

CMP205: Computer Graphics



CAIRO UNIVERSITY

Lecture 1: Line Drawing

Mohamed Alaa El-Dien Aly
Computer Engineering Department
Cairo University
Fall 2013

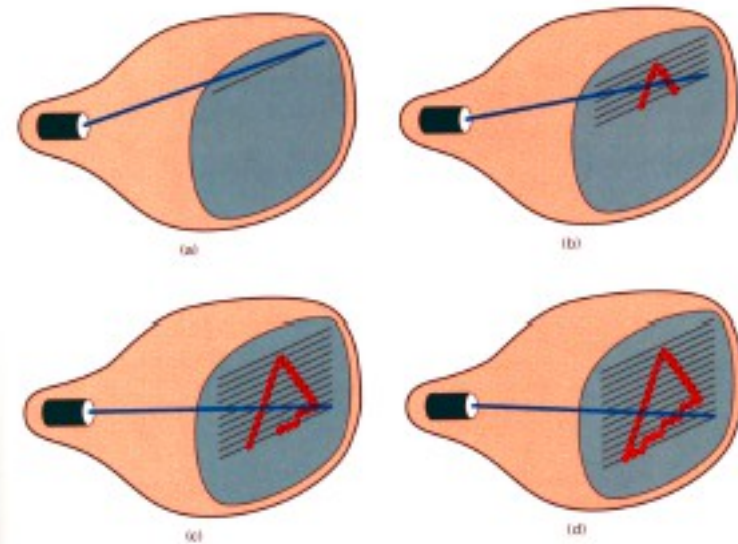
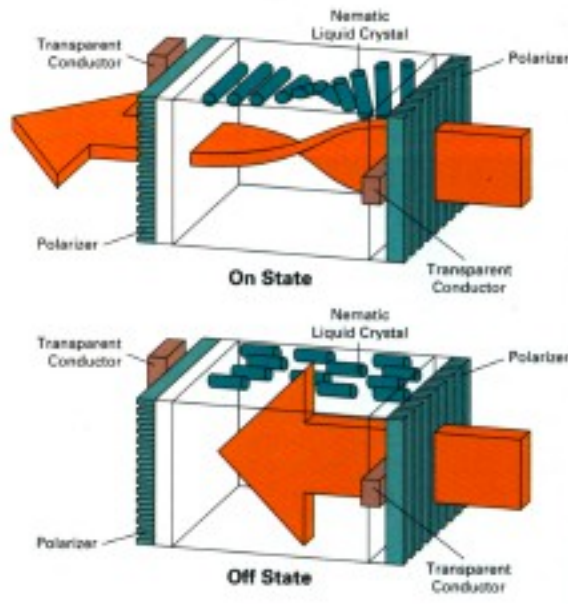
Agenda

- Raster Displays
- Gamma Correction
- RGB Color & Alpha Channel
- Line Drawing

Acknowledgments: Some slides adapted from Steve Marschner and Fredo Durand.

Raster Displays

- Displays that present *raster* images to the user
- LCD, CRT, Projector, ... etc



[H&B fig. 2-7]

Raster Images Vs Vector Images

- Raster Images
 - Made up of pixels with a certain resolution
 - “Pixelization” effect on zoom in
 - E.g. JPEG, BMP, PNG, GIF

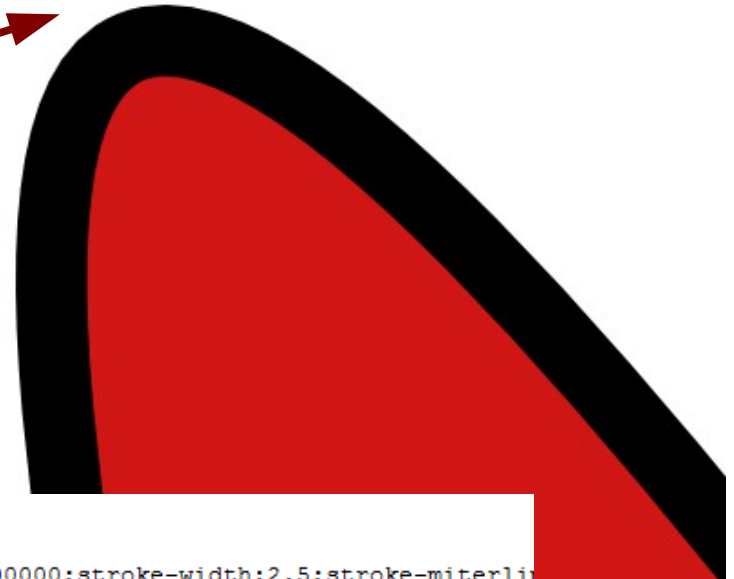


Raster Images Vs Vector Images

- Vector Images
 - Made up of “descriptions” of objects
 - Must be “rasterized” for display
 - Resolution independent
 - E.g. SVG

