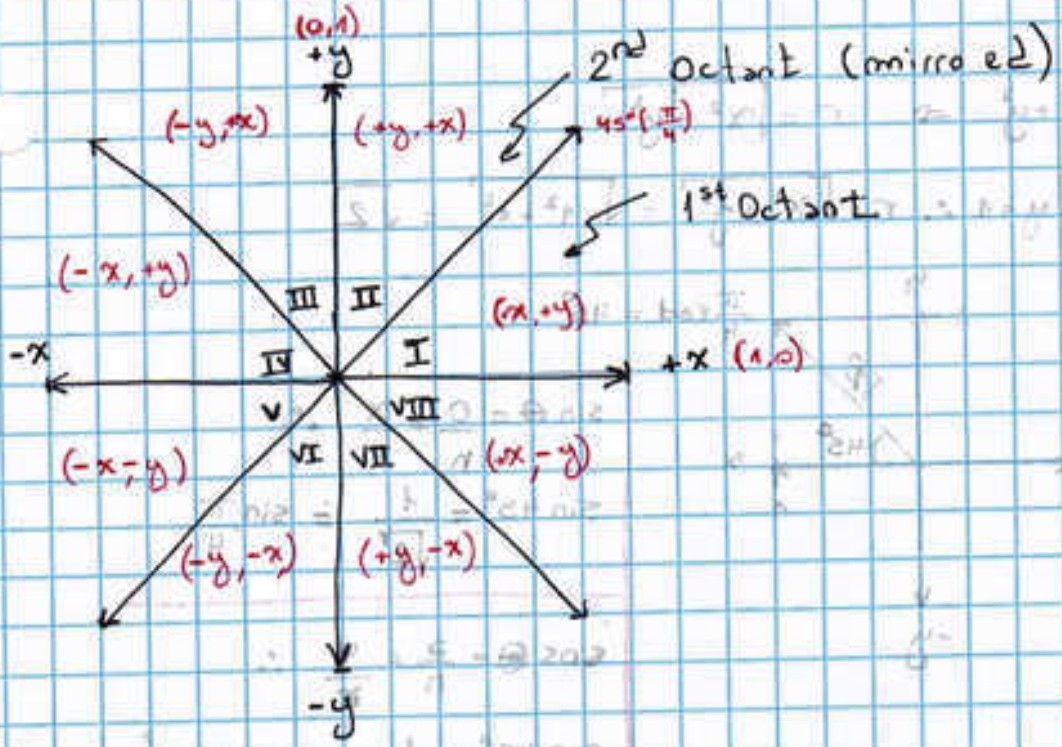
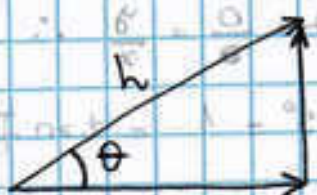


soh cah toA



$\sin\theta = \frac{o}{h} = \frac{y}{r}$

$\cos\theta = \frac{a}{h} = \frac{x}{r}$

$\tan\theta = \frac{o}{a} = \frac{y}{x}$

$h^2 = a^2 + o^2$

$r^2 = x^2 + y^2$

~~$x = \cos\theta$~~

~~$y =$~~

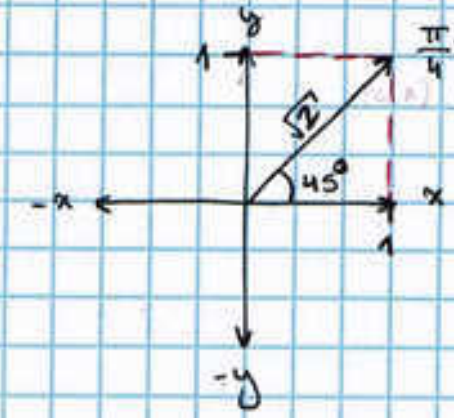
$x = r \cdot \cos\theta = a = h \cdot \cos\theta$

$y = r \cdot \sin\theta = o = h \cdot \sin\theta$

c.

$r^2 = x^2 + y^2 \Rightarrow r = \sqrt{x^2 + y^2}$

if $x = 1$ & $y = 1$ $\therefore$ $r = \sqrt{x^2 + y^2} = \sqrt{1^2 + 1^2} = \sqrt{2}$



