Scan Line

Instructor: Dinesh Maharjan

2022

Graphics Pipeline

- In computer graphics we render 2D images or scenes.
- This involves a set of stage which constitute the graphics pipeline.
- It includes 5 stages.
- Object representation
- Modeling Transformation
- Lighting or coloring
- Viewing pipeline
- Scan conversion or rendering

Object representation

- It defines the object that will be part of scene.
- It involves specifying the vertices or edges with respect to some reference frame.
- Objects are defined in its own or local coordinate.
- Objects can be cubes, spheres, cylinders etc.
- Objects shape, size and position is not important at the point of its representation.

LINE DRAWING

Description: Given the specification for a straight line, find the collection of addressable pixels which most closely approximates this line.

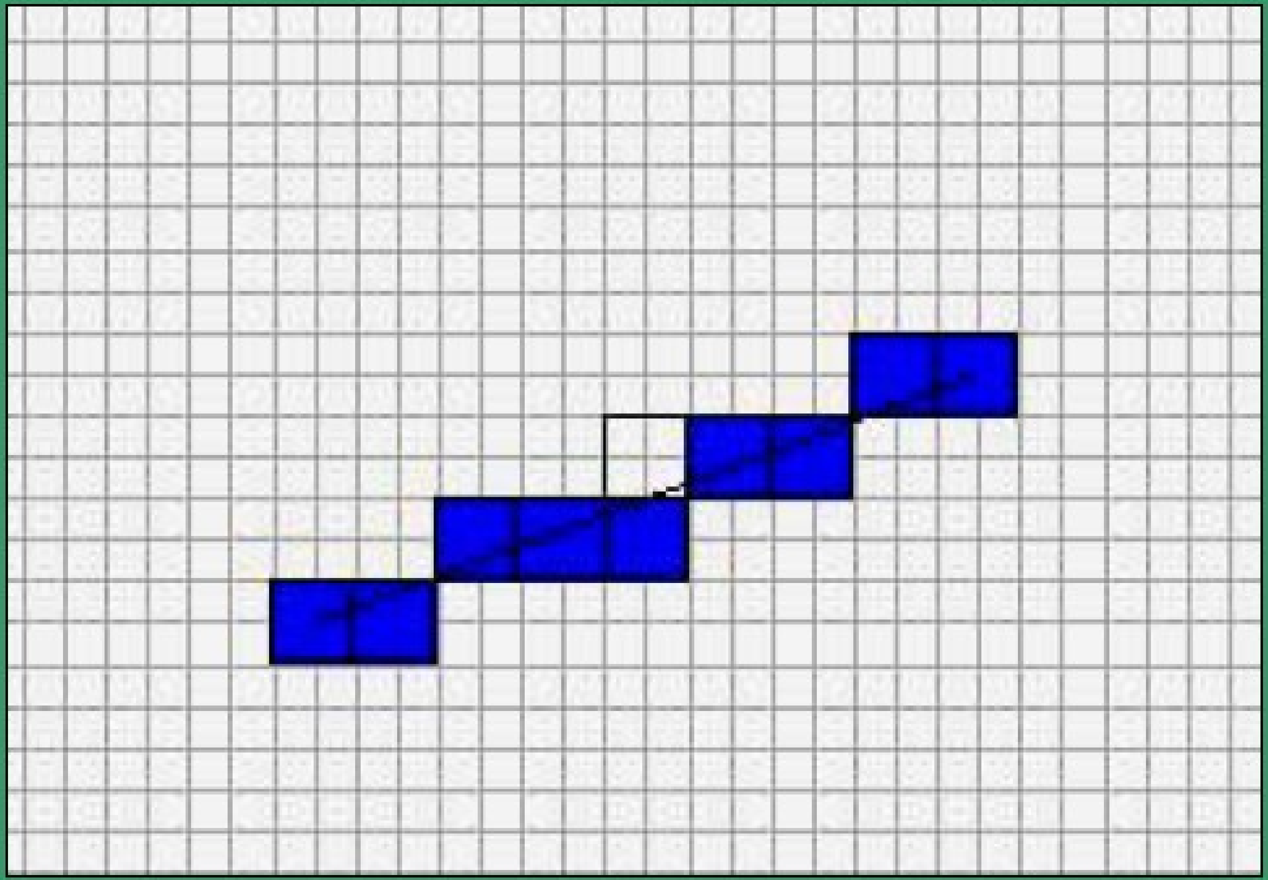
Goals (not all of them are achievable with the discrete space of a raster device):

- Straight lines should appear straight.
- Lines should start and end accurately, matching endpoints with connecting lines.
- Lines should have constant brightness.
- Lines should be drawn as rapidly as possible.

Problems:

- How do we determine which pixels to illuminate to satisfy the above goals?
- Vertical, horizontal, and lines with slope = +/- 1, are easy to draw.
- Others create problems: stair-casing/ jaggies/aliasing.
- Quality of the line drawn depends on the location of the pixels and their brightness

It is difficult to determine whether a pixel belongs to an object



Direct Solution:

Solve $y=mx+b$, where $(0,b)$ is the y-intercept and m is the slope.

Go from x_0 to x_1:

calculate round(y) from the equation.

Take an example, $b = 1$ (starting point $(0,1)$) and $m = 3/5$.

Then $x = 1, y = 2 = round(8/5)$
$x = 2, y = 2 = round(11/5)$
$x = 3, y = 3 = round(14/5)$
$x = 4, y = 3 = round(17/5)$
$x = 5, y = 4 = round(20/5)$

For results, see next slide.

About Slope

- Screen has only positive x axis and positive y axis.
- If abs(dx)>abs(dy), increment x by 1 and workout for y.
- Otherwise increment y by 1 and workout for x

Direct Method

- (x1,y1)=(2,2) and (x2,y2)=(7,5)
- dx=x2-x1=7-2=5
- dy=y2-y1=5-3=2
- m=dy/dx=2/5=0.4
- c=y-mx
- =2-0.4*2
- =1.2
- x=x1=2
- y=y1=2

i	x	y=mx+c	Round(y)	output
0	2	2	2	(2,2)
1	3	2.6	3	(3,3)
2	4	3.2	3	(4,3)
3	5	3.8	4	(5,4)
4	6	4.4	4	(6,4)
5	7	5	5	(7,5)

DDA

- Stands for Digital Differential Analyzer.
- An increment approach to speed up line scan conversion.
- It increments either x or y by 1.
- Then it increment or decrement another co-ordinate by slope or reciprocal of slope.
- The increment or decrement is determined by the direction of change in x or y.

DDA

- j

$$\text{if } |m| <= 1, x_{k+1} = x_k + 1 \text{ and } y_{k+1} = y_k + m$$
$$\text{else, } y_{k+1} = y_k + 1 \text{ and } x_{k+1} = x_k + 1/m$$

DDA Algorithm

- Let initial point (x1,y1) and final (x2, y2)
- dx= x2-x1
- dy=y2-y1
- if abs(dx)>abs(dy), step=abs(dx)
- Else step=abs(dy)
- xinc=dx/step
- yinc=dy/step

DDA Algorithm

- i=0
- While i<=step
- Drawpixel(round(x1),round(x2))
- X1=x1+xinc
- Y1=y1+yinc

DDA

- (x1,y1)=(2,2) and (x2,y2)=(7,5)
- dx=x2-x1=7-2=5
- dy=y2-y1=5-2=3
- Since abs(dx)>abs(dy),
- step=abs(dx)=5
- xinc=dx/step=5/5=1
- yinc=dy/step=3/5=0.6
- x=x1=2
- y=y1=2

i	x	y	output
0	2	2	(2,2)
1	3	2.6	(3,3)
2	4	3.2	(4,3)
3	5	3.8	(5,4)
4	6	4.4	(6,4)
5	7	5	(7,5)

