Rasterization

Raster Display

- The screen is a discrete grid of elements called **pixels**
- Shapes drawn by setting some pixels “on”

Rasterization

- How do we draw geometric primitives?
  - Convert from geometric definition to pixels
  - *rasterization* = selecting the pixels
- Will be done frequently
  - must be fast:
    - use integer arithmetic
    - use addition instead of multiplication

Terminology

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

Terminology

- **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
Today

• Drawing lines

• Filling polygons

**Naïve Algorithm for Lines**

- Line definition: \( ax + by + c = 0 \)
- Also expressed as: \( y(x) = mx + g \)
  - \( m = \text{slope} \)
  - \( g = y(0) \)

for \( x = x_{\text{min}} \) to \( x_{\text{max}} \)

\[
\begin{align*}
y &= mx + g \\
\text{light pixel } (x, y)
\end{align*}
\]

**Slope Dependency**

- Only works with \(-1 \leq m \leq 1\):

\[
\begin{align*}
m &= 3 \\
\text{Extend by symmetry for } m > 1
\end{align*}
\]

Copyright
C. Gotsman, G. Elber, M. Ben-Chen
Computer Science Dept., Technion
Problems

- 2 floating-point operations per pixel
- Improvement:
  \[ y = m \times x_{\min} + g \]
  for \( x_{\min} \to x_{\max} \)
  \[ y = y_{\min} \]
  light pixel \((x, y)\)
end
- Still 1 floating-point operation per pixel
- Compute in floats, pixels in integers

Bresenham Algorithm: Idea

- At each step, choice between 2 pixels \((0 \leq m \leq 1)\)

Bresenham Algorithm

- Need a criterion to choose
- Distance between line and center of pixel:
  the error associated with this pixel
\[ y(x) = mx + g \]

Special Cases

Choose between \(E\) and \(NE\) by sign of \(e = e_1 - e_2\)

\[ y = y_{\min} \]
for \(x_{\min} \to x_{\max}\)
- if \(e < 0\)
  // \(E\)
  update \(e\)  \(e_1 = e_1 + m, e_2 = e_2 - m\)
else
  \(y++\) // \(NE\)
  update \(e\)  \(e_1 = e_1 + m - 1, e_2 = e_2 - (m - 1)\)
light pixel \((x, y)\)
Bresenham Algorithm

\[
y = y_{\text{min}} \\
\text{for } x = x_{\text{min}} \text{ to } x_{\text{max}} \\
\text{if } e < 0 \\
\quad // E \\
\quad e += 2m \\
\text{else} \\
\quad y++ // \text{NE} \\
\quad e += 2m-2 \\
\text{light pixel}(x, y)
\]

Bresenham Algorithm

// Initialize e 
// \( e_1 = m, e_2 = 1 - m \) 
\( e = 2m-1 \) 
\( y = y_{\text{min}} \) 
\text{for } x = x_{\text{min}} \text{ to } x_{\text{max}} \\
\text{if } e < 0 \\
\quad // E \\
\quad e += 2m \\
\text{else} \\
\quad y++ // \text{NE} \\
\quad e += 2m-2 \\
\text{light pixel}(x, y)

Bresenham Algorithm in integers

\[
e = 2m-1 \\
y = y_{\text{min}} \\
\text{for } x = x_{\text{min}} \text{ to } x_{\text{max}} \\
\quad m = \frac{\Delta y}{\Delta x} = \frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \\
\text{define new variable} \quad d = e \Delta x \\
\text{if } e < 0 \\
\quad // E \\
\quad e += 2m \\
\text{else} \\
\quad y++ // \text{NE} \\
\quad e += 2m-2 \\
\text{light pixel}(x, y)
\]

Bresenham Algorithm in integers

\[
d = 2\Delta y - \Delta x \\
y = y_{\text{min}} \\
\text{for } x = x_{\text{min}} \text{ to } x_{\text{max}} \\
\text{if } d < 0 \\
\quad // E \\
\quad d += 2\Delta y \\
\text{else} \\
\quad y++ // \text{NE} \\
\quad d += 2\Delta y - 2\Delta x \\
\text{light pixel}(x, y)
\]

Generalizations

- Circles
- Other algebraic curves
- Line intensity
- Line thickness
- Anti-aliasing

Copyright

C. Gotsman, G. Elber, M. Ben-Chen
Computer Science Dept., Technion

Jacobs Technion-Cornell Institute
Polygon Fill

The Problem

• Problem:
  – Given a closed simple 2D polygon, fill its interior with specified color on graphics display

• Solutions:
  – Flood fill
  – Scan conversion

Flood Fill Algorithm

• Let \( P \) be a polygon whose boundary is already drawn
• Let \( C \) be the color to fill the polygon
• Let \( p = (x, y) \in P \) be a point inside \( P \)

\[
\text{Flood Fill}(\text{Polygon } P, \text{ int } x, \text{ int } y, \text{ Color } C)
\]

if not ((\(x, y\) in \(P\)) or (\(x, y\) on \(P\)))
begin
  PlotPixel(\(x, y, C\));
  FloodFill(\(P, x + 1, y, C\));
  FloodFill(\(P, x, y + 1, C\));
  FloodFill(\(P, x, y - 1, C\));
  FloodFill(\(P, x - 1, y, C\));
end ;

Basic Scan Conversion Algorithm

• Let \( P \) be a polygon with \( n \) vertices \( v_0 \) to \( v_{n-1} \) \((v_n = v_0)\)
• Let \( C \) be the color
• Each intersection of straight line with boundary moves in/out the polygon
• Detect (and set) pixels inside the polygon boundary

Basic Scan Conversion

\[
\text{Scan Convert}(\text{Polygon } P, \text{ Color } C)
\]

for \( y \) := 0 to ScreenYMax do
  \( I \leftarrow \text{Points of intersections of edges of } P \text{ with line } Y = y \);
  Sort \( I \) in increasing \( X \) order and
  Fill with color \( C \) alternating segments ;
end
Special Cases

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</td>
</tr>
<tr>
<td>Requires GetPixelVal system call</td>
<td>screen space</td>
</tr>
<tr>
<td>Requires a seed point</td>
<td>Device independent</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>

Copyright
C. Gotsman, G. Elber, M. Ben-Chen
Computer Science Dept., Technion