$\sin\theta = \frac{o}{h} = \frac{x}{r}$ $\therefore$

$\sin 45° = \frac{1}{\sqrt{2}} = \sin\frac{\pi}{4}$

$\cos\theta = \frac{a}{h} = \frac{y}{r}$ $\therefore$

$\cos 45° = \frac{1}{\sqrt{2}} = \cos\frac{\pi}{4}$

$\tan\theta = \frac{o}{a} = \frac{y}{x}$ $\therefore$

$\tan 45° = 1 = \tan\frac{\pi}{4}$

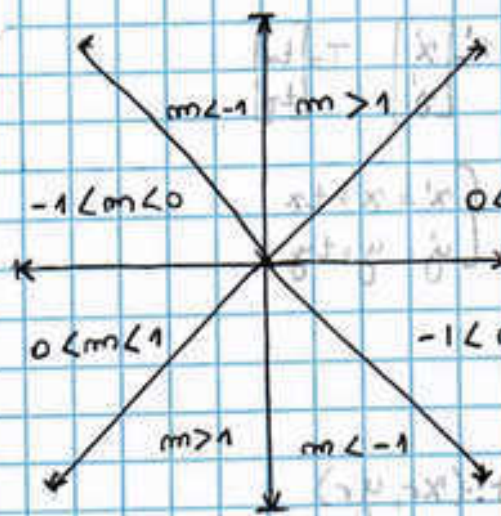
$\tan\theta = \frac{o}{a} = \frac{r \cdot \sin\theta}{r \cdot \cos\theta} = \frac{\sin\theta}{\cos\theta}$

$\tan 45° = \frac{\sin 45°}{\cos 45°} = \frac{\sin\frac{\pi}{4}}{\cos\frac{\pi}{4}} = \frac{1/\sqrt{2}}{1/\sqrt{2}}$

$= \frac{1/\sqrt{2}}{\sqrt{2}/\sqrt{2}} = \frac{1}{\sqrt{2}} \cdot \frac{\sqrt{2}}{1} = 1$