DDA

- (x1,y1)=(7,5) and (x2,y2)=(2,2)
- dx=x2-x1=2-7=-5
- dy=y2-y1=2-5=-3
- Since abs(dx)>abs(dy),
- step=abs(dx)=5
- xinc=dx/step=-5/5=-1
- yinc=dy/step=-3/5=-0.6
- x=x1=7
- y=y1=5

i	x	y	output
0	7	5	(7,5)
1	6	4.4	(6,4)
2	5	3.8	(5,4)
3	4	3.2	(4,2)
4	3	2.6	(3,3)
5	2	2.0	(2,2)

DDA

- (x1,y1)=(2,3) and (x2,y2)=(1,9)
- dx=x2-x1=1-2=-1
- dy=y2-y1=9-3=6
- Since abs(dy)>abs(dx),
- step=abs(dy)=6
- xinc=dx/step=-1/6=-0.167
- yinc=dy/step=6/6=1
- x=x1=2
- y=y1=3

i	x	y	output
0	2.00	3	2,3
1	1.83	4	2,4
2	1.67	5	2,5
3	1.50	6	2,6
4	1.33	7	1,7
5	1.17	8	1,8
6	1.00	9	1.9

DDA

- Eliminates floating point multiplication m.x
- Any point can be initial point.
- Still has to perform float operations and round operations
- Line is not smooth

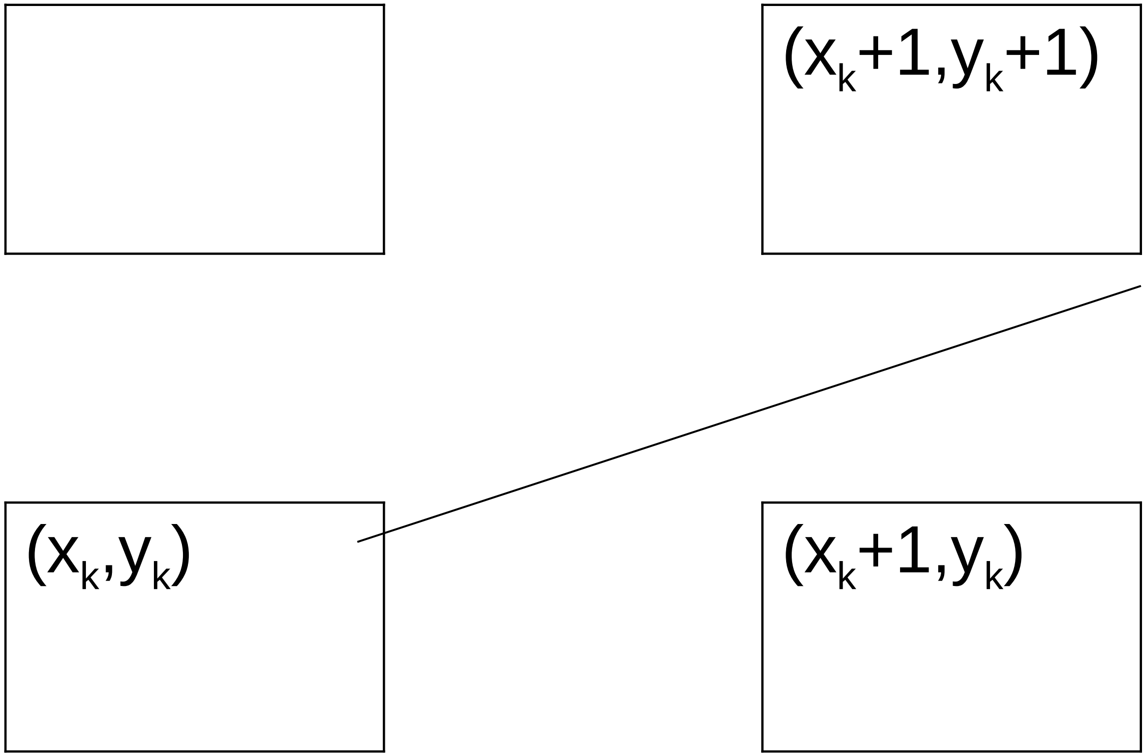
Bresenham

- It starts from the left end pixel
- It samples closest pixel to the line for each increment in x coordinate.
- If (x_1,y_1) is starting point and (x_2,y_2)
- It uses only integers to scan a line.

Bresenham

- Let current point be (x_k,y_k) and final point be (x_n,y_n)
- Let the equation of line be $y=mx+b$
- Since bresenham always increase x by 1,
- Next co-ordinate should be (x_k+1,y) where y is can be either y_k or y_k+1
- Now $y=m(x_k+1)+b$

Bresenham



Bresenham

- Distance between (x_k+1,y_k+1) and (x_k+1,y) is
- d2=y_k+1-y (upper distance)
- Or d2=y_k+1 - m(x_k+1) - b
- Distance between (x_k+1,y_k) and (x_k+1,y) is
- d1=y-y_k (lower distance)
- Or, d1=m(x_k+1)+b-y_k
- Let us evaluate d1-d2
- d1-d2= m(x_k+1)+b-y_k - y_k -1+m(x_k+1)+b
- =2m(x_k+1)+2b -2y_k - 1
- dx(d1-d2)=2dy(x_k+1)+2dx.b – 2dx.y_k - dx

Bresenham

- $dx(d1-d2)= 2dy(x_k+1)+2dx.b – 2dx.y_k$ - dx
- $=2dy.x_k+2dy+2dx.b – 2dx.y_k$ - dx
- Here, 2dy+2dx.b-dx is constant and does not contribute in decision making.
- So can be discarded
- So, $dx.(d1-d2)= 2dy.x_k – 2dx.y_k$
- Let $p_k=dx.(d1-d2) = 2dy.x_k – 2dx.y_k$
- Be current decision parameter and Next is
- $p_{k+1}=2dy.x_{k+1} –2 dx.y_{k+1}$

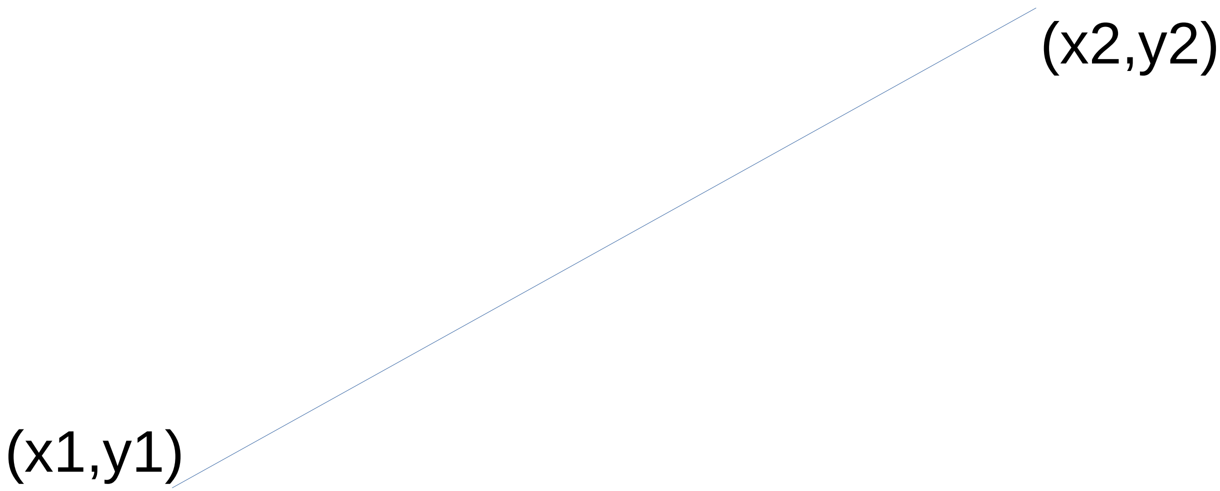
Bresenham

- If we see carefully dx cannot be negative since x2>x1.
- So, if $p_k<0$, then (x_k+1, y_k) is closer to the line, otherwise (x_k+1, y_k+1) is closer.
- If $p_k<0$, then $y_{k+1}=y_k$ and $x_{k+1} = x_k+1$
- So, $p_{k+1}=2dy.(x_k+1) - 2dx.y_k$
- Or $p_{k+1}= 2dy.x_k+ 2dy - 2dx.y_k$
- Or, $p_{k+1}= p_k +2dy$

