Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Computer Graphics

Introduction to Computer Graphics

Edited by
Dr. Md. Manjur Ahmed
Dr. Suryanti Awang
Faculty of Computer Systems and Software Engineering
manjur@ump.edu.my

Chapter Description

- Aims
  - Basic of Computer Graphics.

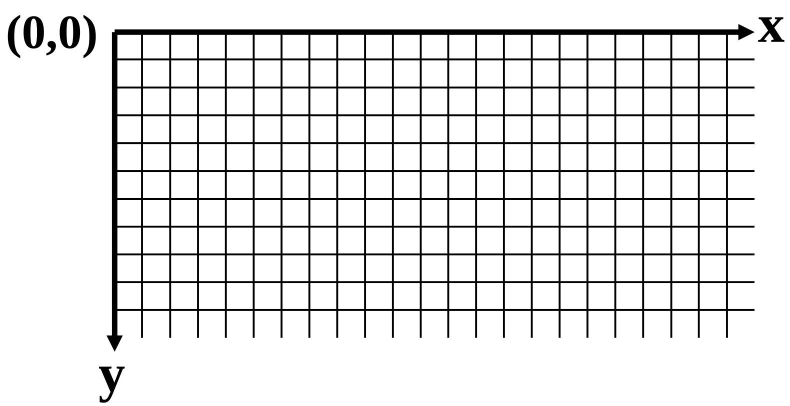
- Expected Outcomes
  - Understand the basic concept of computer graphics. (CO1: Knowledge)
  - Ability to use the computer graphics technology. (CO1: Knowledge)

- References

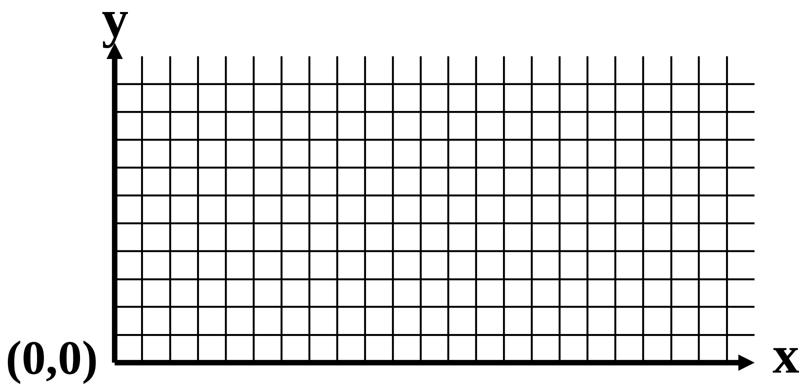
  - Computer Graphics by Zhigang Xiang, Schaum's Outlines.
  - Donald Hearn & M. Pauline Baker, Computer Graphics with OpenGL, 4th Edition, Boston : Addison Wesley, 2011.

COORDINATES SYSTEM

Most windowing systems:



OpenGL framebuffer:



COORDINATES SYSTEM

Does it matter? No, we just need to be aware of the difference:

- Where a pixel in the framebuffer will show up on screen?
- How do we get the pixel address under the mouse pointer?

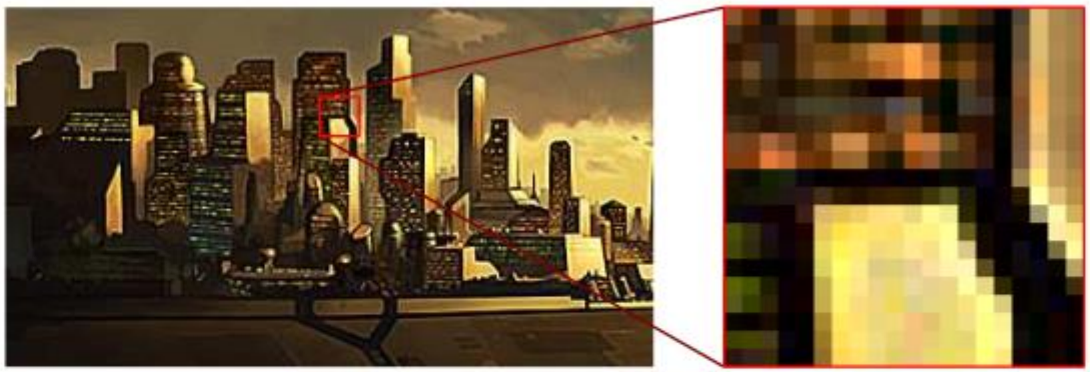
Could some other display library have its framebuffer lay-out match your windowing system? Absolutely. Many do.

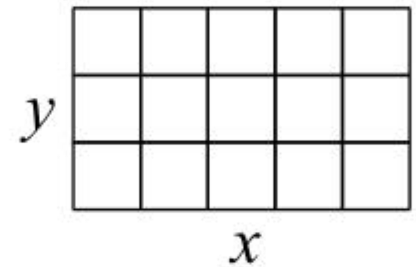
What if all we never directly displayed our framebuffer, but wrote it out as an image for later display?

- Virtually all image formats use screen-space coordinates.

RASTER DISPLAY

- Represented by a 2D array of positions called pixels





Ref

Zooming in on an image made up of pixels

- pixel at location (0,0), lower left corner
- Color frequently requires 1 byte per channel (three color channels per pixel namely R=red, G=green, B=blue).

RASTER DISPLAY

- **Frame Buffer**: ~ is stored the color data, often called an image map or bitmap.

```
setpixel(x, y, color)
```

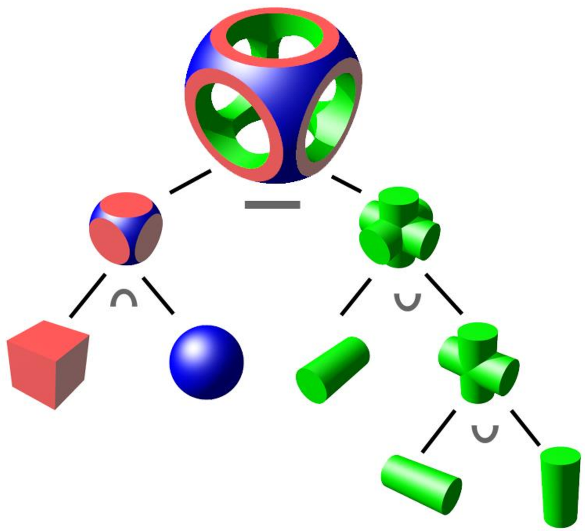
Sets the pixel at position (x, y) to the given color.

```
getpixel(x, y)
```

Gets the color at the pixel at position (x, y).

