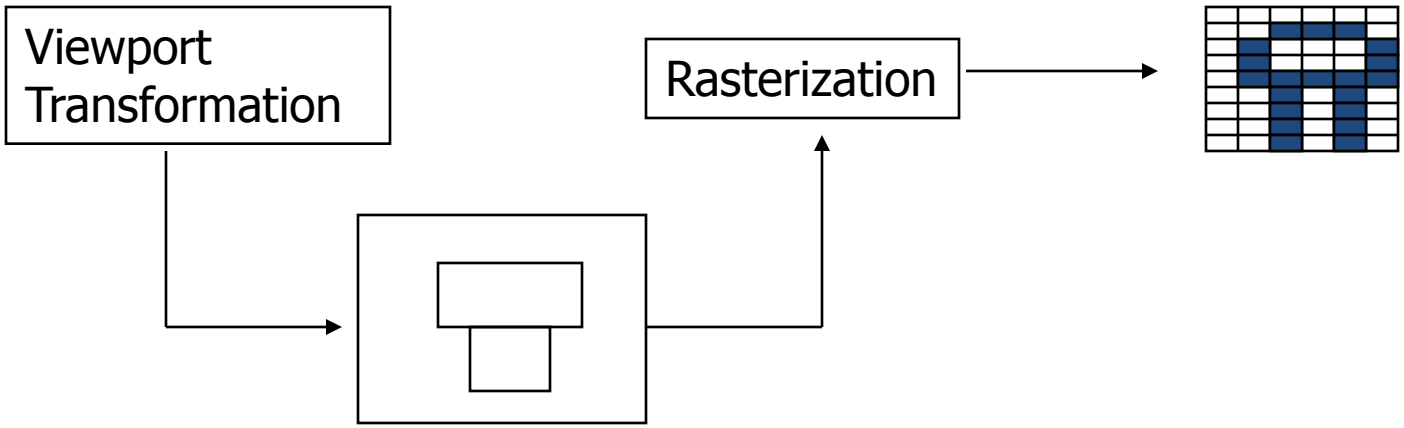
Line Drawing Techniques in Computer Graphics

Jagjit Bhatia

Dept of Computer Science

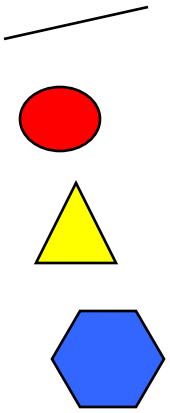
Rasterization (Scan Conversion)

- **Convert high-level geometry description to pixel colors in the frame buffer**
- **Example: given vertex x,y coordinates determine pixel colors to draw line**
- **Two ways to create an image:**
  - **Scan existing photograph**
  - **Procedurally compute values (rendering)**



Rasterization

- A fundamental computer graphics function
- Determine the pixels' colors, illuminations, textures, etc.
- Implemented by graphics hardware
- Rasterization algorithms
  - Lines
  - Circles
  - Triangles
  - Polygons



