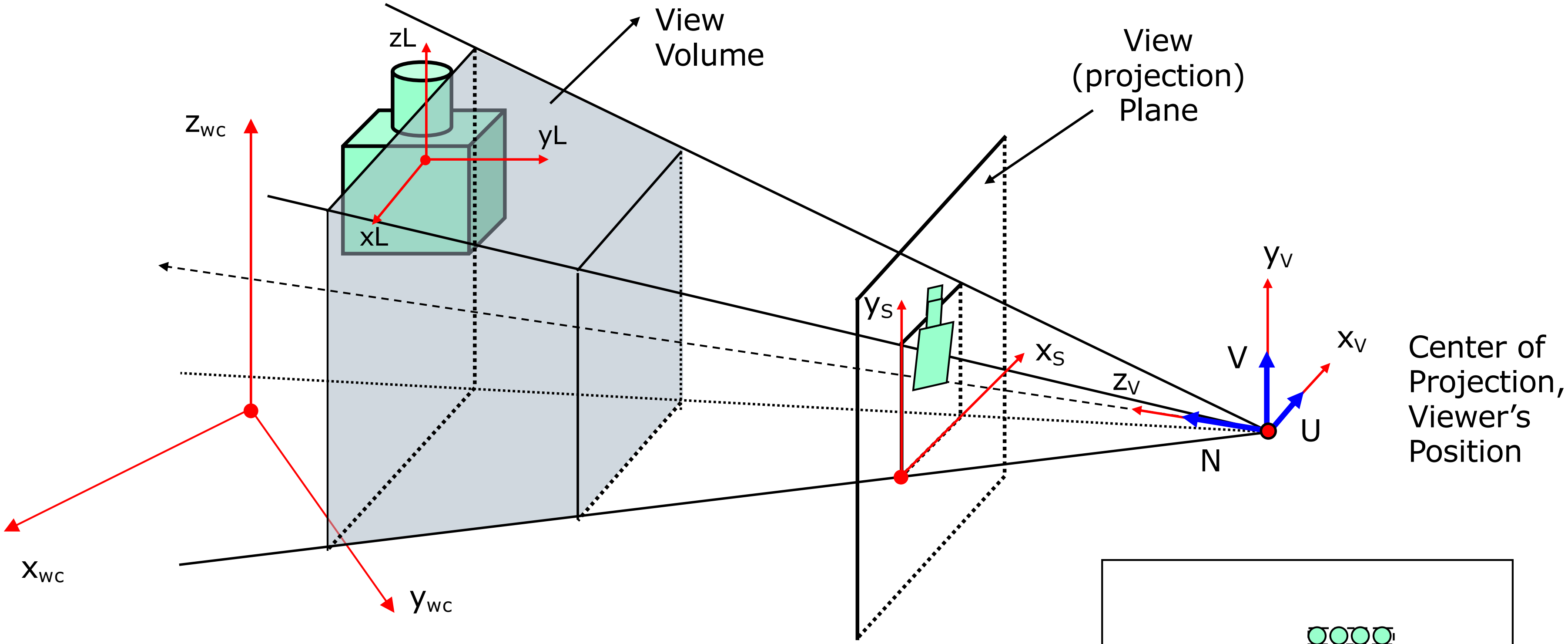
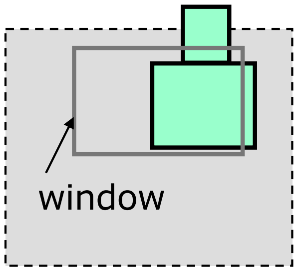
ScanLine Conversion Algorithms

