Scan Conversion
Lines & circles

by Brian Wyvill
With thanks to

Prof. Rich Riesenfeld (University of Utah)
for some slides

Dr. Jack Bresenham
for some interesting discussions
Implicit Vs. Parametric

Implicit Definition

\[ f(x,y) = x^2 + y^2 - r^2 = 0 \]

e.g. \( r = 1 \)
\[ f(0,0) = 0 + 0 - 1 < 0 \] inside
\[ f(0,0) = 1 + 1 - 1 > 0 \] outside

implies search space to find \( x,y \) to satisfy: \( f(x,y) = 0 \)

iso-surface: \( f(x,y) - c = 0 \)

Parametric Definition

\[ x = r \sin(\alpha) \]
\[ y = r \cos(\alpha) \]
\[ 0 \leq \alpha \leq 2\pi \]
Line Characterisations

- **Explicit:** \[ y = mx + B \]
- **Implicit:** \[ F(x, y) = ax + by + c = 0 \]
- **Constant slope:** \[ \frac{\Delta y}{\Delta x} = k \]
- **Constant derivative:** \[ f'(x) = k \]
**Line Characterisations**

- **Parametric:** \[ P(t) = (1-t)P_0 + tP_1 \]
  
  where,
  
  \[ P(0) = P_0 ; \quad P(1) = P_1 \]

- Intersection of 2 planes
- Shortest path between 2 points
- **Convex hull** of 2 discrete points
Discrete Line

- Lines vs. Line Segments
- What is a discrete line segment?
  - This is a relatively recent problem
  - How to generate a discrete line?
A Good Line

- No gaps in adjacent pixels
- Pixels close to ideal line
- Consistent choices; same pixels in same situations
A Good Line

- Smooth looking
- Even brightness in all orientations
- Same line for $P_0 P_1$ as for $P_1 P_0$
- Double pixels stacked up?
“Good” Discrete Line - 2

- Smooth looking
- Even brightness in all orientations
- Same line for $P_0 P_1$ as for $P_1 P_0$
- Double pixels stacked up?
Incremental Fn Eval

- Recall \( f(x_{i+1}) = f(x_i) + \Delta(x_i) \)

- Characteristics
  - Fast
  - Cumulative Error

- Need to define \( f(x_o) \)
Restricted Form

- Line segment in first octant with
  \[ 0 < m < 1 \]
- Let us proceed
Two Line Equations

- **Explicit:** \( y = mx + B \)
- **Implicit:** \( F(x, y) = ax + by + c = 0 \)

Define: \( dy = y_1 - y_0 \) \( dx = x_1 - x_0 \)

Hence, \( y = \left( \frac{dy}{dx} \right)x + B \)
Relating Explicit to Implicit Eq’s

Recall, \[
\frac{dy}{dx} x - y + B = 0
\]

Or, \[
(dy)x + (-dx)y + (dx)B = 0
\]

\[
\therefore F(x, y) = (dy)x + (-dx)y + (dx)B = 0
\]

where, \[
a = (dy); \quad b = -(dx); \quad c = B(dx)
\]
Investigate Sign of \( F \)

Verify that

\[
F(x, y) = \begin{cases} 
+ & \text{below line} \\
0 & \text{on line} \\
- & \text{above line}
\end{cases}
\]

Look at extreme values of \( y \)
Implicit Line Equation

\[ F(x, y) = ax + by + c = 0 \]

\[ a = dy \quad b = -dx \quad \text{and} \quad c = B(dx) \]

\[ dx = 5 \quad dy = 3 \quad B = 0 \]

E.g. line (0,0) to (5,3)

\[ f(3,3) = 3*3 - 5*3 = -6 \quad <0 \]

\[ f(4,2) = 3*4 - 5*2 = 2 \quad >0 \]
“Reasonable assumptions” have reduced the problem to making a binary choice at each pixel:

(Previous)  \[ \rightarrow \]  E (next)

 NE (next)
Decision Variable $d$ (logical)

Define a logical decision variable $d$

- linear in form
- incrementally updated (with addition)
- tells us whether to go $E$ or $NE$
The Picture

ideal line

\[ y = y_p + 1 \]
\[ x = x_p \]

previous

\[ M \quad \text{midpoint} \]

\[ Q \]

\[ NE \]

\[ E \]

\[ x = x_p + 1 \]
Bresenham’s Algorithm (Jack Bresenham 1965) (midpoint technique) Pittaway

Desired Line
Assume $0 \geq \text{slope} \geq 1$

Suppose we have chosen pixel P, which pixel should be next?

