University of British Columbia

Chapter 4

Scan Conversion – Drawing Lines on Raster Display

Scan Conversion - Rasterization

- Convert continuous rendering primitives into discrete fragments/pixels
  - Lines
    - Bresenham
  - Triangles
    - Flood Fill
    - Scanline
    - Implicit formulation

University of British Columbia

Lines and Curves

- Explicit - one coordinate as function of the others

$$y = f(x)$$
$$z = f(x,y)$$

line

$$y = mx + b$$
$$y = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1) + y_1$$

circle

$$y = \pm\sqrt{r^2 - x^2}$$

University of British Columbia

Lines and Curves

- Parametric – all coordinates as functions of common parameter

$$(x,y) = (f_1(t), f_2(t))$$
$$(x,y,z) = (f_1(t), f_2(t), f_3(t))$$

line

$$x(t) = x_0 + t(x_1 - x_0)$$
$$y(t) = y_0 + t(y_1 - y_0)$$
$$t \in [0,1]$$

circle

$$x(\theta) = r\cos(\theta)$$
$$y(\theta) = r\sin(\theta)$$
$$\theta \in [0, 2\pi]$$

University of British Columbia

Lines and Curves

- Implicit - define as "zero set" of function of all the parameters

$$\{(x,y) : F(x,y) = 0\}$$
$$\{(x,y,z) : F(x,y,z) = 0\}$$

- Defines partition of space

$$\{(x,y) : F(x,y) > 0\}, \{(x,y) : F(x,y) = 0\}, \{(x,y) : F(x,y) < 0\}$$

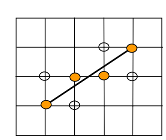
line		circle	
$F(x,y) = (x - x_0)dy - (y - y_0)dx$		$F(x,y) = x^2 + y^2 - r^2$	
$F(x,y) = 0$	**(x,y) is on line**	$F(x,y) = 0$	**(x,y) is on circle**
$F(x,y) > 0$	**(x,y) is below line**	$F(x,y) > 0$	**(x,y) is outside**
$F(x,y) < 0$	**(x,y) is above line**	$F(x,y) < 0$	**(x,y) is inside**

University of British Columbia

Basic Line Drawing

Assume $x_1 < x_2$ & line slope absolute value is ≤ 1

```
Line ( x1, y1, x2, y2 )
begin
float dx, dy, x, y, slope;
dx ⇐ x2 − x1;
dy ⇐ y2 − y1;
slope ⇐ dy/dx;
y ⇐ y1
for x from x1 to x2 do
begin
PlotPixel ( x, Round (y) );
y ⇐ y + slope;
end;
end;
```



Questions:
Can this algorithm use integer arithmetic ?

Does it accumulate error ?

Is the error significant ?

University of British Columbia

Recursive Line Drawing

Simple, recursive, integer, line drawing:

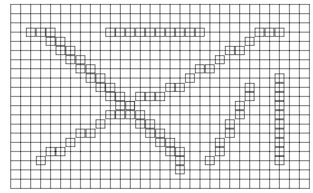
```
Line ( x1, y1, x2, y2 )
begin
int x, y ;
x ⇐ (x1 + x2) / 2 ;
y ⇐ (y1 + y2) / 2;
if ( ( x = x1 and  y = y1 ) or
    ( x = x2 and  y = y2 ) )
  return  ;
else  begin
PlotPixel  ( x, y ) ;
Line  ( x1, y1, x, y ) ;
Line  ( x, y, x2, y2 ) ;
end  ;
end  ;
```

Questions:
Does the algorithm accumulate error ?

Is it significant ?

University of British Columbia

Recursive Line Drawing (cont'd)



- **More Problems:**
  - Line not drawn sequentially
  - Function call for each pixel drawn

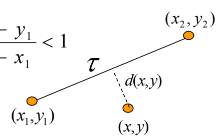
We want a faster algorithm !