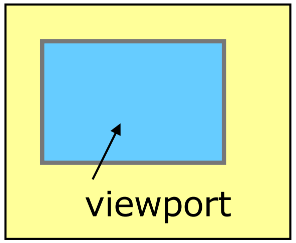
Contents

Real objects to image on the screen

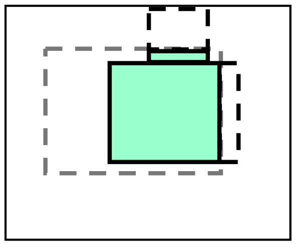




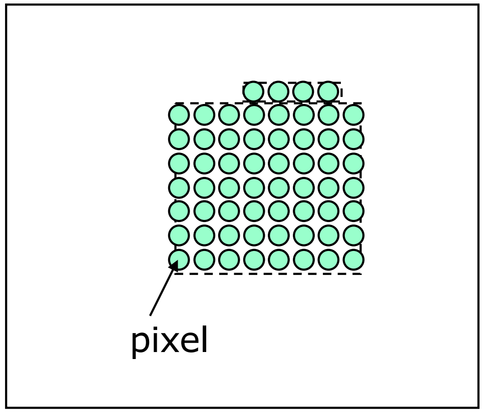
Window definition in projection plane



Viewport definition in projection plane

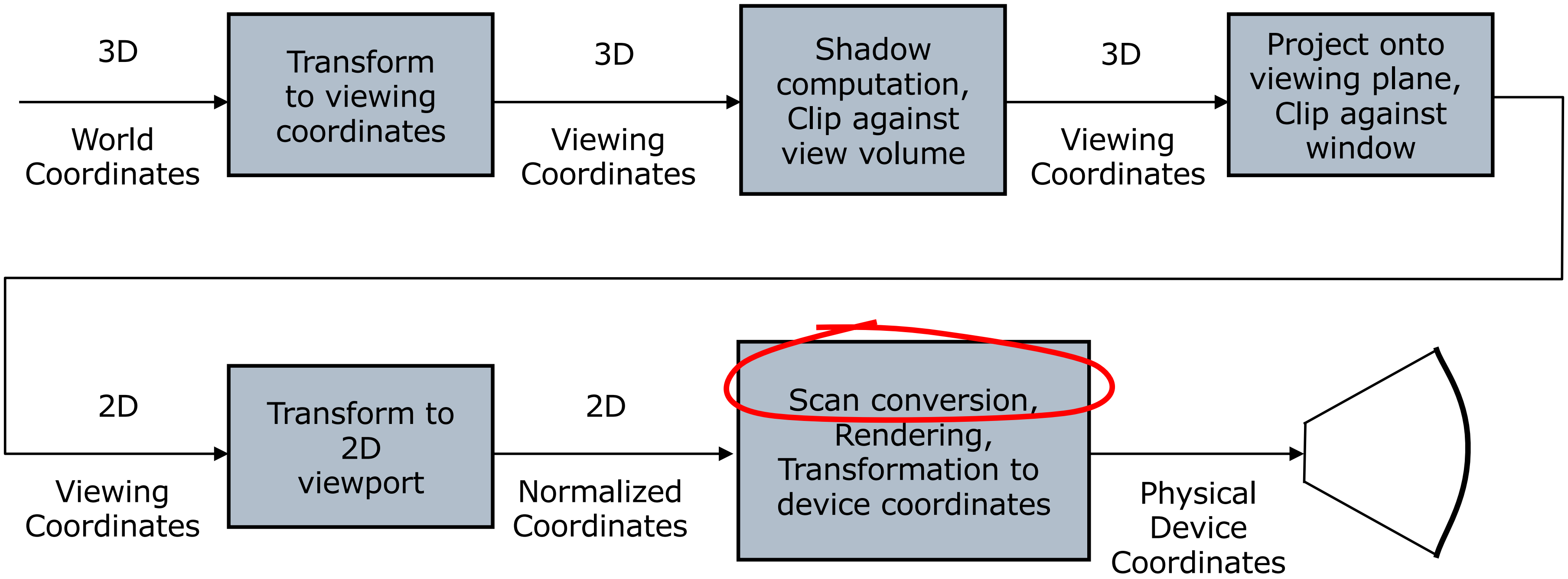


Window's content inside the viewport

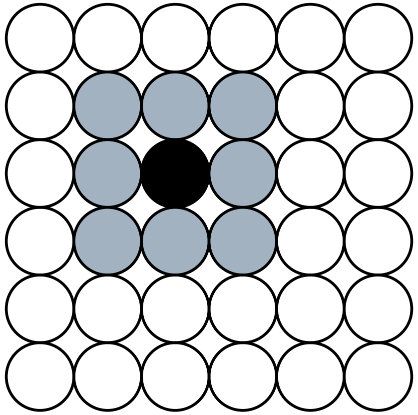


Scan conversion onto the raster screen

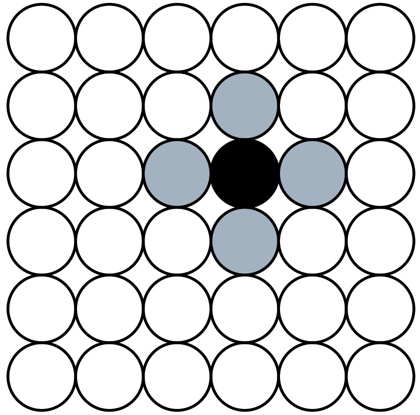
Viewing transformations pipeline



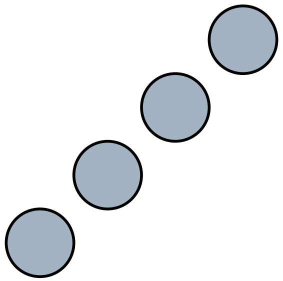
Pixel vicinity and continuity

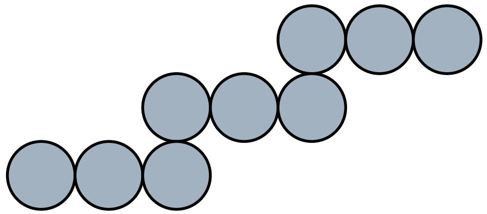


8 neighbors



4 neighbors

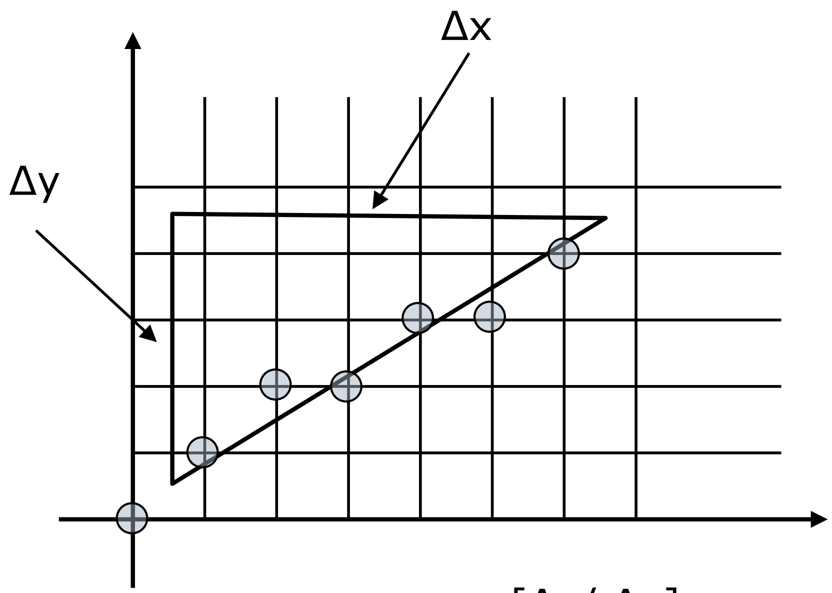




Line drawing

```
var yt : real; Δx, Δy, xi, yi : integer;
for xi := 0 to Δx do begin
      yt := [Δy/Δx]*xi;
      yi := trunc(yt+[1/2]);
      display(xi,yi);
end;
```

Objective: avoid multiplication



y = mx, m = [Δy/ Δx]

Line-drawing algorithms

$y = mx + b$

$b = y1 - mx1$

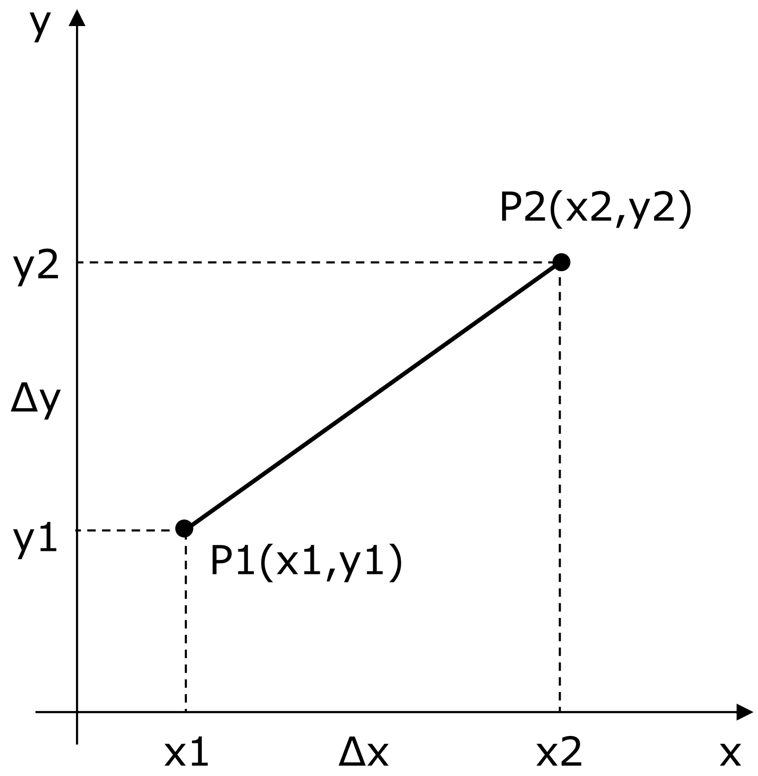
$m = (y2-y1)/(x2-x1) = \Delta y/\Delta x$

$\Delta y = m\Delta x$

Assumptions:

$m > 0$

$X1 < x2$



DDA Algorithm

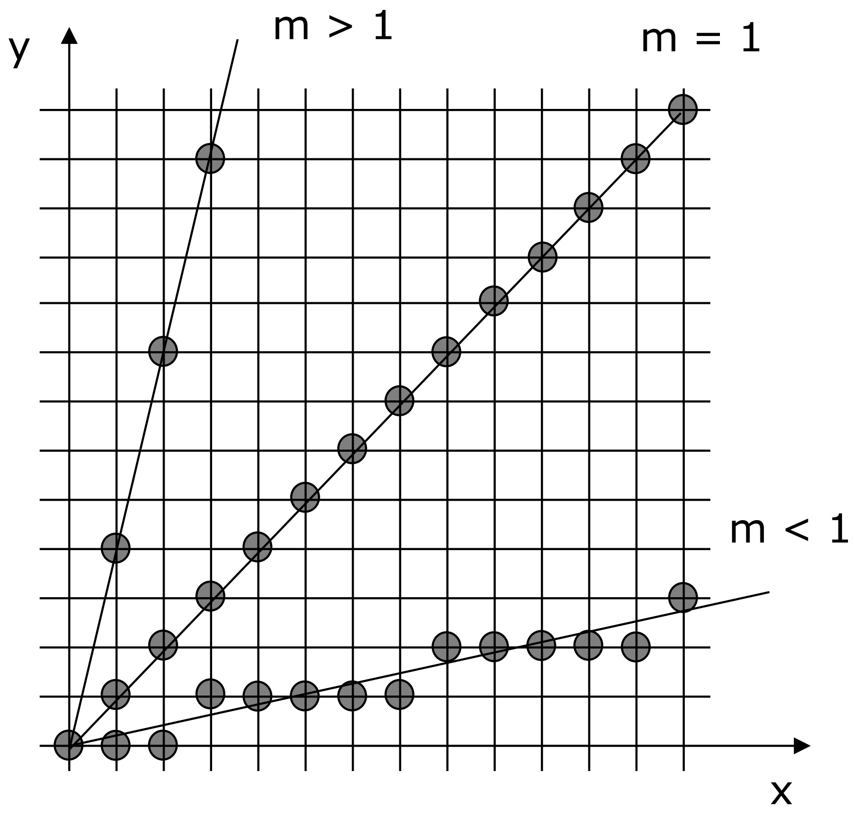
DDA = Digital Differential Analyzer

Basic formula:

$\Delta y = m\Delta x,\ m>0$

$x_{i+1} = x_i + 1$

$y_{i+1} = y_i + m$



Makes discontinuity for another case

Therefore two formulas:

$\Delta y = m\Delta x,\ 0<m<1$

$\Delta x = (1/m)\Delta y,\ 1<m$

DDA Algorithm – two cases

Makes discontinuity for another case

Therefore two formulas:

$\Delta y = m\Delta x$, $0<m<1$

$\Delta x = (1/m)\Delta y$, $1<m$

Case 1: $0<m<1$

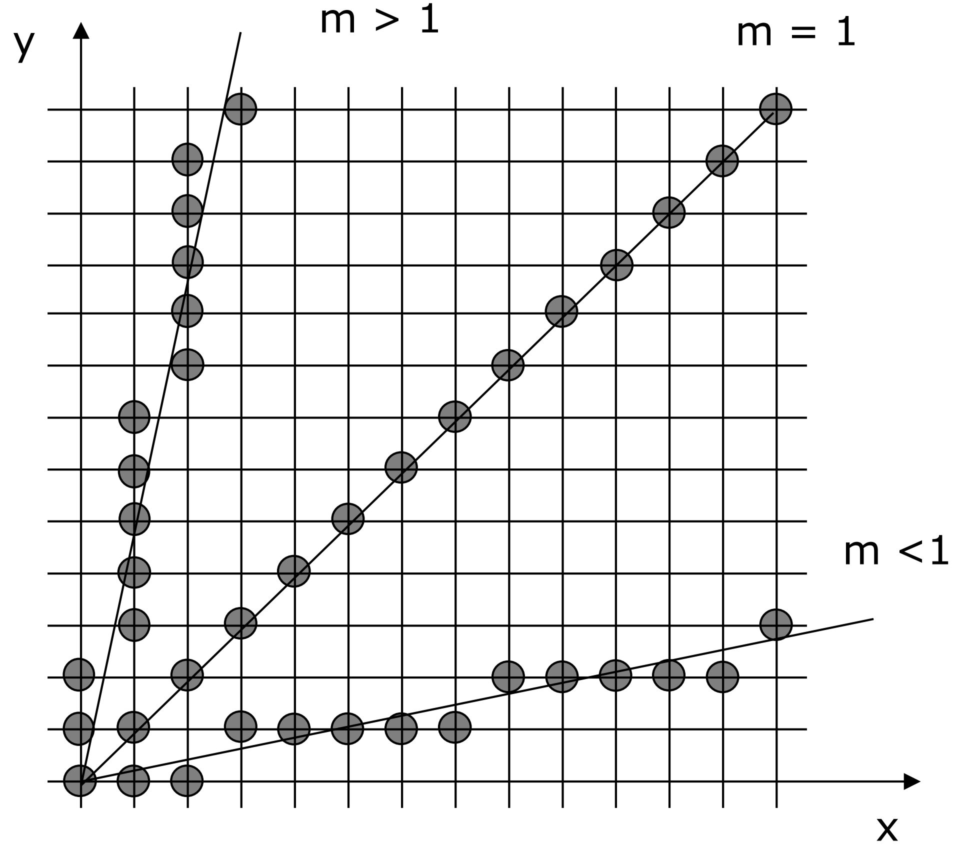
$X_{i+1} = x_i + 1$

$y_{i+1} = y_i + m$

Case 2: $1<m$

$y_{i+1} = y_i + 1$

$x_{i+1} = x_i + 1/m$



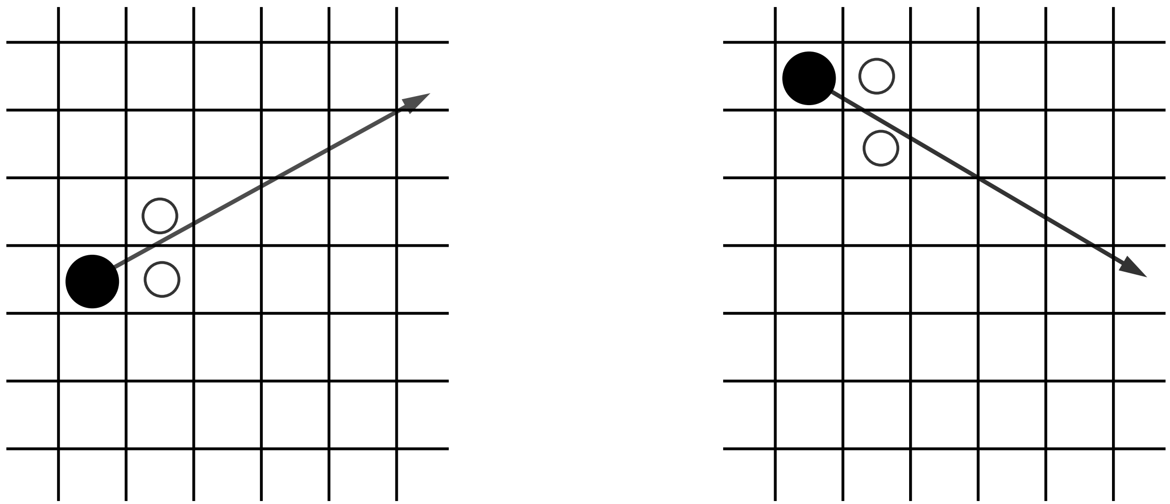
DDA Algorithm – basic case

Assumption : 0 <m<1, x1<x2

```
procedure dda (x1, y1, x2, y2 : integer);
  var
       Δx, Δy, k : integer;
        x, y : real
  begin
       Δx := x2 - x1;
       Δy := y2 - y1;
        x := x1; y := y1;
        display(x,y);
        for k := 1 to Δx do begin
             x := x + 1;
              y := y + [Δy/ Δx];
              display(round(x),round(y));
        end { for k }
   end; { dda }
```

Bresenham's Line Algorithm

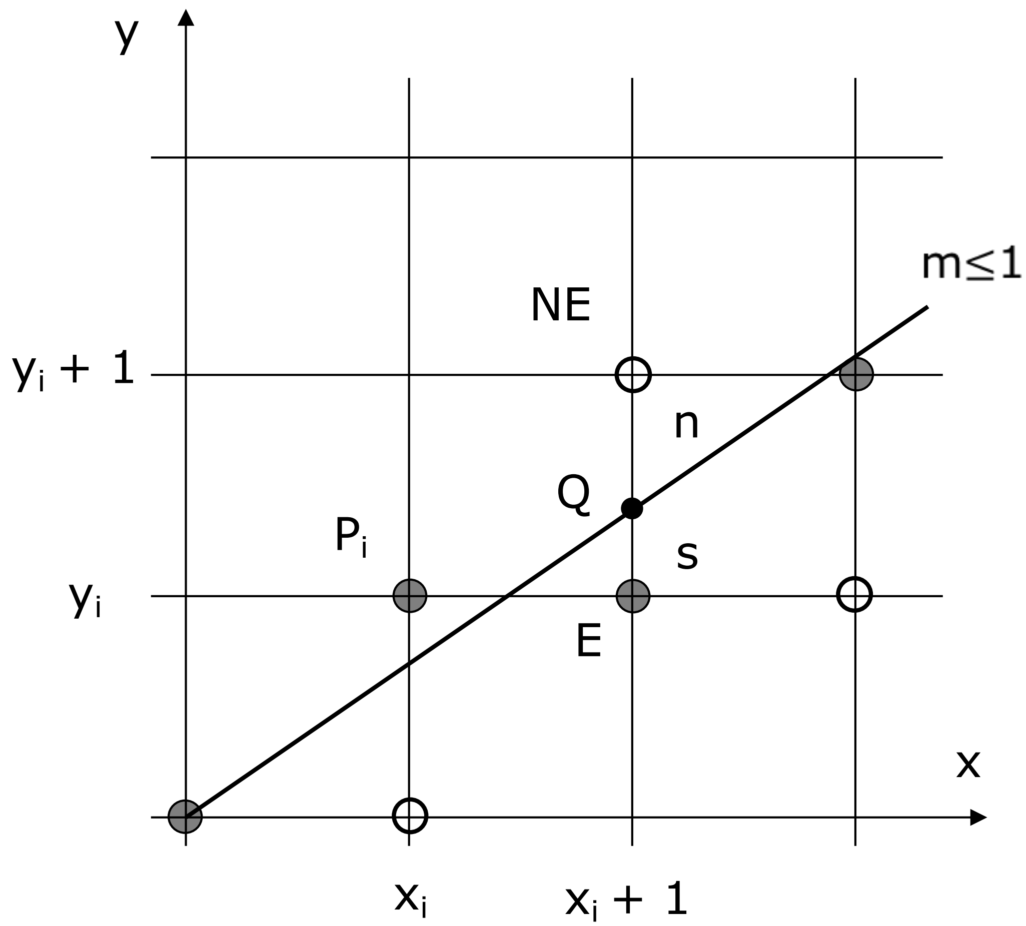
- Basic idea:
  1. Find the closest integer coordinates to the actual line path
  2. Use only integer arithmetic
  3. Compute iteratively the next point, $P_{i+1}=F(P_i)$