Since the slope is between 0 and 1 the next pixel will be either NE or E.
The Picture (again)

ideal line

\((x_p, y_p)\) previous

\((x_p + 1, y_p + 1)\)

\((x_p + 1, y_p + \frac{1}{2})\)

\((x_p + 1, y_p)\)
Observe the relationships

- Suppose $Q$ is above $M$, as before.
- Then $F(M) > 0$, $M$ is below the line
- So, $F(M) > 0$ means line is above $M$,
- Need to move NE, increase $y$ value
Observe the relationships

- Suppose \( Q \) is below \( M \).
- Then \( F(M) < 0 \), implies \( M \) is above the line

- So, \( F(M) < 0 \), means line is below \( M \),
- Need to move to \( E \);

\( don’t \) increase \( y \)
How to choose B or C, is Q closer to NE or E?

Slope = \( \frac{dy}{dx} \)

We need to compute \( F(M) = F(xp+1, yp+\frac{1}{2}) \)

Define discriminant \( d = F(xp+1, yp+\frac{1}{2}) \)

\( d_M = a.(xp+1) + b.(yp+\frac{1}{2}) + c \)

We wish to make an incremental algorithm:

- **If** \( d > 0 \) choose pixel NE, find M2

  \[
  \text{new } d_{M2} = F(xp+2, yp+\frac{3}{2}) = a.(xp+2) + b.(yp+\frac{3}{2}) + c
  \]

- **If** \( d < 0 \) choose pixel E, find M1

  \[
  \text{new } d_{M1} = F(xp+2, yp+\frac{1}{2}) = a.(xp+2) + b.(yp+\frac{1}{2}) + c
  \]
Calculating the discriminant \( d \)

If \( d < 0 \) choose pixel \( E \),

\[
\text{new } d_{M1} = a.(xp+2) + b.(yp+\left\lfloor \frac{1}{2} \right\rfloor) + c
\]

\[
\text{old } d_M = a.(xp+1) + b.(yp+\left\lfloor \frac{1}{2} \right\rfloor) + c
\]

Subtracting: \( d_{M1} = d_M + a \)

but \( a = dy \) known as \( \Delta E \)

If \( d > 0 \) choose pixel \( NE \), find \( M2 \)

\[
\text{new } d_{M2} = F(xp+2, yp+\left\lfloor \frac{3}{2} \right\rfloor) = a.(xp+2) + b.(yp+\left\lfloor \frac{3}{2} \right\rfloor) + c
\]

Subtracting \( d_M \) gives:

\[
d_{M2} = d_M + a + b
\]

\[
\Delta NE = a + b = dy - dx
\]

Initial value of \( d \)

First pixel is endpoint \( xo, yo \)

First midpoint is \((xo+1, yo+\left\lfloor \frac{1}{2} \right\rfloor)\)

Find \( d \) by choosing \( E \) or \( NE \):

\[
F(xo+1, yo+\left\lfloor \frac{1}{2} \right\rfloor) = a(xo+1) + b(yo+\left\lfloor \frac{1}{2} \right\rfloor) + c
\]

\[
= axo + byo + c + a + b/2
\]

\[
= F(xo, yo) + a + b/2
\]

Since \( F(xo, yo) \) is on the line \( F(xo, yo) = 0 \)

Initial value of \( d = a + b/2 = dy - dx/2 \)

Since we don’t want the \( \frac{1}{2} \) in the expression multiply by 2 which has no effect on the sign of \( d \):

