3D Graphics Programming Tools

Revision – Key Concepts
Rasterisation
(Past Exam Questions Review)

Dr. Pengwei Hao, Dr Chao Shu
School of Electronic Engineering and Computer Science
Queen Mary University of London
p.hao@qmul.ac.uk; c.shu@qmul.ac.uk
Nov. 2021

Rasterisation

Bresenham's Midpoint Line Algorithm

b) This question is about rasterisation. Use the line equation for a line from (x0, y0) to (x1, y1) to derive the mid-point algorithm for line generation.

[9 marks]

Solution:

Line equation: (x1 – x0)(y – y0) = (y1 – y0)(x – x0) or F(x,y) = 0 or
F(x,y) = (y1 – y0)(x – x0) – (x1 – x0)(y – y0) = (y1–y0)x–(x1–x0)y – (y1–y0)x0+(x1–x0)y0 (1 mark)
F(x+1, y) – F(x,y) = (y1–y0) (1 mark)
F(x+1, y+1) – F(x,y) = (y1–y0) – (x1–x0) (1 mark)
F(x+1, y+1/2) – F(x,y) = (y1–y0) – (x1–x0)/2 (1 mark)
We use 2F(x,y) = 0, dx=x1–x0, dy=y1–y0, (1 mark) then we have:
d(x0, y0) = 2dy – dx (1 mark)
d(x+1, y+1) = 2F(x+1, y+1) – 2F(x,y) = 2dy – 2dx (1 mark)
d(x+1, y) = 2F(x+1, y) – 2F(x,y) = 2dy (1 mark)
Move from (x0, y0) to (x1, y1) incrementally by x=x+1 (East) or x=x+1 and y=y+1 (North-East). (1 mark)

(total 9 *marks*)

Line Generation with Midpoints

Line equation: $(x1 - x0)(y - y0) = (y1 - y0)(x - x0)$ or $F(x,y) = 0$

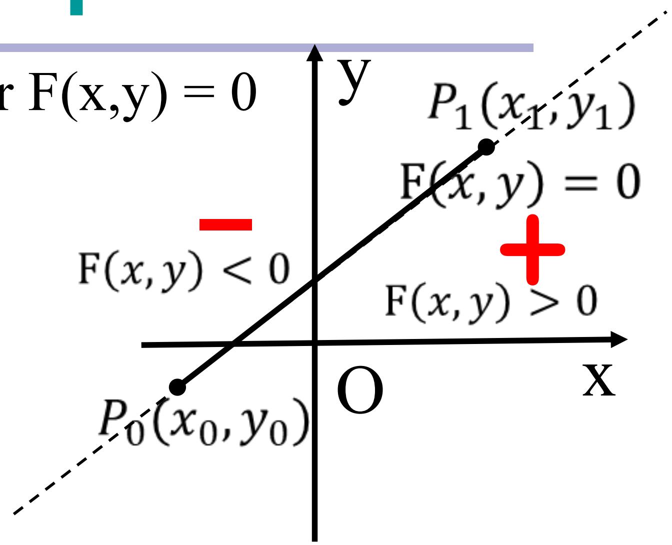
$F(x,y) = (y1 - y0)(x - x0) - (x1 - x0)(y - y0)$

$\quad = (y1-y0)x-(x1-x0)y - (y1-y0)x0+(x1-x0)y0$

$F(x+1, y) - F(x,y) = (y1-y0)$

$F(x+1, y+1) - F(x,y) = (y1-y0) - (x1-x0)$

$F(x+1, y+1/2) - F(x,y) = (y1-y0) - (x1-x0)/2$



If point P(x,y) drawn, the next point is either P(x+1,y) or P(x+1,y+1)
To decide which point, use the relative position of the midpoint M = (x+1, y+1/2) with respect to the line, which half-plane it is, positive or negative.

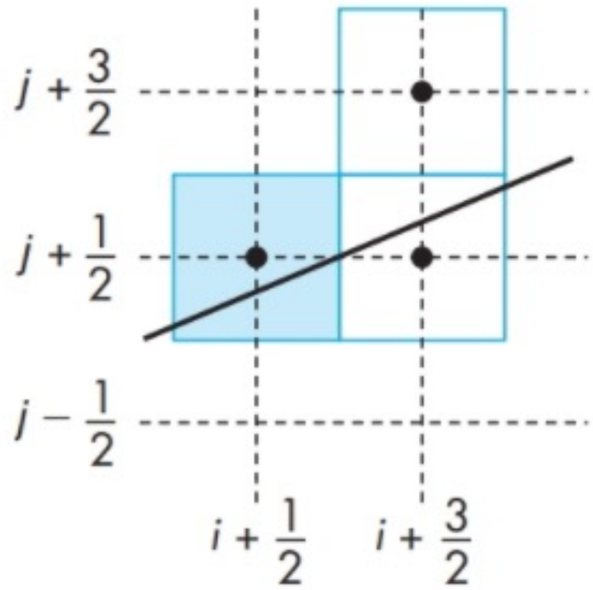
We use $2F(x,y) = 0$, $dx=x1-x0$, $dy=y1-y0$, then we have:

$d(x0, y0) = 2dy - dx$

$d(x+1, y+1) = 2F(x+1, y+1) - 2F(x,y) = 2dy - 2dx$

$d(x+1, y) = 2F(x+1, y) - 2F(x,y) = 2dy$

as the updating for every move.

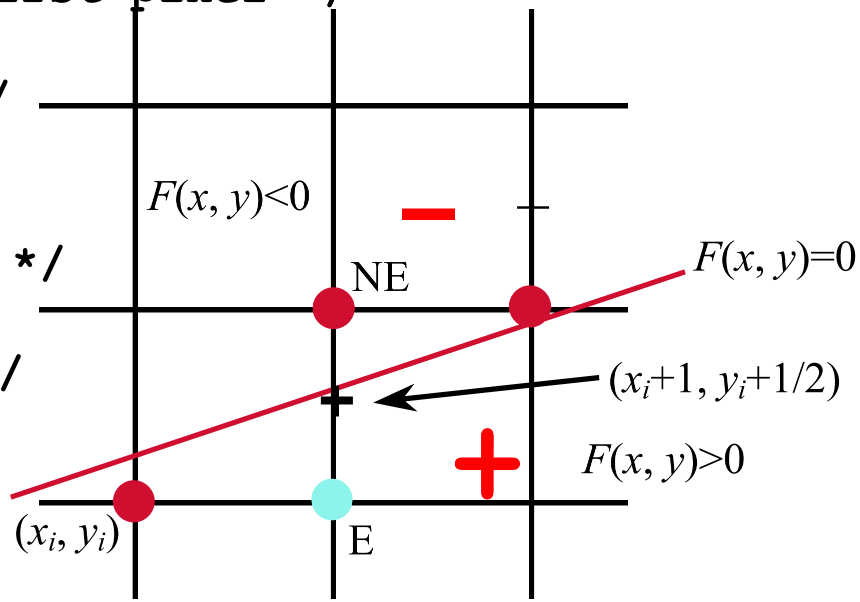


Bresenham's Midpoint Line Algorithm

Bresenham's algorithm:

```
void MidpointLine(int x0, int y0, int x1, int y1)
{   int dx,dy,incrE,incrNE,d,x,y;
    dx=x1-x0;   dy=y1-y0;
    d=2*dy-dx;            /* initial value of d */
    incrE=2*dy;           /* increment for move to E */
    incrNE=2*dy-2*dx;     /* increment for move to NE */
    x=x0;   y=y0;
    DrawPixel(x,y)          /* draw the first pixel */
    while (x<x1){
          if (d<=0){          /* choose E */
                  d+=incrE;
                  x++;        /* move E */
          } else {             /* choose NE */
                  d+=incrNE;
                  x++;   y++; /* move NE */
          }
          SetPixel (x,y);
    }
}
```

Cost: 1 integer add per pixel



Queen Mary
University of London

Rasterisation

Bresenham's Midpoint Line Algorithm

- Prerequisite: 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, $y = \left(\frac{dy}{dx}\right)x + B$ $\Rightarrow$ $\frac{dy}{dx}x - y + B = 0$

$\Downarrow$

Or, $(dy)x + (-dx)y + (dx)B = 0$

Relating explicit to implicit equations $\therefore \quad F(x,y) = (dy)x + (-dx)y + (dx)B = 0$

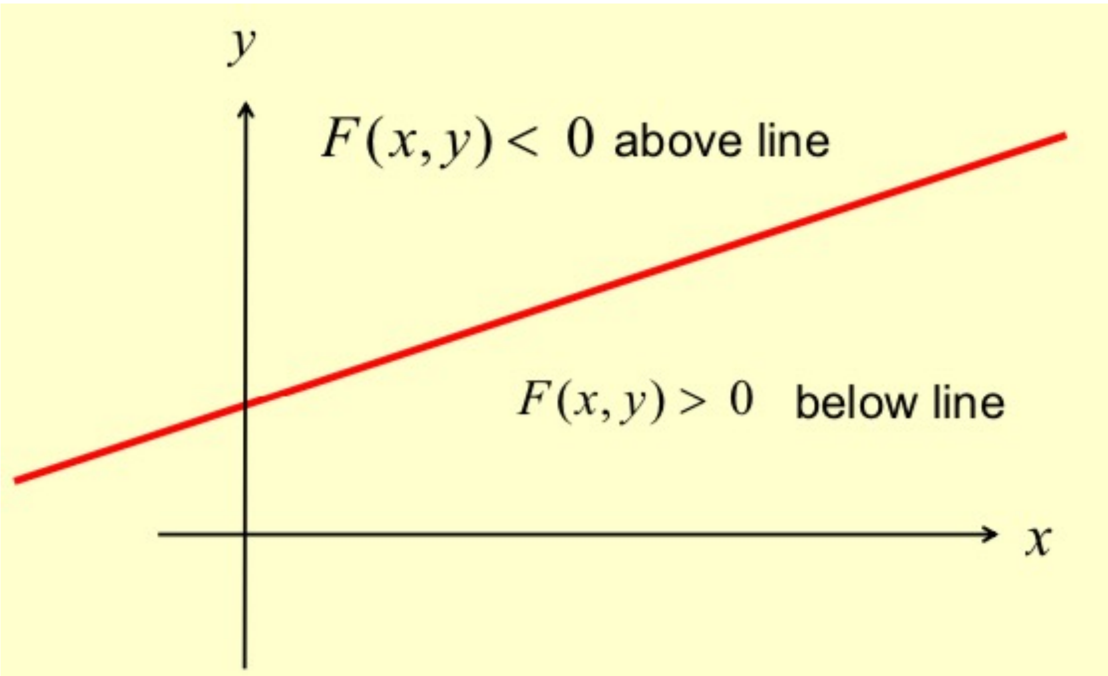
where, $\boxed{a = (dy); \quad b = -(dx); \quad c = B(dx)}$

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Prerequisite: Half-Spaces



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

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- Initial Assumption

- Line segment in *first* octant with

$$0 < m < 1$$

- After we derive this, we'll look at the other cases (other octants)

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

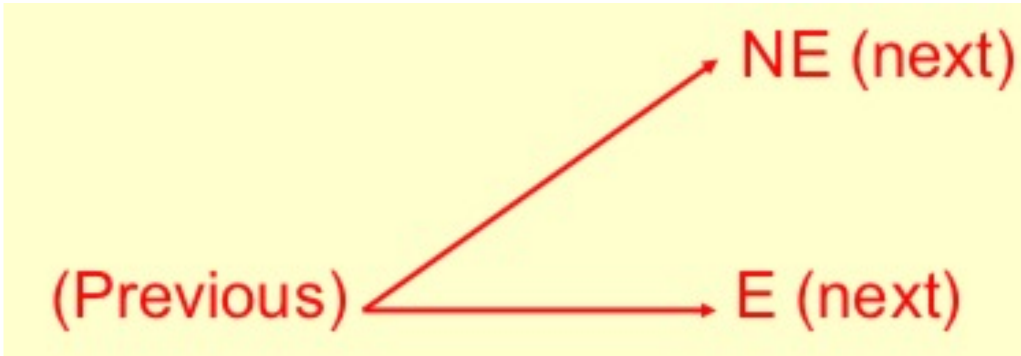
Rasterisation

Bresenham's Midpoint Line Algorithm

- Key to Bresenham's Algorithm

Decision variable d → Make binary choice at each pixel

Define a logical *decision* variable d
- linear in form
- incrementally updated (with addition)
- tells us whether to go *E* or *NE*