$= \frac{1/\sqrt{2}}{\sqrt{2}/1}$

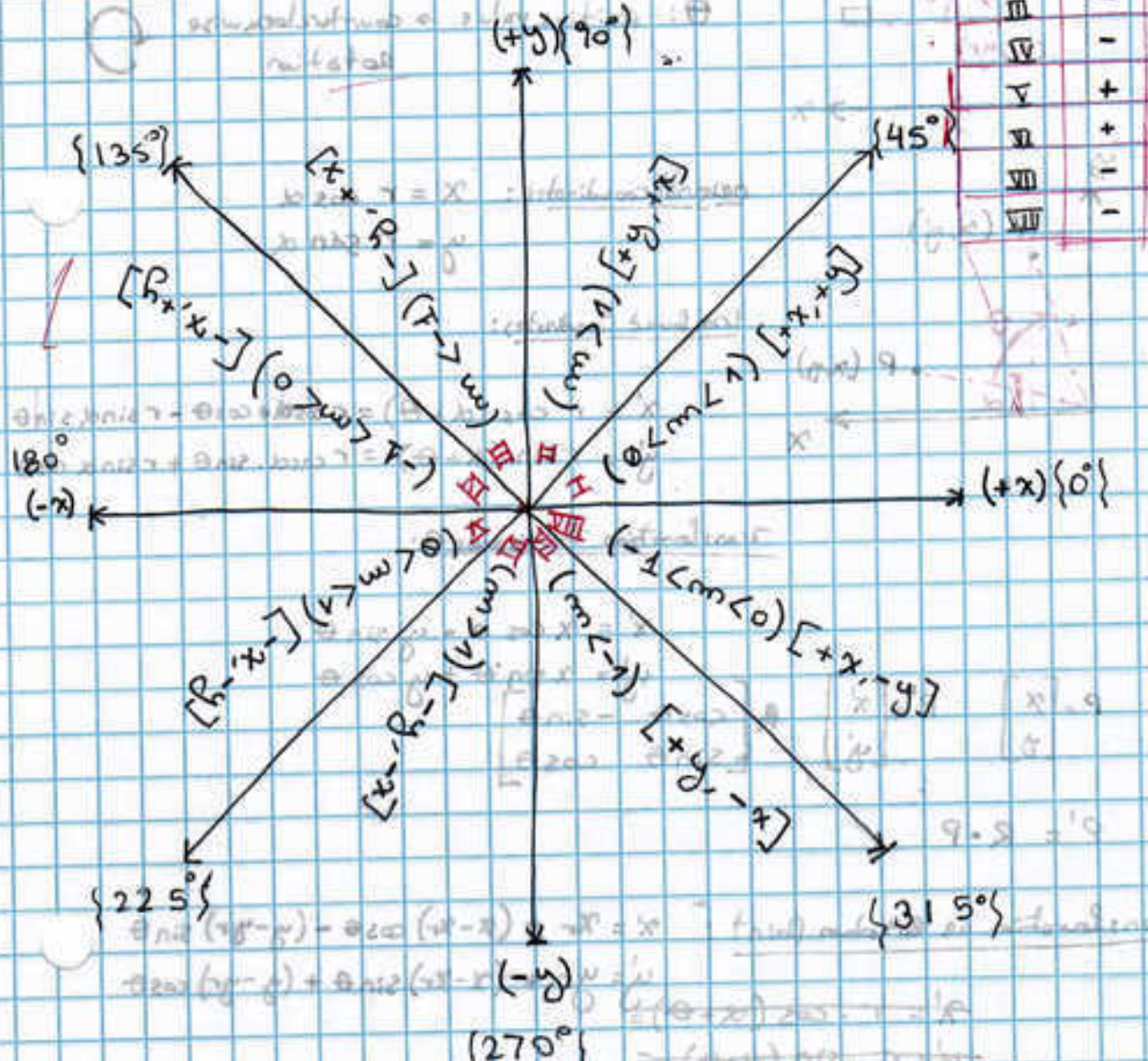
(slope) $m = \frac{dy}{dx}$



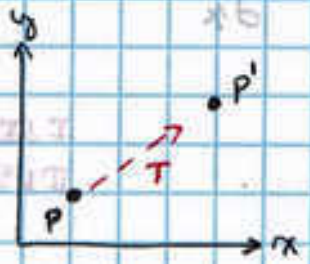
I & IV swap (x,y)(-y,x)

II & III reverse

octant	slope	dx dy ratio
I	+	dy > dx
II	+	dy < dx
III	-	dy < dx
IV	-	dy > dx
V	+	dy > dx
VI	+	dy < dx
VII	-	dy < dx
VIII	-	dy > dx



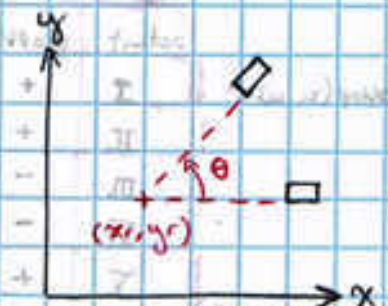
Two Dimensional Translation



$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

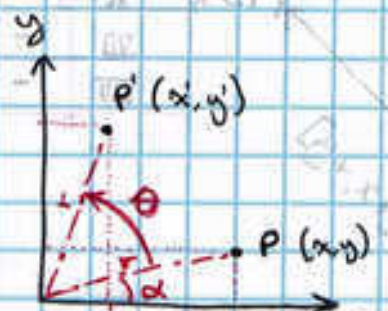
$$P' = P + T \begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

Two Dimensional Rotation



Rotation point: (x_r, y_r)
(pivot point)

θ: positive value → counterclockwise rotation



original coordinates: $x = r\cos\alpha$, $y = r\sin\alpha$

transformed coordinates:

$$x' = r\cos(\alpha+\theta) = r\cos\alpha\cos\theta - r\sin\alpha\sin\theta$$
$$y' = r\sin(\alpha+\theta) = r\cos\alpha\cdot\sin\theta + r\sin\alpha\cos\theta$$

Translation through angle θ:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix} \quad P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

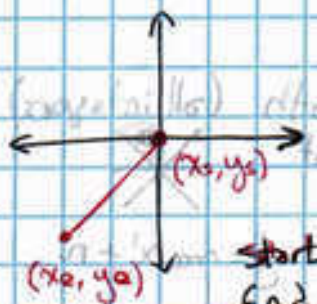
$$P' = R \cdot P$$

Translation on Rotation Point: $x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

~~$x' = r\cdot\cos(\alpha+\theta) =$~~

~~$y' = r\cdot\sin(\alpha+\theta) =$~~

① 

start point : (xs, ys)
End point : (xe, ye)

② Bresenham work on Octant I 

③ Obtain angle ~~tan~~ of line β

④ Calculate β' so β' $0° < \beta' < 45°$

⑤ Use Bresenham Algorith to store all points coordinate

⑥ Use 2D rotation to rotate points to β

sohcahtoa 

$\tan\alpha = \frac{o}{a} = \frac{y}{x}$

Goals

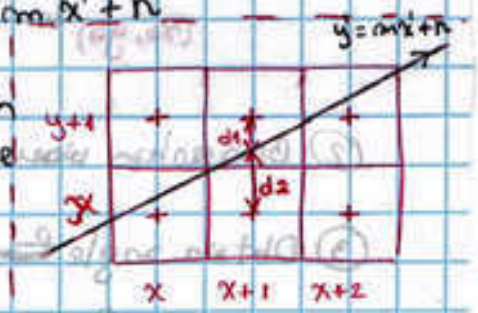
128 erdwell p1 z23, end w 11

82 Bresenham's Line Draw Algorithm

Positive slope

- Draw line without using floating point math (all integers)
- Basic algorithm draws a line at 2nd Octant

1. We wish to draw a line which equation is $y' = mx' + n$



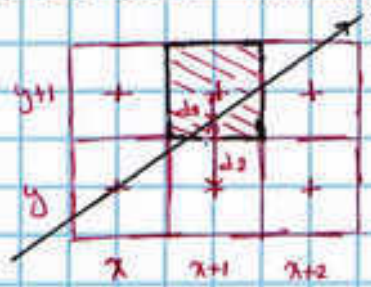
2. Draw a pixel on position (x, y) & verify which closest point pixel to (x', y') belong to the line. $(x', y') \in$ line