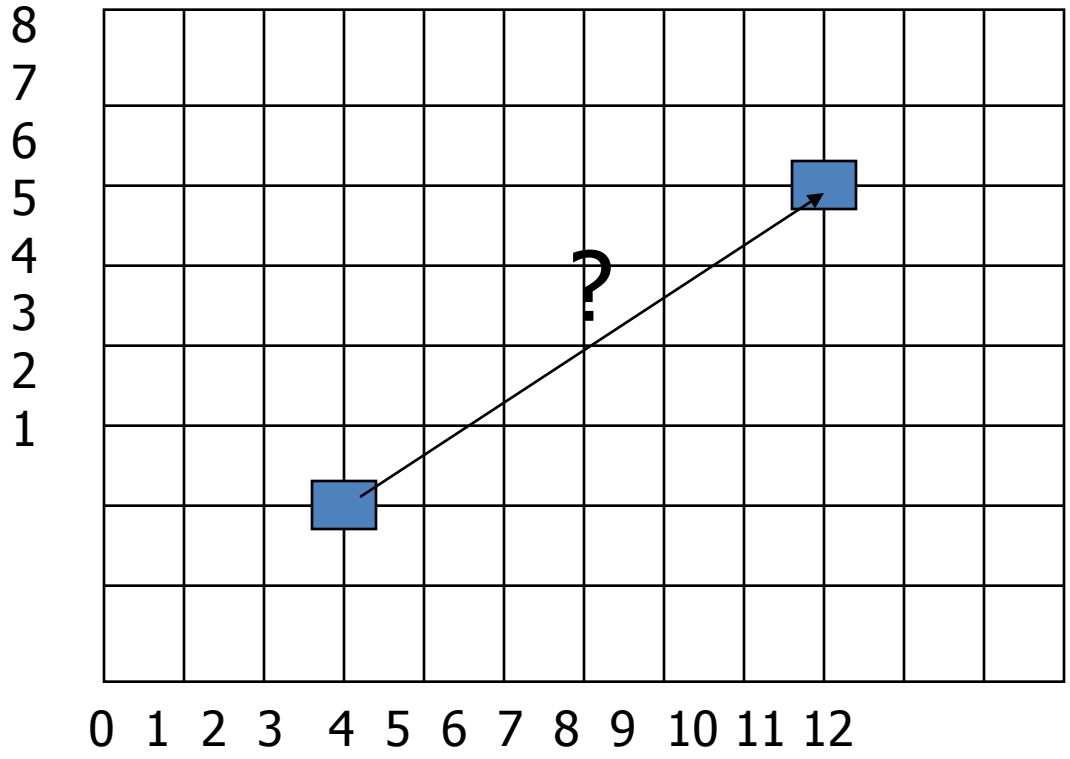
Rasterization Operations

- Drawing lines on the screen
- Manipulating pixel maps (pixmaps): copying, scaling, rotating, etc
- Compositing images, defining and modifying regions
- Drawing and filling polygons
  - Previously gl Begin(GL_POLYGON), etc
- Aliasing and antialiasing methods

Line drawing algorithm

- Programmer specifies (x,y) values of end pixels
- Need algorithm to figure out which intermediate pixels are on line path
- Pixel (x,y) values constrained to integer values
- Actual computed intermediate line values may be floats
- Rounding may be required. E.g. computed point (10.48, 20.51) rounded to (10, 21)
- Rounded pixel value is off actual line path (jaggy!!)
- Sloped lines end up having jaggies
- Vertical, horizontal lines, no jaggies

Line Drawing Algorithm



Line: (3,2) -> (9,6)

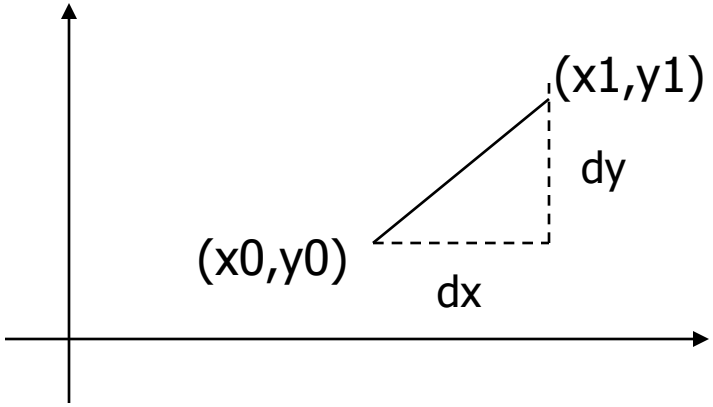
Which intermediate pixels to turn on?

Line Drawing Algorithm

- Slope-intercept line equation
  - y = mx + b
  - Given two end points (x0,y0), (x1, y1), how to compute m and b?

$$m = \frac{dy}{dx} = \frac{y1 - y0}{x1 - x0}$$

$$b = y0 - m * x0$$



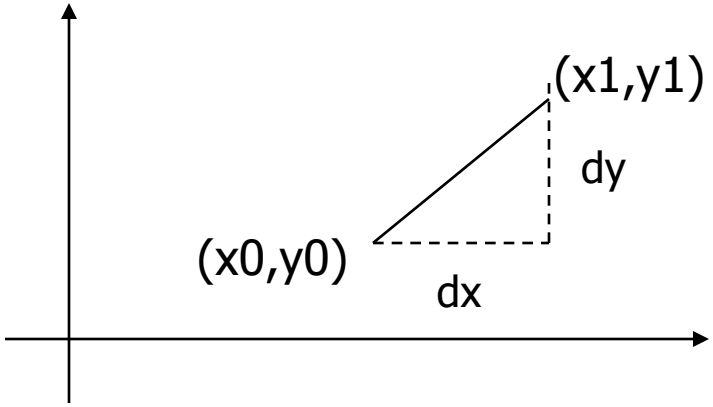
Line Drawing Algorithm

- Numerical example of finding slope m:
- (Ax, Ay) = (23, 41), (Bx, By) = (125, 96)

$$m = \frac{By - Ay}{Bx - Ax} = \frac{96 - 41}{125 - 23} = \frac{55}{102} = 0.5392$$