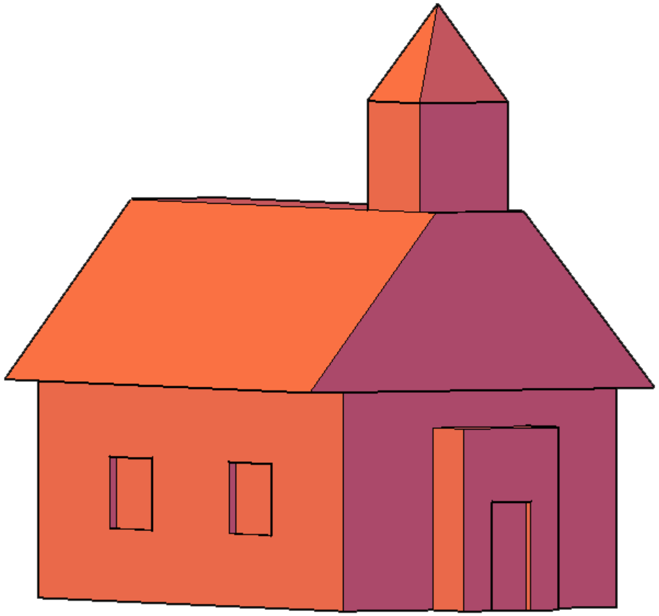
- **Scan conversion:** convert to basic and low level objects into corresponding pixel map depictions.

OUTPUT PRIMITIVES



- A picture consists of a complex objects
- Come from basic geometric structures called Object Primitives

Points and Lines



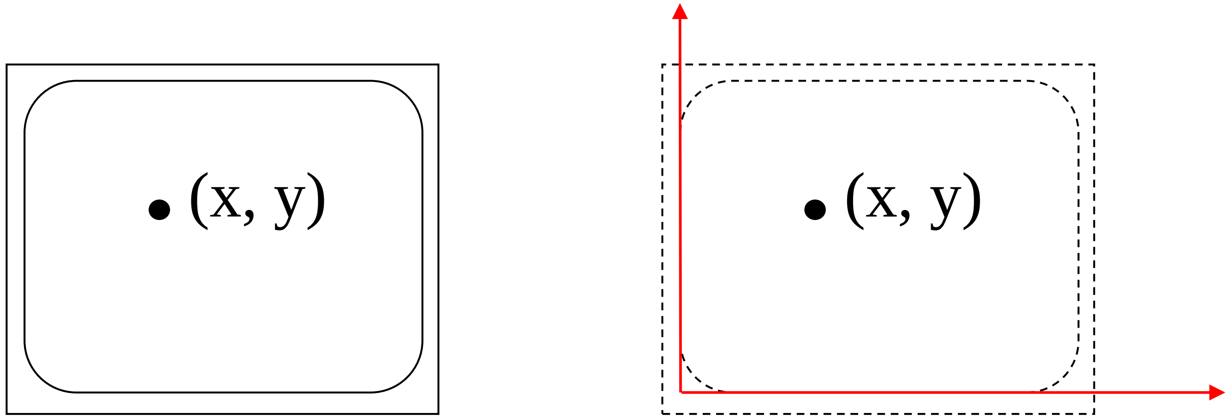
- A line can be completed by calculating the line path between 2 endpoints. Points and Lines

Simple Line

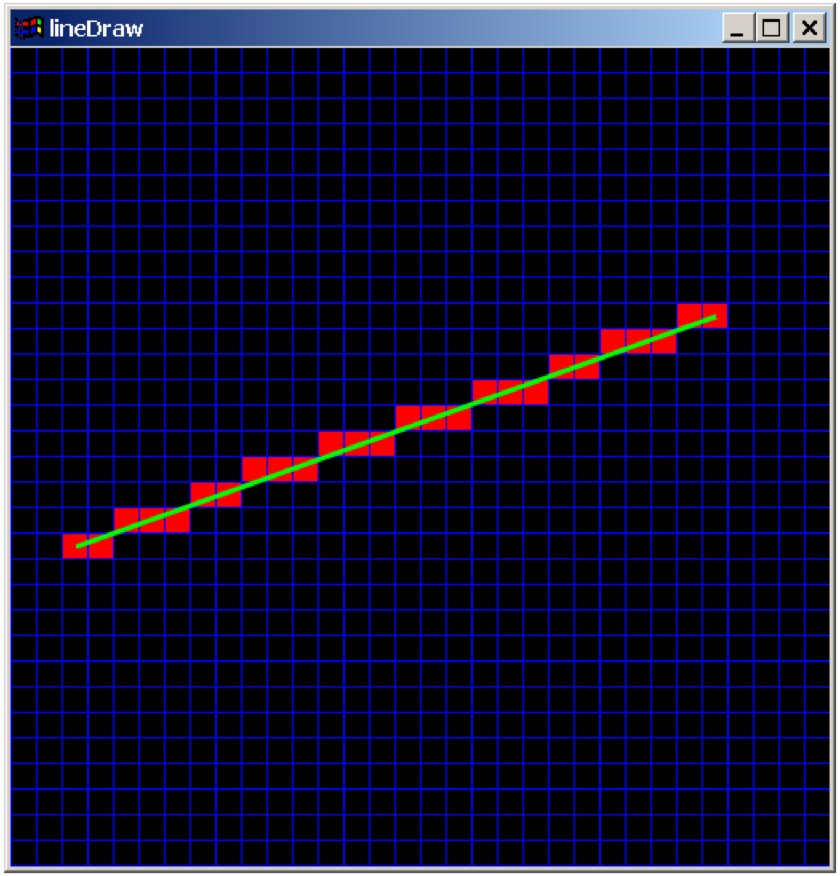
- A point on the screen with position (x, y): plot a pixel with corresponding position

- Sample code:

`SetPixel ( x, y )` → **a function in windows.h**



Simple Line



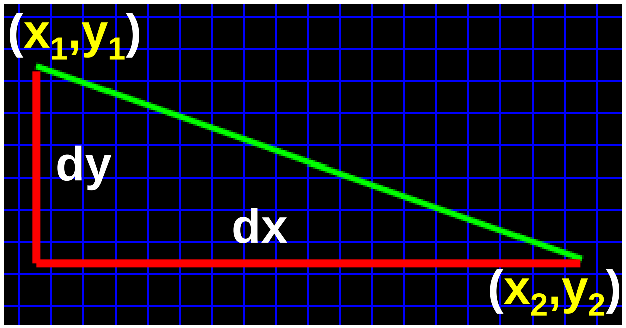
Raster-scan devices address discrete pixels

The endpoints and intermediate points must be set individually

The points are calculated from the slope-intercept equation

Line drawing is a fundamental operation on which many other routines are built

Equation of a line



$$y = m.x + c$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$c = y_1 - m.x_1$$