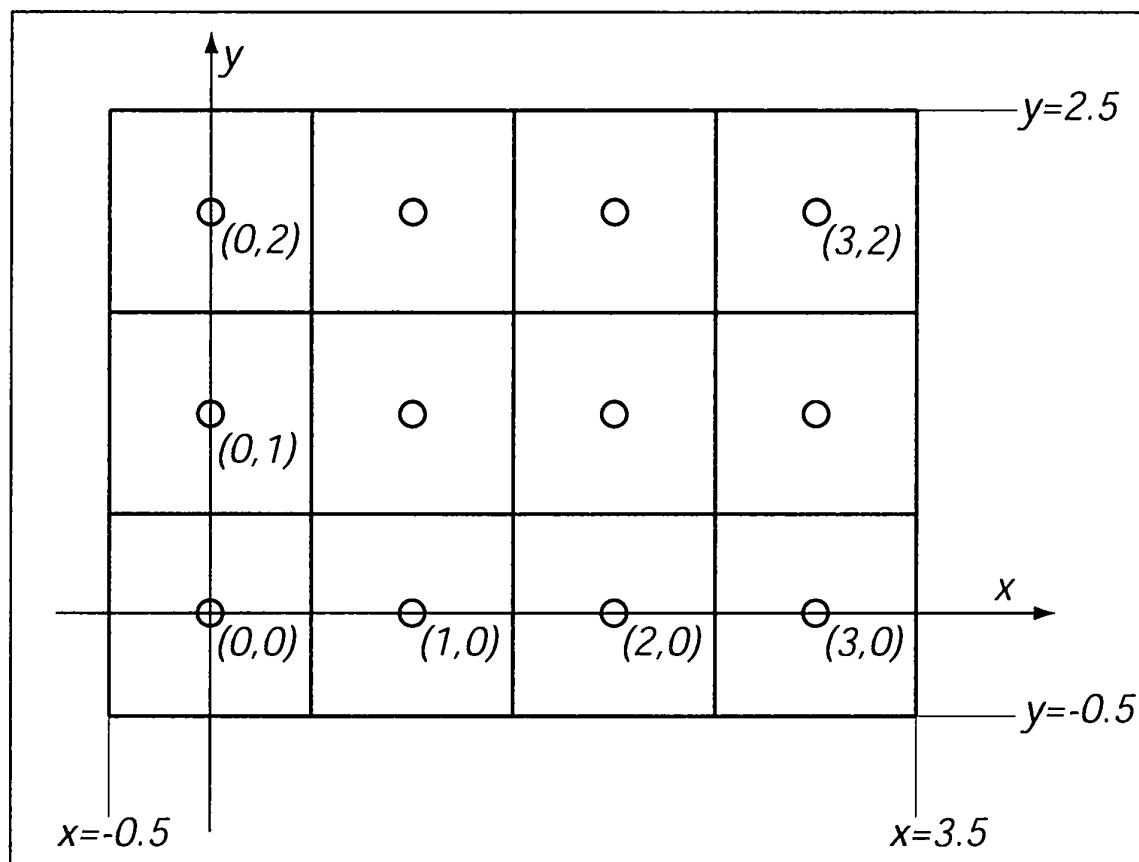


```
<path
  sodipodi:type="star"
  style="fill:#d11616;fill-opacity:1;stroke:#000000;stroke-width:2.5;stroke-miterli
  id="path3003"
  sodipodi:sides="5"
  sodipodi:cx="254.28571"
  sodipodi:cy="448.07648"
  sodipodi:r1="137.35846"
  sodipodi:r2="52.745647"
  sodipodi:arg1="0.52359878"
  sodipodi:arg2="1.1906953"
  inkscape:flatsided="false"
  inkscape:rounded="0.13"
  inkscape:randomized="0.005"
  d="m 372.59548,516.37604 c -7.04466,11.1192 -86.44943,-25.10108 -98.49121,-19.796
  inkscape:transform-center-x="8.8250402"
  inkscape:transform-center-y="4.4368255" />
```



Pixels

- Square Grid of “pixels”
- Each pixel stores color values
- Resolution: size of raster



Pixels at integer coordinates

4 x 3 Pixel Grid

Display Intensity

- How many bits to represent each pixel?
 - Floating point number: 16 or 32 bits
 - Integers: 8, 10, 12, ... bits
- Floating points
 - 0 → black (pixel off)
 - 1 → white (pixel fully on)
 - 0.5 → halfway gray
- Integers: quantized values e.g. 8-bits
 - 0 → black
 - 255 → white

Gamma Correction

- How do displays convert pixel *values* to *intensities*?
 - We would like the screen to:
 - display *white* (intensity 1.0) when we give it 1.0
 - display *black* (intensity 0.0) when we give it 0.0
 - display *grey* (intensity 0.5) when we give it 0.5
- What actually happens?
 - Screens respond non-linearly, e.g. they give intensity
 - 1.0 with input 1.0
 - 0.0 with input 0.0
 - 0.25 with input 0.5

Gamma Correction

This is modeled as:

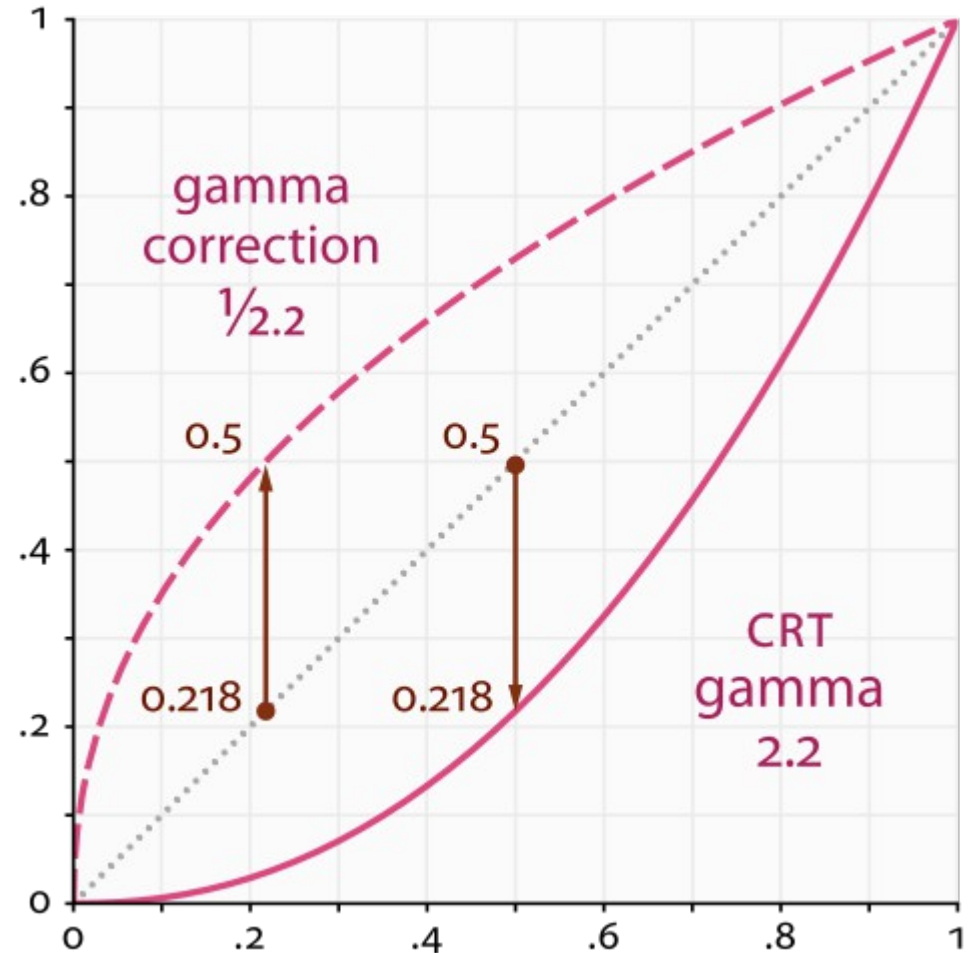
$$I_{out} = I_{max} \times I_{in}^{\gamma}$$