3. Move along x-axis, use algorithm to decide which next point on y-axis to draw.

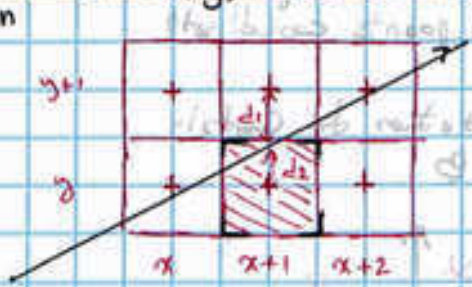
- Do $(x+1, y)$ pixel or $(x+1, y+1)$ pixel belong to line $y = mx' + n$?

- if point is not in the center of the pixel, we we must find the second best point who's distance from the line is the smallest ($d1 > d2$ or $d2 < d1$)

4. if $d1 < d2$ then next point to (x, y) should be $(x+1, y+1)$
   if $d2 > d2$ then next point to (x, y) should be $(x+1, y)$



$d1 < d2$
$(x+1, y+1)$



$d1 > d2$
$(x+1, y)$

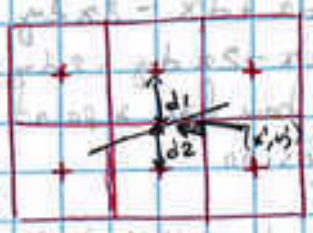
→ For every increment of x coordinate we must check if we should increment y coordinate

5.

Bresenham's Line Drawing Algorithm (continuation) [positive slope]

5. $d_1 = (y+1) - y'$

$d_2 = y' - y$



$$\boxed{d_1 - d_2} = (y+1) - y' - (y' - y)$$
$$= y + 1 - y' - y' + y$$
$$= 2y - 2y' + 1 = \boxed{2y + 1 - 2y'}$$

6. $\begin{cases} d_1 - d_2 = 2y + 1 - 2y' \\ y' = mx' + n \end{cases} \Rightarrow \boxed{d_1 - d_2} = 2y + 1 - 2y'$

$$= \boxed{2y + 1 - 2(mx' + n)}$$

7. For every iteration $\boxed{x' = x+1}$ and slope $\boxed{m = \frac{dy}{dx} = \frac{(y+1) - y}{x+1 - x}}$

$$\begin{cases} d_1 - d_2 = 2y + 1 - 2(mx' + n) \\ x' = x + 1 \end{cases}$$

$$d_1 - d_2 = 2y + 1 - 2(mx' + n)$$
$$= 2y + 1 - 2[m(x+1) + n]$$
$$= 2y + 1 - 2[mx + m + n]$$
$$= 2y + 1 - 2mx - 2m - 2n$$
$$= 2y + 1 - 2m(x+1) - 2n$$

$$\begin{cases} d_1 - d_2 = 2y + 1 - 2m(x+1) - 2n \\ m = \frac{dy}{dx} \end{cases}$$

$$d_1 - d_2 = 2y + 1 - 2\left(\frac{dy}{dx}\right)(x+1) - 2n$$

we wish to eliminate division

$$dx(d_1 - d_2) = dx\left(2y + 1 - 2\frac{dy}{dx}(x+1) - 2n\right)$$

$$\boxed{dx(d_1 - d_2)} = 2y\,dx + dx - 2dy(x+1) - 2n\,dx$$

$$= \boxed{2y\,dx + dx - 2x\,dy - 2dy - 2n\,dx}$$

84 8. Let $\begin{cases} f(x,y) = dx(d_1 - d_2) = \\ dx(d_1 - d_2) = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dx \end{cases}$

$f(x,y) = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dx$

a. $f(x,y)$ determine how far a point is from the line without using division

b. $f(x,y)$ is the function that determine the distance from the pixel we want to draw (x,y) to the line $y = mx + n$

9. Let (x_i, y_i) be the last point we draw

Let (x_{i+1}, y_{i+1}) be the next point we will draw

For each iteration $f(x_{i+1}, y_{i+1}) - f(x_i, y_i)$

$$\begin{cases} f(x_i, y_i) = 2y_i\,dx + dx - 2x_i\,dy - 2dy - 2n\,dx \\ f(x_{i+1}, y_{i+1}) = 2y_{i+1}\,dx + dx - 2x_{i+1}\,dy - 2dy - 2n\,dx \end{cases}$$

~~$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = [2y_{i+1}\,dx - 2x_{i+1}\,dy - 2dy - 2n\,dx] - [2y_i\,dx + dx - 2x_i\,dy - 2dy - 2n\,dx]$~~

