CS-184: Computer Graphics

Lecture #9: Scan Conversion

Prof. James O'Brien
University of California, Berkeley
Today

• 2D Scan Conversion
  • Drawing Lines
  • Drawing Curves
  • Filled Polygons
  • Filling Algorithms
Drawing a Line

• Basically, its easy... but for the details
• Lines are a basic primitive that needs to be done well...
Drawing a Line

• Basically, it's easy... but for the details
• Lines are a basic primitive that needs to be done well...

Drawing a Line

$p_1 = (x_1, y_1)$

$p_2 = (x_2, y_2)$
Drawing a Line
<table>
<thead>
<tr>
<th>Drawing a Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Some things to consider</td>
</tr>
<tr>
<td>• How thick are lines?</td>
</tr>
<tr>
<td>• How should they join up?</td>
</tr>
<tr>
<td>• Which pixels are the right ones?</td>
</tr>
</tbody>
</table>

For example:
Drawing a Line

Inclusive Endpoints

$\mathbf{p}_1 = (x_1, y_1)$

$\mathbf{p}_2 = (x_2, y_2)$
Drawing a Line

\[ y = m \cdot x + b, x \in [x_1, x_2] \]

\[ m = \frac{y_2 - y_1}{x_2 - x_1} \]

\[ b = y_1 - m \cdot x_1 \]
```
Δx = 1
Δy = m \cdot Δx

x = x1
y = y1
while (x <= x2)
   plot(x, y)
   x++
   y += Dy
```
Drawing a Line

\[ \Delta x = 1 \]
\[ \Delta y = m \cdot \Delta x \]

After rounding
Drawing a Line

\[ \Delta x = 1 \]
\[ \Delta y = m \cdot \Delta x \]
\[ y += \Delta y \]

Accumulation of roundoff errors

How slow is float-to-int conversion?
Drawing a Line

\[ |m| \leq 1 \quad |m| > 1 \]

\{ Gap \}
void drawLine(int x1, x2, int y1, y2)

float m = \( \frac{y_2 - y_1}{x_2 - x_1} \)
int x = x1
float y = y1

while (x <= x2)

    setPixel(x, round(y), PIXEL_ON)

    x += 1
    y += m

Not exact math
Accumulates errors
# Drawing a Line

```c
void drawLine(int x1, x2, int y1, y2)
{
    float m = (y2 - y1) / (x2 - x1);
    int x = x1;
    int y = y1;
    float e = 0.0;

    while (x <= x2)
    {
        setPixel(x, y, PIXEL_ON);
        x += 1;
        e += m;
        if (e >= 0.5)
        {
            y += 1;
            e -= 1.0;
        }
    }
}
```

No more rounding
void drawLine(int x1, x2, int y1, y2)

    int x = x1
    int y = y1
    float e = -0.5

    while (x <= x2)
        setPixel(x, y, PIXEL_ON)
        x += 1
        e += float(y2-y1)/(x2-x1)
        if (e >= 0.0)
            y += 1
            e -= 1.0
void drawLine-Error4(int x1,x2, int y1,y2)

    int x = x1
    int y = y1
    float e = -0.5*(x2-x1)          // was -0.5

    while (x <= x2)

        setPixel(x,y,PIXEL_ON)

        x += 1
        e += y2-y1                  // was /(x2-x1)
        if (e >= 0.0)               // no change
            y+=1
        e-=(x2-x1)                // was 1.0
## Drawing a Line

```c
void drawLine(int x1, int x2, int y1, int y2)
{
    int x = x1;
    int y = y1;
    int e = -((x2-x1)); // removed *0.5

    while (x <= x2)
    {
        setPixel(x, y, PIXEL_ON);
        x += 1;
        e += 2*(y2-y1); // added 2*
        if (e >= 0.0) // no change
        {
            y += 1;
            e -= 2*(x2-x1); // added 2*
        }
    }
}
```
Drawing a Line

```c
void drawLine_Bresenham(int x1, x2, int y1, y2) {
    int x = x1;
    int y = y1;
    int e = -(x2 - x1);

    while (x <= x2) {
        setPixel(x, y, PIXEL_ON);
        x += 1;
        e += 2 * (y2 - y1);
        if (e >= 0.0) {
            y += 1;
            e -= 2 * (x2 - x1);
        }
    }
}
```

Faster

Not wrong

\[ |m| \leq 1 \]

\[ x_1 \leq x_2 \]
Drawing Curves

\[ y = f(x) \]

Only one value of \( y \) for each value of \( x \)...
## Drawing Curves

- Parametric curves
  - Both $x$ and $y$ are a function of some third parameter

\[
x = f(u) \\
y = f(u) \\
x = f(u) \\
u \in [u_0 \ldots u_1]
\]
Drawing Curves

\[ x = f(u) \quad u \in [u_0 \ldots u_1] \]
Drawing Curves

- Draw curves by drawing line segments
  - Must take care in computing end points for lines
  - How long should each line segment be?

\[ x = f(u) \quad u \in [u_0 \ldots u_1] \]
Drawing Curves

- Draw curves by drawing line segments
- Must take care in computing end points for lines
- How long should each line segment be?
- Variable spaced points

\[ \mathbf{x} = \mathbf{f}(u) \quad u \in [u_0 \ldots u_1] \]
Drawing Curves

- Midpoint-test subdivision
Drawing Curves

- Midpoint-test subdivision

\[ |f(u_{mid}) - 1(0.5)| \]
Drawing Curves

- Midpoint-test subdivision

\[ |f(u_{mid}) - 1(0.5)| \]
Drawing Curves

- Midpoint-test subdivision
  - Not perfect
  - We need more information for a guarantee...

\[ |f(u_{mid}) - 1(0.5)| \]
Filled Polygons
Filled Polygons
Filled Polygons

Toggle inside/outside flag to "INSIDE"
Filled Polygons

Toggle inside/outside flag to "OUTSIDE"
What happens at these locations?
If we count ONCE...
Filled Polygons

If we count TWICE...
Filled Polygons

Treat (scan \( y = \text{vertex } y \)) as (scan \( y > \text{vertex } y \))
Filled Polygons

Horizontal edges
Filled Polygons

Horizontal edges
Filled Polygons

• “Equality Removal” applies to all vertices
• Both $x$ and $y$ coordinates
Filled Polygons

- Final result:
Who does this pixel belong to?

Filled Polygons
<table>
<thead>
<tr>
<th>Drawing a Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>• How thick?</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>• Ends?</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Butt</td>
</tr>
<tr>
<td>Round</td>
</tr>
<tr>
<td>Square</td>
</tr>
<tr>
<td>Drawing a Line</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>• Joining?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ugly</th>
<th>Bevel</th>
<th>Round</th>
<th>Miter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inside/Outside Testing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Polygon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-exterior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-zero winding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimize for Triangles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Split triangle into two parts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Two edges per part</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Y-span is monotonic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- For each row</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Interpolate span</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Interpolate barycentric coordinates</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Flood Fill
Flood Fill