$$\gamma = 2.2$$

$$I_{in} = 0 \rightarrow I_{out} = 0$$

$$I_{in} = 0.5 \rightarrow I_{out} = 0.218$$

$$I_{in} = 1 \rightarrow I_{out} = 1$$

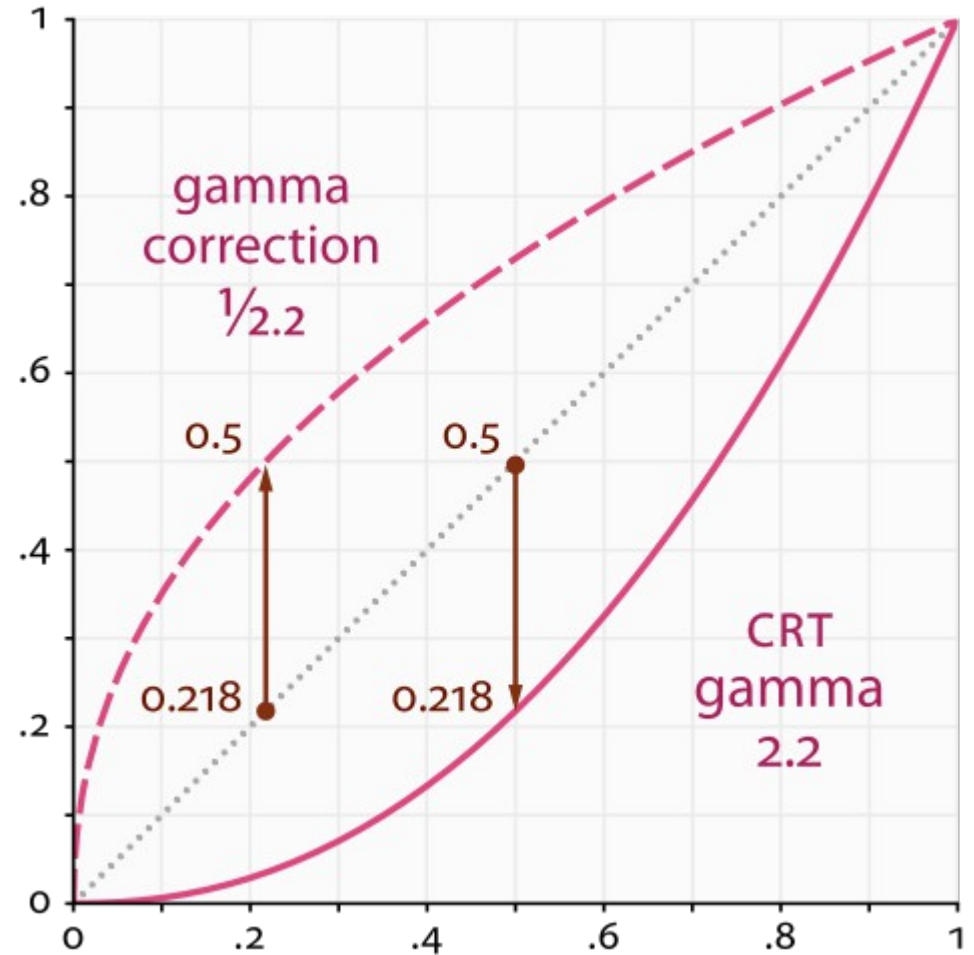


Gamma Correction

To get the correct intensity from the screen, we modify the input:

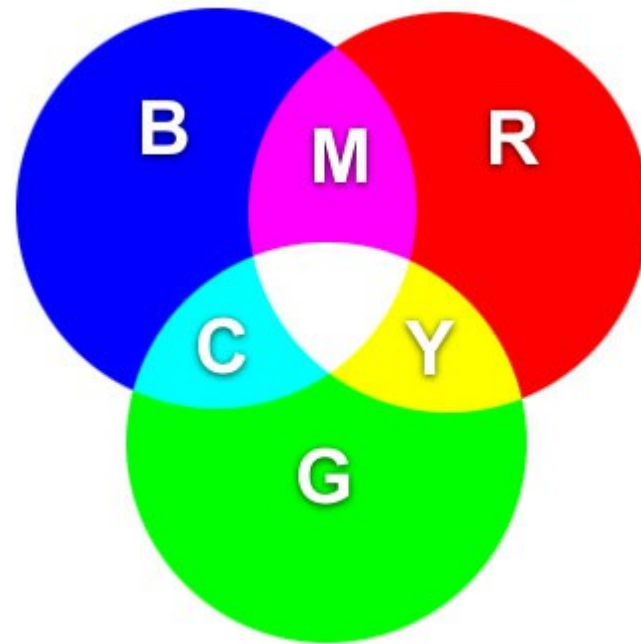
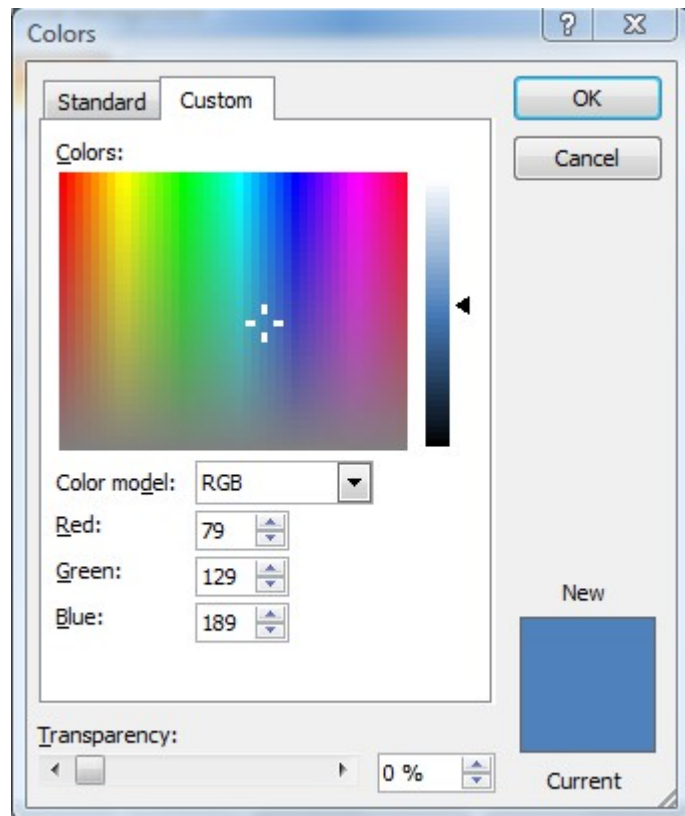
Correction

$$\begin{aligned} I_{out} &= I_{max} \times (I_{in}^{1/\gamma})^\gamma \\ &= I_{max} \times I_{in} \end{aligned}$$