- Based on eq. and positions of 2 endpoints (x1, y1), (x2, y2):

$$m = \frac{y_2 -}{x_2 -}$$

- Therefore,

$$\Delta y = m$$

- and

$$\Delta x = \frac{\Delta}{m}$$

```
int x
float m, y
m = (y1 - y0) / (x1 - x0)
for (x = x0; x <= x1; ++x) {
  y = m * (x - x0) + y0
  setpixel(x, round(y), linecolor)
}
```

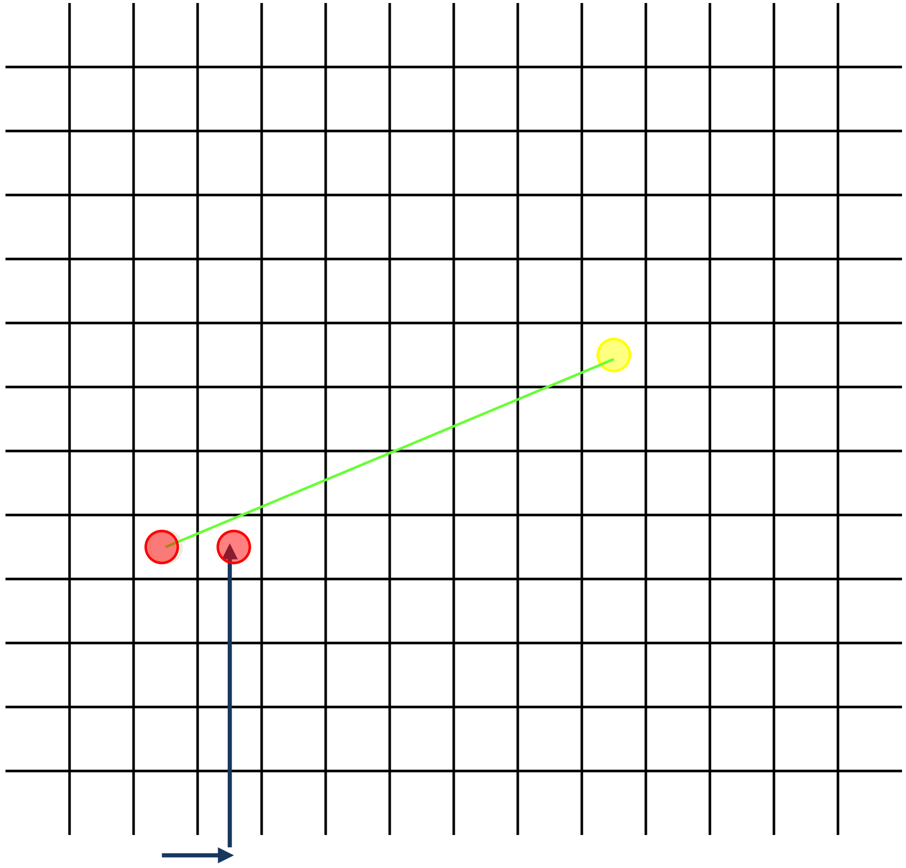
Example
Digital Differential Analyzer (DDA)

