CS 430
Computer Graphics

Line Drawing

Week 1, Lecture 2

David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

1

1

Outline

- Line drawing
- Digital differential analyzer
- Bresenham's algorithm

2

2

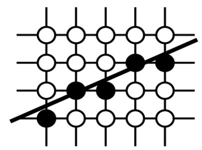
Line Drawing

15

15

Scan-Conversion Algorithms

- Scan-Conversion: Computing pixel coordinates for *ideal* line on 2D raster grid
- Pixels best visualized as circles/dots
  – Why? Monitor hardware



16

16

Drawing a Line

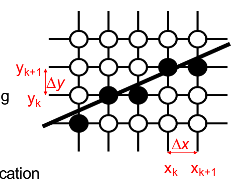
- $y = mx + B$
- $m = \Delta y / \Delta x$
- Start at leftmost x and increment by 1
  $\rightarrow \Delta x = 1$
- $y_i = \text{Round}(mx_i + B)$
- This is expensive and inefficient
- Since $\Delta x = 1$, $y_{i+1} = y_i + \Delta y = y_i + m$
  – No more multiplication!
- This leads to an incremental algorithm

17

17

Digital Differential Analyzer (DDA)

- If |slope| is less then 1
  - $\Delta x = 1$
  - else $\Delta y = 1$
- Check for vertical line
  - $m = \infty$
- Compute corresponding Δy (Δx) = m (1/m)
- $x_{k+1} = x_k + \Delta x$
- $y_{k+1} = y_k + \Delta y$
- Round (x,y) for pixel location
- Issue: Would like to avoid floating point operations



18

18

Generalizing DDA

- If |slope| is less than or equal to 1
  - Ending point should be right of starting point
- If |slope| is greater than 1
  - Ending point should be above starting point
- Keep x and y as floating point values
- Vertical line is a special case

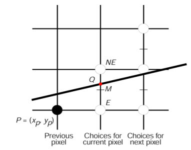
  $\Delta x = 0$

19

19

Bresenham's Algorithm

- 1965 @ IBM
- Basic Idea:
  - Only integer arithmetic
  - Incremental



- Consider the *implicit* equation for a line:

$$f(x,y) = ax + by + c = 0$$

20

1994 Foley/VanDam/Finer/Huges/Phillips ICG

20

The Algorithm

```
void bresenham(IntPoint q, IntPoint r) {
    int dx, dy, D, x, y;
    dx = r.x - q.x;                 // line width and height
    dy = r.y - q.y;
    D = 2*dy - dx;                  // initial decision value
    y = q.y;                        // start at (q.x,q.y)
    for (x = q.x; x <= r.x; x++) {
        writePixel(x, y);
        if (D <= 0) D += 2*dy;      // below midpoint - go to E
        else {                      // above midpoint - go to NE
            D += 2*(dy - dx); y++;
        }
    }
}
```

Assumptions: $q_x < r_x$
$0 \leq$ slope ≤ 1

21

Pics/Math courtesy of Dave Mount @ UMD-CP

21

Bresenham's Algorithm

Given:
implicit line equation: $f(x,y) = ax + by + c = 0$

Let: $d_x = r_x - q_x,\ d_y = r_y - q_y$
where r and q are points on the line and d_x is positive

$$a = d_y,\ b = -d_x,\ c = -(q_x r_y - r_x q_y)$$

Then:
Observe that all of these are integers
and: $f(x,y) < 0$ for points above the line
$f(x,y) > 0$ for points below the line
Now…..

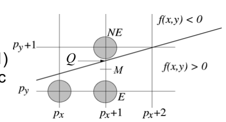
23

Pics/Math courtesy of Dave Mount @ UMD-CP

23

Bresenham's Algorithm

- Suppose we just finished (p_x, p_y)
  - (assume 0 ≤ slope ≤ 1) other cases symmetric
- Which pixel next?
  - *E* or *NE*



$$\text{East } (E = (p_x + 1, p_y))$$
$$\text{NorthEast } (NE = (p_x + 1, p_y + 1))$$