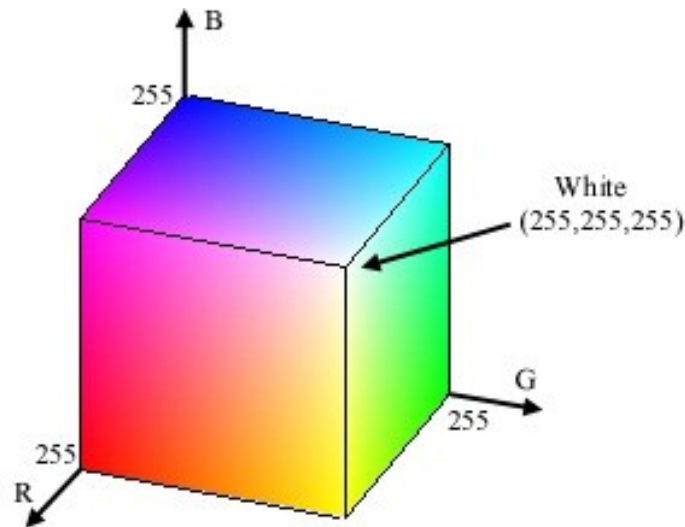
RGB Color

- For color images, each pixel has three values: Red, Green, and Blue Color
- Additive Property

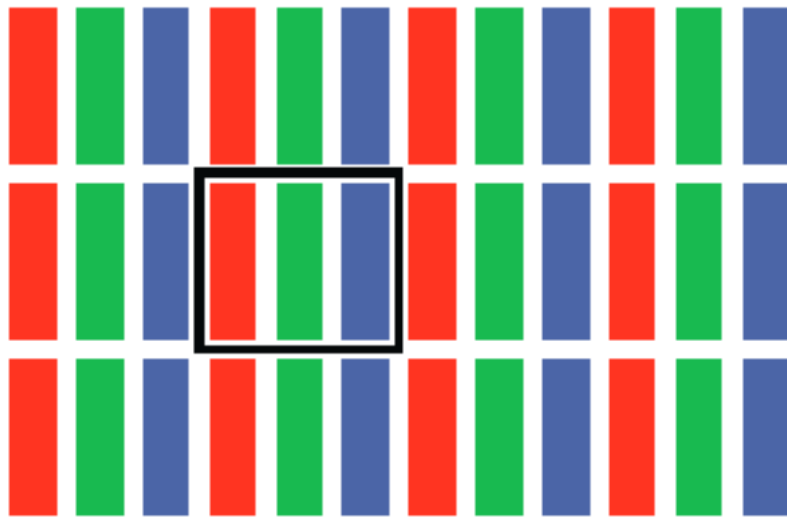


RGB Colors

- Represent colors as 3 dimensional vectors (r, g, b)
 - Black = (0, 0, 0)
 - Red = (1, 0, 0)
 - ...
 - White = (1, 1, 1)

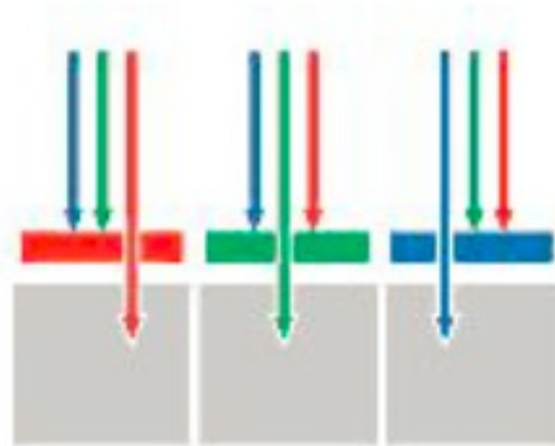


Displays and Cameras



LCD pixels have 3 sub-pixels

Mosaic Capture

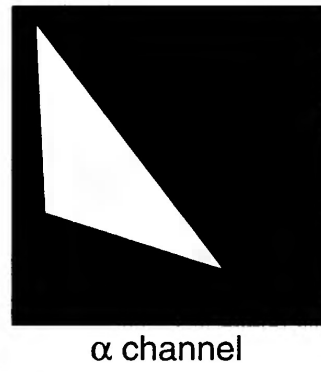
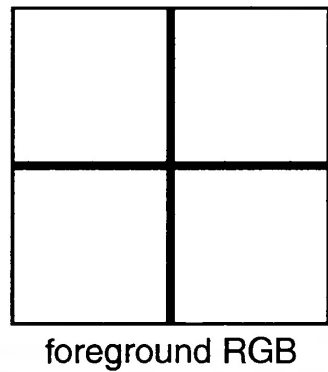
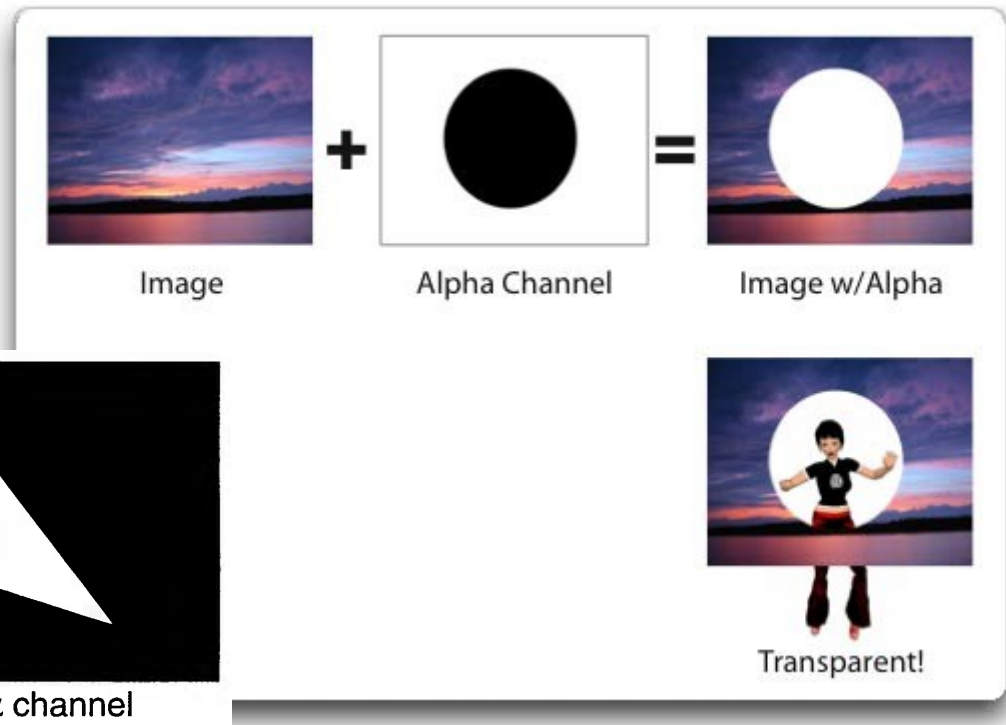