Bresenham method – basic idea

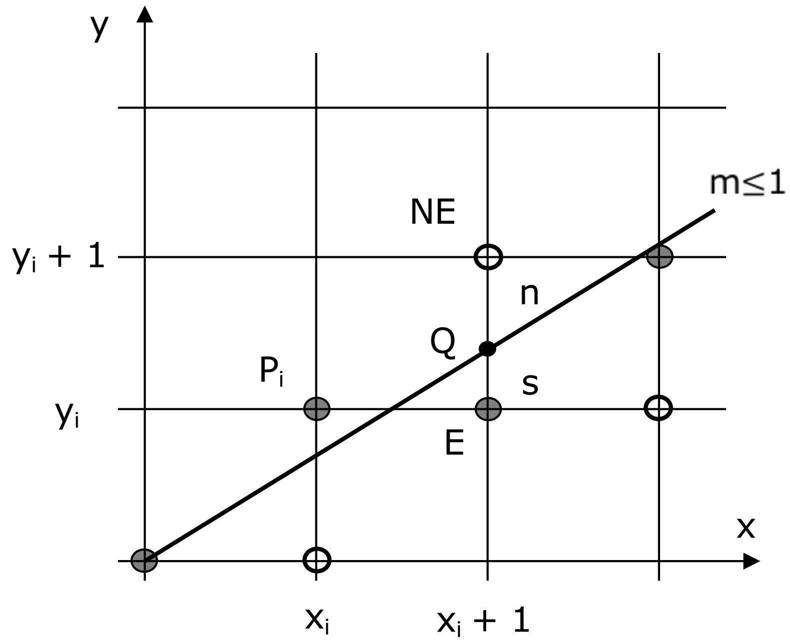
- Basic approach:

1. Compute a decision variable $d_i=f(P_i, d_{i-1})$
2. Depending on the d_i value chose the next position E or NE



Bresenham method - mathematics

Assumptions: $0<m\leq 1$, $x_i<x_j$, $i<j$



$y = mx+b$, $y_i = mx_i$, $m = \Delta y/\Delta x$

$s = y-y_i = m(x_i+1)+b-y_i$

$n = y_i+1-y=y_i+1-m(x_i+1)-b$

$k = s-n = 2m(x_i+1)-2y_i+2b-1$

Let us consider $d_i = k\Delta x = \Delta x(s-n)$

$$
\begin{aligned}
d_i = \Delta x(s-n) &= 2\Delta y(x_i+1)-2\Delta x y_i+\Delta x(2b-1) \\
&= 2\Delta y x_i -2\Delta x y_i +[2\Delta y+\Delta x(2b-1)] \\
&= 2\Delta y x_i -2\Delta x y_i + Const
\end{aligned}
$$

$d_i = 2\Delta y x_i -2\Delta x y_i + Const$

if $d_i < 0$ then $E(x_i+1,y_i)$

if $d_i \geq 0$ then $NE(x_i+1,y_i+1)$

Bresenham method - mathematics

$d_i = 2\Delta y x_i - 2\Delta x y_i + Const$

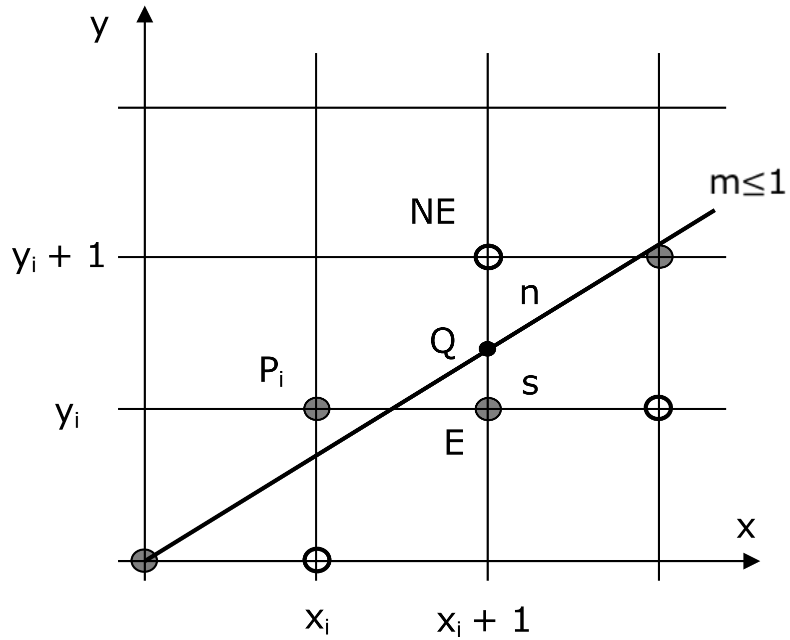
$d_{i+1} = 2\Delta y x_{i+1} - 2\Delta x y_{i+1} + Const$

$d_{i+1} - d_i = 2\Delta y(x_{i+1} - x_i) - 2\Delta x(y_{i+1} - y_i)$

$d_{i+1} = d_i + 2\Delta y - 2\Delta x(y_{i+1} - y_i)$

if $d_i < 0$ then $P_{i+1} = E$, and $y_{i+1} = y_i$

otherwise $P_{i+1} = NE$, and $y_{i+1} = y_i + 1$



Therefore

$d_{i+1} = d_i + 2\Delta y$ if $d_i < 0$

$d_{i+1} = d_i + 2\Delta y - 2\Delta x$ otherwise

The first value of the decision variable:

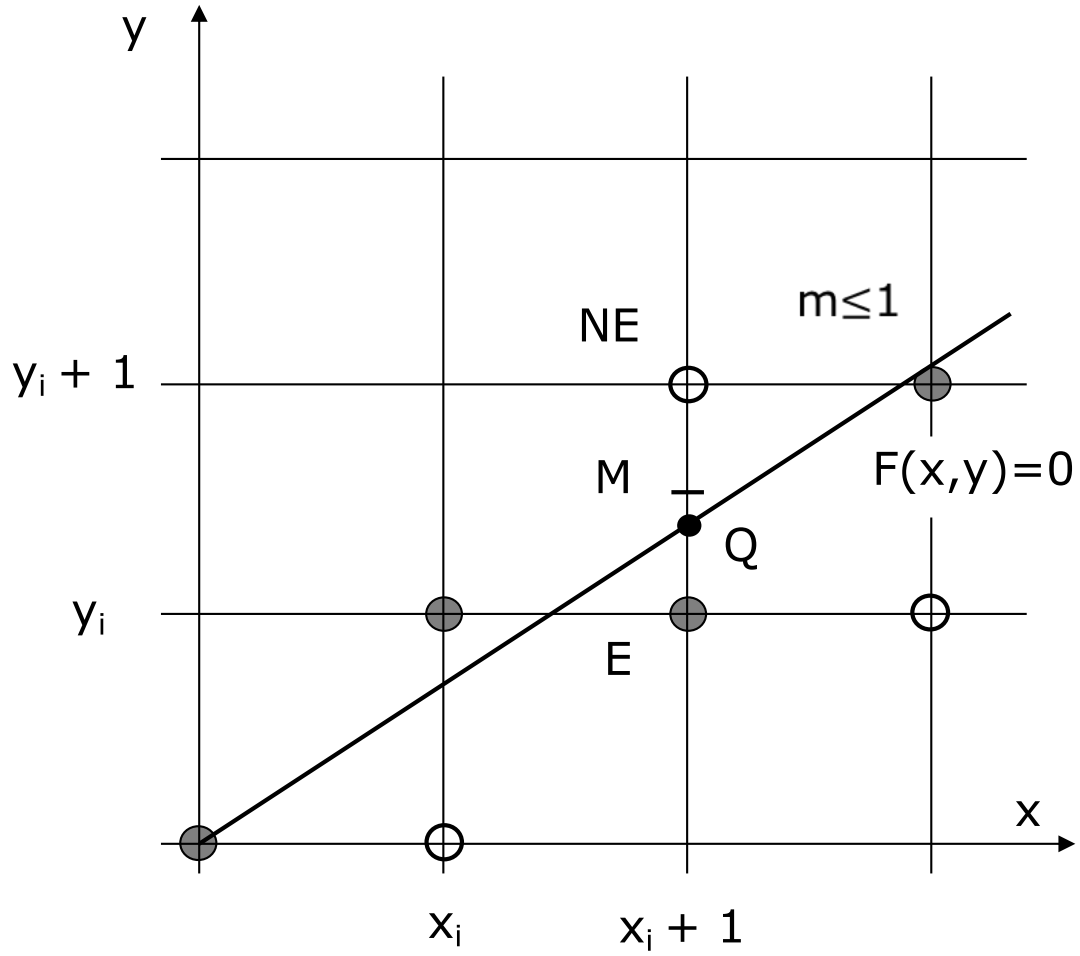
$d_0 = 2\Delta y x_1 - 2\Delta x y_1 + [2\Delta y + \Delta x(2b-1)]$

