Computer Graphics

Z Sweedyk
Lecture 9

Outline

- Today: Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

Graphics Pipeline 5

1. Build Primitives (Model Coordinates)
2. Assemble Scene (World Coordinates)
3. Clip & Project into 2D (Projected Coordinates)
4. Normalize (Normalized Coordinates)
5. Scan Convert with Hidden Surface Removal

Line Segments

- Scan converting line segments
  - Naïve algorithm
  - Midpoint algorithm
  - Bresenham's algorithm

Scan Converting Line Segments

Which pixels should be on?

1-pixel wide lines
1-pixel wide lines: $m \geq 0$

$0 \leq m < 1$: one pixel per row

$1 < m$: one pixel per column

1-pixel wide lines: $m < 0$

$-1 \leq m < 0$: one pixel per column

$1 < m$: one pixel per column

Scan Conversion

• Input: Endpoint Pixels

• Output: Pixels to turn on for a 1-pixel wide line segment

For today...

• A pixel is a little square!

Points and Pixels

Pixel (1,3)

Point (1.5, .5)

Pixel (0,0)

Equivalent Ideal Lines Segments

If that’s not good enough you need better resolution!!!!!
Claim

All we have to do is solve the scan conversion problem for the special case where
1. 0 ≤ m ≤ 1
2. x₀ = y₀ = 0

We’ll prove this later … first we’ll devise an algorithm for the special case.

Line Segments

- Scan converting line segments
  - Naive algorithm (special case)
  - Midpoint algorithm
  - Bresenham’s algorithm

Naïve Algorithm

(Case: x₀ = y₀ = 0, 0 ≤ m ≤ 1)

SpecialCaseNaive(int x₀, int y₀)

int current_x = 0
float current_y = 0,
float m = (float) y₀ / (float) x₀

while (current_x ≤ x₁)

  DrawPixel(current_x, round(current_y))

  current_x += 1, current_y += m

Line Segments

- Scan converting line segments
  - Naïve algorithm
  - Midpoint algorithm
  - Bresenham’s algorithm
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing
Let's try to do better!

• Suppose we've just drawn the \((i,j)\) pixel. How do we choose the next pixel?

How do we choose next pixel?

\(y = mx, 0 \leq m \leq 1\)

• The options are E and NE
  • If midpoint lies below the ideal line: go NE
  • Else: go E

Line in the plane (through the origin)

\((x,y): y > mx\)

\((x,y): y = mx\)

\((x,y): y < mx\)

The Test

\(u = i + 1\)

\(v = j + 1/2\)

Ideal line: \(y = mx^{1/2}\)

If \(v < mu\) then go NE
Else go E

Using Midpoint Test (Case: \(x_0 = y_0 = 0\) and \(0 \leq m \leq 1\))

\(m = y_1 / x_1\)

\([=0, \neq]\)

while \(i < x_1\)

write-pixel \((i,j)\)

if \(j + 1/2 < m(i + 1)\)

\(i += 1, j += 1\) // go NE

else \(i += 1\) // go E

Modified Test

Is \(j + 1/2 < m(i + 1)\)?

Is \(j + 1/2 < (y_1 / x_1) \cdot (i + 1)\)?

Is \(x_1(i + 1) + 1 \leq 2y_1(i + 1)\)?

Important: \(x_1\) and \(y_1\) are integers!
**Midpoint Algorithm**  
(Case: \( x_0=y_0=0 \) and \( 0 \leq m \leq 1 \))

```
SpecialCaseMidpoint(x_1, y_1)

i=0, j=0
while current_i < \( x_1 \)
    writepixel(I, j)
    if \( x_1(2j+1) < 2y_1(i+1) \)
        i+=1, j+=1  // go NE
    else i+=1    // go E
```

**Endpoints (0,0) & (9,4):**

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

If \( 9(2j+1) < 8(i+1) \)  
Go NE  
Else  
Go E

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Line Segments**

- Scan converting line segments
  - Naive algorithm
  - Midpoint algorithm
  - Bresenham's algorithm
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing

**Sleight of hand**

```
if \( x_1(2j+1) < 2y_1(i+1) \)
    go NE
else go E
```

\[ d = x_1(2j+1) - 2y_1(i+1) \]  
if \( d < 0 \)  
    go NE  
else go E
Endpoints (0,0) & (9,4):
\[ d = 9(2j+1) - 8(i+1) \]

If \( d < 0 \)
Go NE
Else Go E

Can we compute \( d \) incrementally?

Special Case Bresenham’s
\((x_0, y_0 = 0 \text{ and } 0 \leq m < 1)\)

\[
\text{SpecialCaseBresenham's}(x, y) \\
\text{i}=0, \text{j}=0 \\
d = x - 2y_1 \\
\text{while } i < x_1 \\
\text{writePixel(i, j)} \\
\text{if } d < 0 \\
\text{i}+=1, \text{j}+=1, \quad d += (2x_1y_1) \\
\text{else} \\
\text{i}+=1, \quad d -= 2y_1 \\
\text{Do as addition}
\]

Claim

All we have to do is solve the problem for the special case where
1. \( 0 \leq m < 1 \)
2. \( x_0 = y_0 = 0 \)

Now we’ll prove this claim

To solve general case:

Translate \((x_0, y_0)\)
endpoint to origin
To solve general case:

If $2^{\text{nd}}/3^{\text{rd}}$ quadrant reflect to $1^{\text{st}}/4^{\text{th}}$

To solve general case:

If $4^{\text{th}}$ quadrant, reflect to first

To solve general case:

If $m \geq 1$, reflect so $0 < m < 1$

To solve general case:

Scan convert with special case algorithm

To solve general case:

Undo $3^{\text{rd}}$ reflection as necessary

To solve general case:

Undo $2^{\text{nd}}$ reflection as necessary
To solve general case:

Undo 1st reflection as necessary

To solve general case:

Undo translation

Example

\((-3,-2), (0,3)\)  \(\rightarrow\)  \((0,0),(3,5)\)  \(\rightarrow\)  \((0,0),(5,3)\)

On Pixels:

\((-3,-2), (-2,-1), (-2,0), (-1,1), (-1,2), (0,3)\)

\((0,0), (1,1), (1,2), (2,3), (2,4), (3,5)\)

\((0,0), (1,1), (2,1), (3,2), (4,2), (5,3)\)

Outline

- Today: Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

Polygon: \(p_1, p_2, p_3, p_4, p_5\)

Polygons: \(p_1, p_3, p_5, p_2, p_4\)

Order Matters
Polygon: Scan Conversion

Polygon \( (p_1, \ldots, p_n) \)
For \( i = 1 \) to \( n-1 \)
DrawLine \( (p_i, p_{i+1}) \)
DrawLine \( (p_n, p_1) \)

Outline

- Today: Scan conversion
  - Lines
  - Polygons
  - Filled polygons
- Next time: Hidden surface removal

Filled Polygon
Which pixels should be on?

Here we get the same size!

Center Claims
Tie Breaker 1: Entering Owns

Tie Breaker 2: Up wins

Exercise

Scan Line Algorithm
1. Compute intersections
2. Order by x-coordinate
3. Use odd-even test to turn on pixels

Odd-Even Test
Odd = in
Even = out

Odd-Even Test
May not be what you’re looking for … but it’s easy to implement
Scan Line Algorithm
How to implement efficiently?

1. Compute intersections
2. Order by x-coordinate
3. Use odd-even test to turn on pixels

Key ideas
- Let $S$ be the set of line segments that intersect scan line $i$. The set of lines that intersect scan line $i+1$ is: $S + \text{new-done}$
- Suppose that line $L=(m,b)$ intersects scan line $i$ at $(x,i)$. If $L$ intersects scan line $i+1$ it does so at: $(x+1/m, i+1)$

Data Structures
- Edge Table
- Active Edge Table

Edge Table (ET)

<table>
<thead>
<tr>
<th>Yval</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>$L_2, L_3$</td>
</tr>
<tr>
<td>3</td>
<td>$L_4$</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>$L_5, L_6$</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Line Segments “beginning” at scan line 3

Example: Edge Table

<table>
<thead>
<tr>
<th>Yval</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>$L_2, L_3$</td>
</tr>
<tr>
<td>3</td>
<td>$L_4$</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>$L_5$</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

Record with info about $L_2$
Active Edge Table (AET)

List of the Line segments intersecting current scan line

Example: Active Edge Table

Scan Line Algorithm

Build ET
Yval=1
Initialize AET=∅
Repeat until ET and AET are empty:
Yval ++
Update info on line segments
Add ET[Yval] to AET
Remove lines from AET that are "done"
Sort lines in AET by x-intercept at y=Yval
Choose pixels based on odd-even test

Scan Line Algorithm

Build ET
Yval=1
Initialize AET=∅
Repeat until ET and AET are empty:
Yval ++
Update info on line segments
Add ET[Yval] to AET
Remove lines from AET that are "done"
Sort lines in AET by x-intercept at y=Yval
Choose pixels based on odd-even test

Line Record

Field 1: ymax

<table>
<thead>
<tr>
<th>line</th>
<th>ymax</th>
</tr>
</thead>
<tbody>
<tr>
<td>L₀</td>
<td>7</td>
</tr>
<tr>
<td>L₁</td>
<td>3</td>
</tr>
<tr>
<td>L₂</td>
<td>7</td>
</tr>
<tr>
<td>L₃</td>
<td>6</td>
</tr>
<tr>
<td>L₄</td>
<td>6</td>
</tr>
</tbody>
</table>
Scan Line Algorithm

Build ET
Yval = 1
Initialize AET = \emptyset
Repeat until ET and AET are empty:
  Yval ++
  Update info on line segments
  Add ET[Yval] to AET
  Remove lines from AET when Yval = ymax
  Sort lines in AET by x-intercept at y = Yval
  Choose pixels based on odd-even test

Line Record: Current Yval = 2

Field 2. Xval

Line Record: Current Yval = 2

Field 3. 1/m

Line Record: Current Yval = 2

Field 3. 1/m
Scan Line Algorithm

Build ET
Yval = 1
Initialize AET = ϕ
Repeat until ET and AET are empty:
Yval ++
Increment xval by 1/m for each line in AET
Add ET[yval] to AET
Remove lines from AET when Yval = ymax
Sort lines in AET by x-intercept at y=yval
Choose pixels based on odd-even test

Initialize Line Records

Initialize Edge Table

Initialize Active Edge Table

Yval = 0

Yval = 1

(sorted) AET
Turn on (i, 1) where: 1 ≤ i ≤ 1/m
Turn on \((i,2)\) where \(4/3 \leq i < 7/2\)

Turn on \((i,3)\) where \(5/3 \leq i < 6\)

Turn on \((i,4)\) where \(2 \leq i < 3\) or \(3 \leq i < 6\)

Turn on \((i,5)\) where \(7/3 \leq i < 3\) and \(9/2 \leq i < 6\)

Turn on \((i,6)\) where \(8/3 \leq i < 3\)
Claims

- AET always contains an even number of lines
- The algorithm implements the correct tie-breaking rules