~~$= 2y_{i+1}\,dx - 2x_{i+1}\,dy - 2dy - 2n\,dx - 2y_i\,dx - dx + 2x_i\,dy + 2dy + 2n\,dx$~~

$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2y_{i+1}\,dx + dx - 2x_{i+1}\,dy - 2dy - 2n\,dx - [2y_i\,dx + dx - 2x_i\,dy - 2dy - 2n\,dx]$

$= 2y_{i+1}\,dx + dx - 2x_{i+1}\,dy - 2dy - 2n\,dx - 2y_i\,dx - dx + 2x_i\,dy + 2dy + 2n\,dx$

$= 2y_{i+1}\,dx - 2x_{i+1}\,dy - 2y_i\,dx + 2x_i\,dy$

$= 2dx(y_{i+1} - y_i) - 2dy(x_{i+1} - x_i)$

Bresenham line drawing Algorithm (continued) [Positive slope]

10. ~~$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y(x_i, y_i)$~~

$$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y(x_i, y_i)(x_{i+1} - x_i)$$

- For each increment of the x coordinate each iteration is $x_{i+1} = x_i + 1$

$$\begin{cases} f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y(x_{i+1} - x_i) \\ x_{i+1} = x_i + 1 \end{cases}$$

$$\boxed{f(x_{i+1}, y_{i+1}) - f(x_i, y_i)} = 2d_x(y_{i+1} - y_i) - 2d_y(x_{i+1} - x_i)$$

$$= 2d_x(y_{i+1} - y_i) - 2d_y(x_i + 1 - x_i)$$

$$= \boxed{2d_x(y_{i+1} - y_i) - 2d_y}$$

11. ~~$f(x_{i+1}, y$~~

$$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y$$

- if $\boxed{d_1 < d_2}$ then next point of (x, y) should be (x_{i+1}, y_{i+1})

therefore $\begin{cases} y_{i+1} = y_i + 1 \\ f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y \end{cases}$

$$\boxed{f(x_{i+1}, y_{i+1}) - f(x_i, y_i)} = 2d_x(y_{i+1} - y_i) - 2d_y$$

$$= 2d_x(y_i + 1 - y_i) - 2d_y$$

$$= \boxed{2d_x - 2d_y}$$

- if $\boxed{d_1 > d_2}$ then next point of (x, y) should be $(x+1, y)$

therefore $\begin{cases} y_{i+1} = y_i \\ f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2d_x(y_{i+1} - y_i) - 2d_y \end{cases}$

$$\boxed{f(x_{i+1}, y_{i+1}) - f(x_i, y_i)} = 2d_x(y_{i+1} - y_i) - 2d_y$$

$$= 2d_x(y_i - y_i) - 2d_y$$

$$= \boxed{-2d_y}$$

12. Find out $f(x,y)$ for the initial start point:

ej Let assume the initial start point is $(0,0)$

(Even if $(0,0)$ is not on the line we only care for the slope and direction)

(A) $f(x,y) = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dy$

(B) $y = mx + n$

(C) $m = \frac{dy}{dx}$

↖ we need to find this

1. $\begin{cases} y = mx + n \\ m = \frac{dy}{dx} \end{cases} \Rightarrow \boxed{y = mx + n} = \boxed{\frac{dy}{dx}\,x + n}$

2. $\begin{cases} y = \frac{dy}{dx}\,x + n \\ f(x,y) = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dy \end{cases}$

1. $\begin{cases} y = mx + n \\ m = \frac{dy}{dx} \end{cases} \Rightarrow \boxed{y = mx + n} = \boxed{\frac{dy}{dx}\,x + n} \Rightarrow \boxed{n = y - \left(\frac{dy}{dx}\right)x}$

2. $\begin{cases} n = y - \left(\frac{dy}{dx}\right)x \\ f(x,y) = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dy \end{cases}$

$\boxed{f(x,y)} = 2y\,dx + dx - 2x\,dy - 2dy - 2n\,dy$

$= 2y\,dx + dx - 2x\,dy - 2dy - 2\left[y - \left(\frac{dy}{dx}\right)x\right] - dy\,dx$

$= 2y\,dx + dx - 2x\,dy - 2dy - 2y\,dx + 2\,dx\left(\frac{dy}{dx}\right)x$

$= 2y\,dx + dx - 2x\,dy - 2dy - 2y\,dx + 2x\,dy$

$= \boxed{dx - 2dy} \Rightarrow \boxed{f(x,y) = f(0,0) = dx - 2dy}$

Bresenham line drawing algorithm (continued) [Positive slope]

```
void draw_bresenham_line (x0, y0, x1, y1)

    dx = x1 - x0
    dy = y1 - y0
    d = dx - 2dy          // f(x,y) = f(0,0)
    y = y0
    for (x = x0 ; x ≤ x1 ; ++x)
        draw pixel (x, y)
        if (d < 0)
            y = y + 1
            d = d + 2dx - 2dy
        else
            d = d - 2dy
```