Bresenham

- For initial decision parameter,
- $p_1 = 2dy - dx$
- This is midpoint theorem

Bresenham



- dx>dy
- Loop will be based on dx.
- x2>x1 and y2>y1
- So, x1 and y1 will increase to x2 and y2.
- Because, Bresenham always starts from (x1,y1)

Bresenham Pseudocode

- Let intial point be (x1,y1) and final point be (x2,y2)
- dx=abs(x2-x1)
- dy=abs(y2-y1)
- p=2.dy-dx
- x=x1
- y=y1
- i=0
- while(i<=dx)
- Drawpoint(x,y)

Bresenham Pseudocode

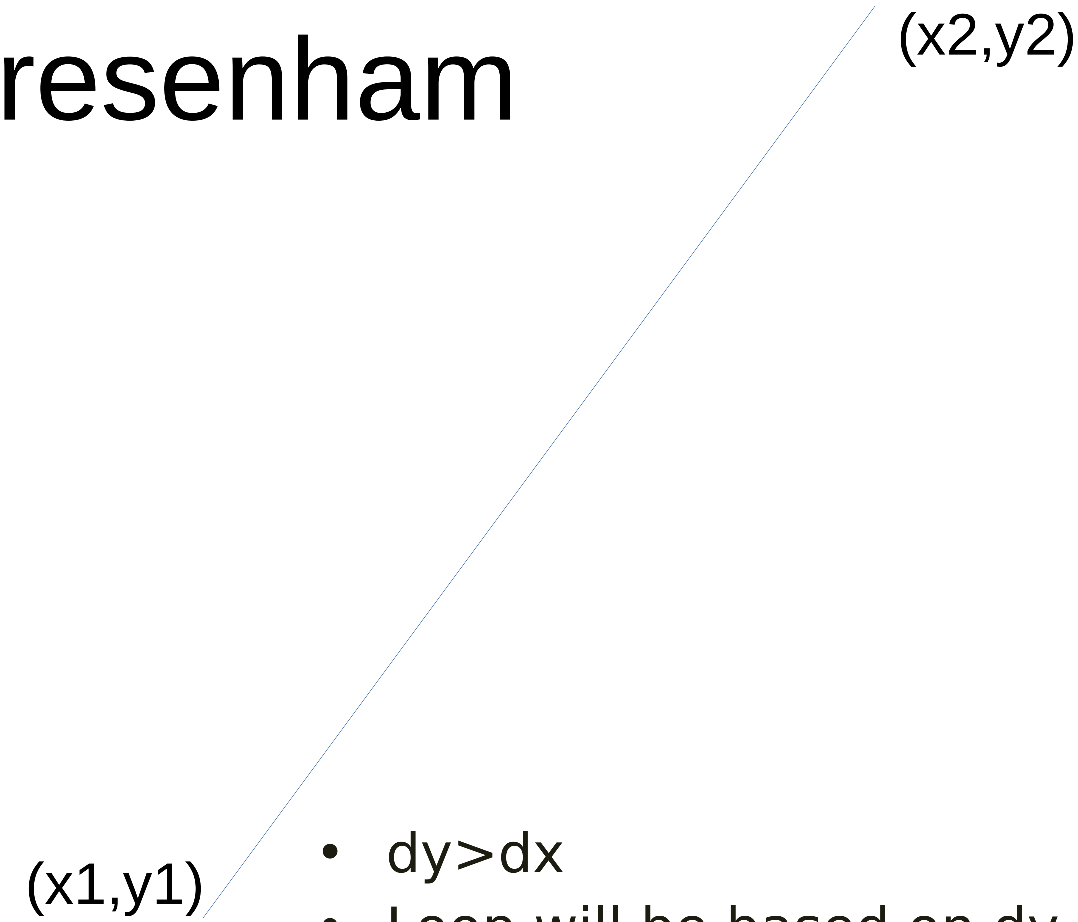
- X=x+1
- If p<0, p=p+2.dy
- Else y=y+1, p=p+2.dy-2dx

Bresenham Example

- (x1,y1)=(2,2) and (x2,y2)=(7,5)
- dx=abs(x2-x1)=abs(7-2)=5
- dy=abs(y2-y1)=abs(5-2)=3
- p=2.dy-dx=1
- x=x1=2
- y=y1=2

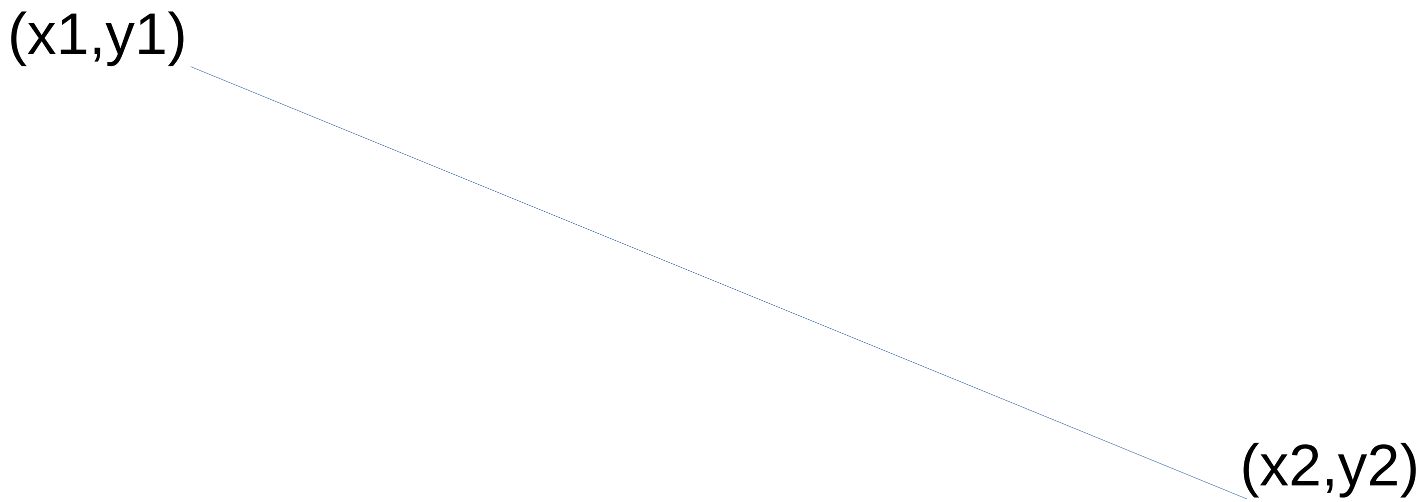
i	x	y	p	output
0	2	2	1	(2,2)
1	3	3	-3	(3,3)
2	4	3	3	(4,3)
3	5	4	-1	(5,4)
4	6	4	5	(6,4)
5	7	5	1	(7,5)

Bresenham



- dy>dx
- Loop will be based on dy.
- x2>x1 and y2>y1
- So, x1 and y1 will increase to x2 and y2.
- Because, Bresenham always starts from (x1,y1)