Bayer Mosaic in Digital Cameras

Alpha Channel

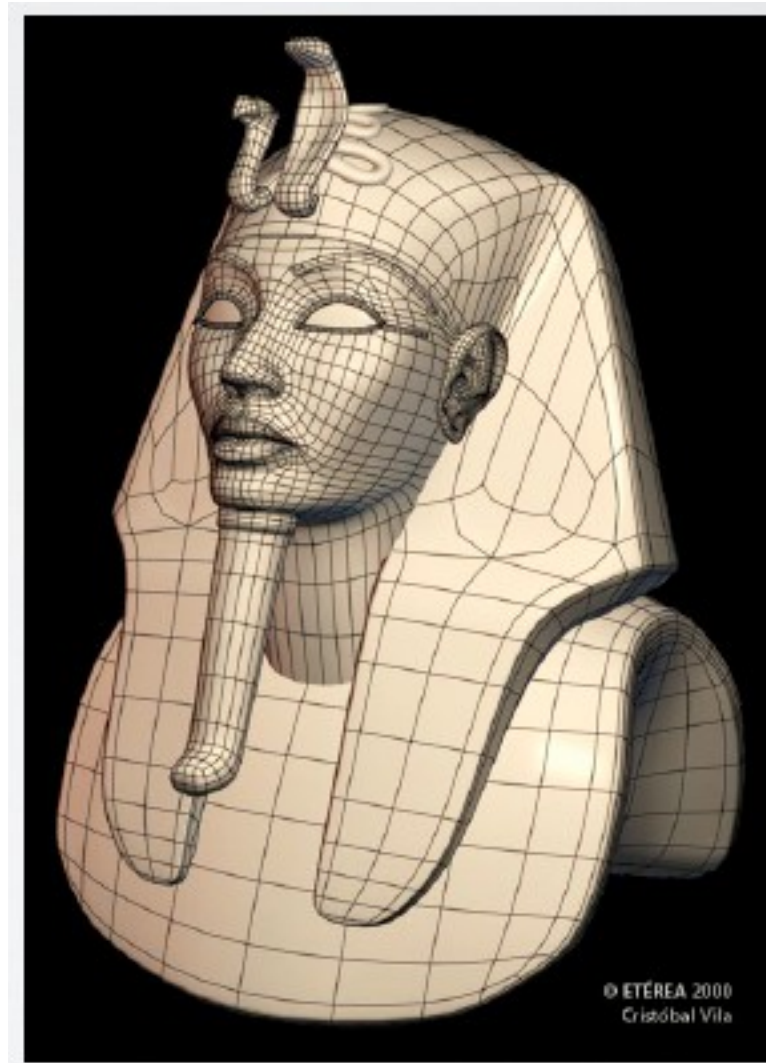
- Handles (partially) transparent objects

$$c = \alpha c_f + (1 - \alpha) c_b$$



Line Drawing

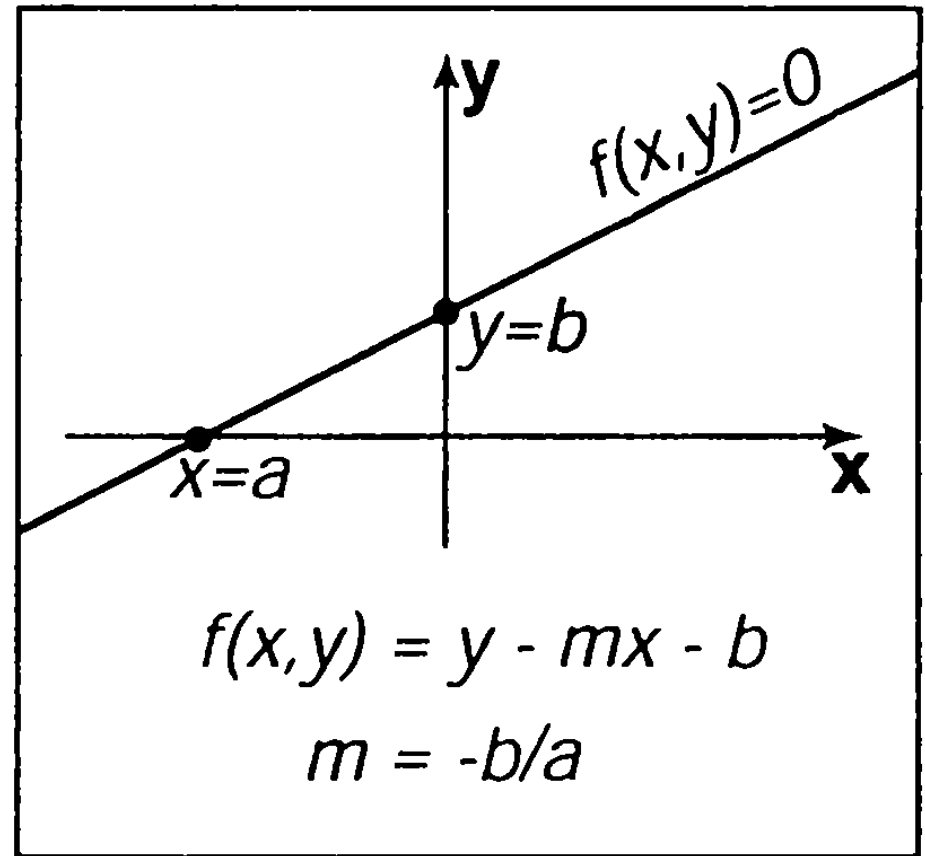
- Basic Operation



Line Representation

- Slope-Intercept
- Problem?
 - Vertical lines!
 - $m = \infty$
- Solution?