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



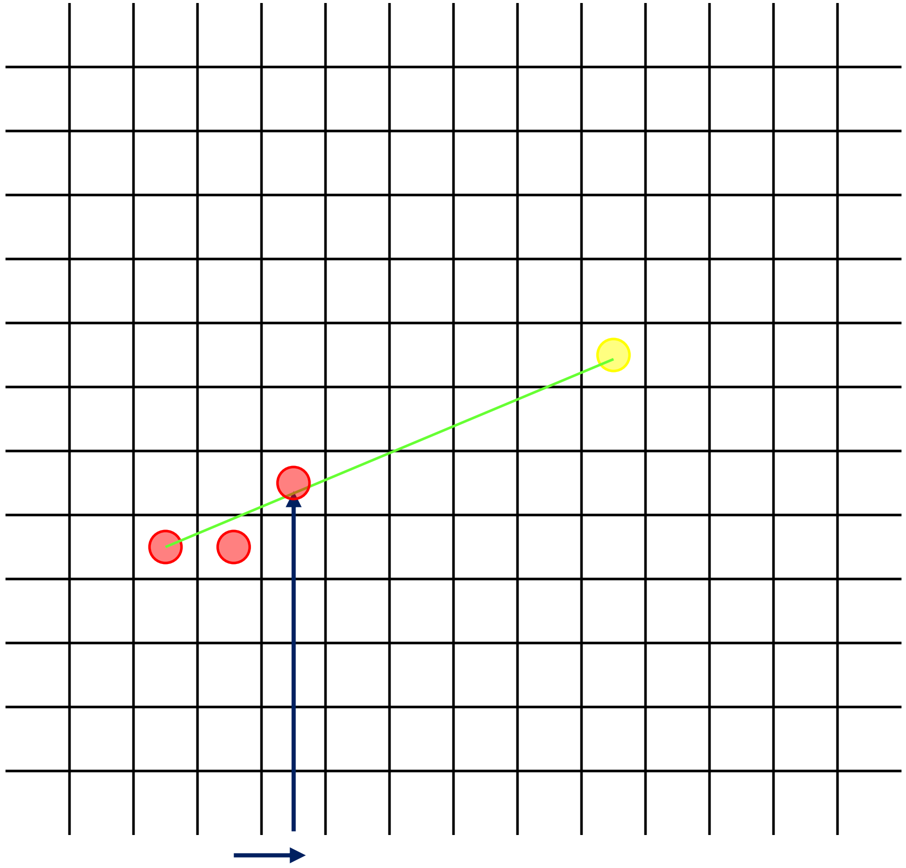
DDA Example

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



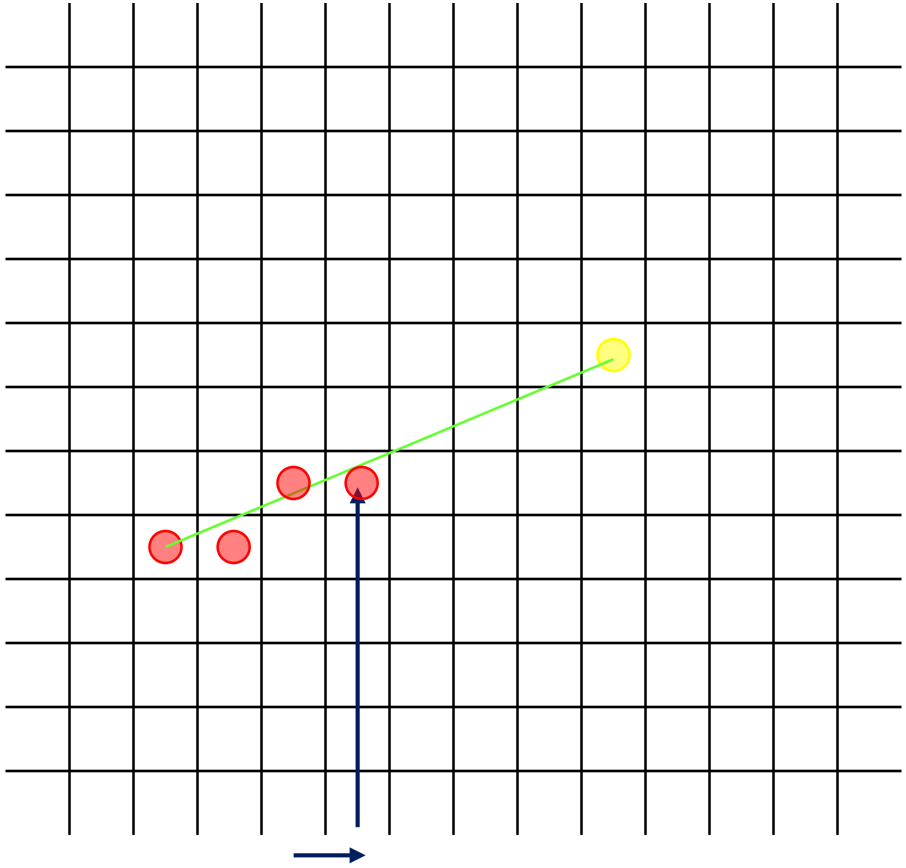
DDA Example

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



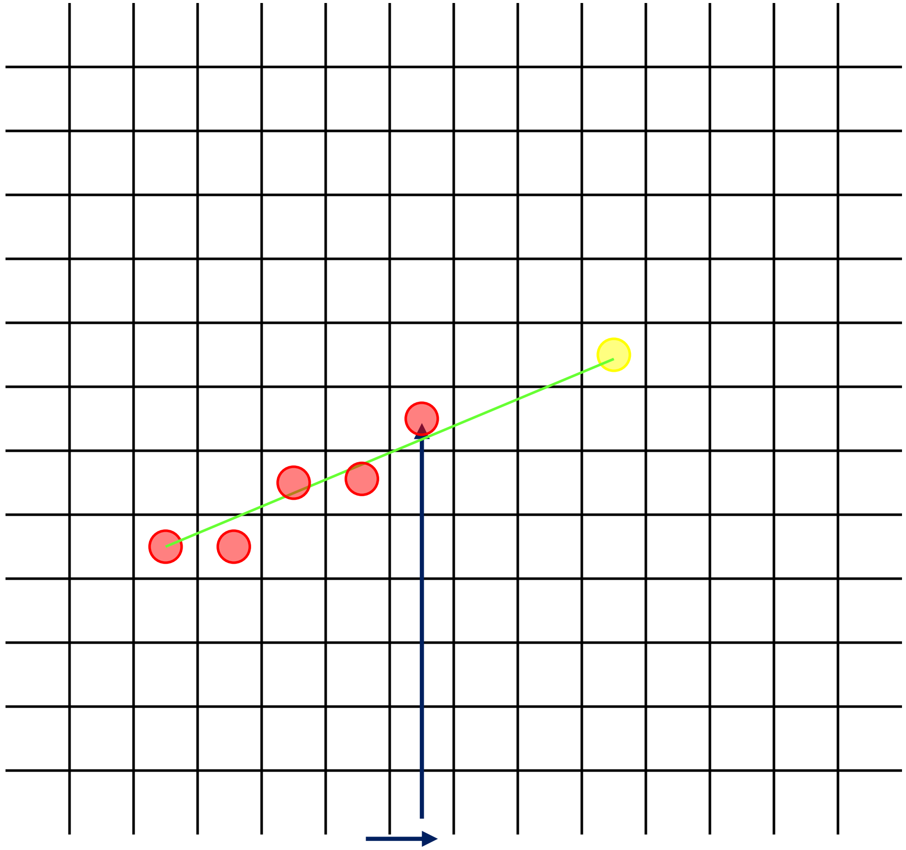
DDA Example

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



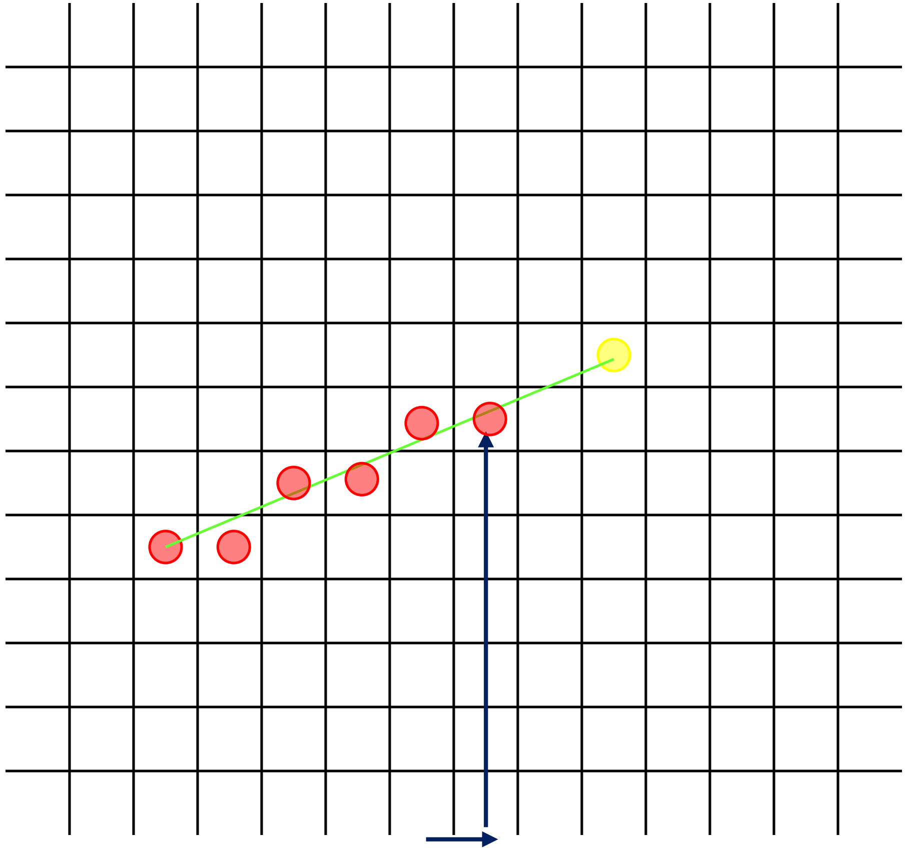
DDA Example

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



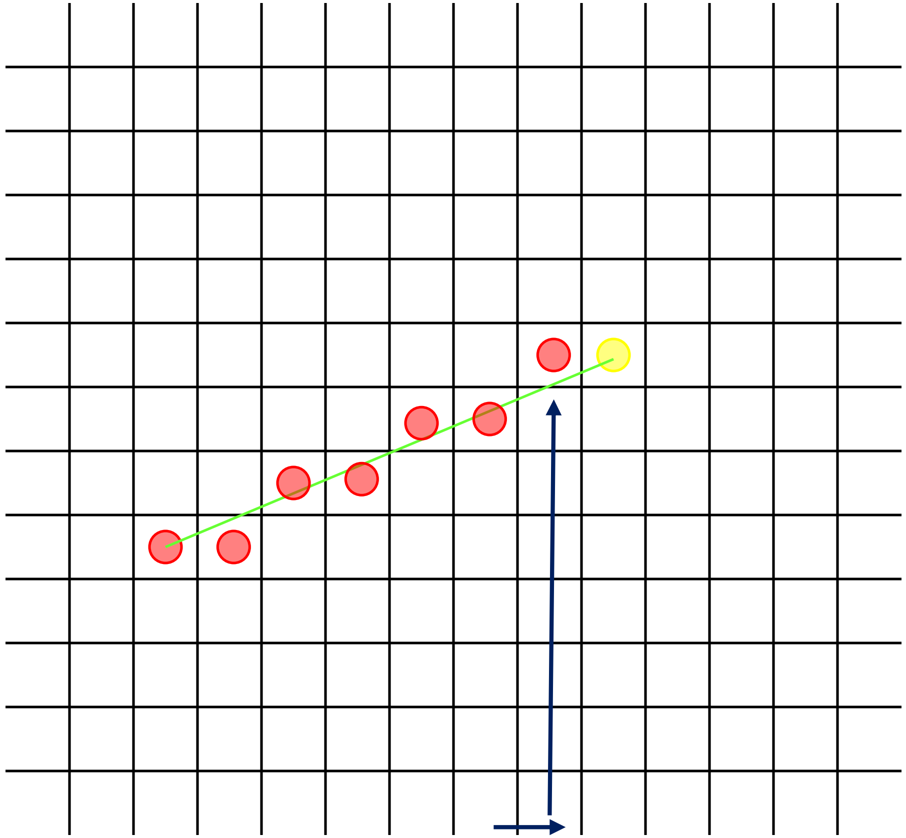
DDA Example

Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$



DDA Example

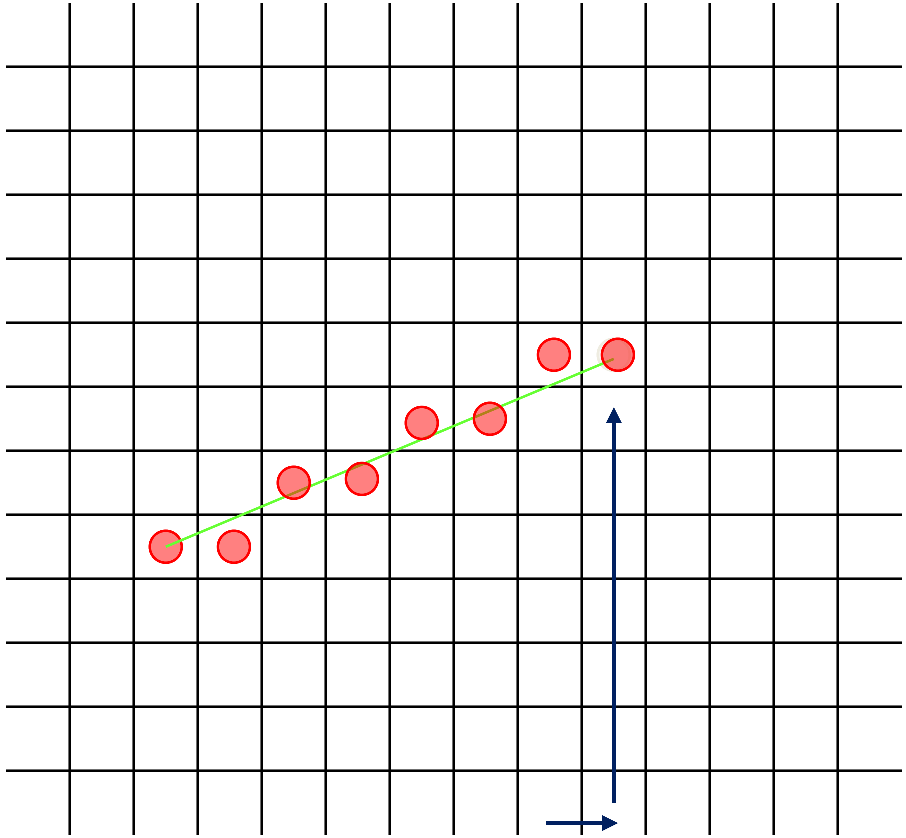
Sample at unit x:

$$x_{k+1} = x_k + \Delta x$$
$$= x_k + 1$$

Corresponding y pos.:

$$y_{k+1} = y_k + \Delta y$$
$$= y_k + m \cdot \Delta x$$
$$= y_k + m \cdot (1)$$

Consider endpoints:
P1(0,0), P2(7,4)



DDA Example

Let $P_1(2,2)$, $P_2(7, 5)$

Calculate the points that made up the line P_1P_2

First work out *m* and *c*:

$$m = \frac{5-2}{7-2} = \frac{3}{5} \qquad c = 2 - \frac{3}{5} * 2 = \frac{4}{5}$$

Now work for each *x* value work out the *y* value:

$$y(3) = \frac{3}{5} * 3 + \frac{4}{5} = 2\frac{3}{5} \approx 3$$

$$y(4) = \frac{3}{5} * 4 + \frac{4}{5} = 3\frac{1}{5} \approx 3$$

$$y(5) = \frac{3}{5} * 5 + \frac{4}{5} = 3\frac{4}{5} \approx 4$$

$$y(6) = \frac{3}{5} * 6 + \frac{4}{5} = 4\frac{2}{5} \approx 4$$

DDA in C

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs  dy);
  xIncrement = dx / (float) steps;
  yIncrement = dy / (float) steps;

  setPixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (ROUND(x), ROUND(y));
  }
}
```

DDA Exercise

1. Consider endpoints:
   $P_1(0,0)$, $P_2(6, 4)$
   Calculate the points that made up the line P_1P_2

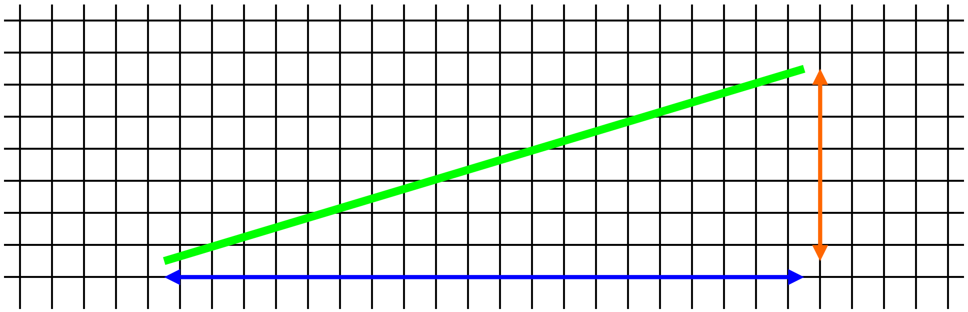
2. Now, consider endpoints:
   $P_1(0,0)$, $P_2(4, 6)$
   Calculate the points that made up the line P_1P_2

Limitation of DDA

Not identify the positions if x1 < x0.

Answer: Shift the order of the points if x1 < x0.

Bresenham's line drawing algorithm



Consider the first condition:

m < 1,   m has a positive value

Bresenham's increments x by 1 and y by 0 or 1

This makes sense for our lines, we want them to be continuous

If the magnitude of the slope were more than 1, we'd swap x & y

Bresenham's line drawing algorithm

Algorithm for |m| < 1:

1. **Take 2 endpoints as input. Assign (x_1, y_1) = first end point.**
2. **Load to frame buffer and plot the first point in display.**
3. **Compute constant values of Δx, Δy, $2\Delta y$, $2\Delta y - 2\Delta x$. Use initial value of decision parameter:**

$$p_1 = 2\Delta y - \Delta x$$

4. Start form $t=1$, for each x_t along the line, test:

if $p_t < 0$, plot (x_{t+1}, y_t) and $p_{t+1} = p_t + 2\Delta y$

else plot (x_{t+1}, y_{t+1}) and $p_{t+1} = p_t + 2\Delta y - 2\Delta x$

5. Continue step 4 Δx times.

Example

Digitize the line with endpoints (20,10) and (25,13).

Plot the line by determining the pixel positions.

$$y = mx + c$$

t	p_t	(x_{t+1}, y_{t+1})
1		
2		
3		
5		

Exercise

Calculate pixel positions that made up the line connecting endpoints: (12, 10) and (17, 14).

1. $(x_1, y_1) = ?$
2. $\Delta x = ?$, $\Delta y = ?$, $2\Delta y = ?$, $2\Delta y - 2\Delta x = ?$
3. $p_1 = 2\Delta y - \Delta x = ?$

t	p_t	(x_{t+1}, y_{t+1})

Exercise

Calculate pixel positions that made up the line connecting endpoints: (12, 10) and (17, 14).

1. (x_1, y_1) = (12, 10)
2. $\Delta x = 5$, $\Delta y = 4$, $2\Delta y = 8$, $2\Delta y - 2\Delta x = -2$
3. $p_1 = 2\Delta y - \Delta x = 3$

k	p_k	(x_{k+1}, y_{k+1})
1	3	
2		
3		

Circle Drawing

Simple way to start

Equation of a circle:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

If we solve for *y*:

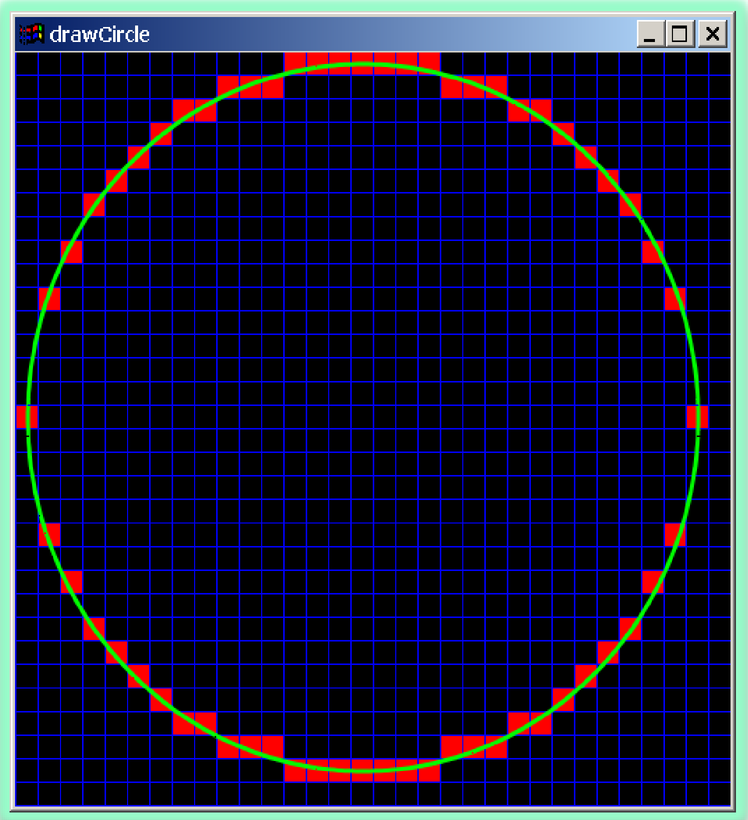
$$y = y_0 \pm \sqrt{r^2 - (x - x_0)^2}$$

Sample C codes:

```
void SIMPLE_CIRCLE(int X_center, int Y_center, int radius_R, Color c) {
  int x, y, r2;
  r2 = radius_R * radius_R;
  for (x = - radius_R; x <= radius_R; x++) {
    y = (int)(sqrt(r2 - x*x) + 0.5);
    setPixel((X_center + x), (Y_center + y, c));
    setPixel((X_center + x), (Y_center - y, c));
  }
}
```

Simple way to start "uncertainty"

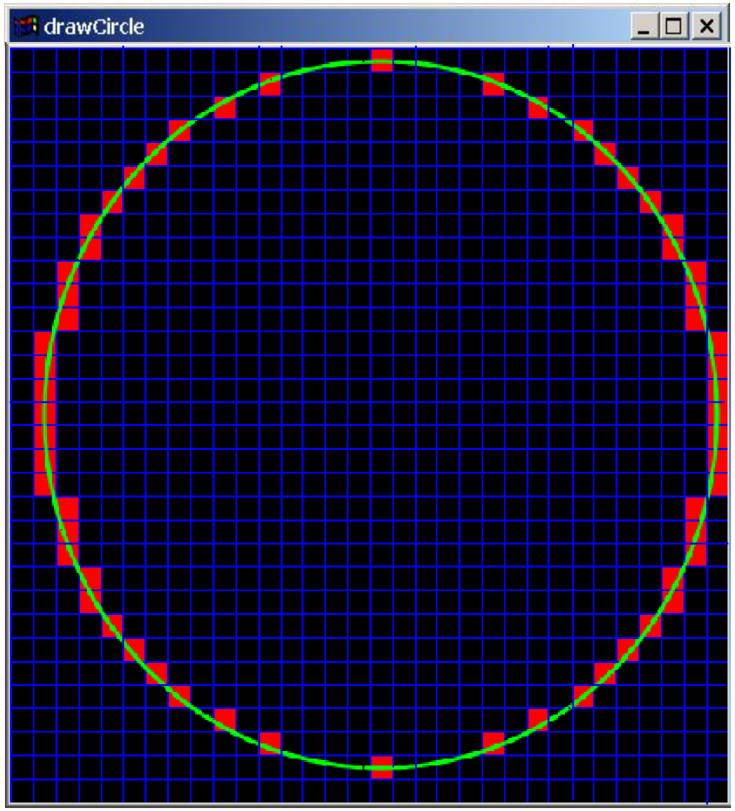
- In certain cases, the slope of the line tangent is greater than 1. Therefore, tangent > 1 shows the uncertainty of this algorithm.
- Looping (or stepping) using *x* won't work here.



Reference: Computer Graphics: Principles and Practice by James D. Foley and et.al.

Circles symmetrical nature

- To solve the issue "slope of the tangent"
- Let's take the advantage of the circle's symmetric nature.
- Consider both positive and negative values for *y* (or x).

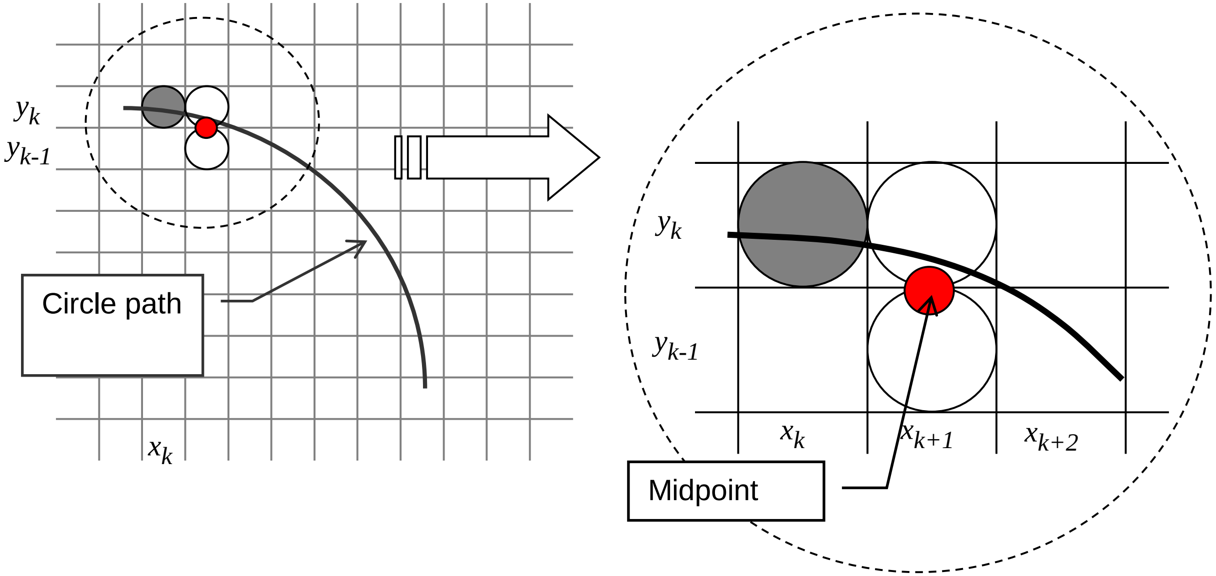


Midpoint Circle Algorithm

- Consider current point is at (x_k, y_k)
- Next point: (x_k+1, y_k), or (x_k+1, y_k-1)?
- Take the midpoint: (x_k+1, y_k-0.5)

•Use the discriminator function to decide:

$$f(x,y) = x^2 + y^2 - r^2$$



Using the circle discriminator

Based on the value return:

$$f(x, y) \begin{cases} < 0 & \text{inside the circle} \\ = 0; & \text{on the circle path} \\ > 0; & \text{outside the circle} \end{cases}$$

By using the midpoint between 2 pixel candidates, we can introduce a decision parameter, p_k , to decide which to plot next:

1)

$$p_k = f\left(x_k + 1, y_k - 0.5\right)$$
$$= \left(x_k + 1\right)^2 + \left(y_k - 0.5\right)^2 - r^2$$

2 and 3)

$$p_k \begin{cases} \text{-ve: midpoint is inside the circle; plot } (x_k+1, y_k) \\ \text{+ve: midpoint is outside the circle; plot } (x_k+1, y_k-1) \end{cases}$$

If the current point is inside the circle ...

If $p_k < 0$

We want to know f(x+1, y) so we can update p:

$$f(x+1, y) = (x + 1)^2 + y^2 - r^2$$

$$f(x+1, y) = (x^2 + 2x + 1) + y^2 - r^2$$

$$\underbrace{f(x+1, y)}_{P_{k+1}} = \underbrace{f(x, y)}_{P_k} + 2x + 1$$

So we increment:

$$p \mathrel{+}= 2x + 1$$

If the current point is outside the circle …

If $p_k > 0$