$y_1 = (\Delta y/\Delta x)x_1 + b$, $\Delta x y_1 = \Delta y x_1 + b\Delta x$

$\Delta x(2b-1) = 2\Delta x y_1 - 2\Delta y x_1 - 2\Delta x$

$d_0 = 2\Delta y - 2\Delta x$

Midpoint line algorithm



Midpoint line algorithm

$y = (\Delta y/\Delta x)x + n$

$F(x,y) = x\Delta y - y\Delta x + n\Delta x$

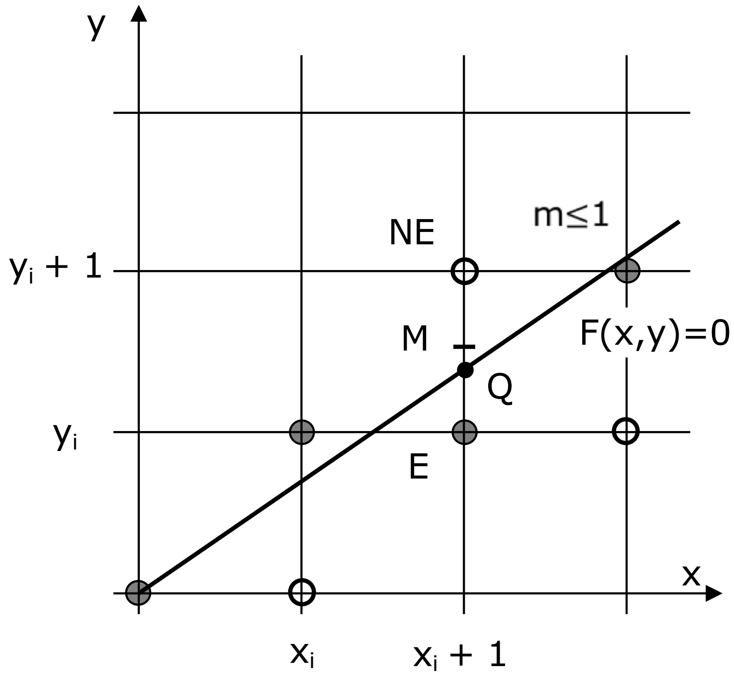
Considering $a=\Delta y$, $b=-\Delta x$, and $c=n\Delta x$

$F(x,y)=ax+by+c=0$, if (x,y) is on the line

Q is closer to

NE if $F(x_i+1, y_i+1/2)<0$

E if $F(x_i+1, y_i+1/2)>0$



Midpoint line algorithm

$F(x_i+1, y_i+1/2)=a(x_i+1)+b(y_i+1/2)+c$

Let $d_i=2F(x_i+1, y_i+1/2)=2a(x_i+1)+b(2y_i+1)+2c$

Since $a=\Delta y$ and $b=-\Delta x$,

$d_i=2\Delta y(x_i+1)-\Delta x(2y_i+1)+2c$

$d_{i+1}=2F(x_{i+1}+1, y_{i+1}+1/2)= 2\Delta y(x_{i+1}+1)-\Delta x(2y_{i+1}+1)+2c$

$x_{i+1}= x_i+1$

$y_{i+1}= y_i+1$, if $d_i \geq 0$

$y_{i+1}= y_i$ otherwise

Therefore

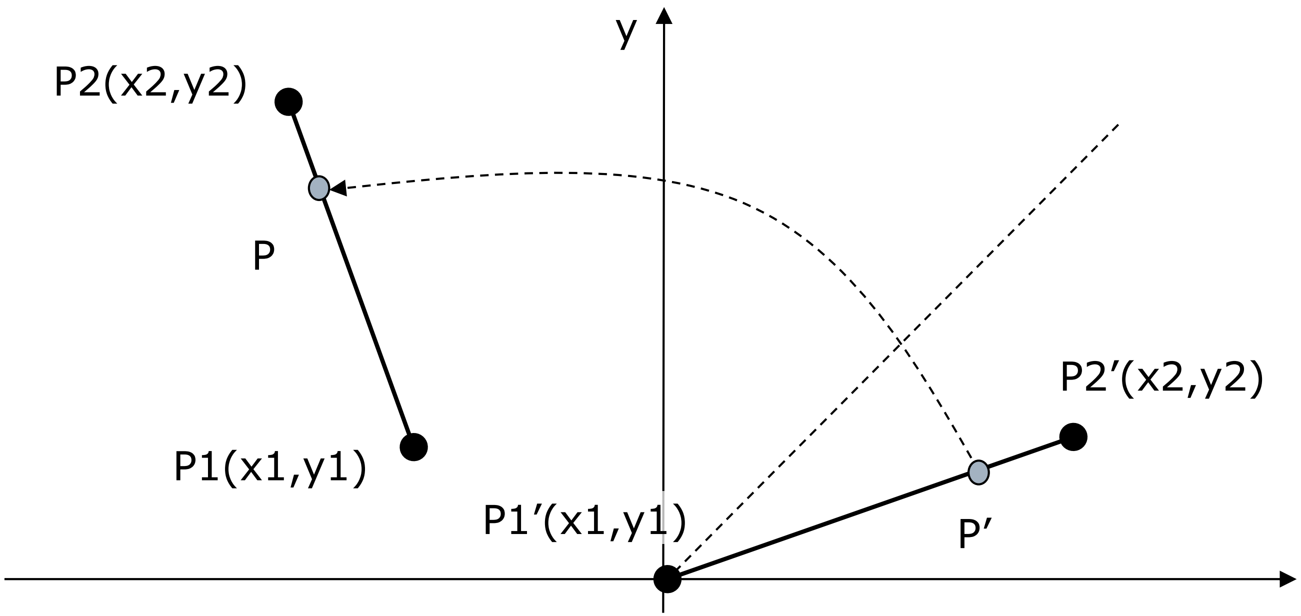
$d_{i+1}= 2\Delta y(x_i+1)-\Delta x(2y_i+1)+2c+2\Delta y-2\Delta x= d_i+2\Delta y-2\Delta x$, if $d_i \geq 0$

$d_{i+1}= d_i+2\Delta y$, otherwise

$d_0=2F(x_0+1, y_0+1/2)= 2(\Delta y x_0-\Delta x y_0+c)+2\Delta y-2\Delta x= 2\Delta y-2\Delta x$

General Bresenham's Line Algorithm

- Basic Bresenham line algorithm is given for line in the first octant.
- To render a general line (A, B):
  1. Transform the line (P1, P2) into the line (P1', P2') in the first octant
  2. Compute the raster line (P1', P2')
     1. For each point P'
        1. Compute point P for the initial line
        2. Render point P