$$y = mx + b$$



Line Representation

- Implicit Form $f(x, y) = Ax + By + C = 0$

Both points are on the line:

$$Ax_0 + By_0 + C = 0$$

$$Ax_1 + By_1 + C = 0$$

The normal vector on the line is:

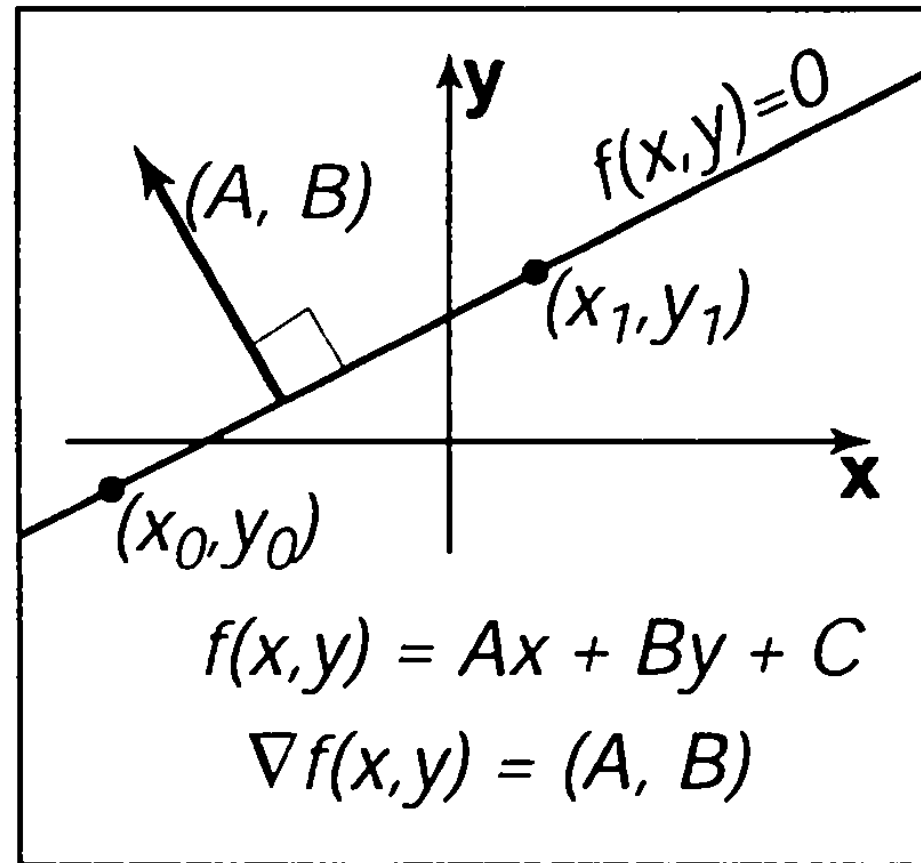
$$(A, B)$$

$$(A, B)^T (x_1 - x_0, y_1 - y_0) = 0$$

why?

One possible (A, B) is:

$$(A, B) = (y_0 - y_1, x_1 - x_0)$$



Line Representation

- Implicit Form $f(x, y) = Ax + By + C = 0$

Plug in (x_0, y_0) we get:

$$(y_0 - y_1)x_0 + (x_1 - x_0)y_0 + C = 0$$

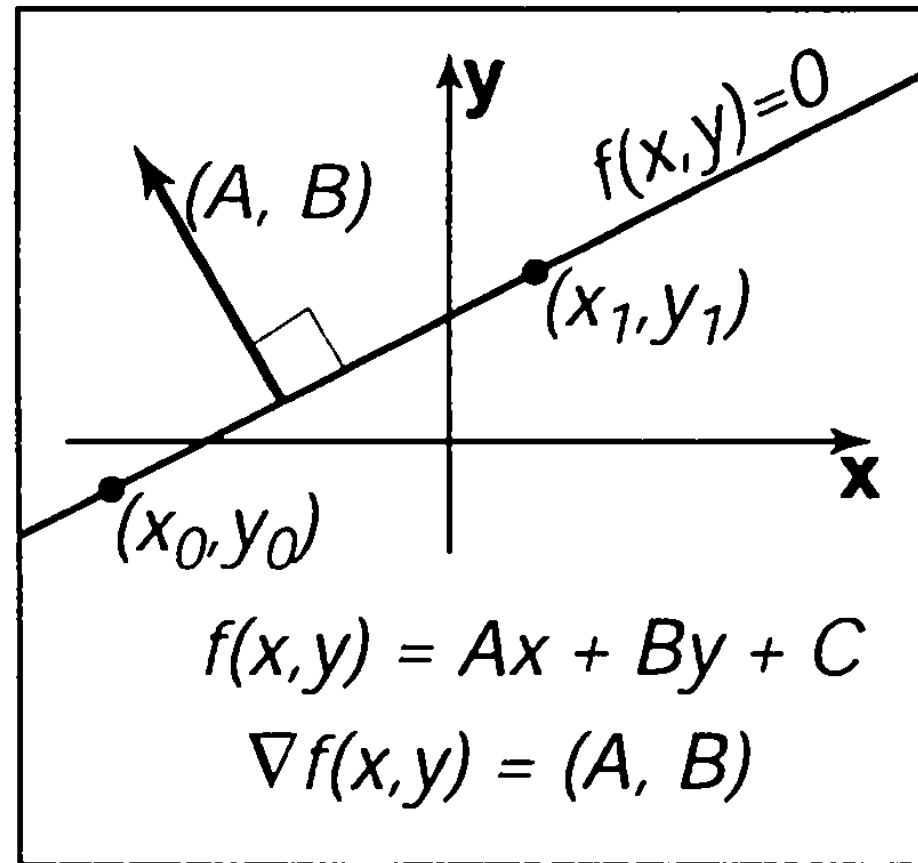
Solve for C :