24

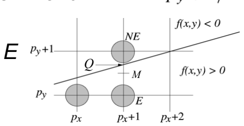
Pics/Math courtesy of Dave Mount @ UMD-CP

24

Bresenham's Algorithm

Assume:
- Q = exact y value at $x = p_x + 1$
- y midway between *E* and *NE*: $M = p_y + 1/2$

Observe:
If $Q < M$, then pick *E*
Else pick *NE*
If $Q = M$,
it doesn't matter



25

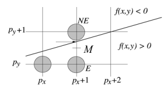
Pics/Math courtesy of Dave Mount @ UMD-CP

25

Bresenham's Algorithm

- Create "modified" implicit function (2x)
  $f(x,y) = 2ax + 2by + 2c = 0$
- Create a *decision variable D* to select, where *D* is the value of *f* at the midpoint:

$$\begin{aligned} D &= f(p_x+1, p_y+(1/2)) \\ &= 2a(p_x+1) + 2b\left(p_y + \frac{1}{2}\right) + 2c \\ &= 2ap_x + 2bp_y + (2a+b+2c) \end{aligned}$$



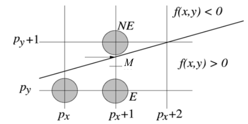
26

Pics/Math courtesy of Dave Mount @ UMD-CP

26

Bresenham's Algorithm

- If $D > 0$ then M is below the line $f(x,y)$
  - *NE* is the closest pixel
- If $D \leq 0$ then M is above the line $f(x,y)$
  - *E* is the closest pixel



27

27

Bresenham's Algorithm

- If $D > 0$ then M is below the line $f(x,y)$
  - *NE* is the closest pixel
- If $D \leq 0$ then M is above the line $f(x,y)$
  - *E* is the closest pixel
- Note: because we multiplied by 2x, *D* is now an integer---which is very good news
- How do we make this incremental??

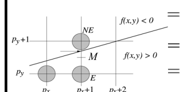
28

28

Case I: When *E* is next

- What increment for computing a new *D*?
- Next midpoint is: $(p_x+2, p_y+(1/2))$

$$\begin{aligned} D_{new} &= f(p_x+2, p_y+(1/2)) \\ &= 2a(p_x+2) + 2b\left(p_y + \frac{1}{2}\right) + 2c \\ &= 2ap_x + 2bp_y + (4a+b+2c) \\ &= 2ap_x + 2bp_y + (2a+b+2c) + 2a \\ &= D + 2a = D + 2d_y \end{aligned}$$



- Hence, increment by: $2d_y$

29

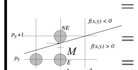
Pics/Math courtesy of Dave Mount @ UMD-CP

29

Case II: When *NE* is next

- What increment for computing a new *D*?
- Next midpoint is: $(p_x+2, p_y+1+(1/2))$

$$\begin{aligned} D_{new} &= f(p_x+2, p_y+1+(1/2)) \\ &= 2a(p_x+2) + 2b\left(p_y + \frac{3}{2}\right) + 2c \\ &= 2ap_x + 2bp_y + (4a+3b+2c) \\ &= 2ap_x + 2bp_y + (2a+b+2c) + (2a+2b) \\ &= D + 2(a+b) = D + 2(d_y - d_x) \end{aligned}$$



- Hence, increment by: $2(d_y - d_x)$

30

Pics/Math courtesy of Dave Mount @ UMD-CP

30

How to get an initial value for *D*?

- Suppose we start at: (q_x, q_y)
- Initial midpoint is: $(q_x+1, q_y+1/2)$

Then:

$$\begin{aligned} D_{init} &= f(q_x+1, q_y+1/2) \\ &= 2a(q_x+1) + 2b\left(q_y + \frac{1}{2}\right) + 2c \\ &= (2aq_x + 2bq_y + 2c) + (2a+b) \\ &= 0 + 2a + b \\ &= 2d_y - d_x \end{aligned}$$