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

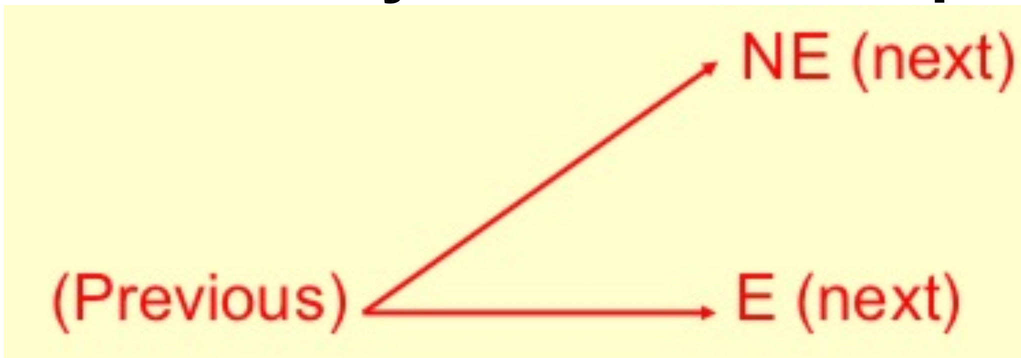
Rasterisation

Bresenham's Midpoint Line Algorithm

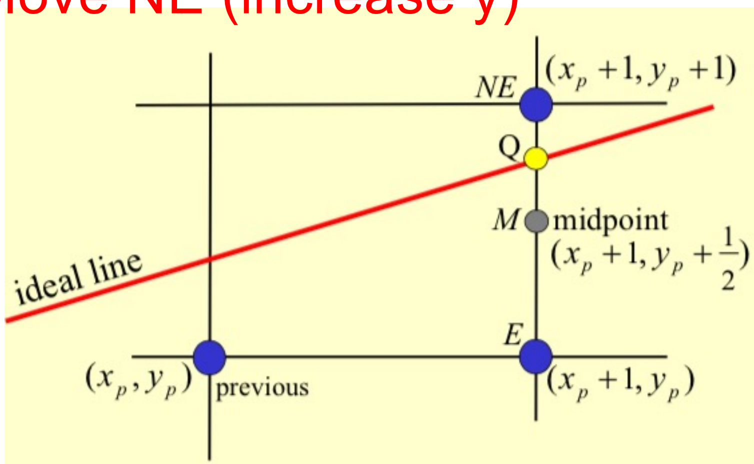
- Key to Bresenham's Algorithm

Decision variable d → Make binary choice at each pixel

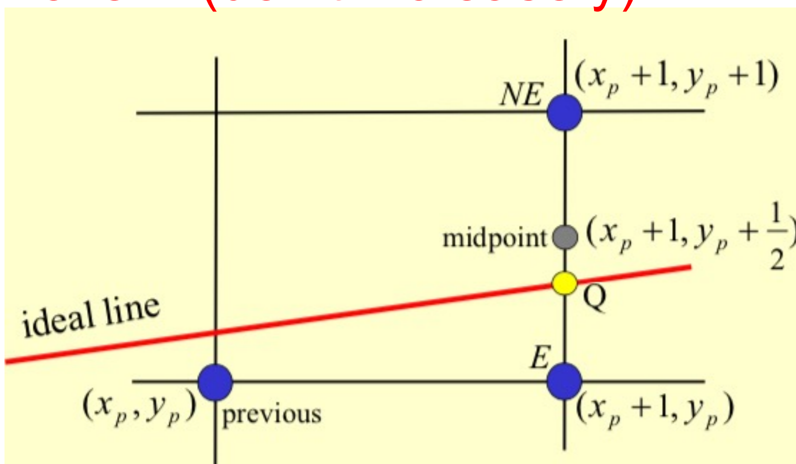
Define a logical *decision* variable d
- linear in form
- incrementally updated (with addition)
- tells us whether to go *E* or *NE*



$F(M) > 0 \Rightarrow M\ is\ blow\ the\ line \Rightarrow$
Move NE (increase y)



$F(M) < 0 \Rightarrow M\ is\ above\ the\ line \Rightarrow$
Move E (don't increase y)



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- Decision Variable d

Let, $$d = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$$

Therefore,

$$d = \begin{cases} > 0 & \Rightarrow NE & \text{(midpoint below ideal line)} \\ < 0 & \Rightarrow E & \text{(midpoint above ideal line)} \\ = 0 & \Rightarrow E & \text{(arbitrary)} \end{cases}$$

Will use an incremental decision variable d (with addition)

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Case E: Suppose E is chosen

- Recall $d_{old} = a(x_p + 1) + b(y_p + \frac{1}{2}) + c$
- $E \Rightarrow: \quad x \leftarrow x+1; \quad y \leftarrow y,$
- $\therefore \ldots d_{new} = F(x_p + 2, y_p + \frac{1}{2})$

$$= a(x_p + 2) + b(y_p + \frac{1}{2}) + c$$

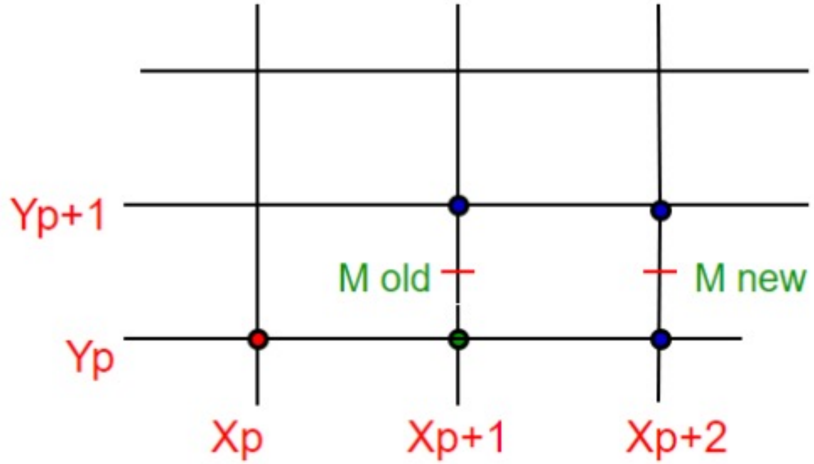
$$d_{new} - d_{old} = \left(a(x_p + 2) + b(y_p + \frac{1}{2}) + c \right)$$

$$- \left(a(x_p + 1) + b(y_p + \frac{1}{2}) + c \right)$$

$$\boxed{d_{new} = d_{old} + a}$$

$$F(x,y) = (dy)x + (-dx)y + (dx)B = 0$$

where, $a = (dy); \quad b = -(dx); \quad c = B(dx)$



$\Delta_E \equiv increment\ we\ add\ to\ d\ if\ E\ is\ chosen$
$\Delta_E = a = dy.$
In this way, $F(M)$ is not evaluated explicitly.
We simply add $\Delta_E = dy$ to update d for E

Queen Mary University of London

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.p

Rasterisation

Bresenham's Midpoint Line Algorithm

- Case NE: Suppose NE is chosen

Recall $d_{old} = a(x_p+1) + b(y_p+\frac{1}{2}) + c$

and, $NE \Rightarrow$: $x \leftarrow x+1$; $y \leftarrow y+1$,

$$\therefore \quad d_{new} = F(x_p+2, y_p+\frac{3}{2})$$

$$= a(x_p+2) + b(y_p+\frac{3}{2}) + c$$

$d_{new} - d_{old} =$

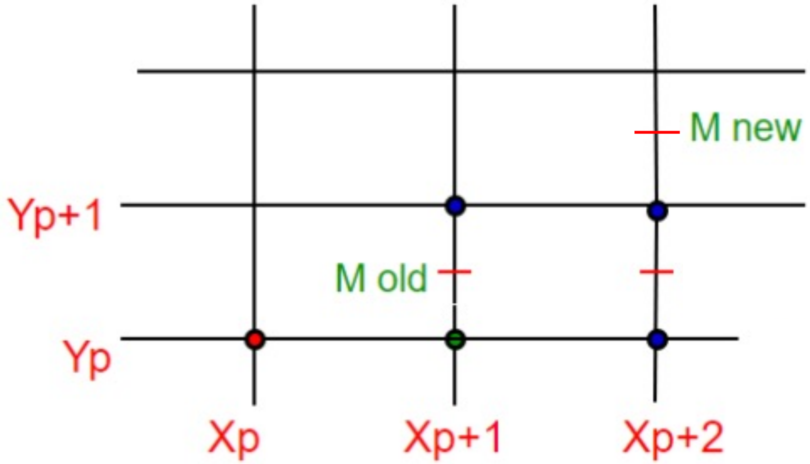
$$= \left(a(x_p+2) + b(y_p+\frac{3}{2}) + c \right)$$

$$- \left(a(x_p+1) + b(y_p+\frac{1}{2}) + c \right)$$

$$\boxed{d_{new} = d_{old} + a + b}$$

$$F(x,y) = (dy)x + (-dx)y + (dx)B = 0$$

where, $a = (dy)$; $b = -(dx)$; $c = B(dx)$



$\Delta_{NE} \equiv increment\ we\ add\ to\ d\ if\ NE\ is\ chosen$

$\Delta_{NE} = a + b = dy - dx.$

In this way, $F(M)$ is not evaluated explicitly.

We simply add $\Delta_{NE} = dy - dx$ to update d for NE

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pd

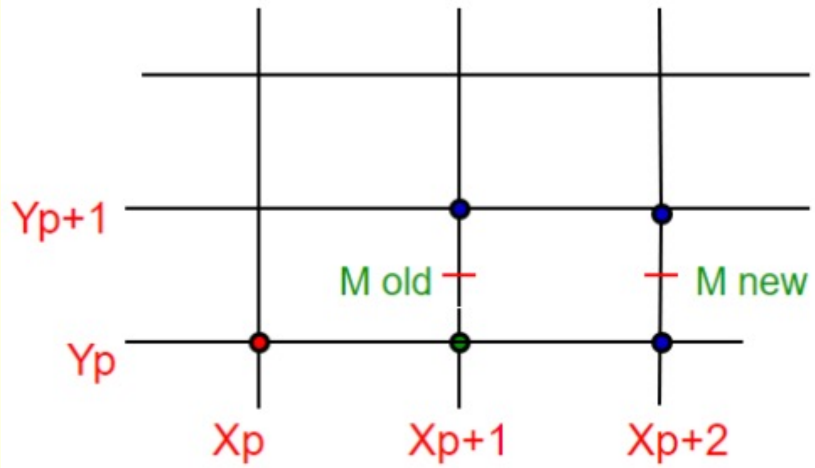
Rasterisation

Bresenham's Midpoint Line Algorithm

- Summary

- At each step of the procedure, we must choose between moving *E* or *NE* based on the sign of the decision variable d
- Then update according to

$$\boxed{d \leftarrow \begin{cases} d+\Delta_E\ , & \text{where } \Delta_E = dy, \text{or} \\ d+\Delta_{NE}\ , & \text{where} \Delta_{NE} = dy-dx \end{cases}}$$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- Initial value of d

- First point is (x_0, y_0)
- First midpoint is $(x_0+1, y_0+\frac{1}{2})$
- What is initial midpoint value?

$$d(x_0+1, y_0+\frac{1}{2}) = F(x_0+1, y_0+\frac{1}{2})$$

$$F(x_0+1, y_0+\frac{1}{2}) = a(x_0+1) + b(y_0+\frac{1}{2}) + c$$
$$= (ax_0 + by_0 + c) + \left(a + \frac{b}{2}\right)$$
$$= F(x_0, y_0) + \left(a + \frac{b}{2}\right)$$

Note, $F(x_0, y_0) = 0$, since (x_0, y_0) is on line.

