Computer Graphics Attributes

– Line and Text Attributes
  • Fonts in Windows

– Area Fill
  • Boundary/Flood Fill Algorithms
  • Scanline Polygon Fill Algorithm
Attributes

- How primitives are to be displayed
- Most systems use modal attributes
  - Values in effect until changed

Text Attributes

- Font (typeface)
  - Character set with particular design style
- Display style
  - underlined, italic, boldface, outlined, strikeout, spacing, etc.
- Color
- Size (width, height)--specified in points
  - Point = 1/72 inch
**Text Attributes, continued**

- **Orientation**—how much character is rotated
- **Escapement**—orientation of line between first & last character in a string

![Character Escapement & Orientation Diagram]

**Line Attributes**

- **Color**
- **Width**
- **Style**—solid, dotted, dashed, etc.
  
  Can be specified by giving a pattern array
  
  e.g., \( \text{pat[ ]} = \{1,1,1,1,1,1,0,0\} \)
  
  Repeat this pattern on entire line:
  
  \( j \text{th pixel along line:} \)
  
  \[
  \text{if (pat[\%8]==1) SetPixel(x,y)}
  \]

  ![Dashed Line Example]

- In Windows, use a pen (CPen)
Area Fill

- Important for any closed output primitive
  - Polygons, Circles, Ellipses, etc.
- Attributes:
  - fill color
  - fill pattern
- 2 Types of area fill algorithms:
  - Boundary/Flood Fill Algorithms
  - Scanline Algorithms

Area Fill Algorithms

- See CS-460/560 Notes Web Page
- Link to:
  - Week 5-BC: Area Fill Algorithms
- URL:
Boundary/Flood Fill Algorithms
- Determine which points are inside from pixel color information
  - e.g., interior color, boundary color, fill color, current pixel color
  - Color the ones that are inside.

Scanline Algorithms
- Examine horizontal scanlines spanning area
- Find intersection points between current scanline and borders
- Color pixels along the scanline between alternate pairs of intersection points
- Especially useful for filling polygons
  - polygon intersection point calculations are very simple and fast
  - Use vertical and horizontal coherence to get new intersection points from old
Boundary/Flood Fill Algorithms

- Determine which points are inside from pixel color information
  - e.g., interior color, boundary color, fill color, current pixel color
  - Color the ones that are inside.
**Connected Area Boundary Fill Algorithm**

- For arbitrary closed areas
- Input:
  - Boundary Color (BC), Fill Color (FC)
  - (x,y) coordinates of seed point known to be inside
- Define a recursive BndFill(x,y,BC,FC) function:
  - If pixel(x,y) not set to BC or FC, then set to FC
  - Call BndFill() recursively for neighboring points

- To be able to implement this, need an inquire function
- e.g., Windows GetPixel(x,y)
  - returns color of pixel at (x,y)
The BndFill() Function

BndFill(x,y,BC,FC)
{
    color = GetPixel(x,y)
    if ( (color != BC) && (color != FC) )
    {
        SetPixel(x,y,FC);
        BndFill(x+1,y,BC,FC);  BndFill(x,y+1,BC,FC);
        BndFill(x-1,y,BC,FC);  BndFill(x,y-1,BC,FC);
    }
}

This would be called by code like:
BndFill(50,100,5,8);  // 5,8 are colors
– Windows GDI: colors are COLORREFs
– RGB() macro could be used

As given, only works with 4-connected regions
Boundary must be of a single color
– Could have multiple interior colors

A 4-connected Region

An 8-connected Region
Flood Fill Algorithm

- A variation Boundary Fill
- Fill area identified by the interior color
  - Instead of boundary color
  - Must have a single interior color
- Good for single colored area with multicolor border

Ups & Downs of Boundary / Flood Fill

- Big Up: Can be used for arbitrary areas!
- BUT-- Deep Recursion so:
  - Uses enormous amounts of stack space
    - (Adjust stack size before building in Windows!)
- Also very slow since:
  - Extensive pushing/popping of stack
  - Pixels may be visited more than once
  - GetPixel() & SetPixel() called for each pixel
    - 2 accesses to frame buffer for each pixel plotted
Adjusting Stack Size in VC++

- ‘Project’ on Main Menu
  - Properties
    - Configuration Properties
      - Linker
        System
        Stack Reserve Size: perhaps 10000000
        Stack Commit Size: perhaps 8000000

Scanline Polygon Fill Algorithm

- Look at individual scan lines
- Compute intersection points with polygon edges
- Fill between alternate pairs of intersection points
More specifically:

- For each scanline spanning the polygon:
  - Find intersection points with all edges the current scanline cuts
  - Sort intersection points by increasing x
  - Turn on all pixels between alternate pairs of intersection points

- But--
  - There may be a problem with intersection points that are polygon vertices
Vertex intersection points that are not local max or min must be preprocessed!

