Computer Graphics

Z Sweedyk
Lecture 2
9/4/00

Line Segments

• Scan converting line segments
  – Naïve algorithm
  – Midpoint algorithm
  – Bresenham’s algorithm
• Clipping line segments (intro)
  – Scissoring
  – Analytical clipping

Line: Set of points \((x,y)\) satisfying \(y = mx + b\)

\[
m = \frac{\Delta y}{\Delta x}
\]

Line through \((x_0,y_0)\) and \((x_1,y_1)\)

\[
m = \frac{y_1 - y_0}{x_1 - x_0}, \quad b = y_1 - mx_1
\]

Line segment with endpoints \((x_0,y_0)\) and \((x_1,y_1)\)

\[
m = \frac{y_1 - y_0}{x_1 - x_0}, \quad b = y_1 - mx_1
\]
Scan Converting Line Segments
Which pixels should be on?

1-pixel wide lines

1-pixel wide lines: m\geq 0

1-pixel wide lines: m < 0

Scan Conversion

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

Points and Pixels
**Points and Pixels**

Point (1,3)
Point (0,0)
Point (6.5,4.5)

**Equivalence Classes of Ideal Lines**

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

**Line Segments**

- **Scan converting line segments**
  - Naïve algorithm
  - Midpoint algorithm
  - Bresenham’s algorithm
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing

**Claim**

All we have to do is solve the scan conversion problem for the special case where

1. \(0 \leq m \leq 1\)
2. \(x_0 = y_0 = 0\)

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

**Ideal lines we’ll scan convert**

**Line Segments**

- **Scan converting line segments**
  - Naïve algorithm (special case)
  - Midpoint algorithm
  - Bresenham’s algorithm
- Clipping line segments
  - Scissoring
  - Analytical clipping
- Antialiasing
Endpoints: (0,0) and (9,4)

\[ y = mx, \quad m = \frac{4}{9} \]

