td>
</tr>
<tr>
<td>Unsuitable for line-based Z-buffer</td>
<td>Suitable for line-based Z-buffer</td>
</tr>
</tbody>
</table>