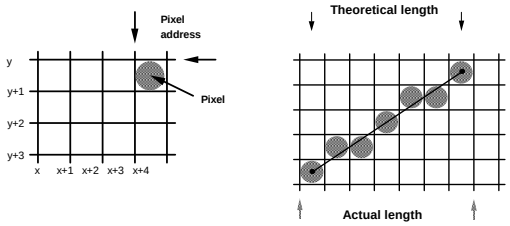
Raster conversion algorithms for line and circle

Introduction
- Pixel addressing
- Primitives and attributes

Line drawing algorithms
- DDA
- Bresehnam

Circle generating algorithms
- Direct method
- Bresenham algorithm

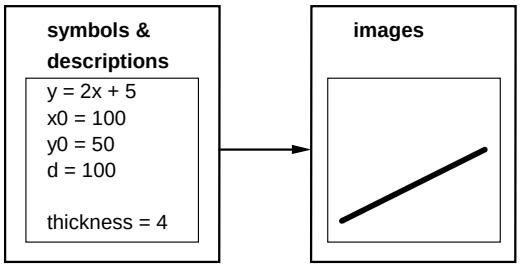
Pixel addressing in raster graphics



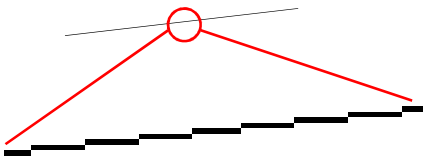
Raster conversion algorithms: requirements

- visual accuracy
- spatial accuracy
- speed

Line (segment) drawing algorithms



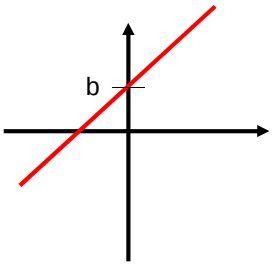
Line – raster representation



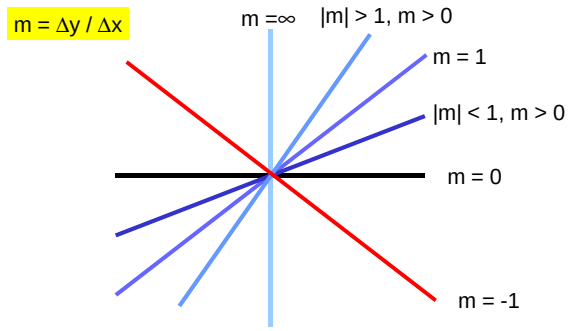
Simple line equation

y = mx + b
m = slope (gradient)
b= intercept with y axis

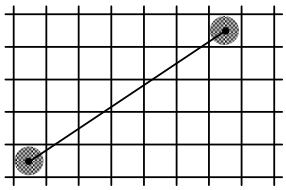
m = Δy / Δx



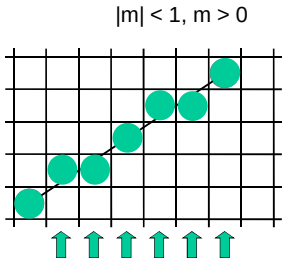
Line gradient (slope)



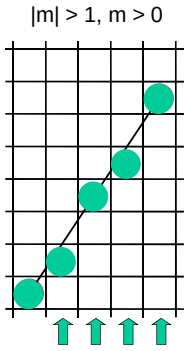
DDA (digital differential algorithm)



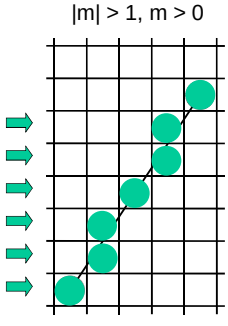
DDA (Digital Differential Algorithm)



DDA (Digital Differential Algorithm)



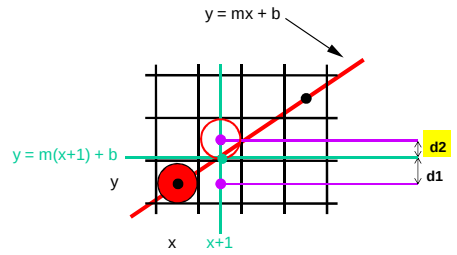
DDA (Digital Differential Algorithm)



DDA (Digital Differential Algorithm)

Derivation

Bresenham's line algorithm



Bresenham's line algorithm (slope ≤ 1)

- input line endpoints, (x_0,y_0) and (x_n, y_n)
- calculate $\Delta x = x_n - x_0$ and $\Delta y = y_n - y_0$
- calculate parameter $p_1 = 2\,\Delta y - \Delta x$
- set pixel at position (x_0,y_0)
- repeat the following steps until (x_n, y_n) is reached:
- if $p_i < 0$
- set the next pixel at position $(x_i + 1, y_i)$
- calculate new $p_{i+1} = p_i + 2\,\Delta y$
- if $p_i \geq 0$
- set the next pixel at position $(x_i + 1, y_i + 1)$
- calculate new $p_{i+1} = p_i + 2(\Delta y - \Delta x)$

For derivation see http://www.cs.bham.ac.uk/~exc/Teaching/Graphics/Bresenham_derivation.pdf

Circle generating algorithms

- Direct
- Polar coordinate based
- Bresenham's

Direct circle algorithm

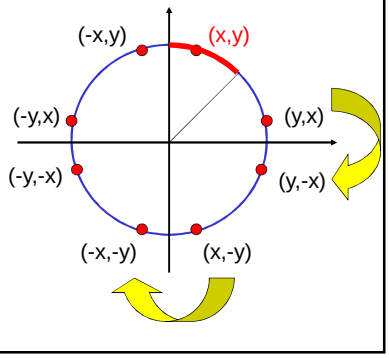
- Cartesian coordinates
- Circle equation:
  $(x - x_c)^2 + (y - y_c)^2 = r^2$
- Step along x axis from $x_c - r$ to $x_c + r$ and calculate
  $y = y_c \pm \sqrt{r^2 - (x - x_c)^2}$

Polar coordinates

- Polar coordinate equation (parametric, w.r.t. φ)
  $x = x_c + r\cos\varphi$
  $y = y_c + r\sin\varphi$
- step through values of φ from 0 to 2π

Optimisation and speed-up

- Symmetry of a circle can be used
- Calculations of point coordinates only for a first one-eighth of a circle
- Assumes $(x_c,y_c)=(0,0)$



Bresenham's circle algorithm

- Assumes centre at (0,0)

Initialisation

- Input radius r
- Plot a point at $x_0=0$, $y_0=r$
- Set k = 0
- Calculate the initial value of the decision parameter $p_0 = 3 - 2r$

While x < y

if $p_k < 0$

$p_{k+1} = p_k + 4x_k + 6$
$x_{k+1} = x + 1$
$y_{k+1} = y_k$

else

$p_{k+1} = p_k + 4(x_k - y_k) + 10$
$x_{k+1} = x_k + 1$
$y_{k+1} = y_k - 1$

plot point at (x_{k+1}, y_{k+1})
increment k

5. Determine symmetry points in the other seven octants and plot points

6. Repeat steps 4 and 5 until $x \geq y$

Homework

- Implement the Bresenham's circle algorithm for a circle with a centre at an arbitrary position (x_c, y_c).
- Using Bresenham's circle algorithm generate circle with centre at (-10,10) and radius r=40
- Extend the Bresenham's line algorithm for lines at an arbitrary slope
- Extend the Bresenham's line algorithm to draw lines of arbitrary thickness (the original algorithm generates lines of nominal thickness of 1)
- Implement an algorithm for drawing anti-aliased lines