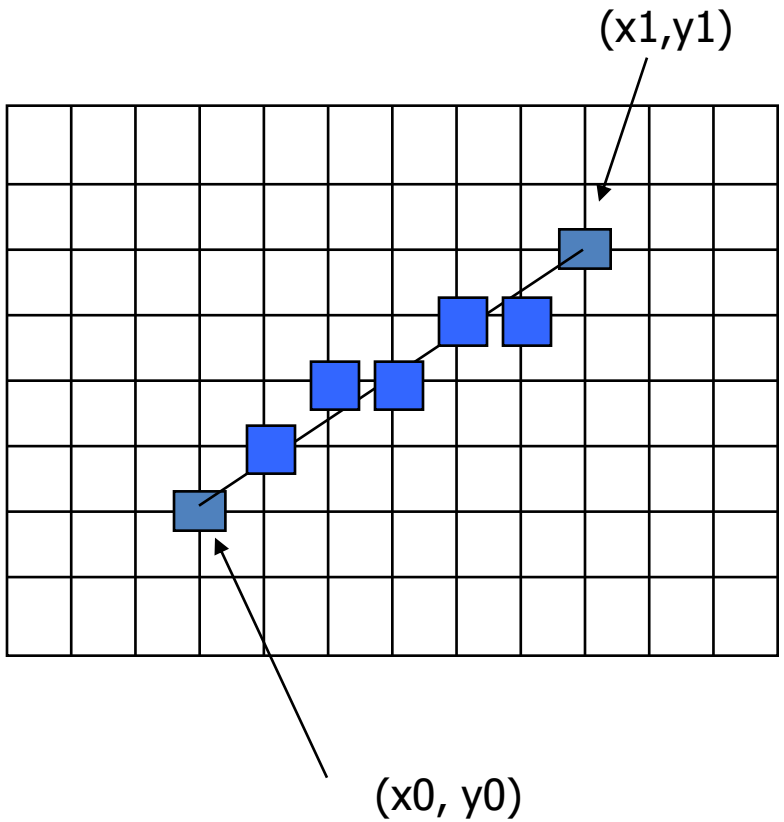
Digital Differential Analyzer (DDA): Line Drawing Algorithm

- Walk through the line, starting at (x0,y0)
- Constrain x, y increments to values in [0,1] range
- Case a: x is incrementing faster (m < 1)
  - Step in x=1 increments, compute and round y
- Case b: y is incrementing faster (m > 1)
  - Step in y=1 increments, compute and round x



DDA Line Drawing Algorithm (Case a: m < 1)

$$y_{k+1} = y_k + m$$



x = x0 y = y0

Illuminate pixel (x, round(y))

x = x0 + 1 y = y0 + 1 * m

Illuminate pixel (x, round(y))

x = x + 1 y = y + 1 * m

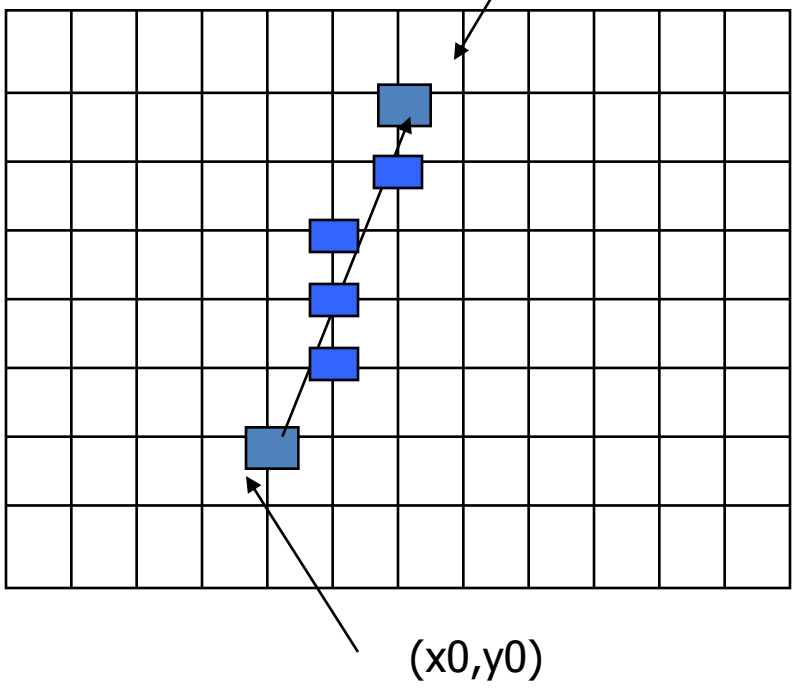
Illuminate pixel (x, round(y))