Hence, $$F(x_0+1, y_0+\frac{1}{2}) = 0 + a + \frac{b}{2}$$
$$= (dy) - \left(\frac{dx}{2}\right)$$

Note, we can clear denominator and not change line,

$$2F(x_0+1, y_0+\frac{1}{2}) = 2(dy) - dx$$

$$2F(x, y) = 2(ax + by + c) = 0$$

So, first value of

$$\boxed{d = 2(dy) - (dx)}$$

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- More Summary

- Initial value $2(dy)-(dx)$
- Choose $\begin{cases} E & \text{if } d \le 0 \\ NE & \text{otherwise} \end{cases}$
- Case E: $d \leftarrow d + \Delta_E$, where $\Delta_E = 2(dy)$
- Case NE: $d \leftarrow d + \Delta_{NE}$,

  where $\Delta_{NE} = 2\{(dy)-(dx)\}$
- Note, all deltas are constants

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Line Generation with Midpoints

Bresenham's Midpoint Line Algorithm

Line equation: (x1 – x0)(y – y0) = (y1 – y0)(x – x0) or F(x,y) = 0

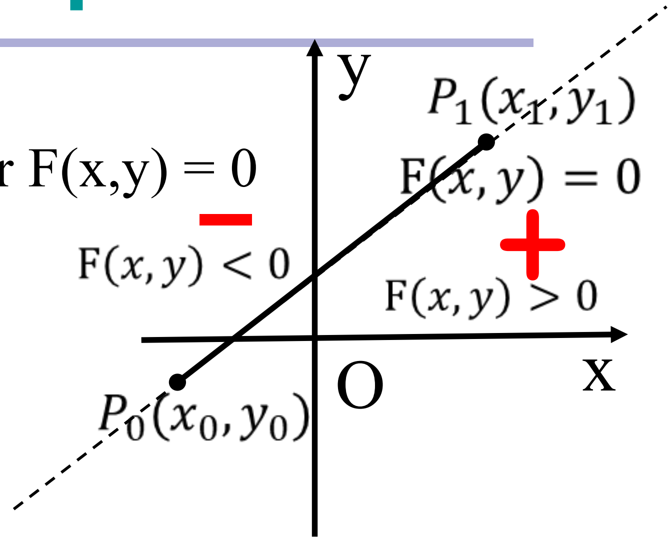
F(x,y) = (y1 – y0)(x – x0) – (x1 – x0)(y – y0)

= (y1–y0)x–(x1–x0)y – (y1–y0)x0+(x1–x0)y0

F(x+1, y) – F(x,y) = (y1–y0)

F(x+1, y+1) – F(x,y) = (y1–y0) – (x1–x0)

F(x+1, y+1/2) – F(x,y) = (y1–y0) – (x1–x0)/2



If point P(x,y) drawn, the next point is either P(x+1,y) or P(x+1,y+1)
To decide which point, use the relative position of the midpoint M = (x+1, y+1/2) with respect to the line, which half-plane it is, positive or negative.

We use 2F(x,y) = 0, dx=x1–x0, dy=y1–y0, then we have:

d(x0, y0) = 2dy – dx

- Initial value $\boxed{d = 2(dy) - (dx)}$

d(x+1, y+1) = 2F(x+1, y+1) – 2F(x,y) = 2dy – 2dx $\boxed{\Delta_{NE} = 2\{(dy) - (dx)\}}$

d(x+1, y) = 2F(x+1, y) – 2F(x,y) = 2dy $\boxed{\Delta_E = 2(dy)}$

as the updating for every move.

Queen Mary
University of London

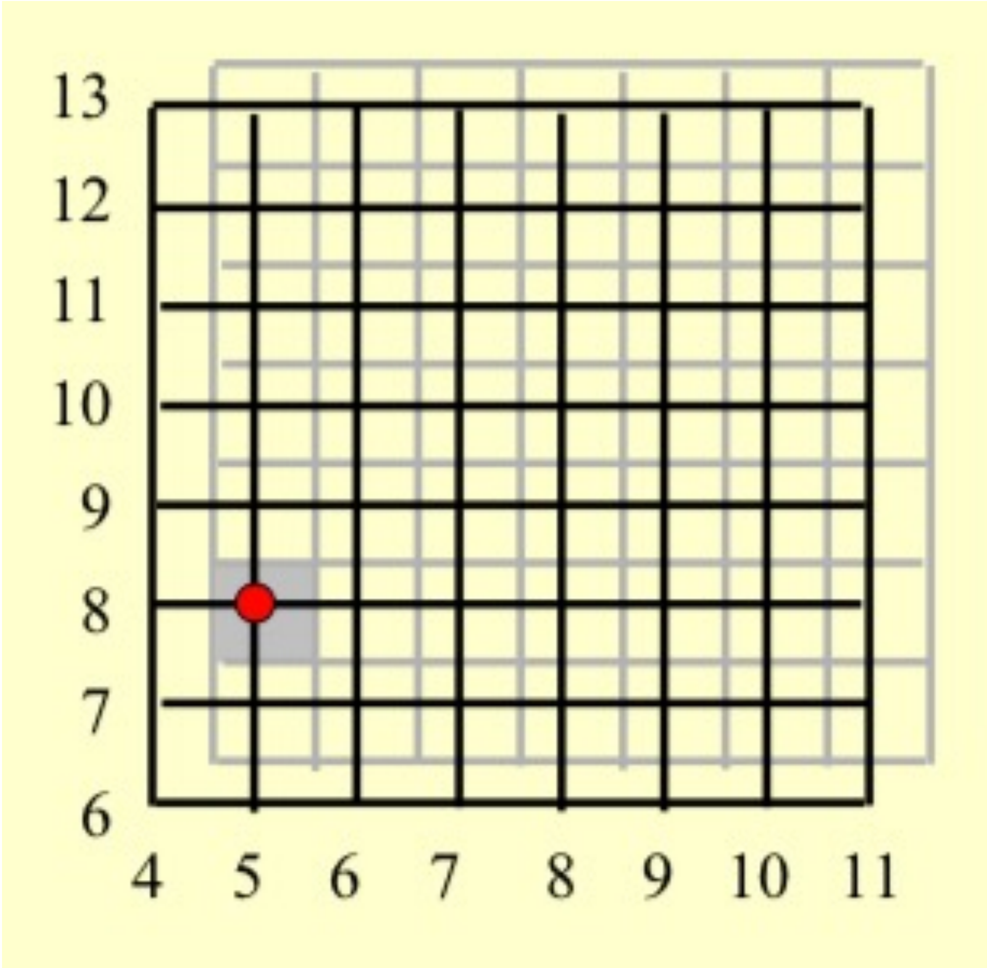
Rasterisation

Bresenham's Midpoint Line Algorithm

- Example
  - Line end points:

$$(x_0, y_0) = (5,8); \quad (x_1, y_1) = (9,11)$$

  - Deltas: $dx = 4; \; dy = 3$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

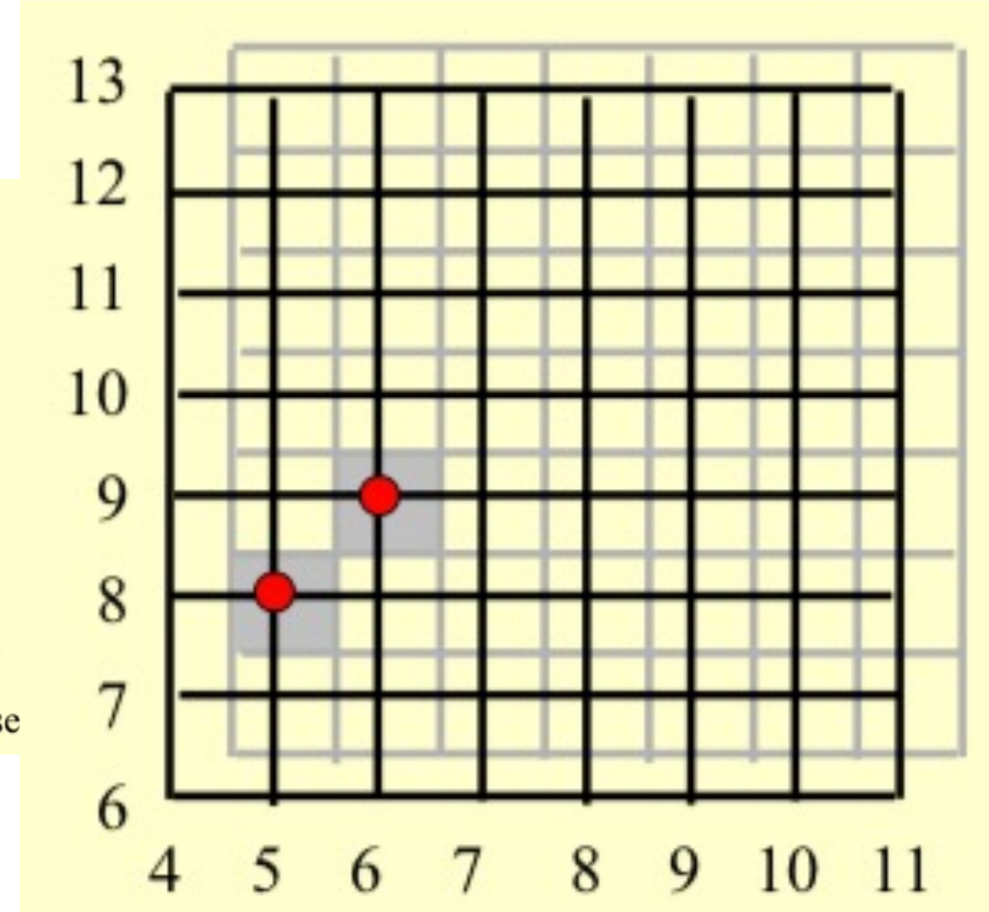
- Example

$$(dx = 4;\ dy = 3)$$

- Initial value of

$$d(5,8) = 2(dy) - (dx)$$
$$= 6 - 4 = 2 > 0$$
$$\boxed{d = 2 \quad \Rightarrow \quad NE}$$

$$d = 2(dy) - (dx) \qquad \begin{cases} E & \text{if } d \le 0 \\ NE & \text{otherwise} \end{cases}$$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Queen Mary University of London

Rasterisation

Bresenham's Midpoint Line Algorithm

- Example

$$(dx = 4;\ dy = 3)$$

- Update value of d

$d \leftarrow d + \Delta_E$, where $\Delta_E = 2(dy)$

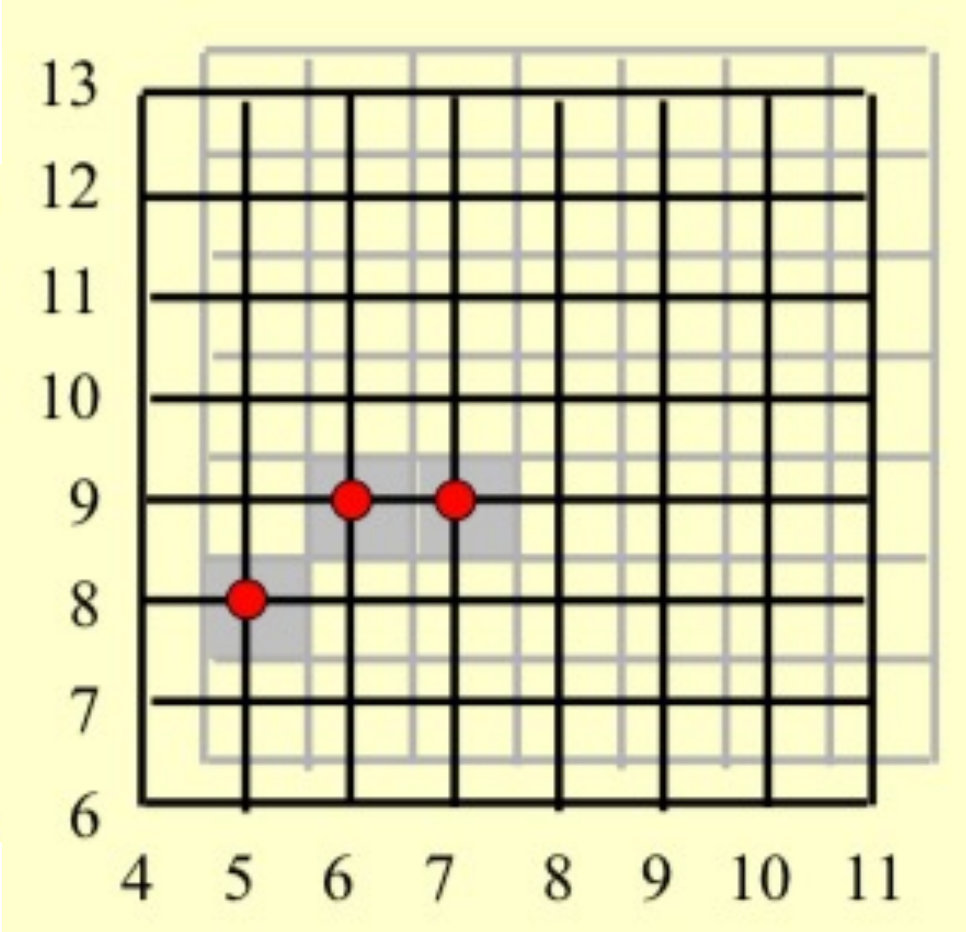
$d \leftarrow d + \Delta_{NE}$,
where $\Delta_{NE} = 2\{(dy) - (dx)\}$