Initial \( d = 2dy - dx \)

Increment of \( \Delta E = 2dy \)

Increment of \( \Delta NE = 2(dy - dx) \)
void midpoint-line(int xo, int yo, int xe, int ye, int value)
{
    int dx = xe - xo;
    int dy = ye - yo;
    int d = 2 * dy - dx;
    int incrE = 2 * dy;
    int incrNE = 2 * (dy - dx);
    int x = xo;
    int y = yo;
    writePixel(xo,yo, value);

    while (x<xe) {
        if (d<=0) {  /* choose E */
            d += incrE;
            x++;
        } else {    /* choose NE */
            d += incrNE;
            x++;y++;
        }
        writePixel(x,y, value);
    } /* end while */
}
/* end midpoint line */
Example

- Line end points:

\[(x_0, y_0) = (5, 8); \hspace{1cm} (x_1, y_1) = (9, 11)\]

- Deltas:

\[dx = 4; \hspace{0.5cm} dy = 3\]
Meeting Bresenham Criteria

• $m = 0; \quad m = 1 \implies \text{trivial cases}$

• $(x_0, y_0) \neq (0, 0) \implies \text{translate}$

• $0 > m > -1 \implies \text{flip about } x\text{-axis}$

• $m > 1 \implies \text{flip about } x = y$
Case 1: Translate to Origin

- \( (x_0, y_0) \)
- \( (x'_0, y'_0) \)
- \( (x'_1, y'_1) \)
- \( (x_1, y_1) \)
Case 0: Trivial Situations

- \( m = 0 \implies \text{horizontal line} \)
- \( m = 1 \implies \text{line } y = x \)

Do not need Bresenham
Case 1: Translate to Origin

- Move \((x_0, y_0)\) to the origin

\[
(x'_0, y'_0) = (0,0);
\]

\[
(x'_1, y'_1) = (x_1 - x_0, y_1 - y_0)
\]

- Need only consider lines emanating from the origin.
Case 2: $m > -1$  
Flip about $x$-axis

\[ (x_0, y_0) \rightarrow (x_1, y_1) \]

\[ (x'_0, y'_0) \rightarrow (x'_1, y'_1) \]
Case 2: Flip about x-axis

- Suppose, $0 > m > -1$,
- Flip about x-axis ($y' = -y$):

$$(x_0', y_0') = (x_0, -y_0);$$

$$(x_1', y_1') = (x_1, -y_1)$$
How do slopes relate?

\[ m = \frac{y_1 - y_0}{x_1 - x_0}; \quad \{ \begin{array}{l} \text{by definition} \\
\end{array} \] 

\[ m' = \frac{y'_1 - y'_0}{x_1 - x_0} \]

Since \( y'_i = -y_i \), \( m' = \frac{-y_1 - (-y_0)}{x_1 - x_0} \)
How do slopes relate?

\[ m' = -\frac{(y_1 - y_0)}{x_1 - x_0} \]

i.e.,

\[ m' = -m \]

\[ \therefore \ 0 > m > -1 \ \Rightarrow \ 0 < m' < 1 \]
Case 3: $m > 1$: Flip about line $y = x$
Case 3: Flip about line $y = x$

$$y = mx + B,$$

swap $x \leftrightarrow y$ and prime them, 

$$x' = my' + B,$$

$$my' = x' - B.$$
**Case 3: \( m' = ? \)**

\[
y' = \left( \frac{1}{m} \right) x' - B,
\]

\[
\therefore m' = \left( \frac{1}{m} \right) \quad \text{and,}
\]

\[
m > 1 \quad \Rightarrow \quad 0 < m' < 1
\]
Example

- **Line end points:**
  
  \[(x_0, y_0) = (5,8); \quad (x_1, y_1) = (9,11)\]

- **Deltas:**
  
  \[dx = 4; \quad dy = 3\]

- **After translation:** \((0,0) \rightarrow (4,3)\)
Graph
Example ( \( dx = 4; \ dy = 3 \) )