University of British Columbia

Midpoint (Bresenham) Algorithm

- **Assumptions:**

$$x_2 > x_1, y_2 > y_1 \, and \; \frac{dy}{dx} = \frac{y_2 - y_1}{x_2 - x_1} < 1$$



- **Idea:**
  - Define error function
  - Proceed along the line incrementally
  - Select direction that minimizes accumulated error

Definitions

$$\tau = \{(x,y) | ax + by + c = xdy - ydx + c = 0\}$$
$$d(x,y) = 2(xdy - ydx + c)$$

University of British Columbia

Bresenham Algorithm

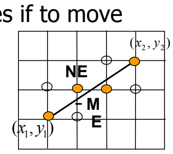
$$d(x,y) \;=\; 2(xdy - ydx + c)$$

- Given point $P = (x,y)$, $d(x,y)$ is signed distance of p to τ (up to normalization factor)
- d is zero for $P \in \tau$
  $\Rightarrow d$ may serve as error function to be minimized
- Starting point satisfies $d(x_1, y_1) = 0$
- Each step moves right (east) or upper right (northeast)

University of British Columbia

Midpoint Line Drawing (cont'd)

- Sign of $d(x_1+1, y_1+\frac{1}{2})$ indicates if to move east or northeast
- At (x_1, y_1)
  $d_{start} = d(x_1+1, y_1+\frac{1}{2}) = 2dy - dx$
- Increment in d (after each step)
  - Move east $\Delta_e = d(x+2, y+\frac{1}{2}) - d(x+1, y+\frac{1}{2}) =$
    $2((x+2)dy - (y+\frac{1}{2})dx + c)) - 2((x+1)dy - (y+\frac{1}{2})dx + c) = 2dy$
  - Move northeast
    $\Delta_{ne} = d(x_1+2, y_1+\frac{3}{2}) - d(x_1+1, y_1+\frac{1}{2}) =$
    $2((x+2)dy - (y+\frac{3}{2})dx + c)) - 2((x+1)dy - (y+\frac{1}{2})dx + c) = 2(dy - dx)$



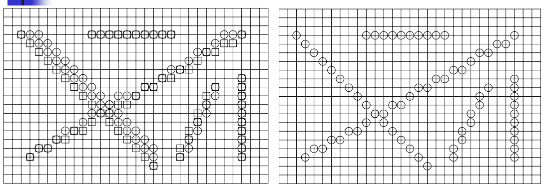
University of British Columbia

Midpoint Line Algorithm

```
Line ( x1, y1, x2, y2 )
begin
int x, y, dx , dy , d , Δe, Δne ;
x ⇐ x1 ;            y ⇐ y1 ;
dx ⇐ x2 − x1 ;      dy ⇐ y2 − y1 ;
d ⇐ 2 * dy − dx ;
Δe ⇐ 2 * dy ;       Δne ⇐ 2 * (dy − dx ) ;
PlotPixel  ( x, y ) ;
while  ( x < x2 )  do
if  ( d < 0 )  then
begin
d ⇐ d + Δe ;
x ⇐ x + 1 ;
end  ;
else  begin
d ⇐ d + Δne  ;
x ⇐ x + 1 ;
y ⇐ y + 1 ;
end  ;
PlotPixel  ( x, y ) ;
end  ;
end  ;
```

University of British Columbia

Midpoint Examples



Midpoint (circles) vs. Recursive (squares) line drawing

- Question: Is there a problem with this algorithm (horizontal vs. diagonal lines)?
- Comment: extends to higher order curves – e.g. circles

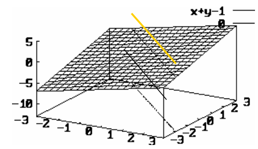
fineline

University of British Columbia

Error Function Intuition

- Error function d can be viewed as explicit surface:

$$d(x,y)=2(xdy-ydx+c)$$



University of British Columbia