Let's drop the subscript for a while…

We want to know $f(x+1, y-1)$ so we can update p:

$$f(x+1, y-1) = (x + 1)^2 + (y - 1)^2 - r^2$$

$$f(x+1, y-1) = (x^2 + 2x + 1) + (y^2 - 2y + 2) - r^2$$

$$\underbrace{f(x+1, y-1)}_{P_{k+1}} = \underbrace{f(x, y)}_{P_k} + 2x - 2y + 2$$

And we increment:

$$p \mathrel{+}= 2x - 2y + 2$$

Where to begin?

We can determine where to go next, how do we start?

We have a variety of choices for our first point on the circle, but we've designed the increments to work from $(0, r)$.

Calculate initial value of p_0 by evaluating:

$$p_0 = f(1, r–0.5) = 1^2 + (r – 0.5)^2 – r^2$$

$$p_0 = f(1, r–0.5) = 1 + (r^2 – r + 0.25) – r^2$$

$$p_0 = 1.25 – r$$

****We want to use integer calculation; you can round p_0**

Midpoint circle algorithm

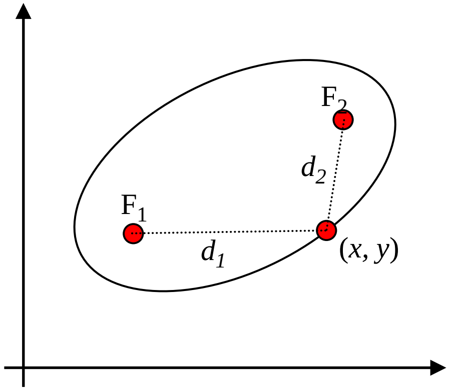
Homework

Ellipse Drawing

Equation

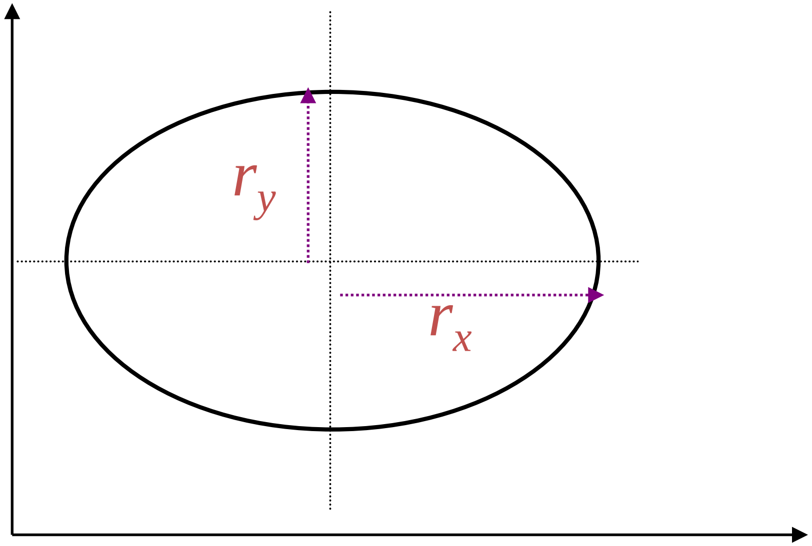
General equation of an ellipse:

$$\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}=\text{constant}$$



$$d_1 + d_2 = constant$$

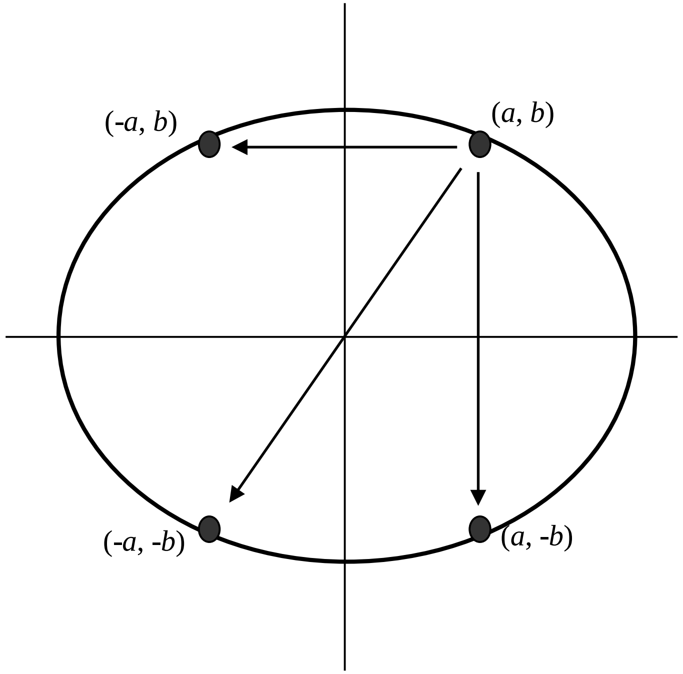
Or,



$$\left(\frac{x-x_c}{r_x}\right)^2+\left(\frac{y-y_c}{r_y}\right)^2=1$$

Symmetry

An ellipse only has a 2-way symmetry.



Equation of an ellipse revisited

Consider an ellipse centered at the origin:

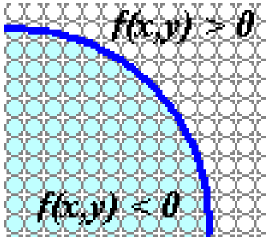
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 = 1$$

What is the discriminator function?

$$f_e(x, y) = r_y{}^2 x^2 + r_x{}^2 y^2 - r_x{}^2 r_y{}^2$$

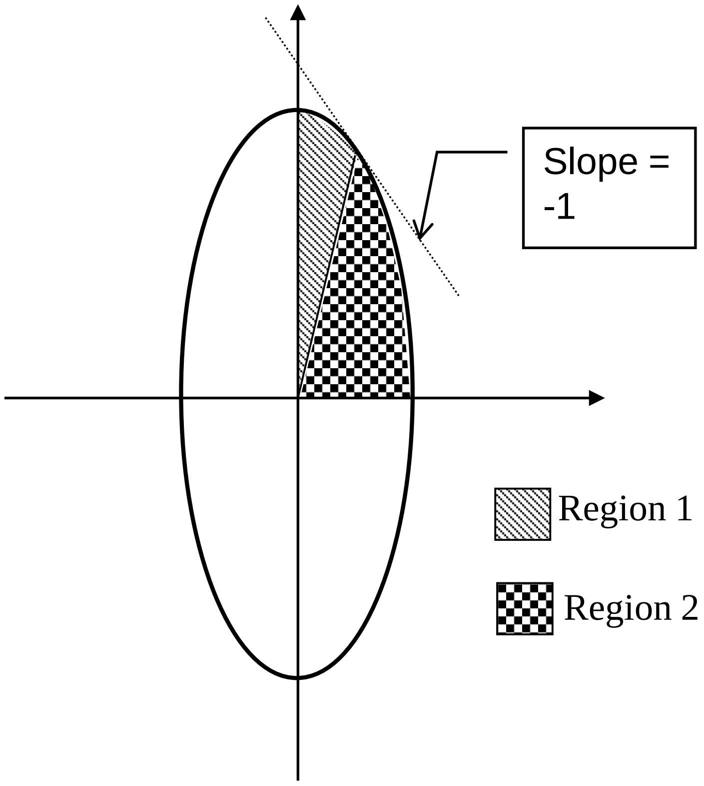
...and its properties:

$f_e(x,y) < 0$ for a point inside the ellipse
$f_e(x,y) > 0$ for a point outside the ellipse
$f_e(x,y) = 0$ for a point on the ellipse



Midpoint Ellipse Algorithm

- Ellipse is different from circle.
- Similar approach with circle, different is sampling direction.
- Region 1:
  - Sampling is at *x* direction
  - Choose between (x_k+1, y_k), or (x_k+1, y_k-1)
  - Midpoint: $(x_k+1, y_k-0.5)$
- Region 2:
  - Sampling is at *y* direction
  - Choose between (x_k, y_k-1), or (x_k+1, y_k-1)
  - Midpoint: $(x_k+0.5, y_k-1)$



Conclusion of The Chapter

- Next Part more in these….