…

Until x == x1

DDA Line Drawing Algorithm (Case b: m > 1)

$$x_{k+1} = x_k + \frac{1}{m}$$



x = x0 y = y0

Illuminate pixel (round(x), y)

y = y0 + 1 x = x0 + 1 * 1/m

Illuminate pixel (round(x), y)

y = y + 1 x = x + 1 /m

Illuminate pixel (round(x), y)

…

Until y == y1

Consider the line from (0,0) to* (4,6). *Use the simple DDA algorithm to rasterize this line.

Sol. Evaluating steps 1 to 5 in the DDA algorithm we have

$X_I = 0$ $Y_1 = 0$

$X_2 = 4$ $Y_2 = 6$

Length = $|Y_2 - Y_1| = 6$

$\Delta X = |X_2 - X_I| / \text{Length}$

$= \frac{4}{6}$

$\Delta Y = |Y_2 - Y_I| / \text{Length}$

$= 6/6 = 1$

Initial value for

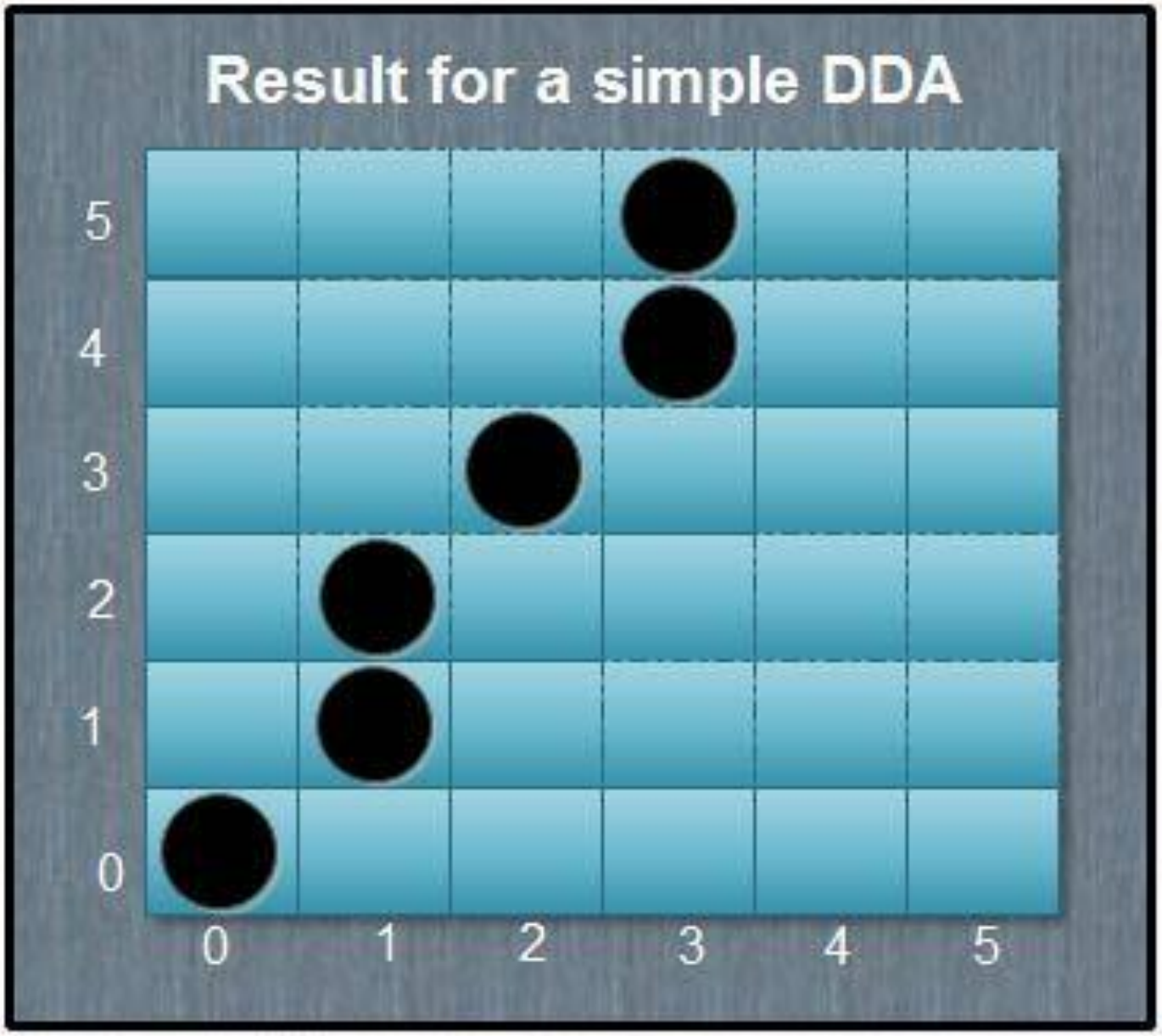
$X = 0 + 0.5 * \text{Sign}(\frac{4}{6}) = 0.5$