Bresenham



- dx>dy
- Loop will be based on dx.
- x2>x1 and y2<y1
- So, x1 will increase and y1 will decrease
- Because, bresenham always starts from (x1,y1) to (x2,y2)

Final Bresenham Pseudocode

- Let intial point be (x1,y1) and final point be (x2,y2)
- xinc=1 and yinc=1
- If x1>x2, then xinc=-1
- If y1>y2, then yinc=-1
- dx=abs(x2-x1)
- dy=abs(y2-y1)
- i=0

Final Bresenham Pseudocode

- if(dx>dy)
  - p=2.dy-dx
  - while(i<=dx)
    - draw(x,y)
    - x=x+xinc
    - If p<0, p=p+2dy
    - else, y=y+yinc, p=p+2dy-2dx

Final Bresenham Pseudocode

- if(dy>dx)
  - p=2.dx-dy
  - while(i<=dy)
    - draw(x,y)
    - y=y+yinc
    - If p<0, p=p+2dx
    - else, x=x+xinc, p=p+2dx-2dy

Bresenham

- All the computations are integer
- All floating points are removed
- No round function.
- Huge improvement in terms of speed of computation which is desirable
- Because screen is to be refreshed at very high speed to avoid flickr.

Drawing Circle

- Let as consider a circle with radius r and center at (0,0)
- The equation is $x^2+y^2=r^2$
- $F(x,y)=x^2+y^2-r^2$
- We can draw circle by computing y for each incremented value of x.
- But, this is very expensive.
- It has to perform square root operations.
- Moreover, the gap between computed pixels is not uniform.

Mid Point Theorem

- Similar to bresham algorithm, mid point theorem samples closest y-coordinate for each incremented x.
- For this it uses mid point of the next possible points.
- Since circle is has 8 symmetric points, it only samples points on first octant and plots remaining octants of the circle by the method of symmetry.

Drawing Circle

CIRCLE DRAWING

Only considers circles centered at the origin with integer radii.

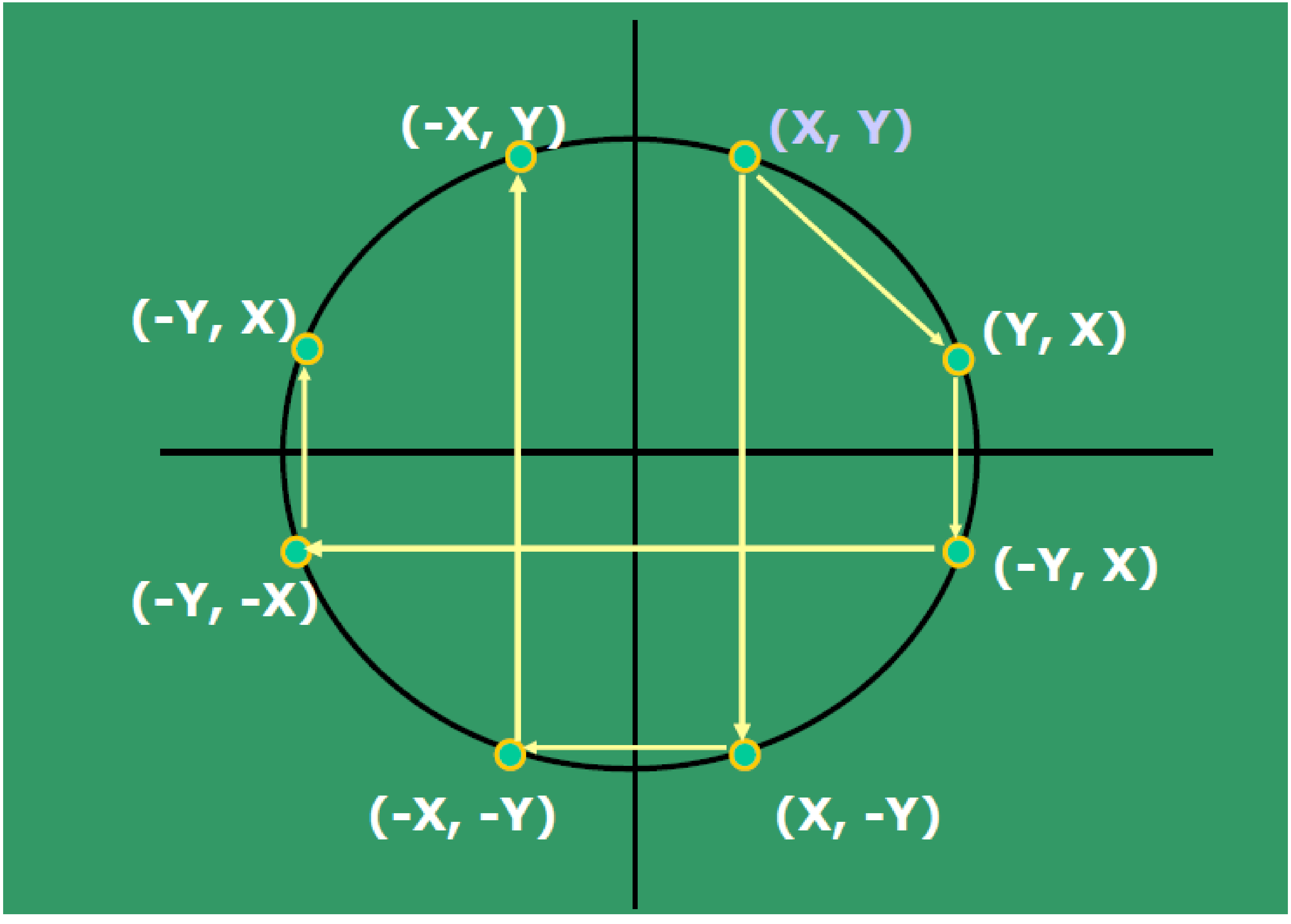
Can apply translations to get non-origin centered circles.

Explicit equation: $y = +/- \mathrm{sqrt}(R^2 - x^2)$

Implicit equation: $F(x,y)= x^2 + y^2 - R^2 =0$

Note: Implicit equations used extensively for advanced modeling

Drawing Circle



Drawing Circle

Use of Symmetry: Only need to calculate one octant. One can get points in the other 7 octants as follows:

```
Draw_circle(x, y)

begin
    Plotpoint (x, y);  Plotpoint (y, x);

    Plotpoint (x, -y); Plotpoint (-y, x);

    Plotpoint (-x, -y) ; Plotpoint (-y, -x);

    Plotpoint (-x, y); Plotpoint (-y, x);
end
```

MIDPOINT CIRCLE ALGORITHM

Will calculate points for the second octant.

Use *draw_circle* procedure to calculate the rest.

Now the choice is between pixels E and SE.