31

Pics/Math courtesy of Dave Mount @ UMD-CP

31

The Algorithm

```
void bresenham(IntPoint q, IntPoint r) {
    int dx, dy, D, x, y;
    dx = r.x - q.x;                    // line width and height
    dy = r.y - q.y;
    D = 2*dy - dx;                     // initial decision value
    y = q.y;                           // start at (q.x,q.y)
    for (x = q.x; x <= r.x; x++) {
        writePixel(x, y);
        if (D <= 0) D += 2*dy;         // below midpoint - go to E
        else {                         // above midpoint - go to NE
            D += 2*(dy - dx); y++;
        }
    }
}
```

Assumptions: $q_x < r_x$
$0 \leq$ slope ≤ 1

Pre-computed: $2d_y$ $\quad 2(d_y - d_x)$

32

Pics/Math courtesy of Dave Mount @ UMD-CP

32

Generalize Algorithm

- If $q_x > r_x$, swap points
- If slope > 1, always increment y, conditionally increment x
- If -1 <= slope < 0, always increment x, conditionally decrement y
- If slope < -1, always decrement y, conditionally increment x
- Rework D increments

33

33

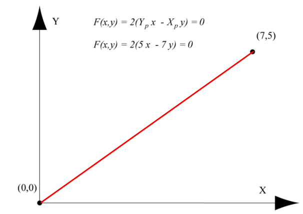
Generalize Algorithm

- Reflect line into first case
- Calculate pixels
- Reflect pixels back into original orientation

34

34

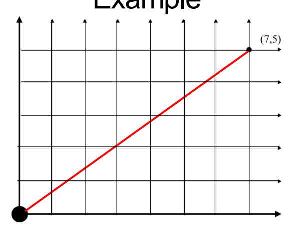
Bresenham's Algorithm: Example



35

35

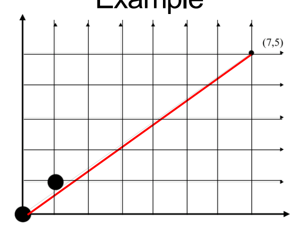
Bresenham's Algorithm: Example



36

36

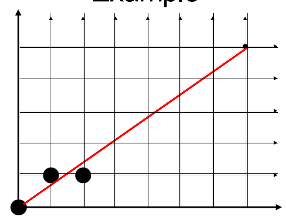
Bresenham's Algorithm: Example



37

37

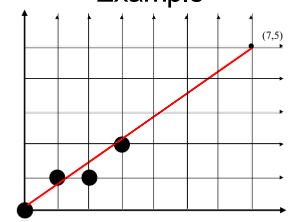
Bresenham's Algorithm: Example



38

38

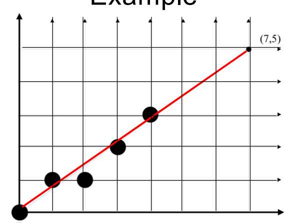
Bresenham's Algorithm: Example



39

39

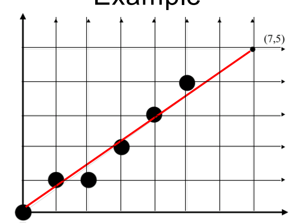
Bresenham's Algorithm: Example



40

40

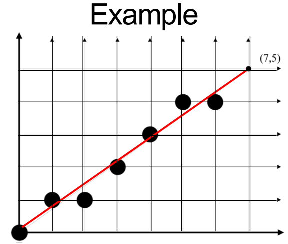
Bresenham's Algorithm: Example



41

41

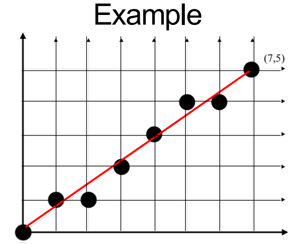
Bresenham's Algorithm: Example



42

42

Bresenham's Algorithm: Example

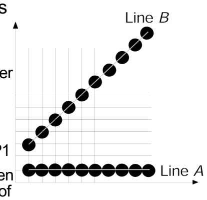


43

43

Some issues with Bresenham’s Algorithms

- Pixel ‘density’ varies based on slope
  - straight lines look darker, more pixels per unit length
  - Endpoint order
  - Line from P1 to P2 should match P2 to P1
  - Always choose *E* when hitting *M*, regardless of direction



44

44

Questions?

Go to Assignment 1

57

57