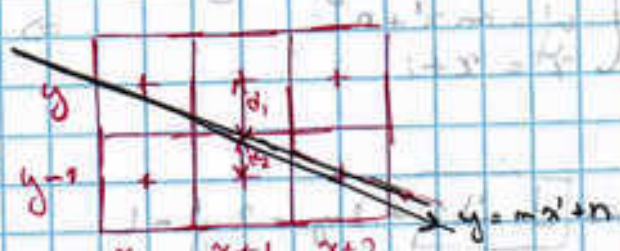
(note: this Algorithm is working octants 1 or octant II only) *

Bresenham line Drawing Algorithm for negative slope

1. We increment x value and decide if y value should be decremented or not

$d_1 < d_2$

$(x+1, y)$

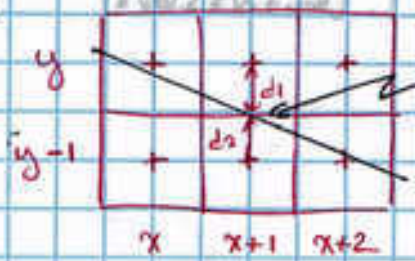


$d_1 > d_2$

$(x+1, y-1)$

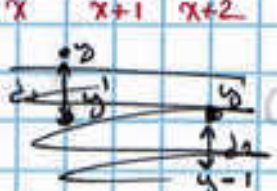
- if $d_1 < d_2$ then the next point to (x,y) should be $(x+1, y)$
- if $d_1 > d_2$ then the next point to (x,y) should be $(x+1, y-1)$

88 2.



$d_1 = y - y'$

$d_2 = y' - (y-1)$



3. $d_1 = y - y'$

$d_2 = y' - (y-1)$

$\boxed{d_1 - d_2} = y - y' - [y' - (y-1)]$

$= y - y' - y' + (y-1)$

$= y - y' - y' + y - 1$

$= \boxed{2y - 2y' - 1}$

4. For the next increment of x, $\underline{x' = x+1}$

$$\begin{cases} d_1 - d_2 = 2y - 2y' - 1 \\ y' = mx' + n \\ x' = x+1 \end{cases} \Rightarrow \boxed{\begin{array}{l} d_1 - d_2 = 2y - 2y' - 1 \\ y' = mx' + n \\ \quad = m(x+1) + n \end{array}}$$

$\Rightarrow \boxed{d_1 - d_2} = 2y - 2y' - 1$

$= 2y - 2[m(x+1) + n] - 1$

$= 2y - 2m(x+1) - 2n - 1$

$= \boxed{2y - 2mx + 2m - 2n - 1}$

$\Rightarrow$

Bresenham line Drawing Algorith for negative slope (continued)

4 (continued)

$$\begin{cases} d_1 - d_2 = 2y - 2mx + 2m - 2n - 1 \\ m = \frac{dy}{dx} \end{cases}$$

$$\boxed{d_1 - d_2} = 2y - 2mx + 2m - 2n - 1$$

$$= \boxed{2y - 2\left(\frac{dy}{dx}\right)x + 2\left(\frac{dy}{dx}\right) - 2n - 1}$$

we can't want divisions so multiply both side by dx

$$dx\left[d_1 - d_2\right] = dx\left\{2y - 2\left(\frac{dy}{dx}\right)x + 2\left(\frac{dy}{dx}\right) - 2n - 1\right\}$$

$$= 2y\,dx - 2x\,dy + 2\,dy - 2n\,dx - dx$$

$$\begin{cases} f(x,y) = dx(d_1 - d_2) \\ dx(d_1 - d_2) = 2y\,dx - 2x\,dy + 2\,dy - 2n\,dx - dx \end{cases}$$

$$\boxed{f(x,y) = 2y\,dx - 2x\,dx + 2dy - 2n\,dx - dx}$$

$\Rightarrow$

90 5. Let $f(x_i, y_i)$ be the last point we draw and let $f(x_{i+1}, y_{i+1})$ be the next point to draw so the iteration for the start point to the next point is $f(x_{i+1}, y_{i+1}) - f(x_i, y_i)$

$$f(x_i, y_i) = 2y_i dx - 2x_i dy + 2dy - 2n dx - dx$$

$$f(x_{i+1}, y_{i+1}) = 2y_{i+1} dx - 2x_{i+1} dy + 2dy - 2n dx - dx$$

$$f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2y_{i+1} dx - 2x_{i+1} dy + 2dy - 2n dx - dx - [2y_i dx - 2x_i dy + 2dy - 2n dx - dx]$$

$$= 2y_{i+1} dx - 2x_{i+1} dy + 2dy - 2n dx - dx - 2y_i dx + 2x_i dy - 2dy + 2n dx + dx$$