**Dealing With Vertex Intersection Points**

### Preprocessing non-max/min intersection points

- Move lower endpoint of upper edge up by one pixel
- i.e., \( y \leftarrow y + 1 \)
- What about \( x \)?
  - \( m = \frac{y}{x} \), so \( \frac{x}{m} = \frac{1}{y} \)
  - But \( y = 1 \), so:
  - \( x \leftarrow x + \frac{1}{m} \)
Active Edge

- A polygon edge intersected by the current scanline
- As polygon is scanned, edges will become active and inactive.
- Criterion for activating an edge:
  \[ y_{sl} = y_{min} \text{ of the edge} \]
  (Here \( y_{sl} \) = \( y \) of current scanline)
- Criterion for deactivating an edge:
  \[ y_{sl} = y_{max} \text{ of the edge} \]
Vertical & Horizontal Coherence

Moving from one scanline to next:

- $y = y + 1$
- If edge remains active, new intersection point coordinates will be:
  - $y_{\text{new}} = y_{\text{old}} + 1$
  - $x_{\text{new}} = x_{\text{old}} + \frac{1}{m}$
  
  ($\frac{1}{m} = \text{inverse slope of edge}$)

Scanline Polygon Fill Algorithm Input

- List of polygon vertices $(x_i, y_i)$
Scanline Polygon Fill Algorithm
Data Structures

1. Edge table:
   – For each edge: edge #, ymin, ymax, x, 1/m

2. Activation Table:
   – (y, edge number activated at y)
     • Provides edge(s) activated for each new scanline
     • Constructed by doing a "bin" or "bucket" sort

3. Active Edge List (AEL):
   – Active edge numbers sorted on x
     • A dynamic data structure

Bin Sort for Activation Table

<table>
<thead>
<tr>
<th>y</th>
<th>activated edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1 2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>0 4</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
</tbody>
</table>

Edge Table

<table>
<thead>
<tr>
<th>e</th>
<th>ymin</th>
<th>ymax</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>10</td>
</tr>
</tbody>
</table>
Scanline Polygon Fill Algorithm

1. Set up edge table from vertex list; determine range of scanlines spanning polygon (miny, maxy)
2. Preprocess edges with nonlocal max/min endpoints
3. Set up activation table (bin sort)
4. For each scanline spanned by polygon:
   - Add new active edges to AEL using activation table
   - Sort active edge list on x
   - Fill between alternate pairs of points (x,y) in order of sorted active edges
   - For each edge e in active edge list:
     If (y != ymax[e]) Compute & store new x (x+=1/m)
     Else Delete edge e from the active edge list

Scanline Polygon Fill Algorithm Example

poly={1,1, 2,5, 5,4, 0,7, 10,4, 10,2, 1,1}
### Scanline Poly Fill Alg. (with example Data)

**Edge Table (As Algorithm Executes)**

<table>
<thead>
<tr>
<th>Edge</th>
<th>1/m</th>
<th>ymax</th>
<th>ymin</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1/4</td>
<td>5</td>
<td>1</td>
<td>1, 1.25, 1.5, 1.75, 2</td>
</tr>
<tr>
<td>1</td>
<td>-3</td>
<td>5</td>
<td>4</td>
<td>5, 2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>4</td>
<td>5, 6, 7, 8</td>
</tr>
<tr>
<td>3</td>
<td>-2/3</td>
<td>7</td>
<td>5</td>
<td>9.33, 8.67, 8</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>10, 10</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1, 10</td>
</tr>
</tbody>
</table>

**Active Edge List (As it develops)**

<table>
<thead>
<tr>
<th>y</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Edges</td>
<td>0,5</td>
<td>0,5</td>
<td>0,4</td>
<td>0,1,2,4</td>
<td>0,1,2,3</td>
<td>2,3</td>
<td>2,3</td>
</tr>
<tr>
<td>Fill between</td>
<td>1-1</td>
<td>1-10</td>
<td>2-10</td>
<td>2-5,5-10</td>
<td>2-2,6-9</td>
<td>7-9</td>
<td>8-8</td>
</tr>
</tbody>
</table>