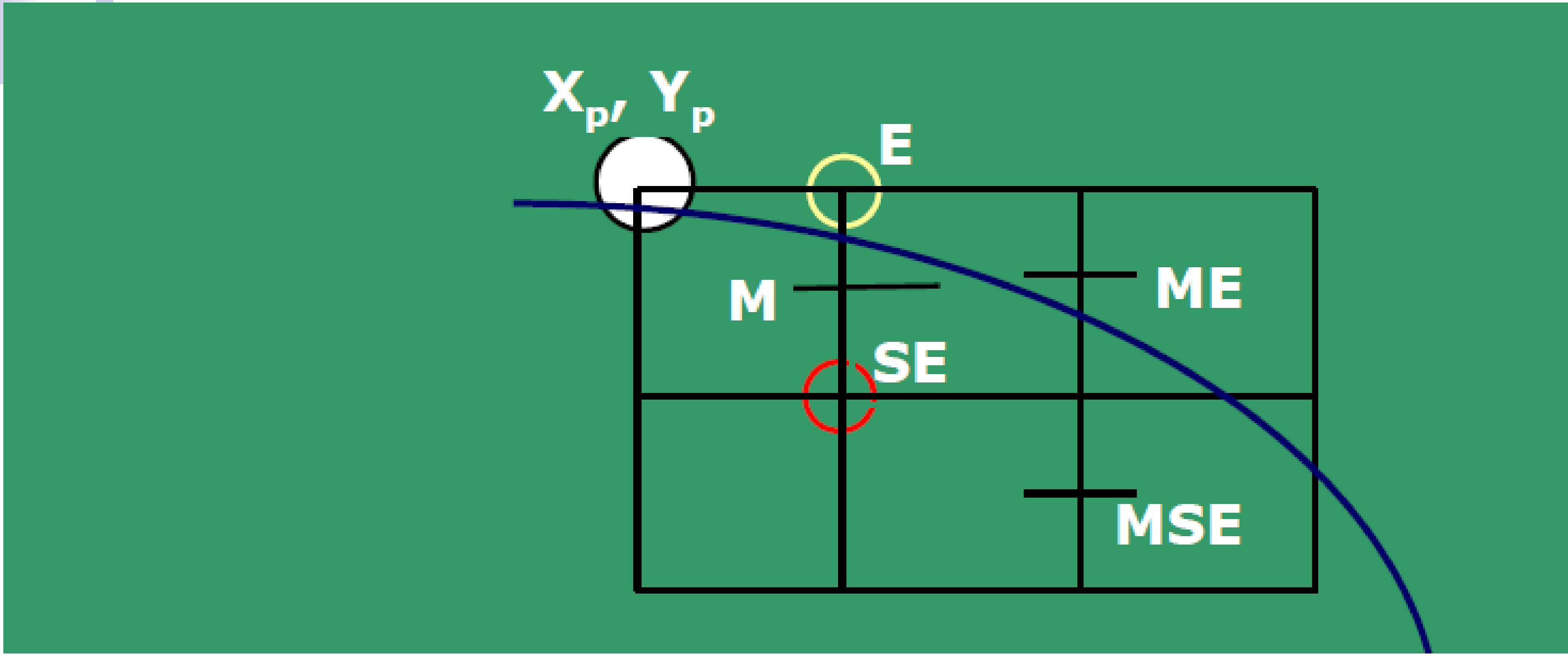
$F(x, y) = x^2 + y^2 - R^2 = 0$

$F(x, y) > 0$ if point is outside the circle

$F(x, y) < 0$ if point inside the circle.

Mid Point Theorem

- Let as consider a circle with radius r and center at (0,0)
- Let current point on circle be (x_k, y_k)
- Two possible next points are $E(x_k+1, y_k)$ and $SE(x_k+1, y_k-1)$
- Mid point of these two points are $M(x_k+1,\ y_k-0.5)$



X_p, Y_p
E
M
ME
SE
MSE

Mid Point Theorem

- If $F(x_p+1,y_p-0.5)>0$, M lies outside of circle.
- It means SE is closer to circle and we chose SE.
- If $F(x_p+1,y_p-0.5)<0$, M lies inside circle.
- It means E is closer to circle and we chose E.

Mid Point Theorem

- $F(x_p+1, y_p-1/2)=(x_p+1)^2+(y_p-1/2)^2 - r^2$
- Let decision parameter be
- $d_p= (x_p+1)^2+(y_p-1/2)^2 – r^2$
- $=x^2{}_p+2.x_p+1+y^2{}_p - y_p+1/4 - r^2$
- $d_{p+1}=x^2{}_{p+1}+2x_{p+1}+1+y^2{}_{p+1}-y_{p+1}+1/4 - r^2$

Mid Point Theorem

- If $d_p >= 0$, then
- $y_{p+1} = y_p - 1$ and $x_{p+1} = x_p + 1$
- $d_{p+1} = (x_p+1)^2 + 2(x_p+1) + 1 + (y_p-1)^2 - (y_p-1) + 1/4 - r^2$
- $= (x_p+1)^2 + 2(x_p+1) + 1 + y^2 p - 2y_p + 1 - y_p + 1 + 1/4 - r^2$
- $= (x_p+1)^2 + y^2_p - y_p + 1/4 - r^2 + 2(x_p+1) + 1 - 2y_p + 1 + 1$
- $= (x_p+1)^2 + (y_p - 1/2)^2 - r^2 + 2(x_p+1) + 1 - 2y_p + 1 + 1$
- $= d_p + 2(x_p+1) + 1 - 2y_p + 1 + 1$
- $= d_p + 2x_p - 2y_p + 5$

Mid Point Theorem

- If dp<0, then
- $y_{p+1}=y_p$ and $x_{p+1}=x_p+1$
- $d_{p+1}= (x_p+1)^2+2(x_p+1)+1+y^2{}_p -y_p+1/4 - r^2$
- $= (x_p+1)^2 +2(x_p+1)+1+(y_p -1/2)^2 - r^2$
- $= (x_p+1)^2 +(y_p -1/2)^2-r^2+2(x_p+1)+1$
- $=d_p+2(x_p+1)+1$
- $=d_p+2x_p+3$

Mid Point Theorem

- If starting point is (0,r), then
- $x_{0=}=0$ and $y_0=r$
- $d_0= (x_0+1)^2+(y_0-1/2)^2 – r^2$
- $= (0+1)^2 +(r-1/2)^2 – r^2$
- $= 1+r^2 – r + ¼ - r^2$
- $= 5/4 - r$

Mid Point Algorithm

- Let us draw a circle with center (xc, yc) and radius r.
- Let (0,r) be starting point.
- X=0, y=r
- p= 1– r
- Putpixels four symmetric axis points.
- While x<=y
- drawpixel(xc,yc,x,y)
- X=x+1
- If p<0, p=p + 2x +3
- Else, p = p + 2(x – y) +5 and y= y- 1