- Last move was *NE*, so

$$\Delta_{NE} = 2(dy - dx)$$
$$= 2(3 - 4) = -2$$

$$\boxed{d = 2 - 2 = 0 \quad \Rightarrow \quad E}$$

$$\begin{cases} E & \text{if } d \le 0 \\ NE & \text{otherwise} \end{cases}$$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Example

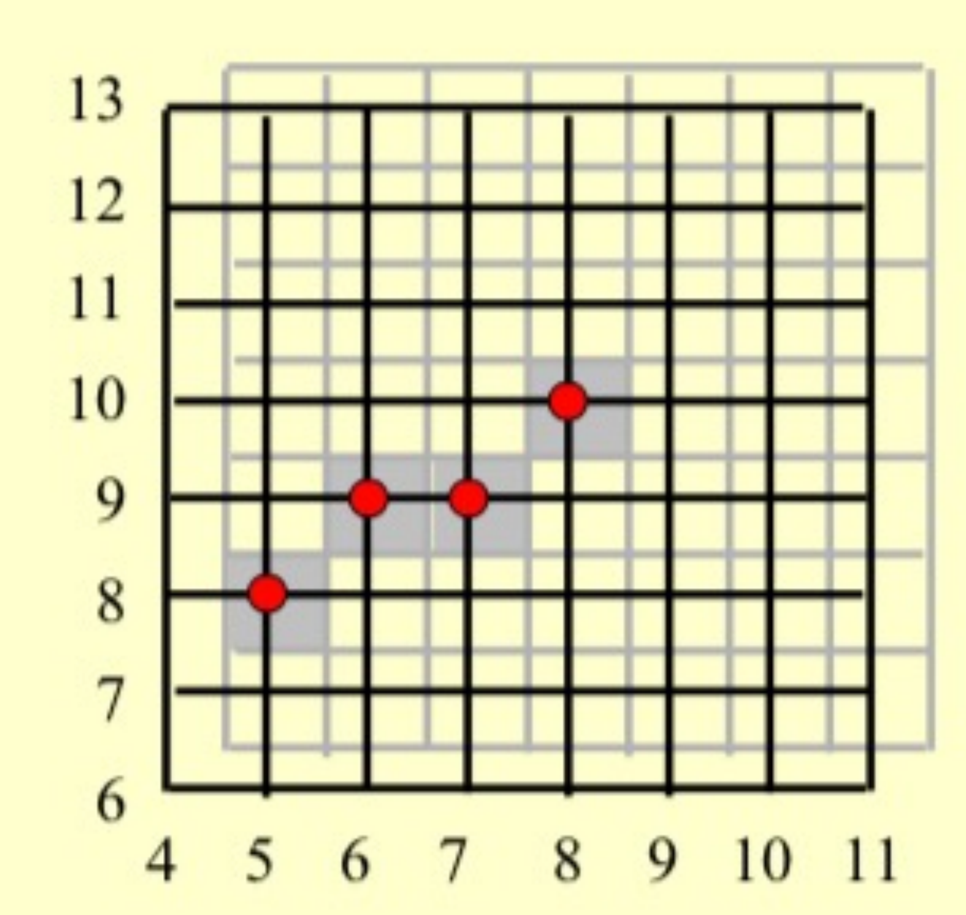
$$(dx = 4;\ \ dy = 3)$$

$d \leftarrow d + \Delta_E$, where $\Delta_E = 2(dy)$

- Previous move was E $d \leftarrow d + \Delta_{NE}$, where $\Delta_{NE} = 2\{(dy) - (dx)\}$

$$\Delta_E = 2(dy)$$
$$= 2(3) = 6$$
$$\boxed{d = 0 + 6 > 0 \ \Rightarrow \ NE}$$

$$\begin{cases} E & \text{if } d \le 0 \\ NE & \text{otherwise} \end{cases}$$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Example

$$(dx = 4;\ \ dy = 3)$$

$d \leftarrow d + \Delta_E$, where $\Delta_E = 2(dy)$

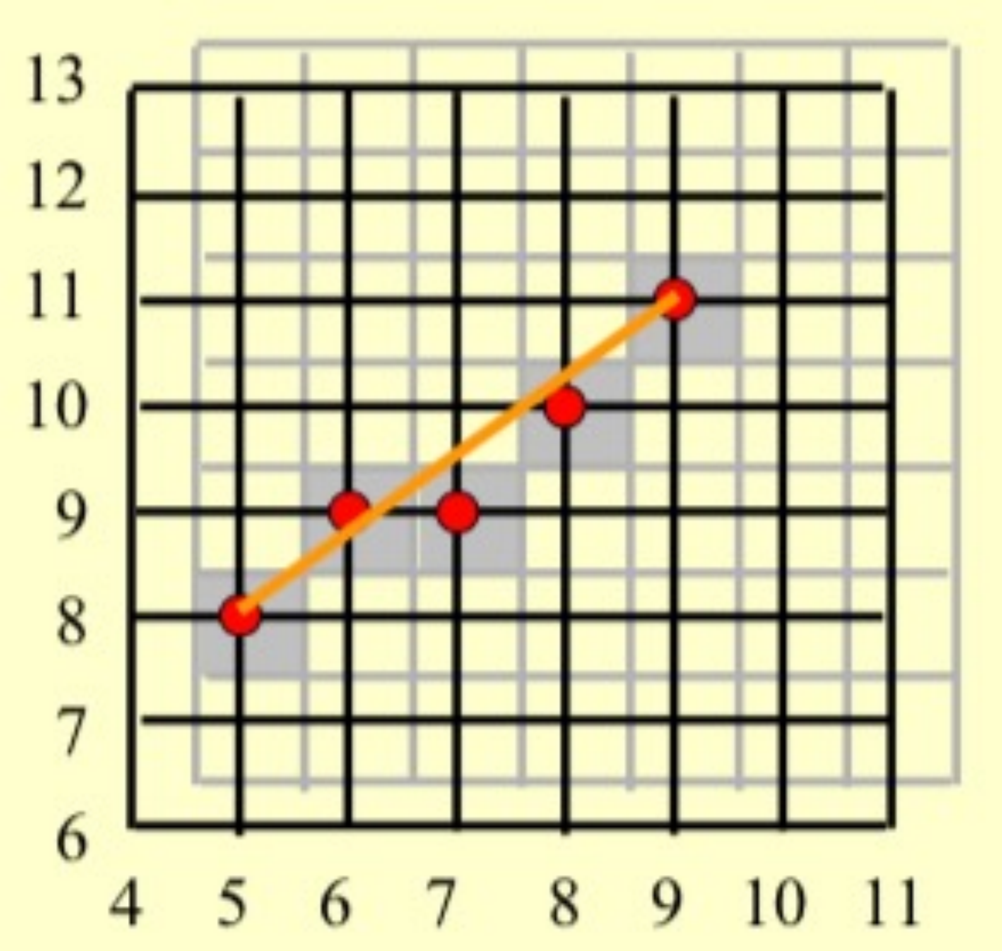
- Previous move was *NE*

$d \leftarrow d + \Delta_{NE}$,
where $\Delta_{NE} = 2\{(dy) - (dx)\}$

$$\Delta_{NE} = 2(dy - dx)$$
$$= 2(3 - 4) = -2$$

$$\boxed{d = 6 - 2 = 4 \quad \Rightarrow \quad NE}$$

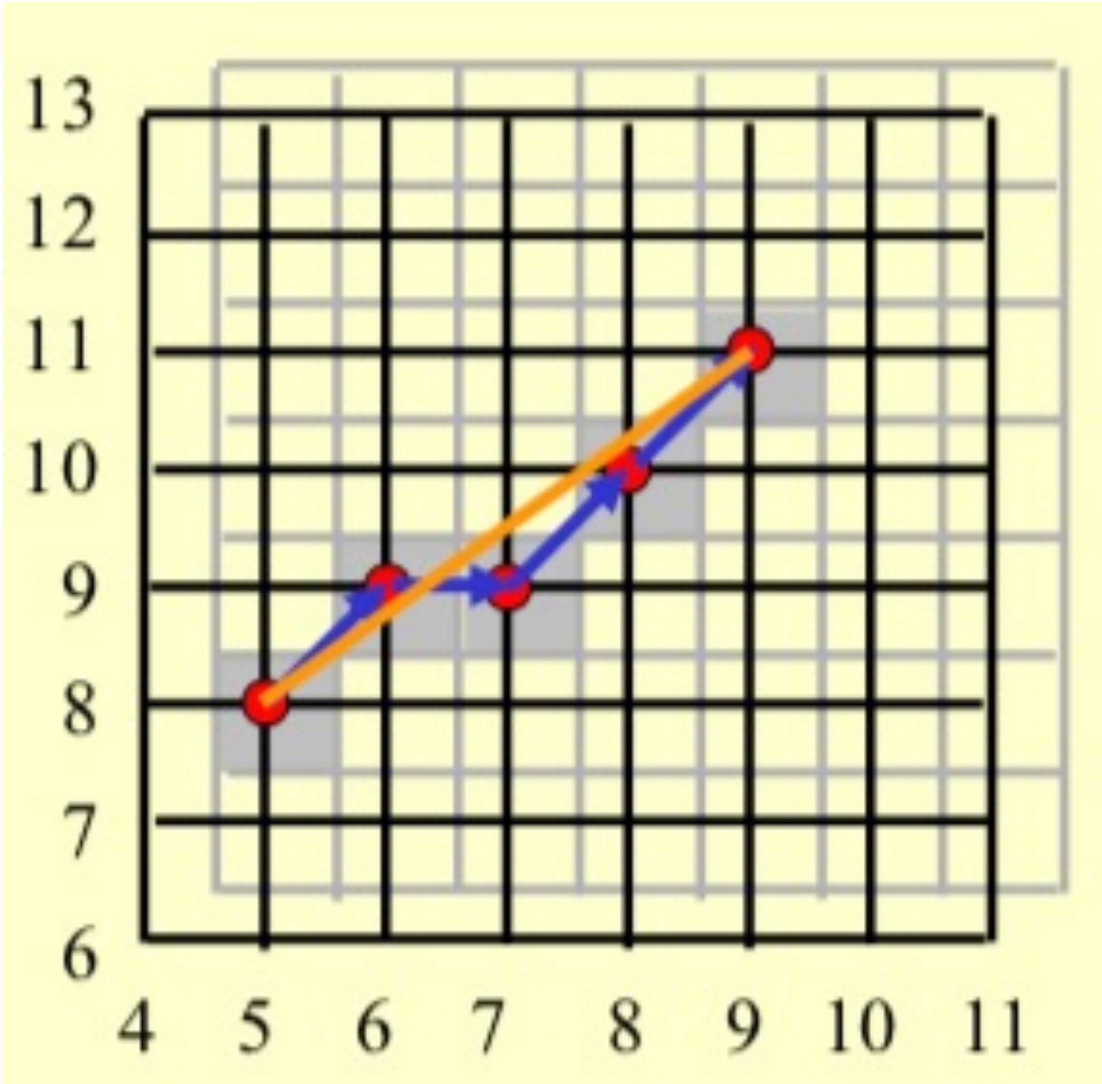
$$\begin{cases} E & \text{if } d \le 0 \\ NE & \text{otherwise} \end{cases}$$