### Special Case Line Algorithm 1

\[ x_0 = y_0 = 0, \quad 0 \leq m \leq 1 \]

```python
SpecialCaseLine1(int x1, int y1)
  int current_x = 0
  float m = (float) y1 / (float) x1
  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

• Choose pixel whose center is closest to the ideal line
How do we choose next pixel?

\[ y = mx, \quad 0 \leq m \leq 1 \]

Define \((u, v)\) to be the midpoint of the line segment between the centers of NE and E.

If \((u, v)\) lies below the ideal line: go NE
Else: go E

Line in the plane (through the origin)

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

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

The Test

\[
\begin{align*}
  u &= i + 1 \\
  v &= j + 1/2 \\
  \text{Ideal line: } y &= mx^{1/2} \\
  \text{If } v < mu & \text{ then go NE} \\
  \text{Else go E}
\end{align*}
\]

Special Case Line Algorithm 2

\((x_0, y_0) = (0, 0) \text{ and } 0 \leq m \leq 1\)

\[
\text{SpecialCaseLine2}(x_1, y_1) \\
i=0, j=0 \\
\text{while } i < x_1 \\
\text{write-pixel}(i, j) \\
\text{if } j + 1/2 < \left( y_1 / x_1 \right) \cdot (i+1) \\
\quad i+=1, j+=1 \quad \text{// go NE} \\
\text{else } i+=1 \quad \text{// go E}
\]

Modified Test

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

Is \( j + 1/2 \frac{1}{2} < \left( y_1 / x_1 \right) \cdot (i+1) \)?

Is \( x_i(2) + 1 \) \( < 2y_j(i+1) \)?

Important: \( x_i \) and \( y_j \) are integers!

Special Case Line Algorithm 3

\((x_0, y_0) = (0, 0) \text{ and } 0 \leq m \leq 1\)

\[
\text{SpecialCaseLine3}(x_1, y_1) \\
i=0, j=0 \\
\text{while } \text{current}_i < x_1 \\
\text{write-pixel}(i) \\
\text{if } x_i(2) + 1 < 2y_j(i+1) \\
\quad i+=1, j+=1 \quad \text{// go NE} \\
\text{else } i+=1 \quad \text{// go E}
\]
Endpoints (0,0) & (9,4): 

\[
\begin{align*}
\text{If } & \quad 9(2j + 1) < 8(i+1) \\
\text{Go NE} \\
\text{Else} & \quad \text{Go E}
\end{align*}
\]

Special Case Line Algorithm 3 
(x_0=y_0=0 and 0 \leq m \leq 1) 

\[
\begin{align*}
i = 0, j = 0 \\
\text{while } \text{current}_i < \infty, \\
\text{writepixel}(i) \\
\text{if } x_1(2j+1) < 2y_1(i+1) \\
\quad i += 1; j += 1 \quad \text{// go NE} \\
\text{else } i += 1 \quad \text{// go E}
\end{align*}
\]

Line Segments 

- Scan converting line segments 
  - Naïve algorithm 
  - Midpoint algorithm 
  - Bresenham’s algorithm 
- Clipping line segments 
  - Scissoring 
  - Analytical clipping 
- Antialiasing 

Sleight of hand 

\[
\begin{align*}
\text{if } x_1(2j+1) < 2y_1(i+1) \\
\quad \text{go NE} \\
\text{else} & \quad \text{go E}
\end{align*}
\]

\[
d = x_1(2j+1) - 2y_1(i+1) \quad \text{if } d < 0 \\
\text{go NE} \\
\text{else} & \quad \text{go E}
\]
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
\( (x_0, y_0) = (0, 0) \) and \( 0 \leq m \leq 1 \)

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 2nd/3rd quadrant reflect to 1st/4th
To solve general case:

- If in the 4th quadrant, reflect to the first quadrant.

To solve general case:

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

To solve general case:

- Scan convert with the special case algorithm.

To solve general case:

- Undo the 3rd reflection as necessary.

To solve general case:

- Undo the 2nd reflection as necessary.

To solve general case:

- Undo the 1st reflection as necessary.
To solve general case:

Example

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

On Pixels:
(-3, -2), (-2, -1), (-2, 0), (-1, 1), (-1, 2), (0, 3)
(0, 0), (1, 1), (2, 1), (3, 2), (4, 2), (5, 3)

Implementation

- Think it through
- Get special case working first
- Endpoint order MATTERS!

Endpoint Order

- Why it matters

Implementation??

- Modified special case Bresenham’s for different slopes/endpoint orders
- Reduction (in this case Bresenham’s should return an array of y-values)

Line Segments

- Scan converting line segments
  - Naïve algorithm
  - Midpoint algorithm
  - Bresenham’s algorithm
- Clipping line segments (intro)
  - Scissoring
  - Analytical clipping
Answer 1: Scissoring

- Compute pixels in world grid
- Then check chosen pixels against viewport boundaries

Problem: Inefficient

Answer 2: Analytical Clipping

- Compute intersection points
- Scan convert line segment between intersection points

But … a’ and b’ are not likely to have integer coordinates

1. We could go back to using floating point operations.
2. We could round the endpoint coordinates and scan convert.
3. We could do it the right way.

The right way

SpecialCaseBresenham’s(x₀,y₀,x₁,x₂,y₁,y₂)

m = y₁/x₁
i = x₁, j = round(m * x₁)
d = x₁(2j+1)-2y₁(i+1)
while i < x₂
    write_pixel(i,j)
    if d < 0
        i += 1, j += 1, d += 2(x₁y₁)
    else
        i += 1, d -= 2y₁
Answer 2: Analytical Clipping

• Compute intersection points
• Scan convert line segment between intersection points

Intersection of lines: \( y = mx \) and \( x = c \)

Or Intersection of lines: \( y = mx \) and \( y = d \)

Or

Etc.

Next time

• Clipping line segments concluded.
• Polygons