$Y = 0 + 0.5 * \text{Sign}(1) = 0.5$

Tabulating the results of each iteration in the step 6 we get,

i	Plot	x	y
		0.5	0.5
1	(0, 0)		
		1.167	1.5
2	(1,1)		
		1.833	2.5
3	(1,2)		
		2.5	3.5
4	(2,3)		

		3.167	4.5
5	(3,4)		
		3.833	5.5
6	(3,5)		
		4.5	6.5


Result for a simple DDA
5
4
3
2
1
0
0
1
2
3
4
5

Example 2 Consider the line from (0,* 0) *to* (-6, -6). *Use the simple* DDA *algorithm line.

Sol. Evaluating steps 1 to 5 in the DDA algorithm we have

$X_l = 0$ $Y_1 = 0$

$X_2 = -6$ $Y_2 = -6$

Length = $l\, X_2 - X_1 l$ $|Y_2 - Y_1 l = 6$

$\Delta X = \Delta Y = -1$

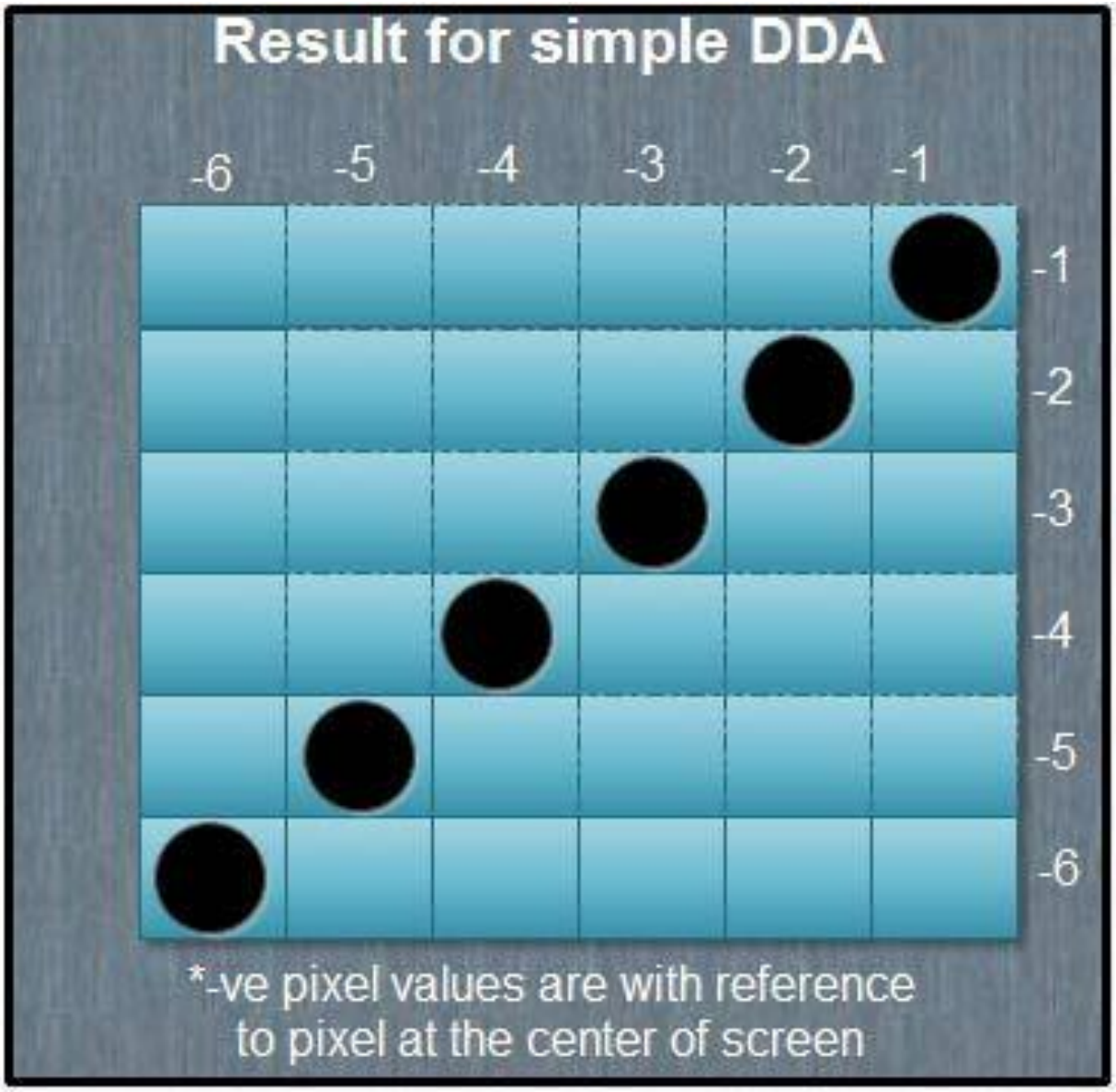
Initial values for

$X = 0 + 0.5 * \text{Sign}(-1) = -0.5$

$Y = 0 + 0.5 * \text{Sign}(-1) = -0.5$

Tabulating the results of each iteration in the step 6 we get,

i	Plot	x	y
		–0.5	–0.5
1	(–1, –1)		
		–1.5	–1.5
2	(–2, –2)		
		–2.5	–2.5
3	(–3, –3)		
		–3.5	–3.5
4	(–4, –4)		
		–4.5	–4.5
5	(–5, –5)		
		–5.5	–5.5
6	(–6, –6)		
		–6.5	–6.5


Result for simple DDA
-6
-5
-4
-3
-2
-1
-1
-2
-3
-4
-5
-6
*-ve pixel values are with reference
to pixel at the center of screen

DDA Line Drawing Algorithm

Step 1: Read the input of the 2 end points of the line as (x1, y1) & (x2, y2) such that x1 != x2 and y1 != y2

Step 2: Calculate dx = x2 – x1 and dy = y2 – y1

Step 3:

```
if(dx>=dy)
step=dx
else
step=dy
```

Step 4: xin = dx / step & yin = dy / step

Step 5: x = x1 + 0.5 & y = y1 + 0.5

Step 6:

```
for(k = 0; k < step; k++)