Rasterisation

Bresenham's Midpoint Line Algorithm

- Example



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- Other cases

Case 0: $m = 0; \quad m = 1 \;\Rightarrow\;$ trivial cases

Case 1: $0 > m > -1 \;\Rightarrow\;$ flip about x-axis

Case 2: $m > 1 \;\Rightarrow\;$ flip about $x = y$

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Bresenham's Midpoint Line Algorithm

- Other cases

  - Case 0: Trivial Situations

    - $m = 0 \quad \Rightarrow \quad$ horizontal line
    - $m = 1 \quad \Rightarrow \quad$ line $y = x$
    - Do not need Bresenham

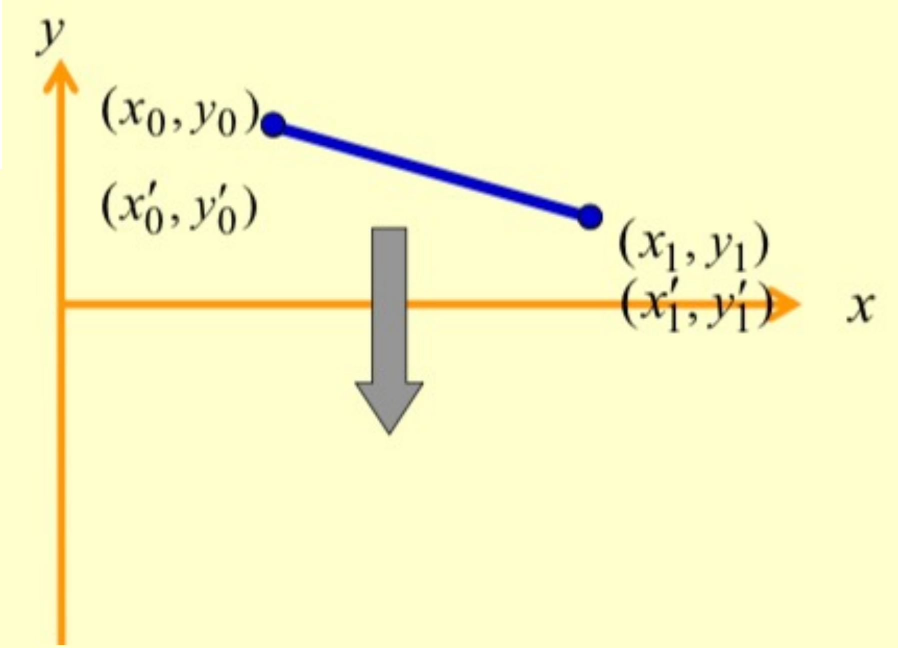
Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Queen Mary University of London

Rasterisation

Bresenham's Midpoint Line Algorithm

- Other cases
  - Case 1: Flip about x-axis



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

$$(x'_0, y'_0) = (x_0, -y_0);$$
$$(x'_1, y'_1) = (x_1, -y_1)$$

$$\left.\begin{array}{l} m = \dfrac{y_1 - y_0}{x_1 - x_0}; \\ \\ m' = \dfrac{y'_1 - y'_0}{x_1 - x_0} \end{array}\right\} \text{ by definition} \qquad \text{i.e.,} \qquad \begin{array}{l} m' = -\dfrac{(y_1 - y_0)}{x_1 - x_0} \\ m' = -m \end{array}$$

Since $y'_i = -y_i$, $m' = \dfrac{-y_1 - (-y_0)}{x_1 - x_0}$ $\boxed{\therefore \quad 0 > m > -1 \quad \Rightarrow \quad 0 < m' < 1}$

Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Other cases
  - Case 2: Flip about line $y=x$

$$y = mx + B,$$

swap $x \leftrightarrow y$ and prime them ,

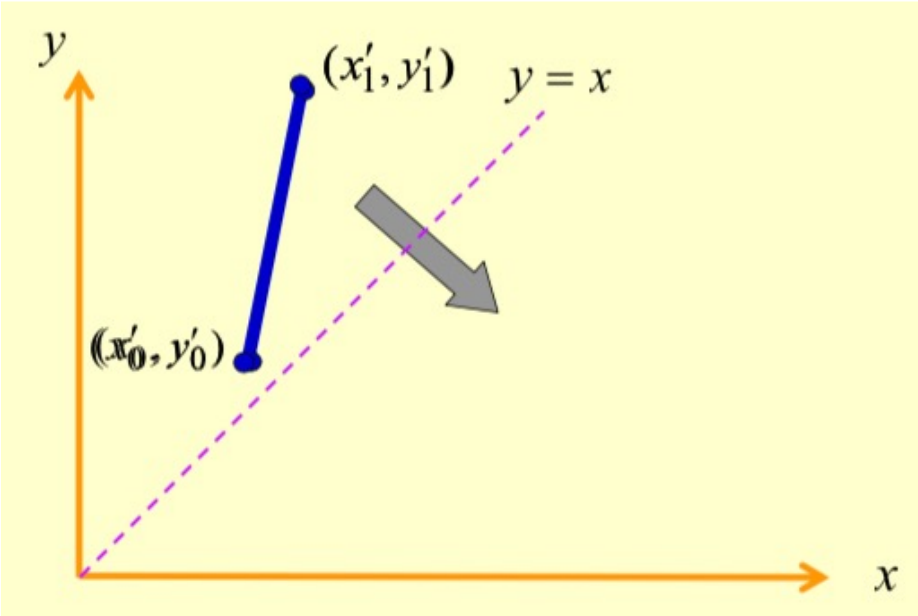
$$x' = my' + B,$$

$$my' = x' - B$$

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

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

$$\boxed{m > 1 \quad \Rightarrow \quad 0 < m' < 1}$$



Source: http://www.eng.utah.edu/~cs5600/slides/Wk%202%20Lec02_Bresenham.pdf

Rasterisation

Bresenham's Midpoint Line Algorithm

- Demonstration

```python
def midPoint(x1, y1, x2, y2):
    """Bresenham's Midpoint Line Algorithm"""

    # Make sure x will be increasing
    if (x1 > x2):
        x1, x2 = x2, x1
        y1, y2 = y2, y1

    # calculate dx & dy
    dx = x2 - x1
    dy = y2 - y1

    # When -1 < slope < 0
    if (dy < 0):
        slope = -1
        dy = -dy
    else:
        slope = 1

    # When
    if (abs(dy) > abs(dx)):
        dx, dy = dy, dx
        x1, y1 = y1, x1
        x2, y2 = y2, x2
        #x = y1
        #y = x1
        swap = True
        #print("swap", "x1=", x1, "y1=", y1)
    else:
        swap = False

    x = x1
    y = y1


    # initial value of decision parameter d
    d = 2 * dy - dx


    # Calculate incremental value for going E and going NE
    incr_e = 2 * dy
    incr_ne = (2 * dy - 2 * dx)

    glBegin(GL_POINTS)
    if swap:
        glVertex2f(y, x)
    else:
        glVertex2f(x, y)
    glEnd()

    # Plot initial given point
    if swap:
        print(y,",",x)
    else:
        print(x,",",y)
    print("d=", d)

    x_coord_lst = [x]
    y_coord_lst = [y]


    while (x < x2):

        # Increase x
        x += 1

        # Make decision to move E or NE
        if (d <= 0):   # Move E
            d += incr_e
            print("Move E", "\n")
        else:    # Move NE
            d += incr_ne
            y += slope
            print("Move NE", "\n")

        # Set the point
        glBegin(GL_POINTS)
        if swap:
            glVertex2f(y, x)
        else:
            glVertex2f(x, y)
        glEnd()

        # Plot initial given point
        if swap:
            print(y,",",x)
        else:
            print(x,",",y)
        print("d=", d)

        x_coord_lst.append(x)
        y_coord_lst.append(y)


    if swap:
        plt.plot(y_coord_lst, x_coord_lst, 'o--')
        plt.plot([y1, y2], [x1, x2], '-')
    else:
        plt.plot(x_coord_lst, y_coord_lst, 'o--')
        plt.plot([x1, x2], [y1, y2], '-')

    plt.axis('equal')
    plt.grid(True)
    plt.show()
```