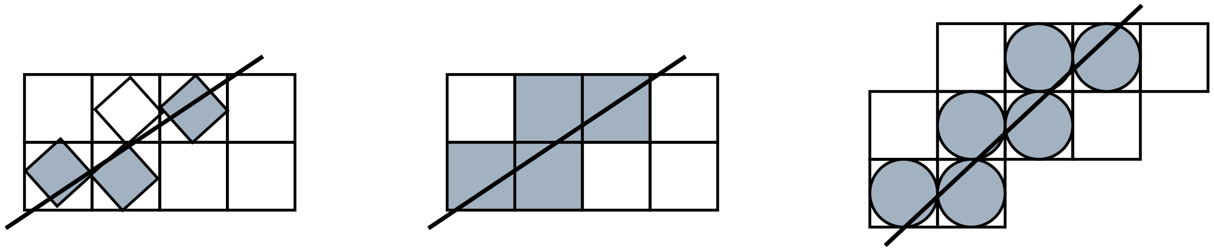
General Bresenham's Line Algorithm

Exercises:

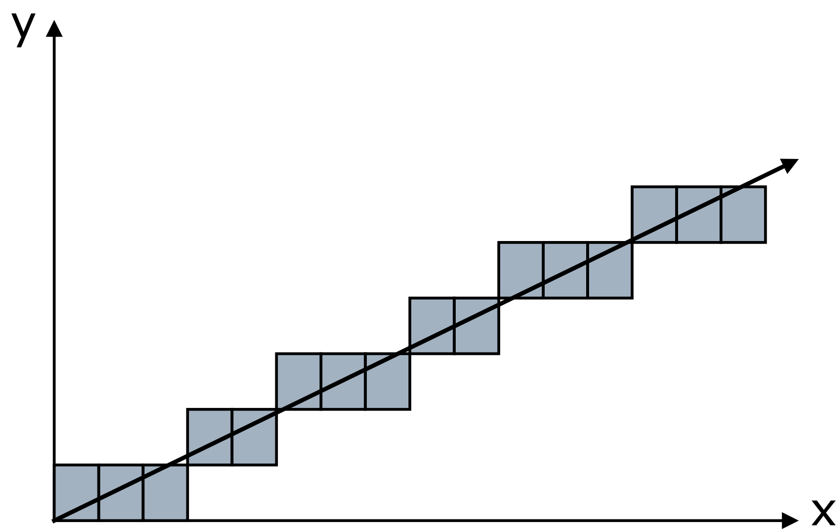
Render and explain the Bresenham algorithm transformations for the following lines:

a. A(7, 5), B(15, 10)
b. A(15, 10), B(7, 5)
c. A(5, 7), B(10, 15)
d. A(10, 15), B(5, 7)
e. A(-5, 7), B(-10, 15)
f. A(-7, 5), B(-15, 10)
g. A(-15, 10), B(-7, 5)
h. A(-15, -10), B(-7, -5)
i. A(-5, -7), B(-10, -15)
j. A(7, -5), B(15, -10)
k. A(15, -10), B(7, -5)

Antialiasing

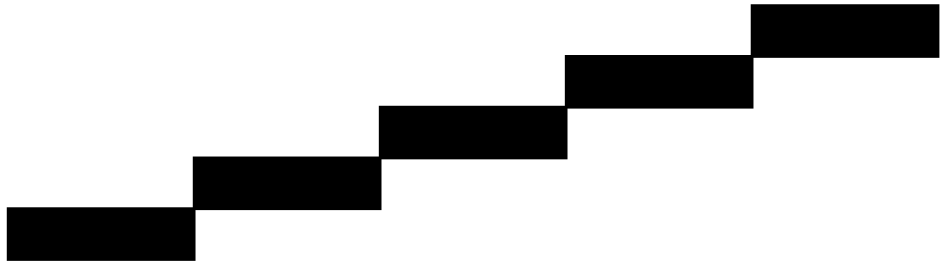


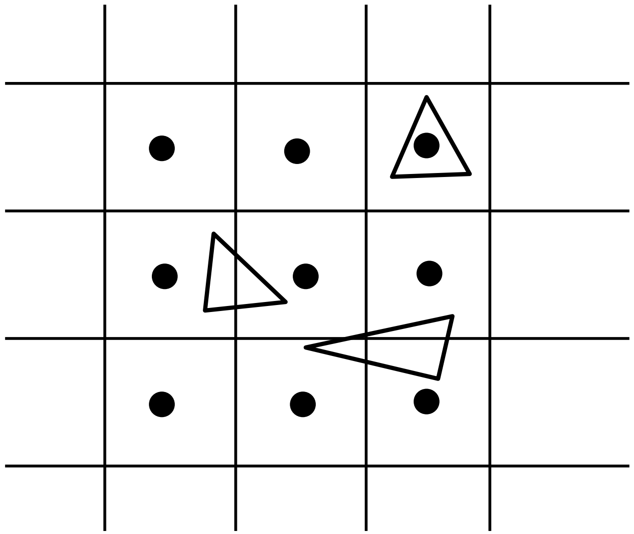
- Removing the stairstep appearance of a line
- Staircase for raster effect
- Need some compensation in line-drawing algorithm for the raster effect

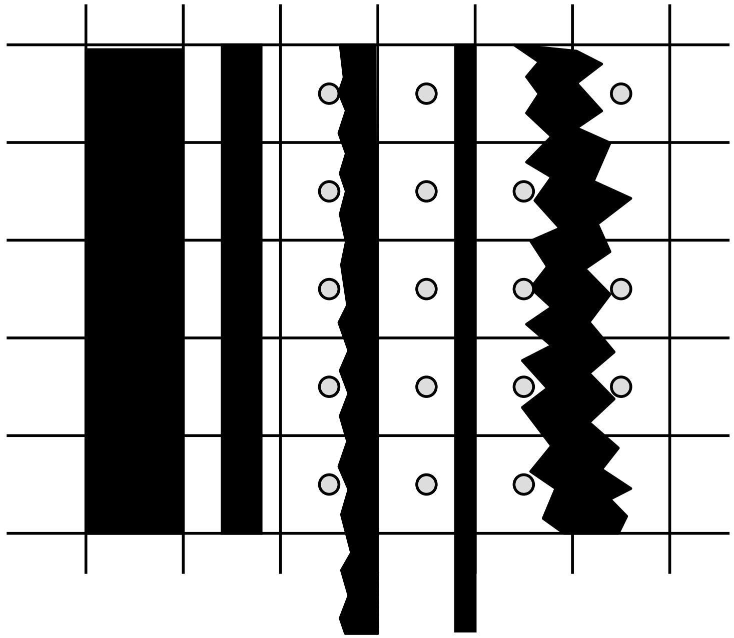


Antialiasing

- Jugged edges
- Small objects
  - unvisible, disturbed
- Textures
  - toothed edges

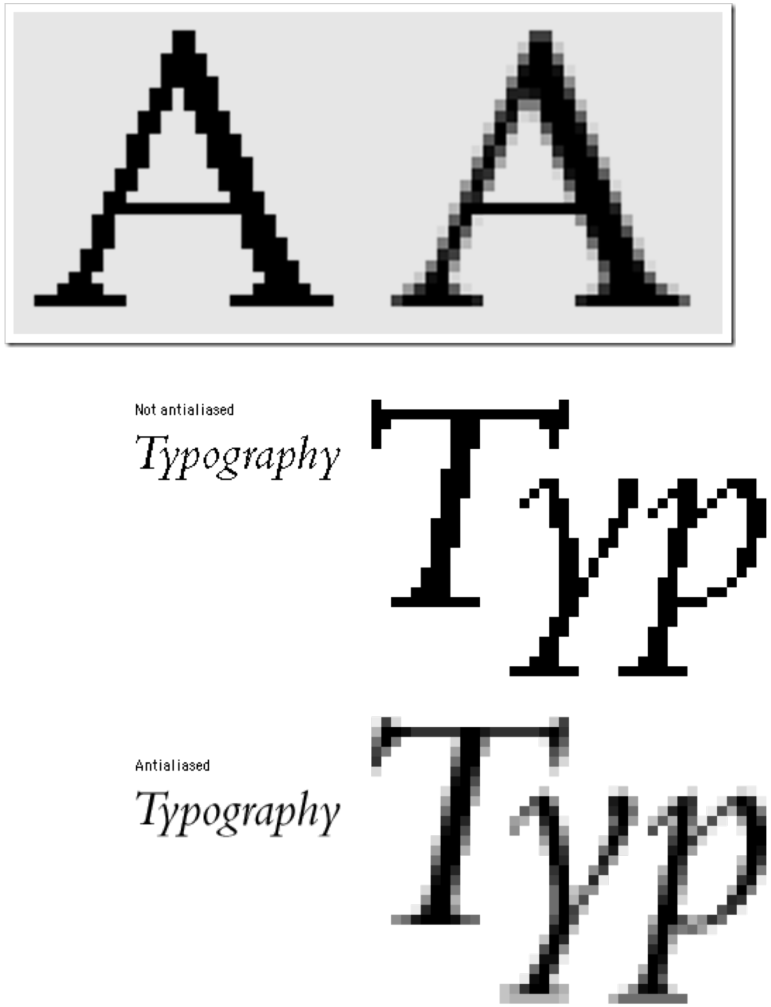






Aliasing effect

Anti-aliasing on the outer edges is lost with 1 bit alpha because pixels cannot be partially transparent.