$$= 2y_{i+1} dx - 2x_{i+1} dy - 2y_i dx + 2x_i dy$$

$$= \boxed{2dx(y_{i+1} - y_i) - 2dy(x_{i+1} - x_i)}$$

6. For every iteration in x coordinate such that $x_{i+1} = x_i + 1$

$$\begin{cases} x_{i+1} = x_i + 1 \\ f(x_{i+1}, y_{i+1}) - f(x_i, y_i) = 2dx(y_{i+1} - y_i) - 2dy(x_{i+1} - x_i) \end{cases}$$

$$= 2dx(y_{i+1} - y_i) - 2dy(x_{i+1} - x_i)$$

$$= 2dx(y_{i+1} - y_i) - 2dy$$

7. If $d_1 < d_2$ then the next point of (x, y) should be (x_{+1}, y_i) so $y_{i+1} = y_i$

$$\boxed{f(x_{i+1}, y_{i+1}) - f(x_i, y_i)} = 2dx(y_i - y_i) - 2dy$$

$$= \boxed{-2dy}$$

If $d_1 > d_2$ then the next point of (x, y) should be $(x_{+1}, y_i - 1)$ so $y_{i+1} = y_i - 1$

$$\boxed{f(x_{i+1}, y_{i+1}) - f(x_i, y_i)} = 2dx(y_i - 1 - y_i) - 2dy$$

$$= \boxed{-2dx - 2dy}$$

Bresenham line drawing Algorithm for negative slope (continued)

8. Find at initial start point $f(x,y)$

Lets assume $f(x,y)$ start at point $(0,0)$

(A) $f(x,y) = 2y\,dx - dx - 2x\,dy - dy - 2n\,dx - dx$

(B) $y = mx + n$

(C) $m = \frac{dy}{dx}$

$$\begin{cases} y = mx + n \\ m = \frac{dy}{dx} \end{cases} \Rightarrow y = \left(\frac{dy}{dx}\right)x + n \Rightarrow \boxed{n = y - \left(\frac{dy}{dx}\right)x}$$

$$\boxed{f(x,y)} = 2y\,dx - dx - 2x\,dy - dy - 2n\,dx - dx$$

$$= 2y\,dx - dx - 2x\,dy - dy - 2\left[y - \left(\frac{dy}{dx}\right)x\right]dx - dx$$

$$= 2y\,dx - dx - 2x\,dy - dy - 2y\,dx + 2\left(\frac{dy}{dx}\right)x\,dx - dx$$

$$= 2y\,dx - dx - 2x\,dy - dy - 2y\,dx + 2x\,dy - dx$$

$$\boxed{= -2\,dy - dx} \qquad \boxed{-2\,dx - dy}$$

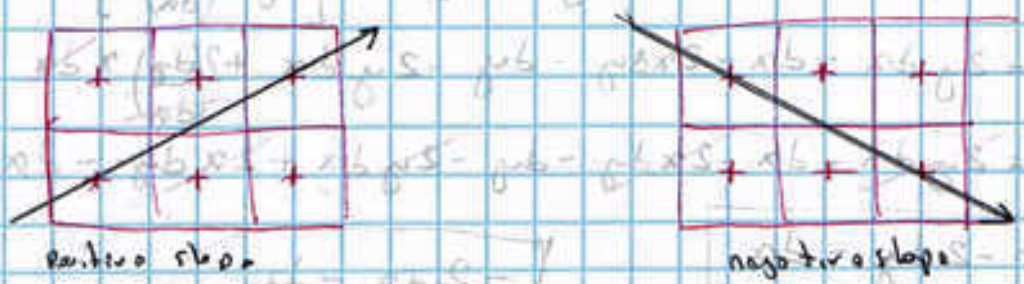
Therefore, for $f(x,y) = -2\,dy - dx$

$f(0,0) = -2\,dy - dx$

draw_bresenham_line_negative (x0, y0, x1, y1)

```
dx = x1 - x0
dy = y1 - y0
d = -dx - 2dy   // f(x,y)
y = y0
for (x = x0 ; x <= x1 ; ++x)
    putpixel (x,y)
    if (d > 0)
        y = y - 1
        d = d - 2dx - 2dy
    else
        d = d - 2dy
```

Bresenham line drawing for positive and negative slope



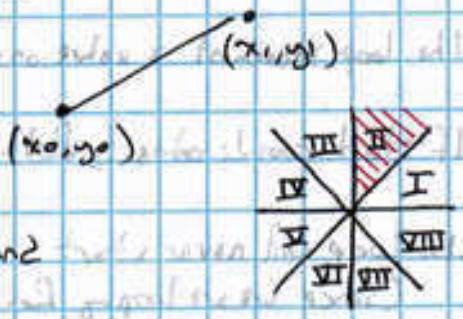
$m = \frac{dy}{dx} = \frac{y_1 - y_0}{x_1 - x_0}$