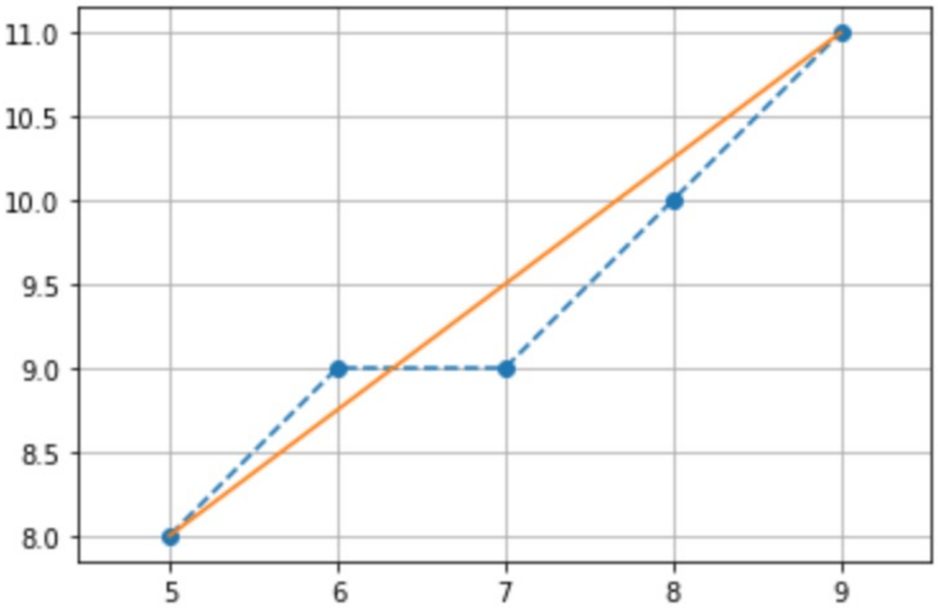
Rasterisation

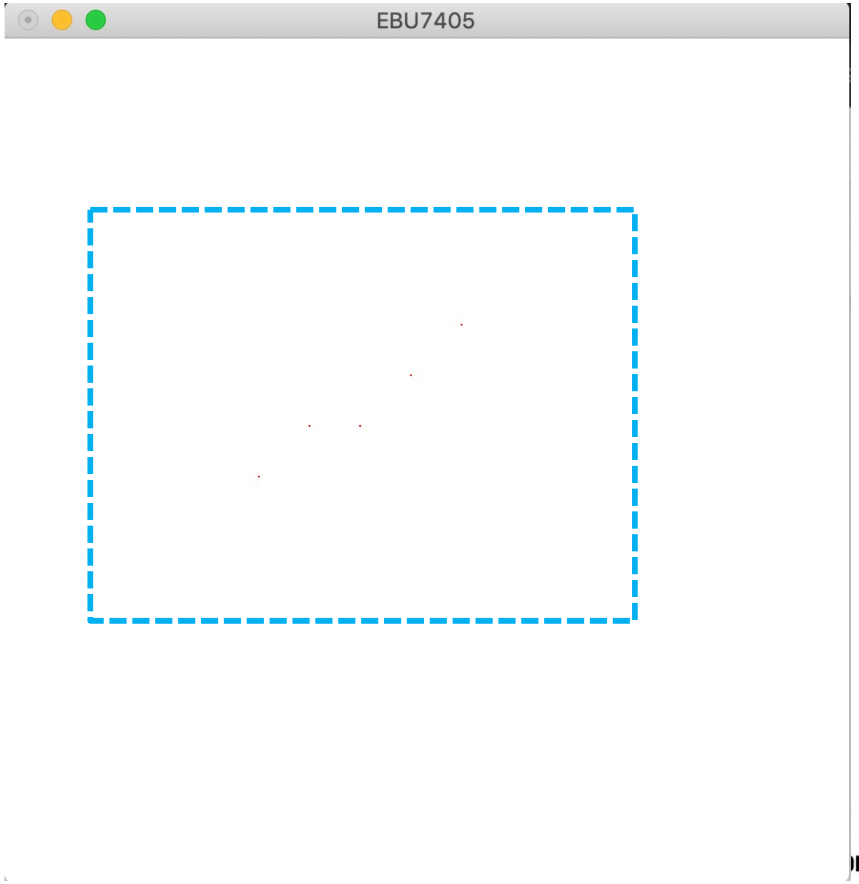
Bresenham's Midpoint Line Algorithm

- Demonstration

$$0 < m < 1$$

$$(x0, y0) = (5, 8); \qquad (x1, y1) = (9, 11)$$





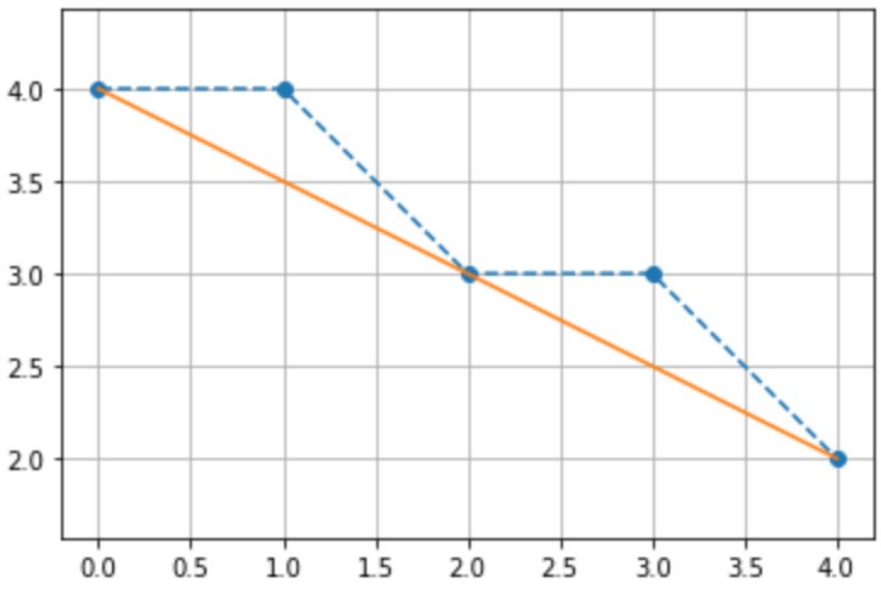
Mary
ndon

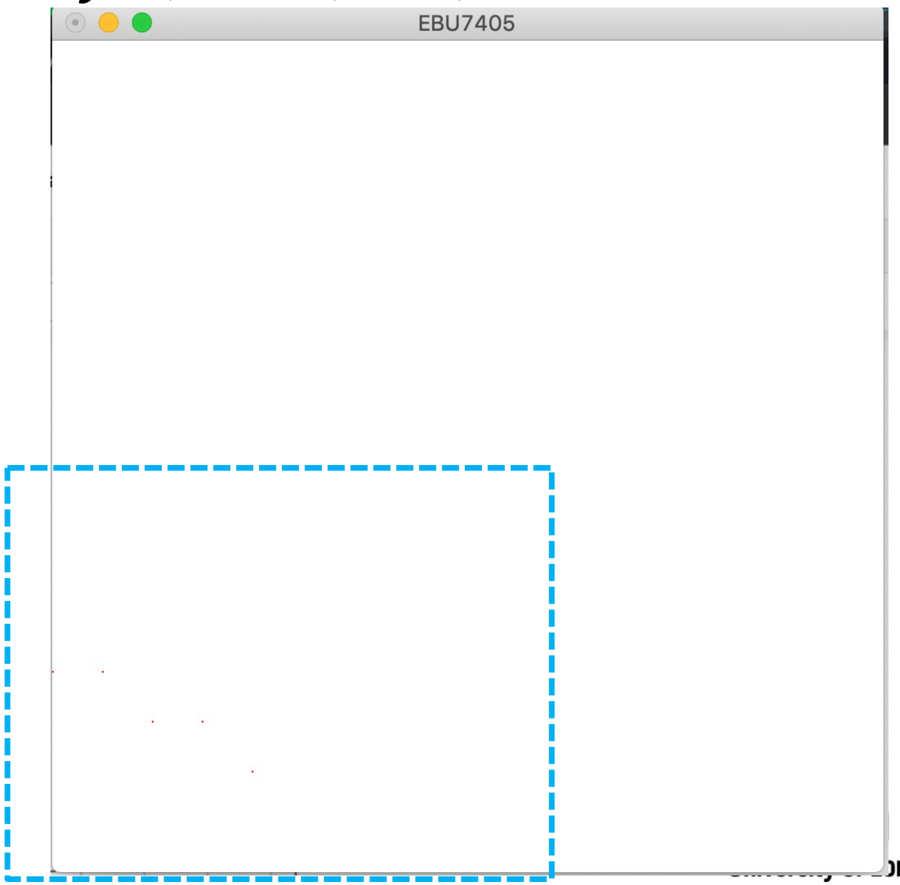
Bresenham's Midpoint Line Algorithm

- Demonstration

$$-1 < m < 0$$

$$(x0, y0) = (0, 4); \qquad (x1, y1) = (4, 2)$$





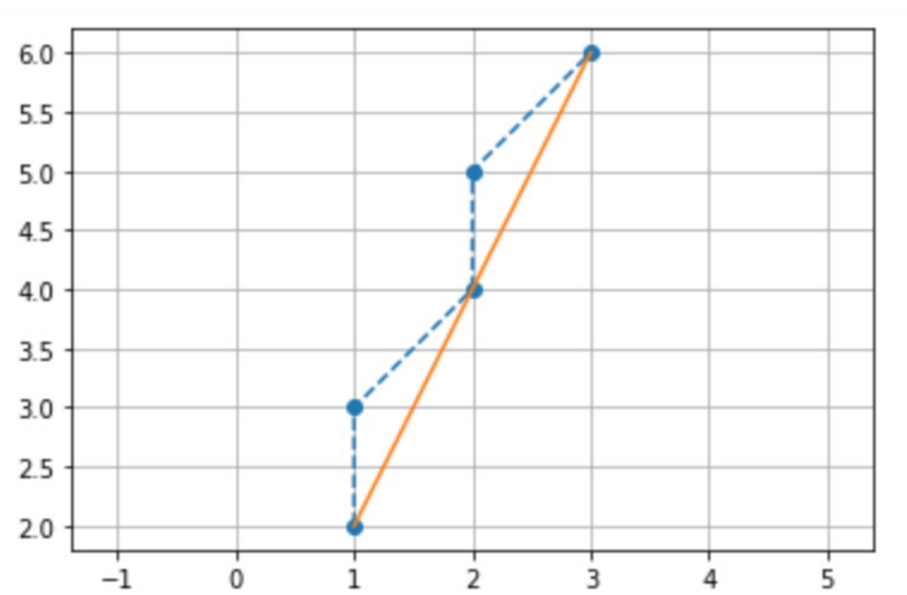
Rasterisation

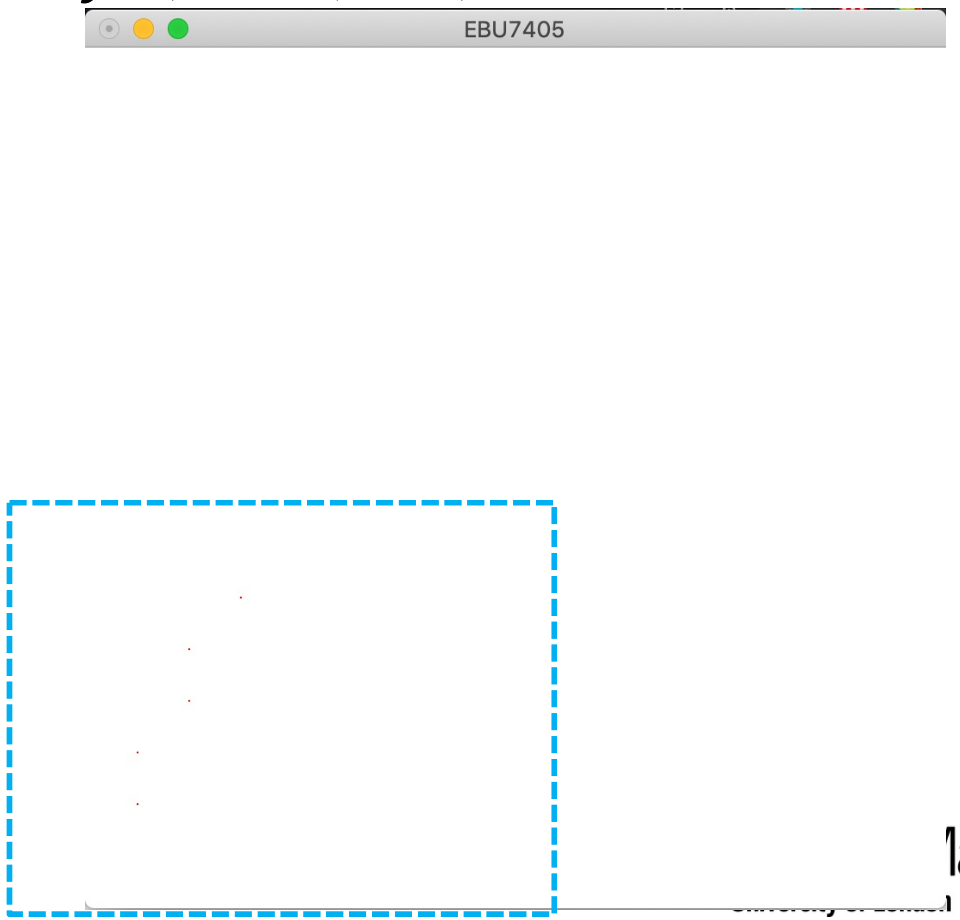
Bresenham's Midpoint Line Algorithm

- Demonstration

$$m > 1$$

$$(x0, y0) = (1, 2); \quad (x1, y1) = (3, 6)$$





Rasterisation

Edge Equations

b) This question is about rasterisation.

[9 marks]

i) For a line from (x0, y0) to (x1, y1), give the implicit function as the line equation F(x,y)=0. Give the normal vector of the line and show which half-plane makes F(x,y)>0 and which half-plane makes F(x,y)<0.

(5 marks)

Solution:

F(x,y) = (y1 – y0)(x – x0) – (x1 – x0)(y – y0) (2 marks)

The normal vector of the line is [(y1 – y0), –(x1 – x0)]. (1 mark)

In the normal vector positive side is positive: dot product of the normal vector and the vector from a point on the line to a point in the half-plane, F(x,y)>0. (1 mark)

In the normal vector negative side is negative: dot product of the normal vector and the vector from a point on the line to a point in the half-plane, F(x,y)<0. (1 mark)

(total 5 *marks*)

ii) Give a method to test if a point is inside or outside a triangle.

(4 marks)

Solution:

3 steps:

Put the vertices in a right order such that the inside is always in positive (or negative) half-plane. (2 marks)

Test if the point makes the three line equations of the three edges positive (or negative). (1 mark)

If all positive (or negative), it is inside, otherwise it is outside the triangle. (1 mark)

(total 4 *marks*)

Edge Equations

Edge equations

- Edge equation → the equation of the line defining that edge
  - Implicit equation of a line

    $$Ax + By + C = 0$$

  - Given a point (**x**,**y**), plugging **x** & **y** into this equation tells us whether the point is:
    - on the line: $Ax + By + C = 0$
    - "above" the line: $Ax + By + C > 0$
    - "below" the line: $Ax + By + C < 0$

Rasterisation

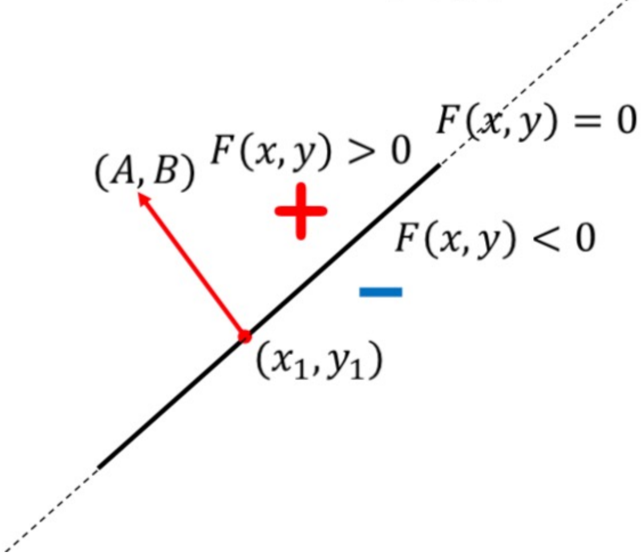
Edge Equations

Edge equations

- Edge equations thus define two *half-spaces*:

2D line equation:

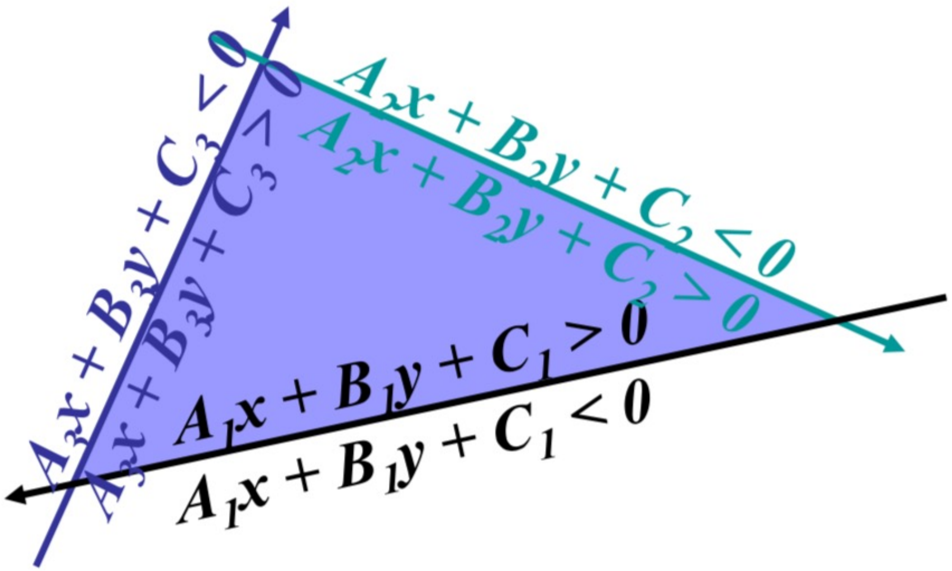
$$F(x, y) = A(x - x_1) + B(y - y_1) = 0$$

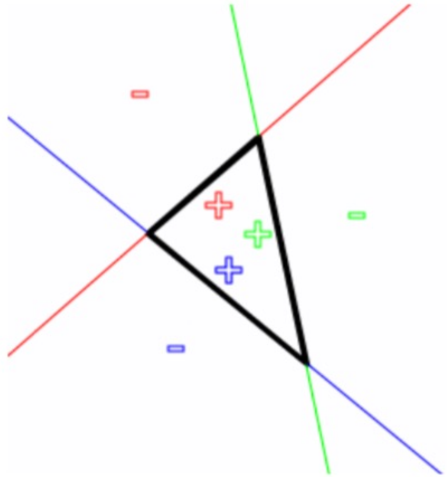


Edge Equations

Edge equations

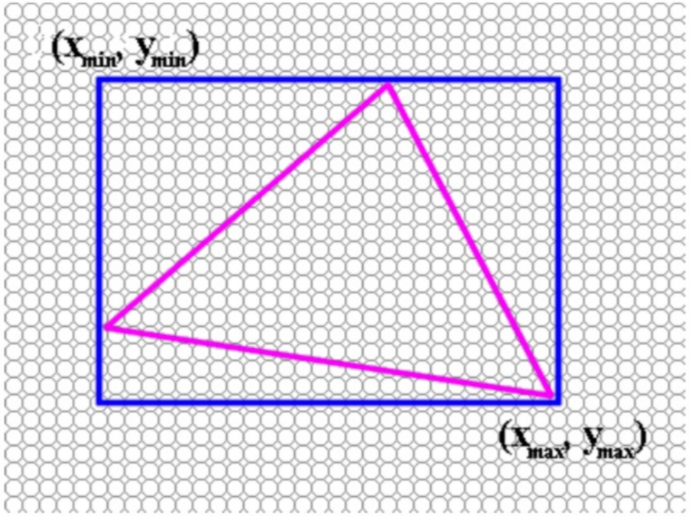
- For an edge of 2 vertices, take the third vertex as in the positive half-space, a triangle can be defined as the intersection of three positive half-spaces





Edge Equations

Edge equations



- We can find edge equation from two vertices
- Given three corners $\mathbf{P_0, P_1, P_2}$ of a triangle, what are our three edges?
- *To make sure that the half-spaces defined by the edge equations all share the same sign on the interior of the triangle*

  → Be consistent (Ex: $[\mathbf{P_0\,P_1}], [\mathbf{P_1\,P_2}], [\mathbf{P_2\,P_0}]$)
- *To make sure that sign is positive?* $Ax + By + C = 0$

  → Test, and flip if needed ($A= -A$, $B= -B$, $C= -C$)

Rasterisation

Edge Equations

- Prerequisite: Line Equations
  - Slope (constant for lines)

    a = rise / run

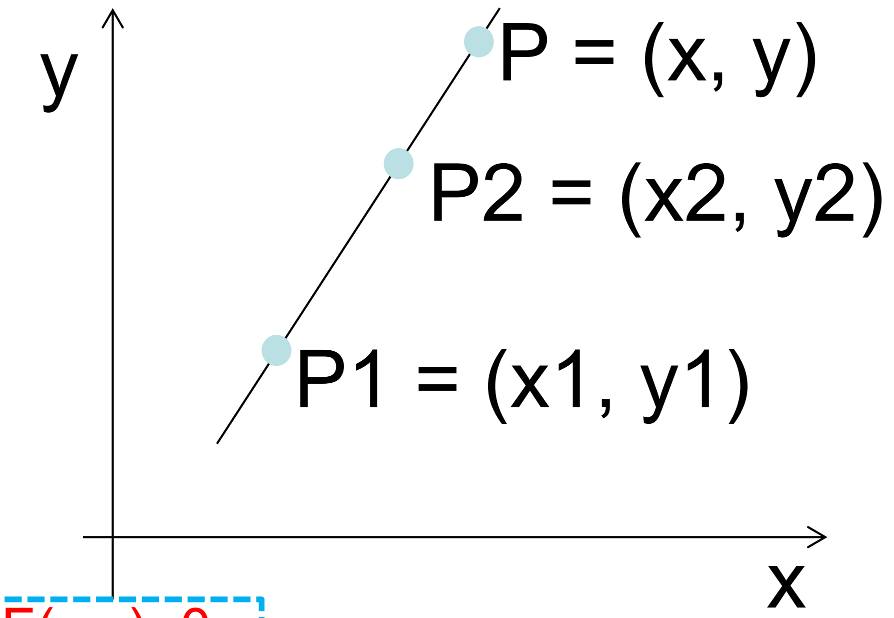
    $a = (y - y1) / (x - x1)$

    $a = (y2 - y1) / (x2 - x1)$
  - Implicit form

$(y - y1) / (x - x1) = (y2 - y1) / (x2 - x1)$

$(x2 - x1)(y - y1) - (y2 - y1)(x - x1) = 0$, or $F(x,y)=0$

if the coordinates of the points are all integers, the coefficients of F(x,y) can all be integers.



  - Explicit form by solving for y

$y = ((y2 - y1) / (x2 - x1)) (x – x1) + y1$

$y = ((y2 - y1)/(x2 - x1))x – ((y2-y1)/(x2 - x1))x1 + y1$

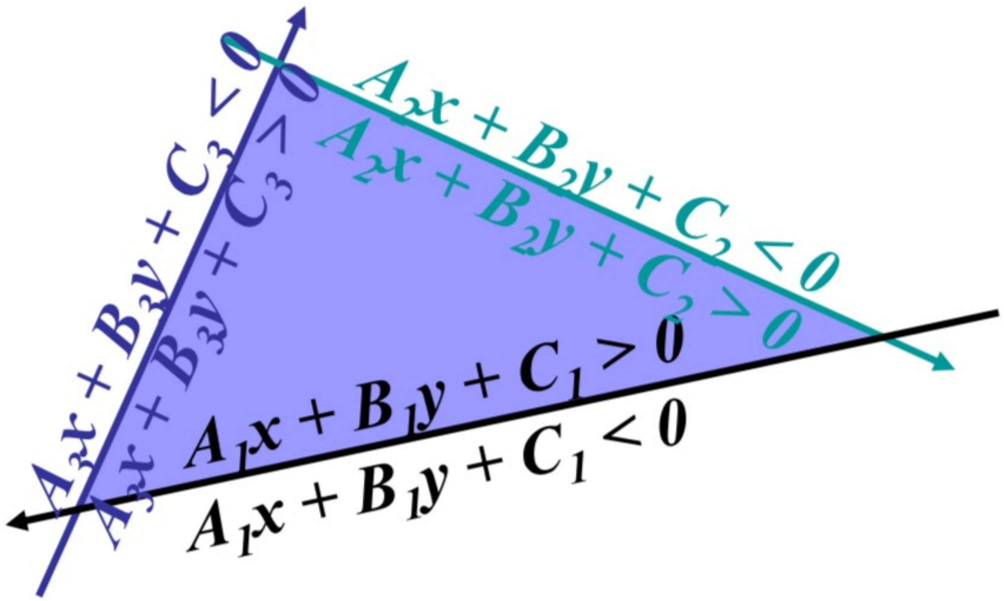
or **$y = ax + b$** where $a = (y2 – y1) / (x2 – x1)$

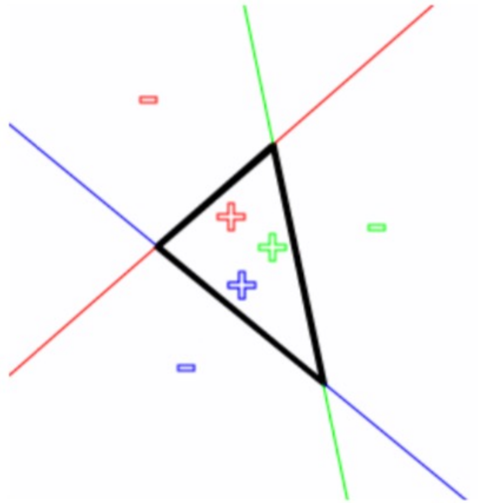
and $b = – ax1 + y1$

Rasterisation

Edge Equations

- Prerequisite: Half-Spaces
  - For an edge of 2 vertices, take the third vertex as in the positive half-space, a triangle can be defined as the intersection of three positive half-spaces





Rasterisation

Edge Equations

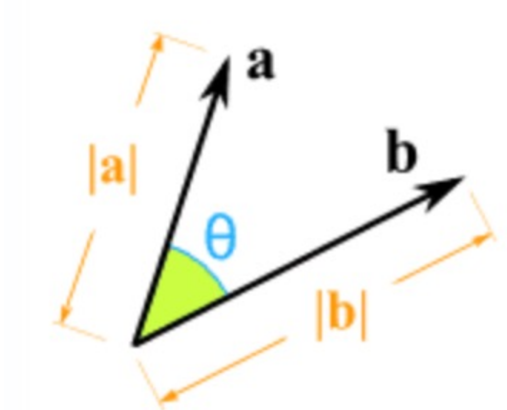
- Prerequisite: Vector Dot Product

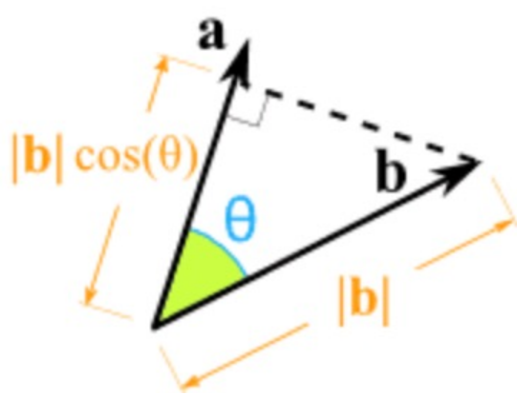
a • b

Calculate in an algebraic way $\quad \mathbf{a} \cdot \mathbf{b} = a_x \times b_x + a_y \times b_y + a_z \times b_z$

The fact that we know **a · b** can be calculated in two ways could be useful!