Mid Point Algorithm

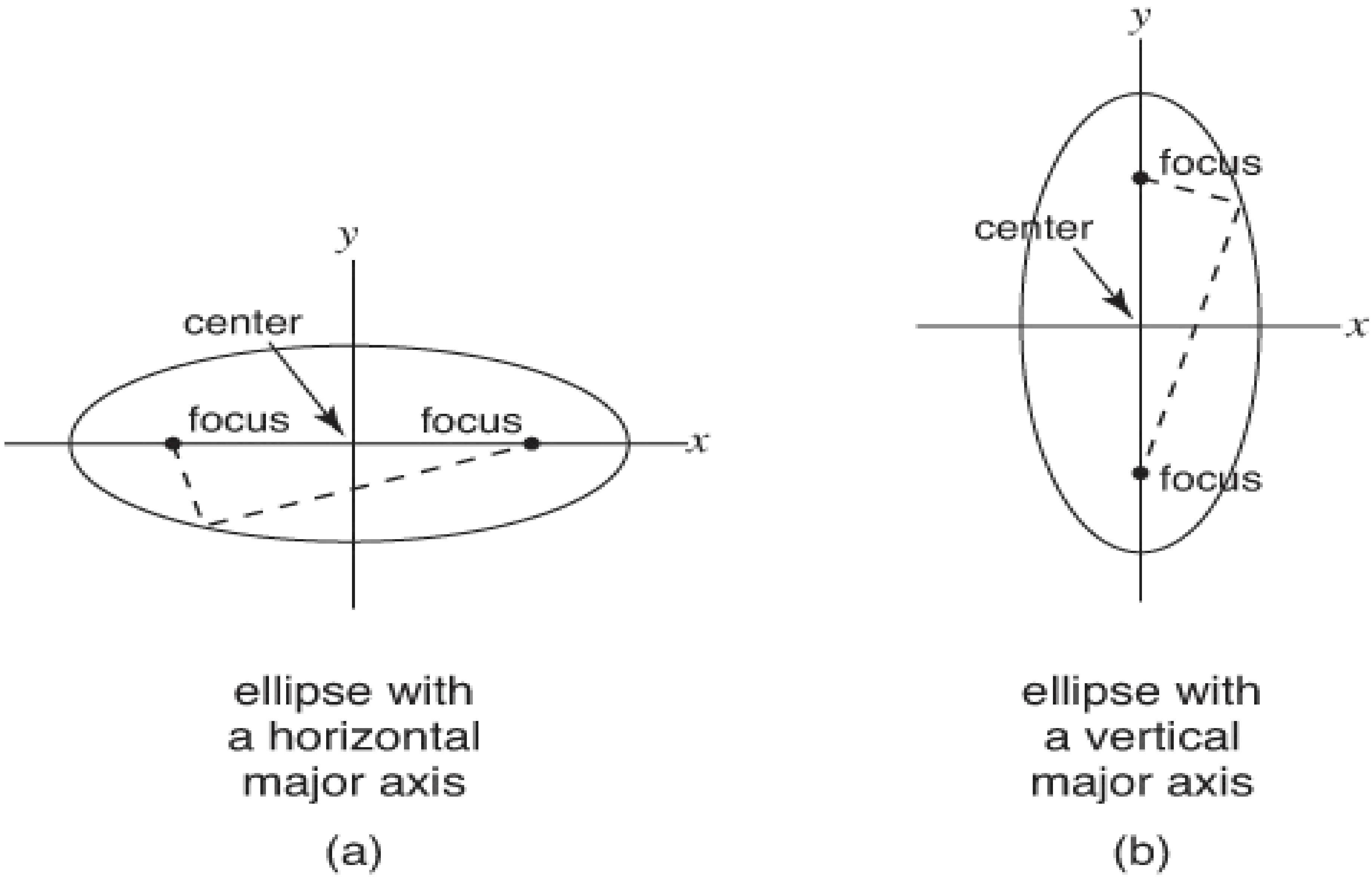
- drawpixel(xc,yc,x,y)
- putpixel(x+xc,y+yc)
- putpixel(x+xc,-y+yc)
- putpixel(-x+xc,y+yc)
- putpixel(-x+xc,-y+yc)
- putpixel(y+xc,x+yc)
- putpixel(y+xc,-x+yc)
- putpixel(-y+xc,x+yc)
- putpixel(-y+xc,-x+yc)

Example

- $xc=0$, $yc=0$
- $r=10$
- $x=0$, $y=r=10$
- $p=1-10=-9$

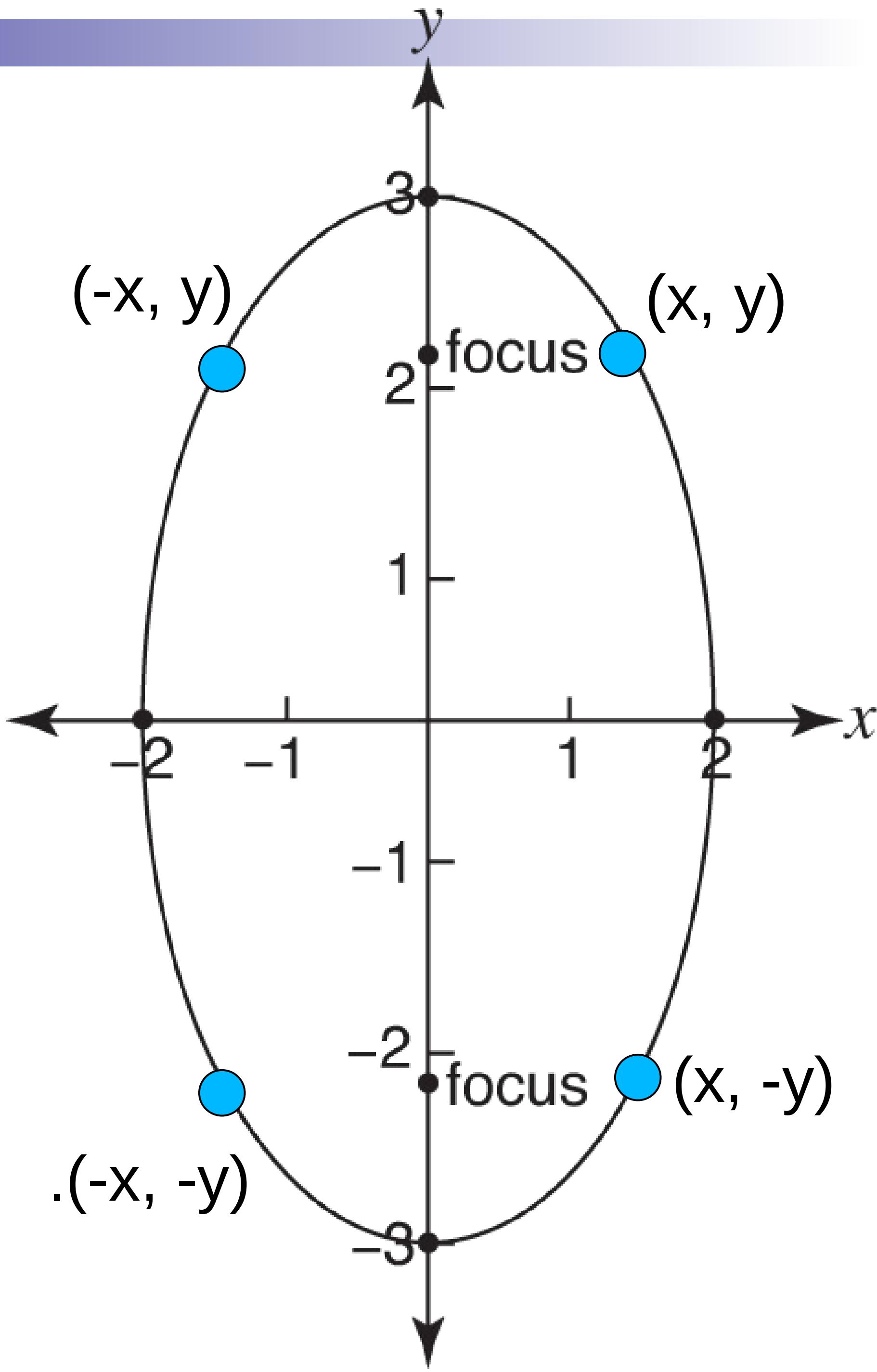
x	y	p	Output
0	10	-9	(0,10)
1	10	-4	(1,10)
2	10	3	(2,10)
3	9	-6	(3,9)
4	9	5	(4,9)
5	8	2	(5,8)
6	7	3	(6,7)
7	6	8	(7,6)
8	5	17	(8,5)
9	4	30	(9,4)
10	3	47	(10,3)