$$C = x_0 y_1 - x_1 y_0$$

$$A = y_0 - y_1$$

$$B = x_1 - x_0$$

$$C = x_0 y_1 - x_1 y_0$$



Line Representation

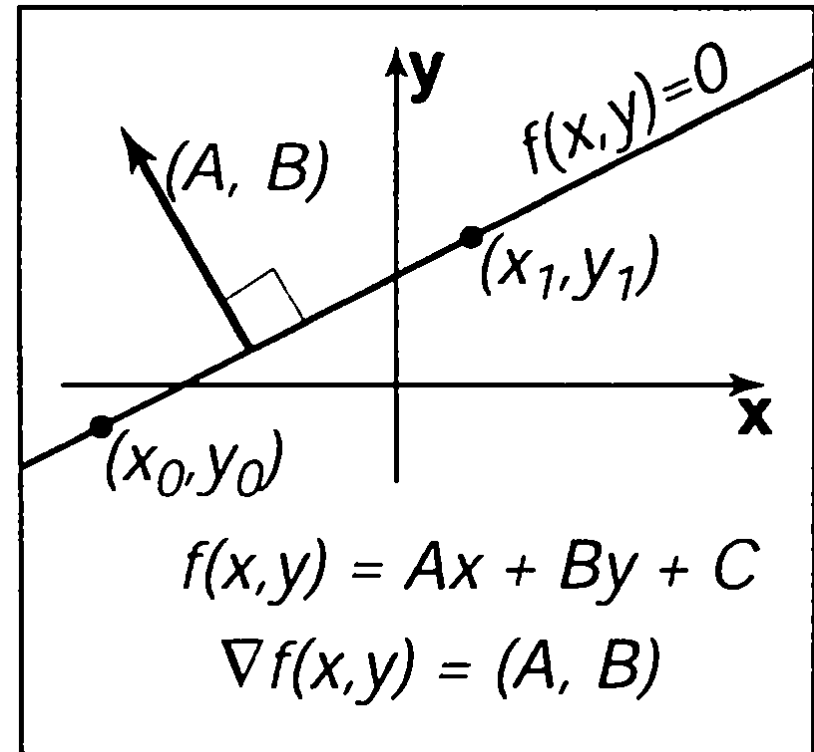
$$f(x, y) = Ax + By + C = 0$$

$f(x, y)$ is the signed scaled distance from the point (x, y) to the line

$f(x, y) = 0 \rightarrow$ On the line

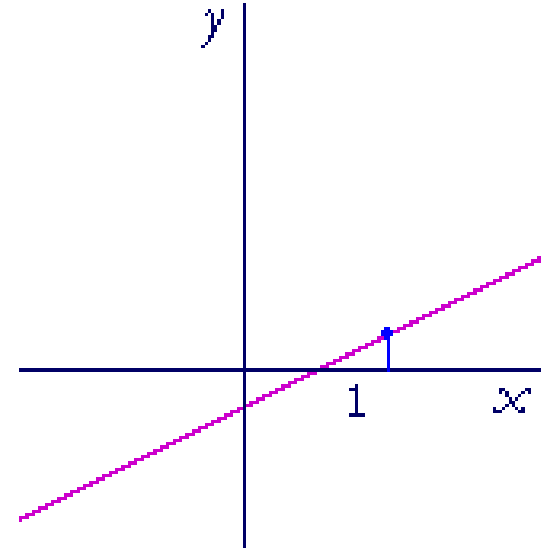
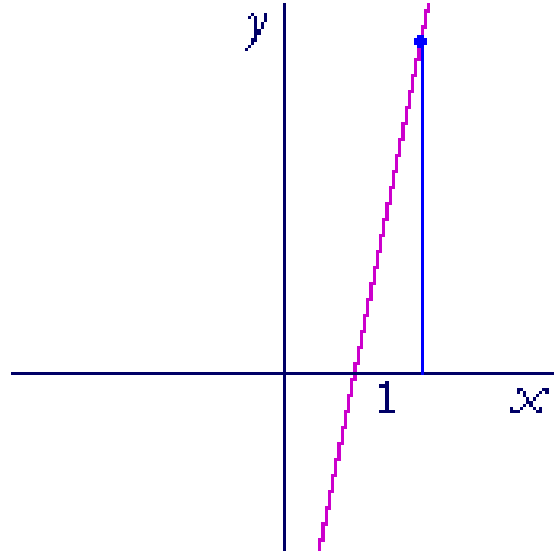
$f(x, y) > 0 \rightarrow$ Above the line

$f(x, y) < 0 \rightarrow$ Below the line



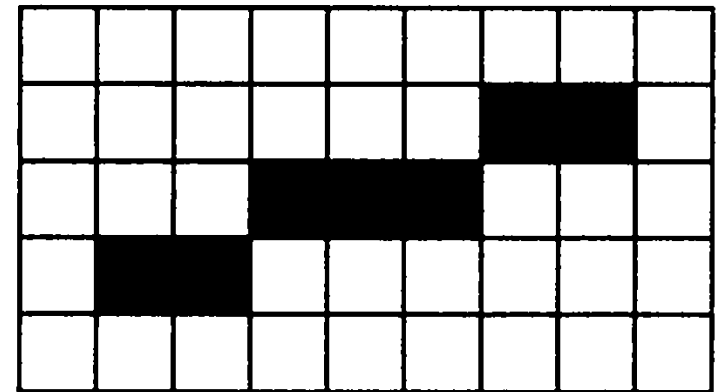
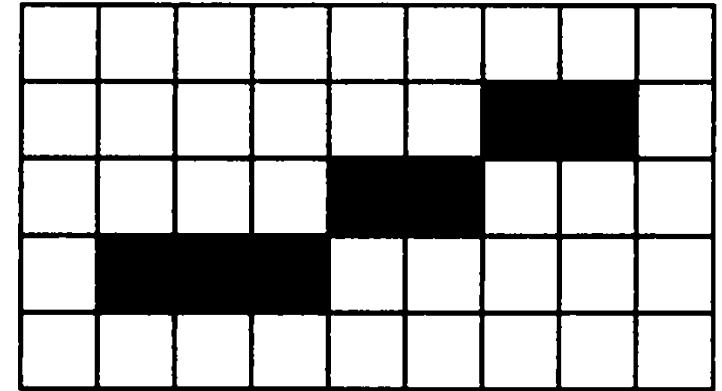
Midpoint Algorithm

- Line Drawing Algorithm
- Uses the Implicit Equation
- Similar to Bresenham's Algorithm
- Integer end-points
- Four Cases:
 - $0 < m \leq 1$
 - $1 < m < \infty$
 - $-1 < m \leq 0$
 - $-\infty < m \leq -1$

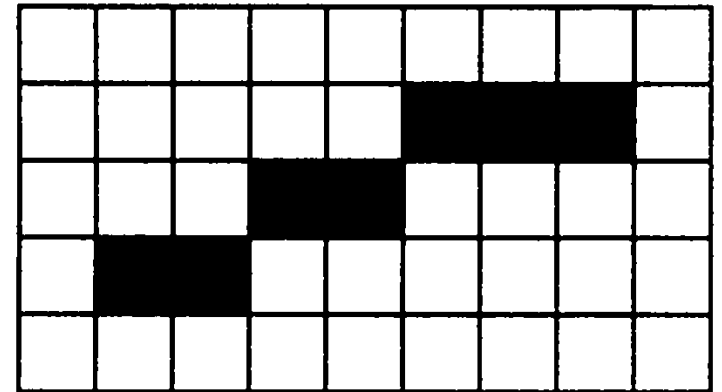


Midpoint Algorithm

- First case:
 - more “run” than “rise”
 - Assume moving right



- Which one is better?