Calculate in a geometric way $\quad \mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \times |\mathbf{b}| \times \cos(\theta)$





$$cos\theta = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} \in [-1, 1]$$

Similarity between two vectors

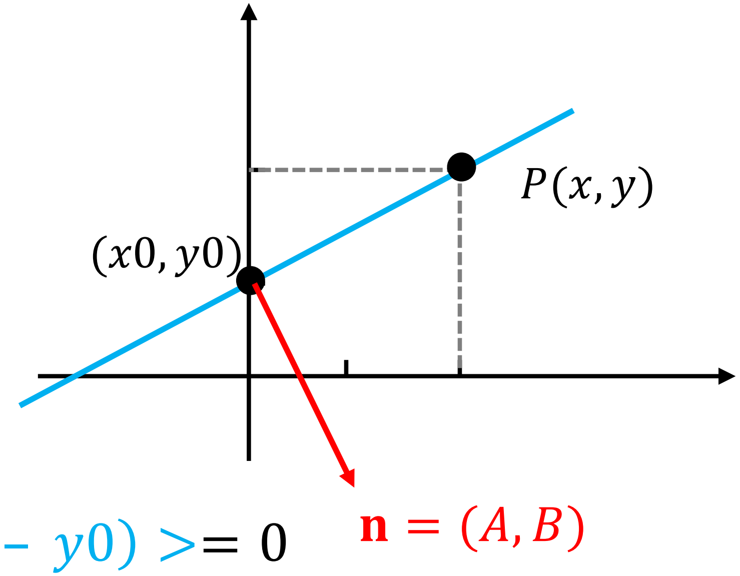
Queen Mary
University of London

Rasterisation

Edge Equations

- Normal Vector of a Line

$$F(x,y) = \underline{(y1 - y0)}\underline{(x - x0)} - \underline{(x1 - x0)}\underline{(y - y0)} = 0$$



It can also be written as

$$F(x,y) = \underbrace{< (y1 - y0), -(x1 - x0) >}_{\text{Normal vector the line}} \cdot \underbrace{< (x - x0), (y - y0) >}_{\text{Vector along the line}} = 0$$

$F(x,y) = Ax + By + C = 0$ Normal vector is $\mathbf{n} = (A, B)$

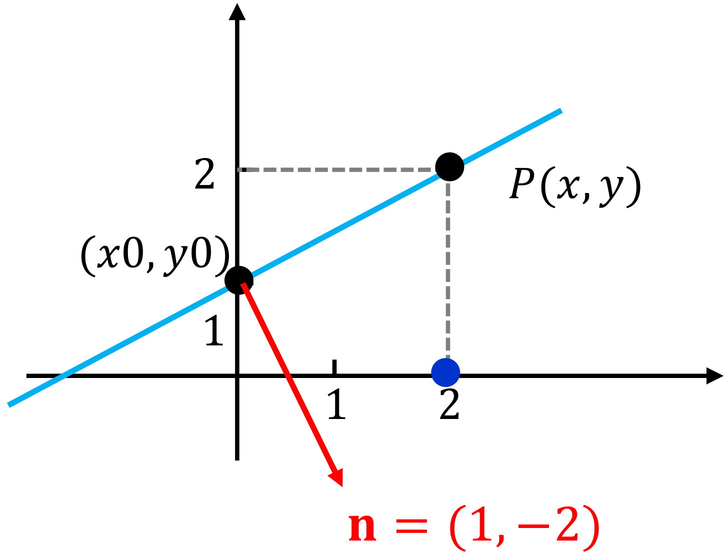
Rasterisation

Edge Equations

- Normal Vector of a Line
  - Example

$$y = \frac{1}{2}x + 1 \qquad F(x,y) = x - 2y + 2 = 0$$

$$\boldsymbol{n} = (1,-2) \qquad F(x,y) = x - 2y + 2 = 0$$
$$or\ \boldsymbol{n} = (-1,2) \qquad F(x,y) = -x + 2y - 2 = 0$$

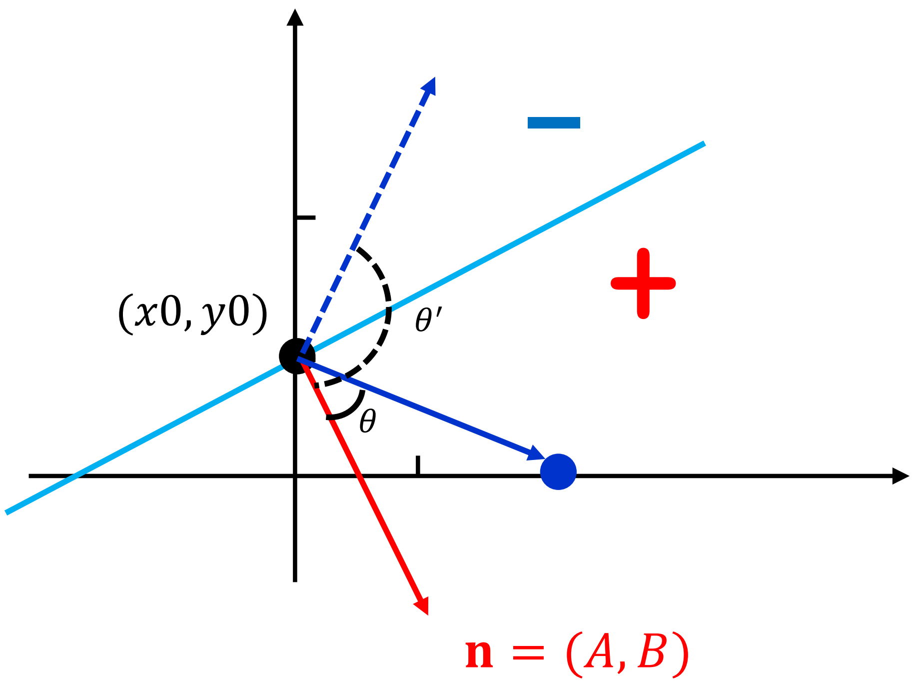


- If $\boldsymbol{n} = (1,-2)$, for point (2, 0), $F(x,y) = 4 > 0$, Positive half-space
- If $\boldsymbol{n} = (-1,2)$, for point (2, 0), $F(x,y) = -4 < 0$, Negative half-space

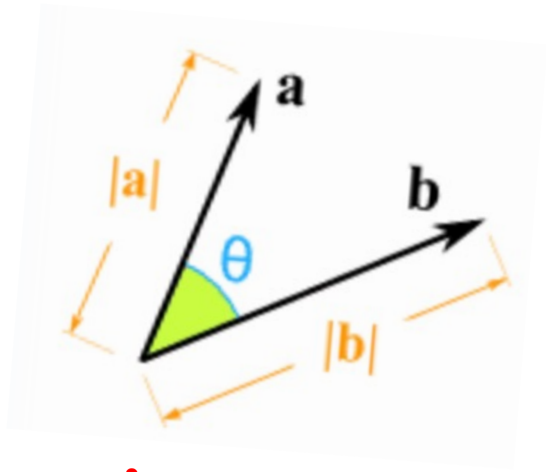
Rasterisation

Edge Equations

- Determine if a point is in the positive half-space



$$\mathbf{a} \cdot \mathbf{b} = a_x \times b_x + a_y \times b_y + a_z \times b_z$$



$$cos\theta = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|} \in [-1, 1]$$

+ $\theta < 90°$: $\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|cos\theta > 0$

− $\theta > 90°$: $\boldsymbol{a} \cdot \boldsymbol{b} = |\boldsymbol{a}||\boldsymbol{b}|cos\theta < 0$

Queen Mary University of London

Rasterisation

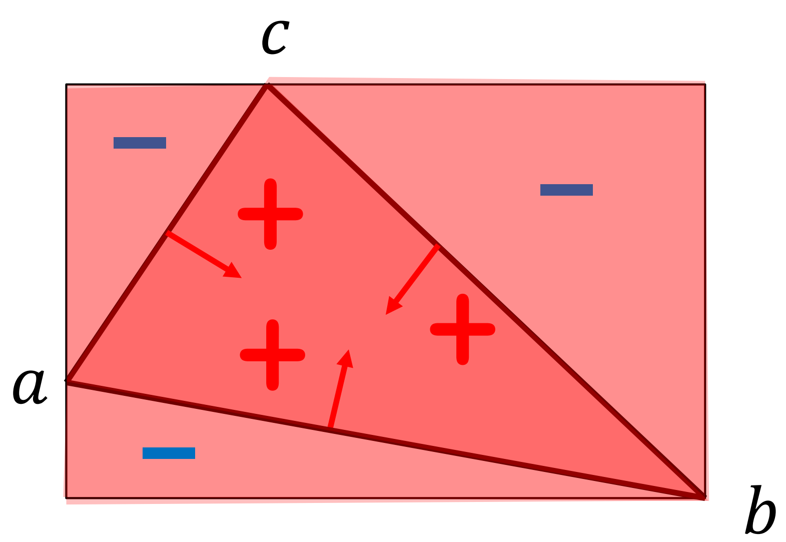
Edge Equations

- Determine if a point is inside a triangle

$$(\mathbf{x}-\mathbf{a})\cdot(\mathbf{b}-\mathbf{a})^{\perp}>0$$

$$(\mathbf{x}-\mathbf{b})\cdot(\mathbf{c}-\mathbf{b})^{\perp}>0$$

$$(\mathbf{x}-\mathbf{c})\cdot(\mathbf{a}-\mathbf{c})^{\perp}>0$$



Rasterisation

Edge Equations

- Revisit the solution

b) This question is about rasterisation.

[9 marks]

i) For a line from (x0, y0) to (x1, y1), give the implicit function as the line equation F(x,y)=0. Give the normal vector of the line and show which half-plane makes F(x,y)>0 and which half-plane makes F(x,y)<0.

(5 marks)

Solution:

F(x,y) = (y1 – y0)(x – x0) – (x1 – x0)(y – y0) (2 marks)

The normal vector of the line is [(y1 – y0), –(x1 – x0)]. (1 mark)

In the normal vector positive side is positive: dot product of the normal vector and the vector from a point on the line to a point in the half-plane, F(x,y)>0. (1 mark)

In the normal vector negative side is negative: dot product of the normal vector and the vector from a point on the line to a point in the half-plane, F(x,y)<0. (1 mark)

(total 5 *marks*)

ii) Give a method to test if a point is inside or outside a triangle.

(4 marks)

Solution:

3 steps:

Put the vertices in a right order such that the inside is always in positive (or negative) half-plane. (2 marks)

Test if the point makes the three line equations of the three edges positive (or negative). (1 mark)

If all positive (or negative), it is inside, otherwise it is outside the triangle. (1 mark)

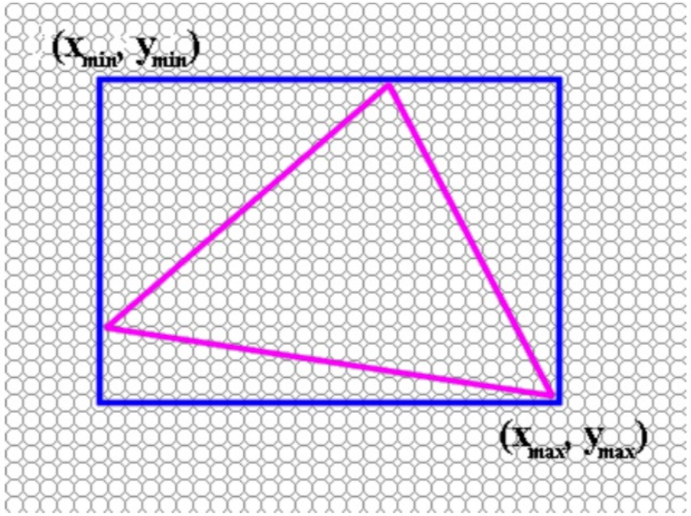
(total 4 *marks*)

Rasterisation

Edge Equations

- An Edge Equation Rasteriser

Edge equations



- We can find edge equation from two vertices
- Given three corners $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$ of a triangle, what are our three edges?
- *To make sure that the half-spaces defined by the edge equations all share the same sign on the interior of the triangle*

  → Be consistent (Ex: $[\mathbf{P}_0\,\mathbf{P}_1], [\mathbf{P}_1\,\mathbf{P}_2], [\mathbf{P}_2\,\mathbf{P}_0]$)
- *To make sure that sign is positive?* $Ax + By + C = 0$

  → Test, and flip if needed ($A= -A, B= -B, C= -C$)

Queen Mary University of London

Questions

c.shu@qmul.ac.uk

Queen Mary
University of London