{
x = x + xin
y = y + yin
putpixel(x, y)
}
```

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>

void main( )
{
float x,y,x1,y1,x2,y2,dx,dy,step;
int i,gd=DETECT,gm;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

printf("Enter the value of x1 and y1 : ");
scanf("%f%f",&x1,&y1);
printf("Enter the value of x2 and y2: ");
scanf("%f%f",&x2,&y2);

dx=abs(x2-x1);
dy=abs(y2-y1);
```

```
if(dx>=dy)
step=dx;
else
step=dy;
 dx=dx/step;
dy=dy/step;
 x=x1;
y=y1;

i=1;
while(i<=step)
{
putpixel(x,y,5);
x=x+dx;
y=y+dy;
i=i+1;
delay(100);
}
 closegraph();
}
```

Advantages of DDA Algorithm

1. It is the simplest algorithm and it does not require special skills for implementation.
2. It is a faster method for calculating pixel positions than the direct use of equation y = mx + b. It eliminates the multiplication in the equation by making use of raster characteristics, so that appropriate increments are applied in the x or v direction to find the pixel positions along the line path

Line Drawing Algorithm Drawbacks

- DDA is the simplest line drawing algorithm
  - Not very efficient
  - Round operation is expensive
- Optimized algorithms typically used.
  - Integer DDA
  - E.g.Bresenham algorithm (Hill, 10.4.1)
- Bresenham algorithm
  - Incremental algorithm: current value uses previous value
  - Integers only: avoid floating point arithmetic
  - Several versions of algorithm: we'll describe midpoint version of algorithm