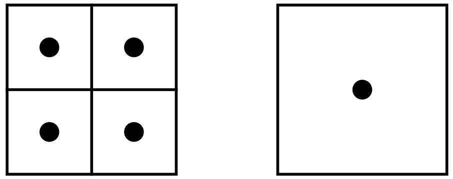


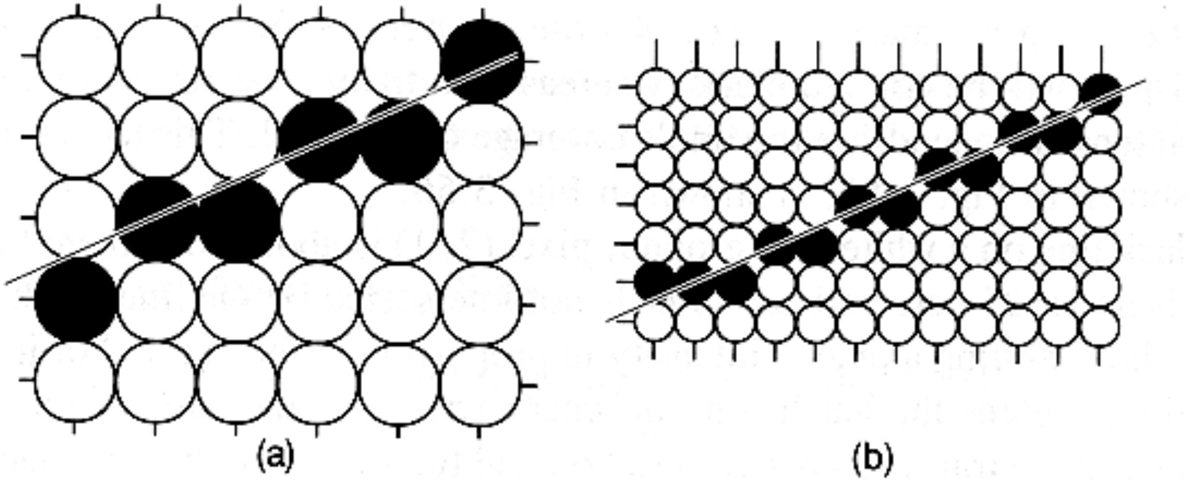
Antialiasing

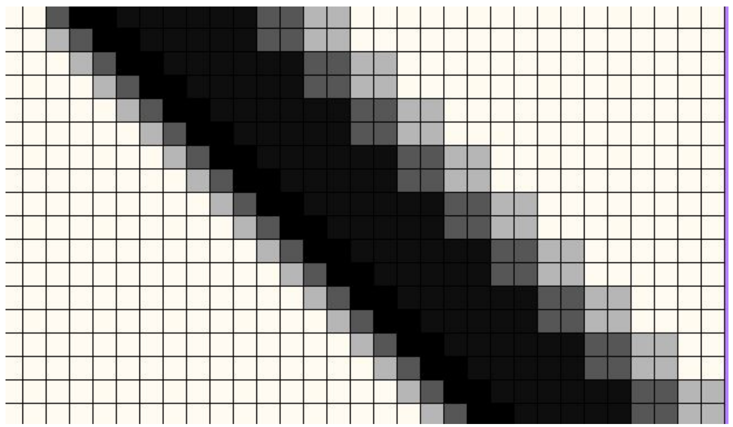
- Supersampling
  - Postfiltering
- Area sampling
  - Prefiltering
- Stochastic sampling

Supersampling (Postfiltering)

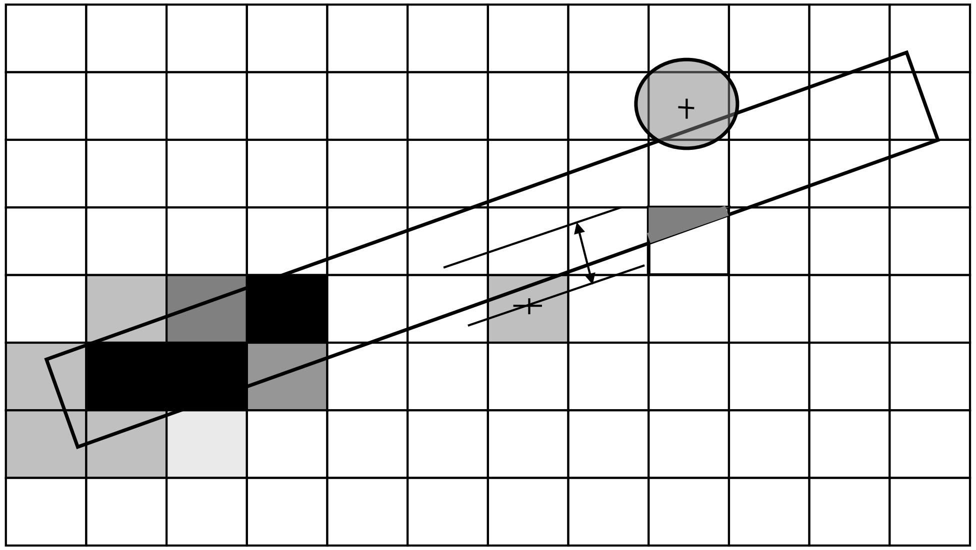
- Increasing resolution

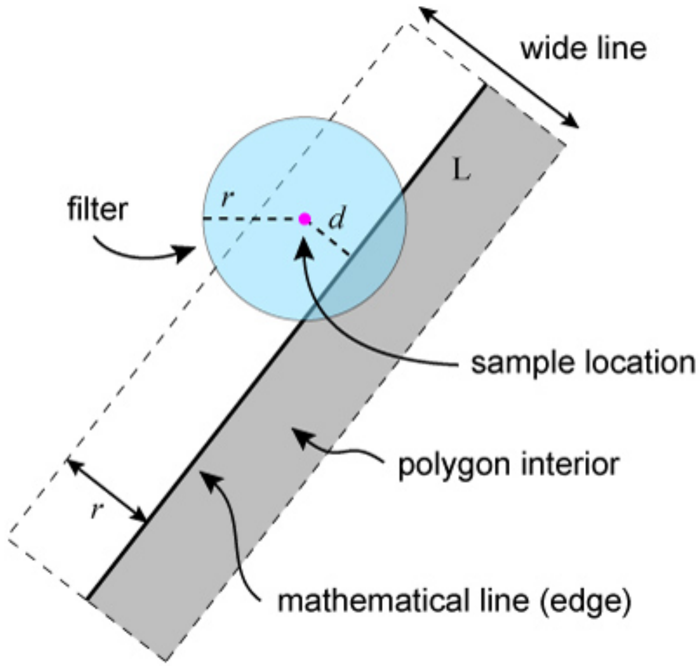


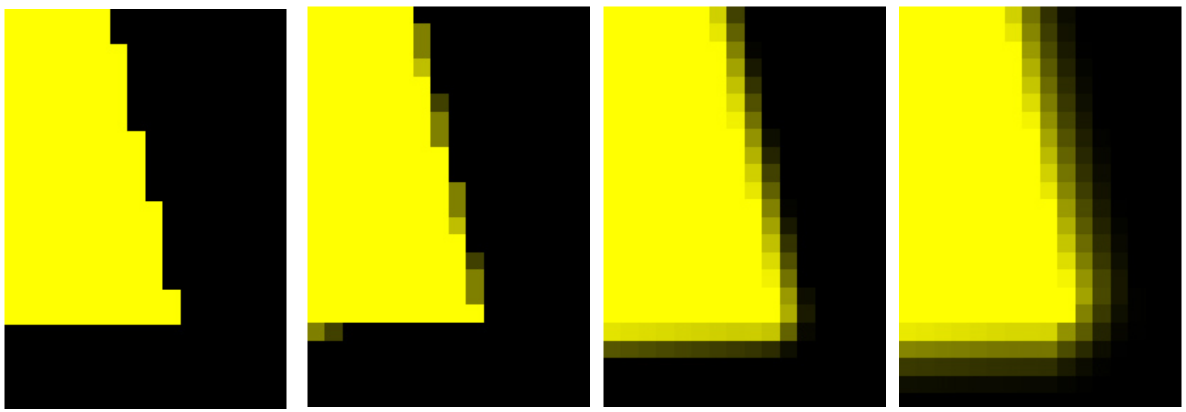




Area Sampling (Prefiltering)