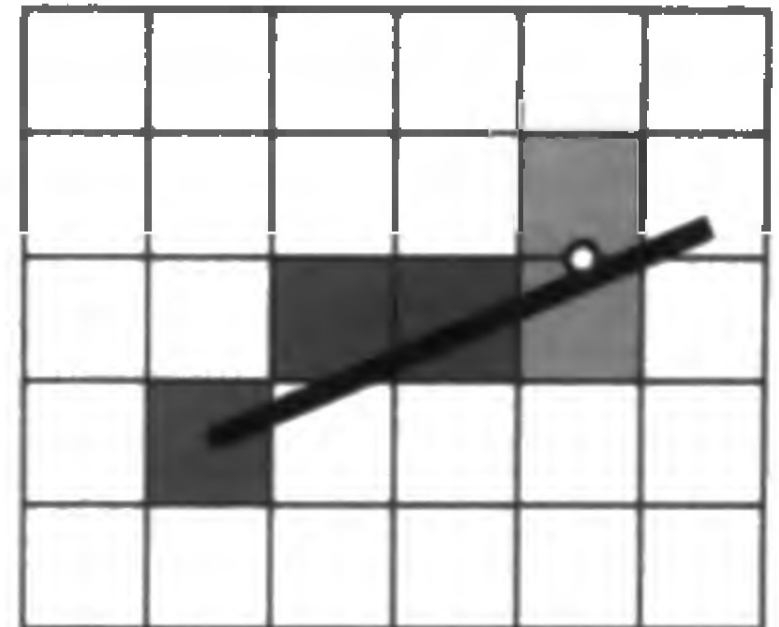
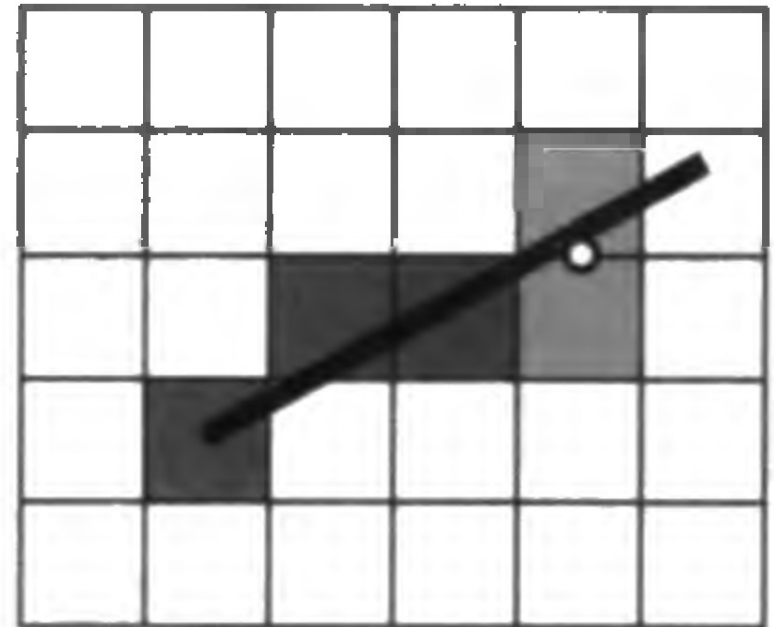


Midpoint Algorithm

- First version

```
 $y = y_0$   
for  $x = x_0 : x_1$   
   $draw(x, y)$   
  if  $f(x+1, y+0.5) < 0$   
     $y = y + 1$ 
```

- Optimizations?



Midpoint Algorithm

- Optimizations
 - Incremental Calculations
 - Integer Operations

```
 $y = y_0$   
for  $x = x_0 : x_1$   
   $draw(x, y)$   
  if  $f(x + 1, y + 0.5) < 0$   
     $y = y + 1$ 
```

Midpoint Algorithm

- Incremental Calculations

- Note that:

$$f(x, y) = (y_0 - y_1)x + (x_1 - x_0)y + C = 0$$

- Which implies that:

$$f(x + 1, y) = f(x, y) + (y_0 - y_1)$$

$$f(x + 1, y + 1) = f(x, y) + (y_0 - y_1) + (x_1 - x_0)$$

Midpoint Algorithm

- Second Pass

```
 $y = y_0$   
 $d = f(x_0 + 1, y_0 + 0.5)$   
for  $x = x_0 : x_1$   
   $draw(x, y)$   
  if  $d < 0$   
     $y = y + 1$   
     $d = d + (x_1 - x_0) + (y_0 - y_1)$   
  else  
     $d = d + (y_0 - y_1)$ 
```

Midpoint Algorithm

- Integer Operations
 - Note that:

$$f(x, y) = 2f(x, y) = 0$$

- Which implies:

$$d = 2f(x_0 + 1, y_0 + 0.5)$$

$$d = 2(y_0 - y_1)(x_0 + 1) + (x_1 - x_0)(2y_0 + 1) + 2x_0y_1 - 2x_1y_0$$

Midpoint Algorithm

- Third Pass

```
 $y = y_0$   
 $d = 2(y_0 - y_1)(x_0 + 1) + (x_1 - x_0)(2y_0 + 1) + 2x_0y_1 - 2x_1y_0$   
for  $x = x_0 : x_1$   
   $draw(x, y)$   
  if  $d < 0$   
     $y = y + 1$   
     $d = d + 2(x_1 - x_0) + 2(y_0 - y_1)$   
  else  
     $d = d + 2(y_0 - y_1)$ 
```

Midpoint Algorithm

- Homework
 - Other three cases
 - Read about Bresenham's Algorithm

Recap

- Raster Displays
- Alpha Channels
- Gamma Correction
- Line Drawing