- if $dy > 0$ and $dx > 0$ or $dy < 0$ and $dx < 0$ slope m is positive
- if $dy < 0$ and $dx > 0$ or $dy > 0$ and $dx < 0$ then slope is negative

```
draw_bresenham_line (x0, y0, x1, y1)
    int dy
    if dy = y1 - y0;
       dx = x1 - x0;
    if((dy > 0 && dx > 0) || (dy < 0 && dx < 0)
        draw_bresenham_line(x0, y0, x1, y1)
    else
        draw_bresenham_line_negative (x0, y0, x1, y1)
```

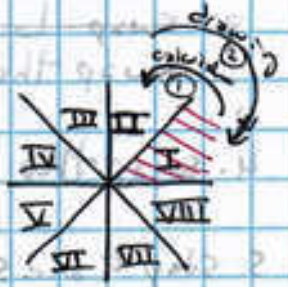
Bresenham line drawing w/ octants

1. start point (x_0, y_0)
   end point (y_0, y_1)



2. Bresenham line algorithm for positive and negative was for octant III

- ## Bresenham line drawing in First Octants



1. line needs to be transformed to the second octant for calculation and transformed back for drawing

2.a. line is in the first octant * $dy > dx$

2.b. Algorithm increment on each iteration and increment the value of y only if needed $(x, y) \Rightarrow (x_{+1}, y_{+1})$
$(x, y) \Rightarrow (x_{+1}, y)$

2c. Algorithm cannot run when $dy > dx$ because loop will end before it get to last y value because # of iteration in x-axis is shorter than

3. to overcome this we need to swap x and y transform the line to 2nd Octant so $dy < dx$ but

4. However, when drawing x and y should be swapped back to represent real line.

```
swapped = 0;
if (abs(y1 - y0) > abs(x1 - x0)
    swap (x0, y0);
    swap (x1, y1);
    swapped = 1;
```

} calculation

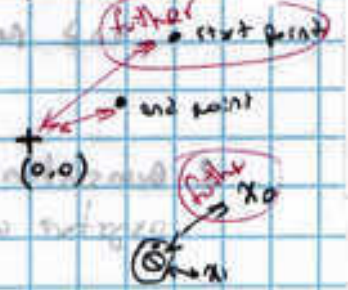
```
if (!swapped) putPixel (x,y)
else          putPixel (y,x)
```

} drawing

- Bresenham line drawing when start coordinates is futhr than end coordinate

1. The loop increment x value on each iteration

2. If start coordinates is futher than the end coordinate



2b. loop will never start
(since we are looping for start point to end point)

3. ~~swap loop condition~~
swap the x values & keep incrementing the x value

4. swap the y values ~~to match~~ as well

5. Step 4 and 2 5 obtain the same line w/
the starting point closer than the end point

6. All lines can be treated as lines in the
1st octant or 2nd Octant after making sure
x coordinate of the starting point is
closer to 0 (zero) than the x coordinate
of the ~~end~~ point ending point

- Bresenham line drawing for Vertical and Horizontal lines

```
if (x0 == x1)
    if y0 > y1
        swap (y0, y1)
    for (y = y0; y <= y1; y++)
        put pixel (x0, y);   return;
if (y0 == y1)
    if x0 > x1
        swap (x0, x1)
    for (x = x0 ; x <= x1 ; ++x)
        put pixel (x, y0)
return;
```

- Optimizing Bresenham line Drawing

1. Notice that in the main loop 2dx and 2dy is used a lot

```
for (x = x0; x <= x1; ++x)
    put pixel (x,y)
    if (d < 0)
        y = y + 1;
        d = d + 2dx - 2dy;
    else
        d = d - 2dy
```

```
twoDx = dx << 1   // shift left
twoDy = dy << 1

for (x=x0; x < x1; ++x)
    put pixel (x,y)
    if (d < 0)
        y++;
        d = d + twoDx - twoDy;
    else
        d = d - twoDy
```

~~for (x=x0; x <= x1; x++)~~
~~put pixel (x,y)~~

```
twoDx = dx << 1
twDy = dy << 1
for (x = x0; x <= x1; x++)
    put pixel (x,y)
    d -= twoDy
    if (d < 0)
        y++;
        d += twoDx
```

2. We can improve the tests of negative slope
   A. since we are dealing w/ signed integers we can use the most significant bit (MSB) to know if the value is positive or negative
   B. ~~if MSB is negative~~ if MSB is set (1) then negative else positive
   C. XORing dx with dy would set the MSB if dx^dy is negative
   D. ANDing it with 0x80000000 will zero all bits except the MSB

A	O	Z
0	0	0
0	1	1
1	0	1
1	1	0

isNegativeSlope = (dy>0 && dx>0)||(dy<0 && dx<0)

isNegativeSlope = (dx ^ dy) & 0x80000000