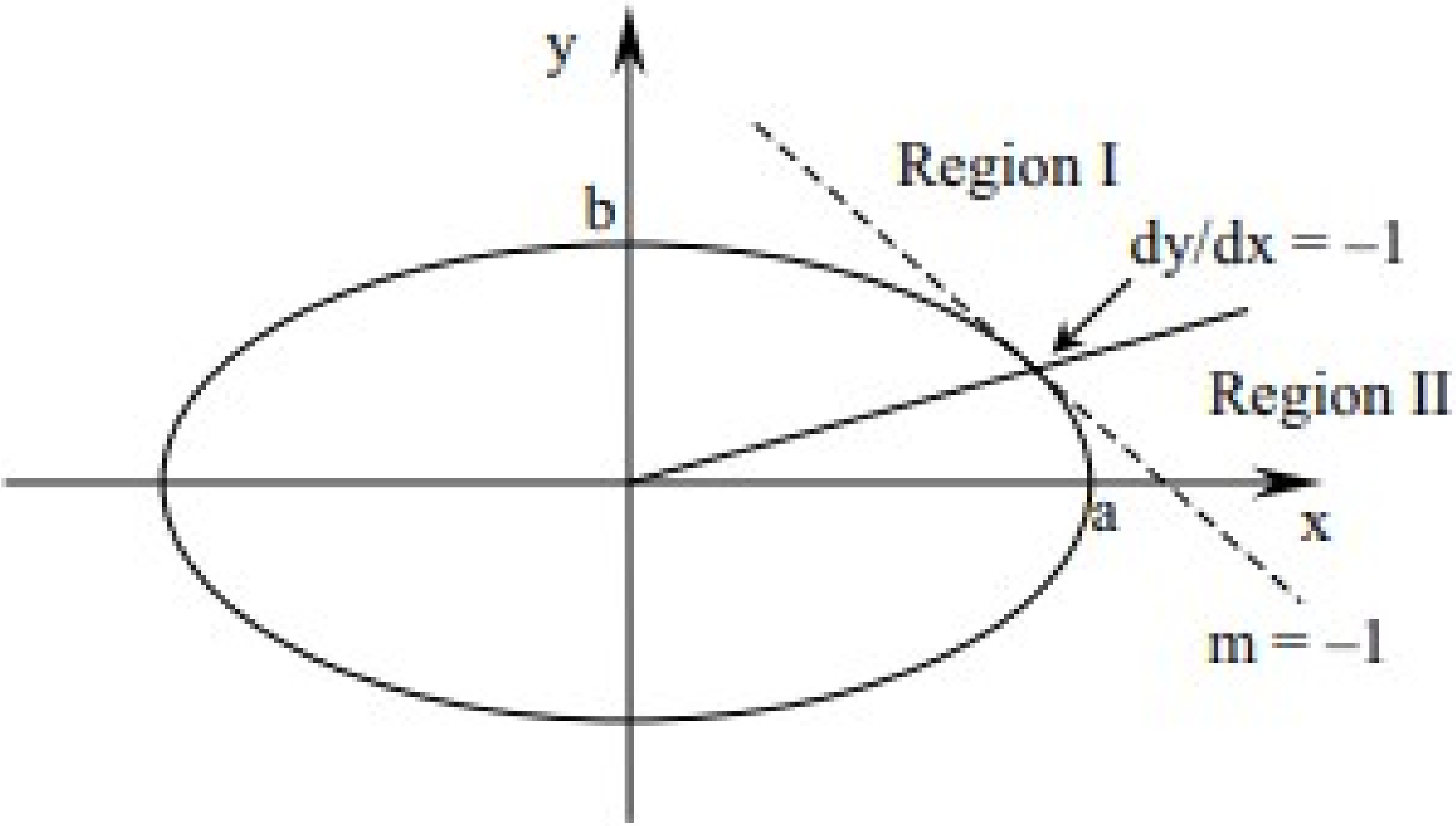
Elipse Algorithm



Elipse Algorithm

- Ellipse is elongated circle.
- Sum of the distances from foci to any point on ellipse is always constant





- Slope at point separating two octants is -1

Center of ellipse $(0, 0)$

Primary axis, a

Secondary axis, b

The equation is

$b^2x^2 + a^2y^2 - a^2b^2 = 0$

Let $f(x, y) = b^2x^2 + a^2y^2 - a^2b^2$

if $f(x, y) > 0, then(x, y)$ lies outside of ellipse.

if $f(x, y) = 0, then(x, y)$ lies on the ellipse.

if $f(x, y) < 0, then(x, y)$ lies inside the ellipse.

The slope at the point that separates two regions is

$$\frac{\partial f(x,y)}{\partial y} = \frac{\partial (b^2x^2+a^2y^2-a^2b^2)}{\partial x} = 0$$

$$b^2 2x + a^2 2y \frac{\partial y}{\partial x} = 0$$

$$\frac{\partial y}{\partial x} = \frac{-b^2 2x}{a^2 2y}$$

if (x, y) is the point that separates two regions, then slope is -1.

$$\frac{\partial y}{\partial x} = \frac{-b^2 2x}{a^2 2y} = -1$$

$$2xb^2 = 2ya^2$$

let current point (x_k, y_k) be on the first region.
Now next possible points are $E(x_k + 1, y_k)$ and $SE(x_k + 1, y_k - 1)$.
The midpoint of these two $M(x_k + 1, y_k - 1/2)$.
let $p_k^1 = f(x_k + 1, y_k - 1/2) = b^2(x_k + 1)^2 + a^2(y_k - 1/2)^2 - a^2b^2$
be decision parameter for the first region.
Let $p_{k+1}^1 = b^2(x_{k+1} + 1)^2 + a^2(y_{k+1} - 1/2)^2 - a^2b^2$.
be next decision parameter.

if $p_k^1 > 0, then M$ lies outside, so SE is closer to ellipse.
So, $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$

$p_{k+1}^1 = b^2(x_{k+1} + 1)^2 + a^2(y_{k+1} - 1/2)^2 - a^2b^2$

$p_{k+1}^1 = b^2(x_k + 1 + 1)^2 + a^2(y_k - 1 - 1/2)^2 - a^2b^2$

$= b^2((x_k + 1)^2 + 2(x_k + 1) + 1) + a^2((y_k - 1/2)^2 - 2(y_k - 1/2) + 1) - a^2b^2$

$= b^2(x_k + 1)^2 + a^2(y_k - 1/2)^2 - a^2b^2 + 2b^2(x_k + 1) + b^2 - 2a^2(y_k - 1/2) + a^2$

$p_{k+1}^1 = p_k^1 + 2b^2(x_k + 1) + b^2 - 2a^2(y_k - 1/2) + a^2$

$p_{k+1}^1 = p_k^1 + 2b^2(x_k + 1) + b^2 - 2a^2y_k + a^2 + a^2$

$p_{k+1}^1 = p_k^1 + 2b^2(x_k + 1) + b^2 - 2a^2(y_k - 1)$

$p_{k+1}^1 = p_k^1 + 2b^2x_{k+1} + b^2 - 2a^2y_{k+1}$

if $p_k^1 < 0, then M$ lies inside, so S is closer to ellipse.
So, $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k$

$$p_{k+1}^1 = b^2(x_{k+1} + 1)^2 + a^2(y_{k+1} - 1/2)^2 - a^2b^2$$
$$p_{k+1}^1 = b^2(x_k + 1 + 1)^2 + a^2(y_k - 1/2)^2 - a^2b^2$$
$$p_{k+1}^1 = b^2((x_k + 1)^2 + 2(x_k + 1) + 1) + a^2(y_k - 1/2)^2 - a^2b^2$$
$$p_{k+1}^1 = b^2(x_k + 1)^2 + 2b^2(x_k + 1) + b^2 + a^2(y_k - 1/2)^2 - a^2b^2$$
$$p_{k+1}^1 = p_k^1 + 2b^2(x_k + 1) + b^2$$
$$p_{k+1}^1 = p_k^1 + 2b^2x_{k+1} + b^2$$

For initial decision parameter, we use initial point $(0, b)$.
$p_0^1 = b^2(x_0 + 1)^2 + a^2(y_0 - 1/2)^2 - a^2b^2$
$p_0^1 = b^2(0 + 1)^2 + a^2(b - 1/2)^2 - a^2b^2$
$p_0^1 = b^2 + a^2b^2 - a^2b + a^2/4 - a^2b^2$
$p_0^1 = b^2 - a^2b + a^2/4$