- Initial value of \( dx = 4; \ dy = 3 \)

\[
d = 2dy - dx \\
d = 2*3 - 4 = 2 > 0
\]

\[
\text{incrE} = 2*dy = 6 \\
\text{incrNE} = 2*(-1) = -2
\]

so \( \text{NE} \) is first move

```c
void midpoint-line(int xo, int yo, int xe, int ye, int value)
{
    int dx = xe - xo;
    int dy = ye - yo;
    int d = 2 * dy - dx;
    int incrE = 2 * dy;
    int incrNE = 2 * (dy - dx);
    int x = xo;
    int y = yo;
    writePixel(xo,yo, value);
}
```
Graph

![Graph Image]
Example (dx = 4; dy = 3)

- Last move was NE
- Update value of d
- \( d = 2 + \text{incrNE} = 2 - 2 = 0 \)
- So move East

```c
while (x<xe) {
    if (d<=0) { /* choose E */
        d+=incrE;
        x++;
    } else { /* choose NE */
        d += incrNE;
        x++;
        y++;
    }
    writePixel(x,y, value);
} /* end while */
/* end midPoint line */
```
Graph
Example (dx = 4; dy = 3)

- Last move was E
- Update value of d
- d = 0 + incrE = 6
- So move NE

```c
while (x<xe) {
    if (d<0) { /* choose E */
        d += incrE;
        x++;
    } else { /* choose NE */
        d += incrNE;
        x++;
        y++;
    }
    writePixel(x, y, value);
} /* end while */

} /* end midpoint line */
```
Example ( \( dx = 4; \ dy = 3 \) )

- Last move was NE
- Update value of \( d \)
- \( d = 6 + \text{incrNE} = 4 \)

So move NE

```java
while (x<=xe) {
    if (d<=0) { /* choose E */
        d+=incrE;
        x++;
    } else { /* choose NE */
        d += incrNE;
        x++;
        y++;
    }
    writePixel(x,y, value);
} /* end while */

} /* end midPoint line */
```
Graph
Graph
Graph
More Raster Line Issues

- Fat lines with multiple pixel width
- Symmetric lines
- How should end pt geometry look?
- Generating curves, e.g., circles, etc.
- Jaggies, staircase effect, aliasing...
Pixel Space

![Pixel Space Diagram](image-url)
Example
Example
Double Step Line Algorithm X Wu (1987)

Symmetric Variation (blob 1987)

Pattern 1

Pattern 2

Pattern 3

Pattern 4

Bresenham’s chooses next pixel with one discriminator.
Wu chooses next pattern of two pixels with one discriminator.
By drawing from both ends of line at once 4 pixels can be chosen
with one discriminator.

Observe that for $0 < \text{slope} \leq 1/2$ choose pattern 1 or (pattern 2 or 3)
for $1/2 < \text{slope} \leq 1$ choose pattern 4 or (pattern 2 or 3)

This can be done with only one discriminator calculated in the loop.
See code in course directory. For a full derivation see:
Fast Line Scan-Conversion, J.G. Rokne and B. Wyvill and X. Wu,
**Double Step code for slope < 1/2**

```c
for (i = 0; i < xend; i++) {
    /* plotting loop */
    ++x;
    --x1;
    if (D < 0) {
        /* pattern 1 forwards */
        plot(x, y, reverse);
        plot(++x, y, reverse);
        /* pattern 1 backwards */
        plot(x1, y1, reverse);
        plot(--x1, y1, reverse);
        D += incr1;
    } else {
        if (D < c) {
            /* pattern 2 forwards */
            plot(x, y, reverse);
            plot(++x, y += step, reverse);
            /* pattern 2 backwards */
            plot(x1, y1, reverse);
            plot(--x1, y1 -= step, reverse);
        } else {
            /* pattern 3 forwards */
            plot(x, y += step, reverse);
            plot(++x, y, reverse);
            /* pattern 3 backwards */
            plot(x1, y1 -= step, reverse);
            plot(--x1, y1, reverse);
        }
    }
    D += incr2; /* only one discriminator incremented */
}
} /* end for */
```