Area Sampling - solutions

1. Unweighted area sampling

   I = f(d)

   - Intensity of a pixel decreases as the distance between the pixel center and the edge increases
   - A primitive does not influence the intensity of a pixel at all if there is no intersection
   - Equal areas contribute equal intensity

2. Weighted area sampling

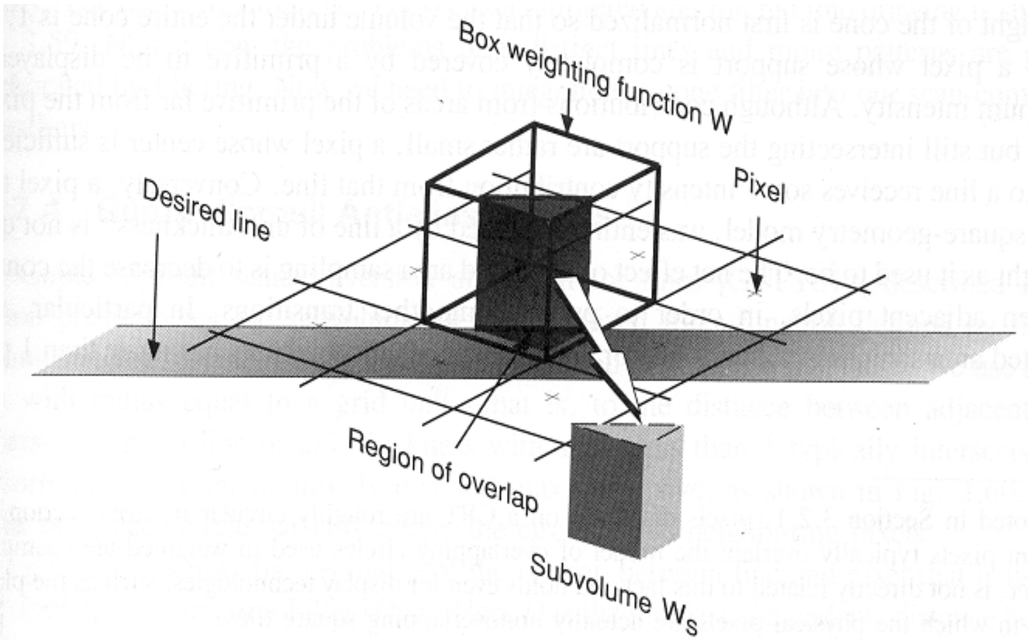
   I = f (ΔA, d), *weighting function*, *filter function*

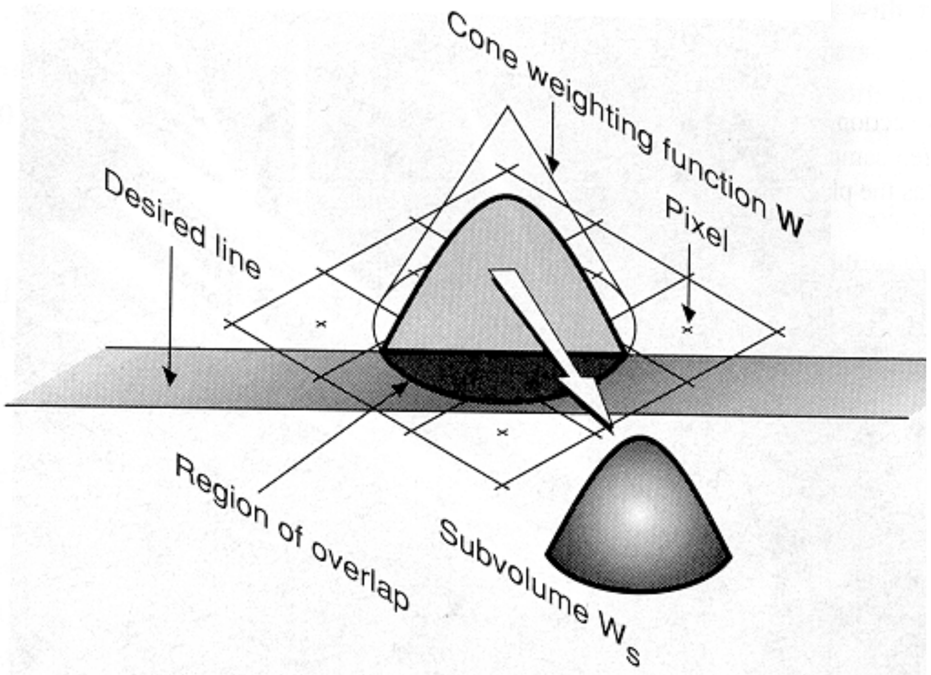
   - The pixel represent a circular area larger than the square tile
   - The primitive intersects the circular area
   - The intersection area contributes to the intensity

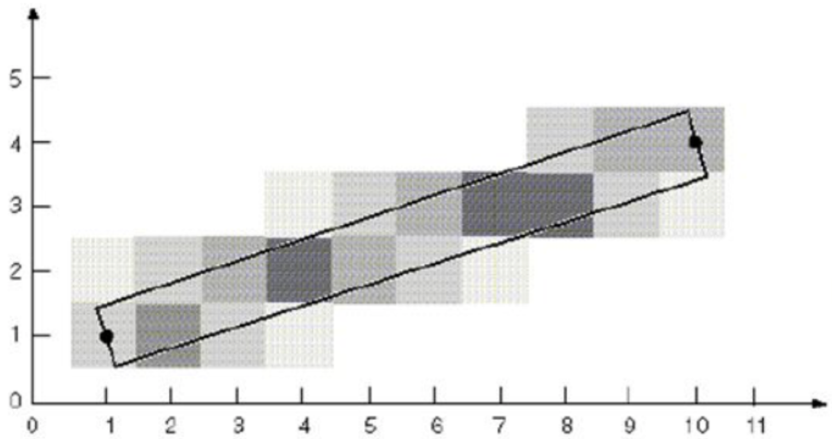
   e.g. Goupta-Sproull incremental method for antialiasing lines

Filters

- Box filter
- Cone filter







Source: Foley, VanDam, Feiner, Hughes - Computer Graphics, Second Edition, Addison Wesley

Questions and proposed problems

1. Why the integer arithmetic is used in graphics algorithms? Why it is imposed in the scan conversion algorithms?
2. How the DDA algorithm works above the first bisection? Is it efficient?
3. Why in the Bresenham algorithm for a line the next pixel could be just E and NE, rather than SE, S, SV, and N?
4. How does the Bresenham algorithm for a line work if the starting point is another than the origin of the coordinate axes?
5. What are the advantages of the Midpoint line algorithm against the Bresenham line algorithm?
6. Render and explain the Bresenham algorithm transformations for the following lines AB: A(10, 15), B(5, 7); A(-5, 7), B(-10, 15); A(-7, 5), B(-15, 10); A(- 15, 10), B(-7, 5). For each case compute the global transformation matrix.
7. Why the computation of the decision variable is faster than analytic computation?

Questions and proposed problems

8. Explain why the Bresenham algorithm is faster or slower than the Midpoint algorithm.
9. Explain how the supersampling approach may improve the quality of graphics rendering? What is the relationship between object point and pixel?
10. Explain why rasterization loses information about the real object?