For second region, let current point be (x_k, y_k)
the possible next points are $(x_k, y_k - 1)$ and $(x_k + 1, y_k - 1)$
The mid point is $(x_k + 1/2, y_k - 1)$
The decision parameter is
$p_k^2 = b^2(x_k + 1/2)^2 + a^2(y_k - 1)^2 - a^2b^2$
The next decision parameter is
$p_{k+1}^2 = b^2(x_{k+1} + 1/2)^2 + a^2(y_{k+1} - 1)^2 - a^2b^2$

for initial decision parameter for second region,
We need to use the last point of the first region on,

$$p_0^2 = b^2(x_0 + 1/2)^2 + a^2(y_0 - 1)^2 - a^2b^2$$

if $p_k^2 > 0$, then $x_{k+1} = x_k$ and $y_{k+1} = y_k - 1$.

$$p_{k+1}^2 = b^2(x_{k+1} + 1/2)^2 + a^2(y_{k+1} - 1)^2 - a^2b^2$$
$$p_{k+1}^2 = b^2(x_k + 1/2)^2 + a^2(y_k - 1 - 1)^2 - a^2b^2$$
$$p_{k+1}^2 = b^2(x_k + 1/2)^2 + a^2((y_k - 1)^2 - 2(y_k - 1) + 1) - a^2b^2$$
$$p_{k+1}^2 = b^2(x_k + 1/2)^2 + a^2(y_k - 1)^2 - a^2b^2 - 2a^2(y_k - 1) + a^2$$
$$p_{k+1}^2 = p_k^2 - 2a^2y_{k+1} + a^2$$

if $p_k^2 < 0$, then $x_{k+1} = x_k + 1$ and $y_{k+1} = y_k - 1$.

$$p_{k+1}^2 = b^2(x_{k+1} + 1/2)^2 + a^2(y_{k+1} - 1)^2 - a^2b^2$$
$$p_{k+1}^2 = b^2(x_k + 1 + 1/2)^2 + a^2(y_k - 1 - 1)^2 - a^2b^2$$
$$p_{k+1}^2 = b^2(x_k + 1/2)^2 + 2b^2(x_k + 1/2) + b^2 + a^2(y_k - 1)^2 - 2a^2(y_k - 1) + a^2 - a^2b^2$$
$$p_{k+1}^2 = p_k^2 + 2b^2(x_k + 1/2) + b^2 - 2a^2(y_k - 1) + a^2$$
$$p_{k+1}^2 = p_k^2 + 2b^2x_k + 2b^2 - 2a^2(y_k - 1) + a^2$$
$$p_{k+1}^2 = p_k^2 + 2b^2(x_k + 1) - 2a^2(y_k - 1) + a^2 + 2b^2$$
$$p_{k+1}^2 = p_k^2 + 2b^2x_{k+1} - 2a^2y_{k+1} + a^2$$

1. start $(0, b)$
2. calculate $p = b^2 - a^2 b$
3. $x = 0, y = b, xc, yc$
4. Repeat:
5. drawellipse(xc, yc, x, y)
6. $x = x + 1$
6. if $p < 0, p = p + 2b^2 x + b^2$
7. else, $y = y - 1, p = p + 2b^2 x + b^2 - 2a^2 y$
8. while $(2b^2 x < 2a^2 y)$
9. $p = b^2 x^2 + a^2 (y - 1)^2 - a^2 b^2$
10. Repeat:
11. drawellipse(xc,yc,x,y)
12. $y = y - 1$
13. if $p > 0, p = p - 2a^2 y + a^2$
14. $else, x = x + 1, p = p + 2b^2 x - 2a^2 y + a^2$
15. while $y >= 0$

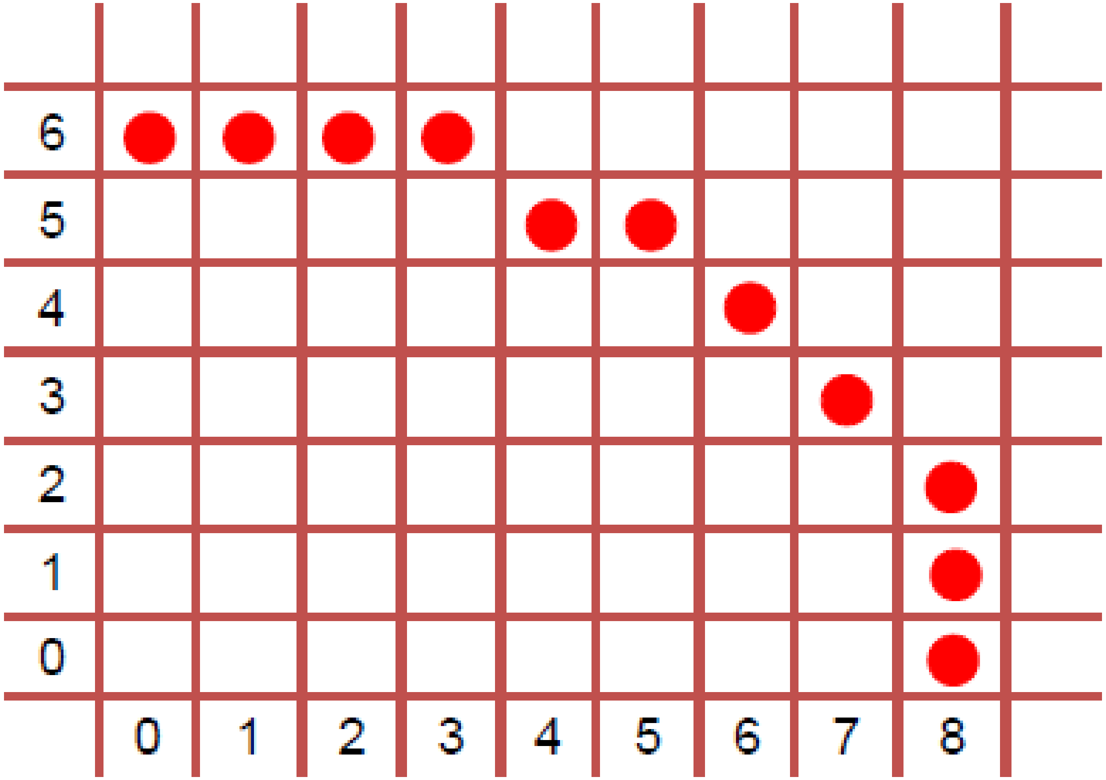
drawellipse(xc, yc, x,y)

1. putpixel(xc+x,yc+y)
2. putpixel(xc+x,yc-y)
3. putpixel(xc-x,yc+y)
4. putpixel(xc-x,yc-y)

Region 1						
a	b	x	y	p	$2b^2x$	$2a^2y$
8	6	0	6	-348	0	768
		1	6	-240	72	768
		2	6	-60	144	768
		3	6	192	216	768
		4	5	-124	288	640
		5	5	272	360	640
		6	4	228	432	512
		7	3	384	504	384

Region 2				
a	b	x	y	p
8	6	7	3	-284
		8	2	-44
		9	1	252
		9	0	316

Plot pixel positions



- Symmetric point calculation:
  - For (0, 6) : (0, 6), (0, -6), (0, -6)
  - For (1,6) : (-1, 6), (-1,-6), (1, -6)
  - For (2,6): (-2,6), (-2, -6), (2,-6)
  - For (3,6): (-3,6), (-3, -6), (3, -6)
  - For(4,5): (-4, 5), (-4,-5) (4,-5)
  - For (5, 5): (-5,5), (-5,-5) (5,-5)
  - For (6,4):(-6,4) (-6,-4) (6,-4)