Fundamentals of Computer Graphics

Lecture 2. Begin OpenGL

Yong-Jin Liu
liuyongjin@tsinghua.edu.cn

Outline

- What is OpenGL
- Raster graphics
- Draw an elemental shape
- Event-driven programming
- Use GLUT advanced library

What is OpenGL

A software API with many functions that allow communication with graphics hardware

- Cross-platform
- Provide a number of functions to draw things
- Common used in many graphics applications
- OpenGL is open source (free)

What is OpenGL

Device Independent Graphics

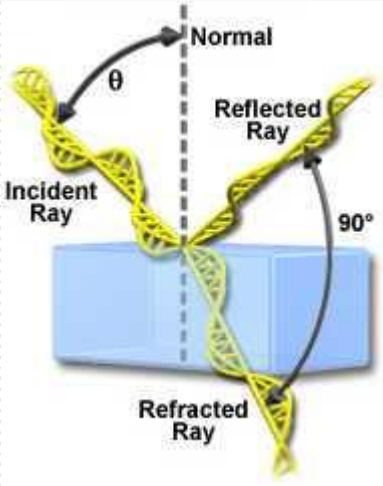
- Cross-platform: Allows same graphics program to be run on many different machine types with nearly identical output (.dll files must be with program)
- OpenGL is an API: it controls whatever hardware you are using, and you use its functions instead of controlling the hardware directly

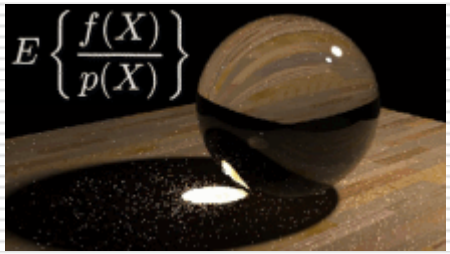
Design and limitation

OpenGL is designed to produce reasonably looking 3D images quickly and simply

- A lots of its design is a rough approximation of how visual phenomena behave in the real world

Design and limitation





Simple lighting
heuristic rules

Global
illumination

Physics
Optics

Computer
Science

Realistic(Accurate)
Very time consuming

Fast
Real time

Design and limitation

OpenGL is designed to produce reasonably looking 3D images quickly and simply

- Meant to only handle rendering, nothing else

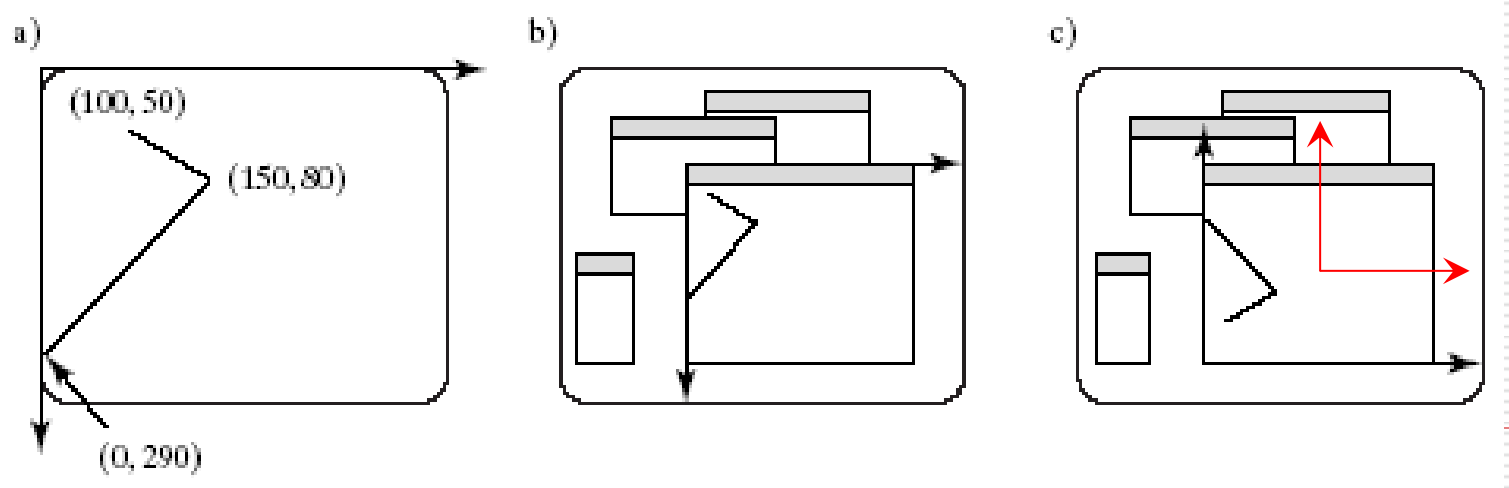
  So no window management, event-handling, etc…

- Meant to abstract away graphics hardware into a standard, clean interface

Getting Started Making Pictures

Graphics display: (a) Entire screen; (b) windows system

- Build-in coordinates in OS: a, b
- Coordinates in OpenGL: c



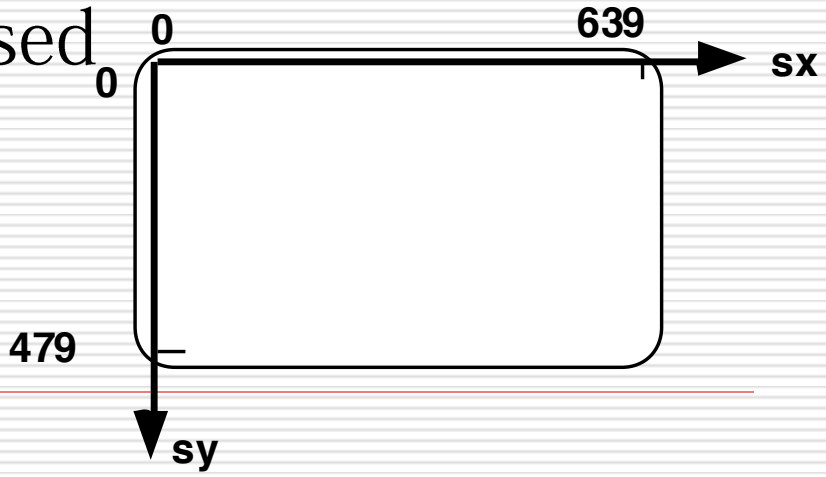
Graphics Display Devices

Graphics displays are either line-drawing devices or raster displays

- Line-drawing devices
  - Vector based
- Raster devices
  - Point based

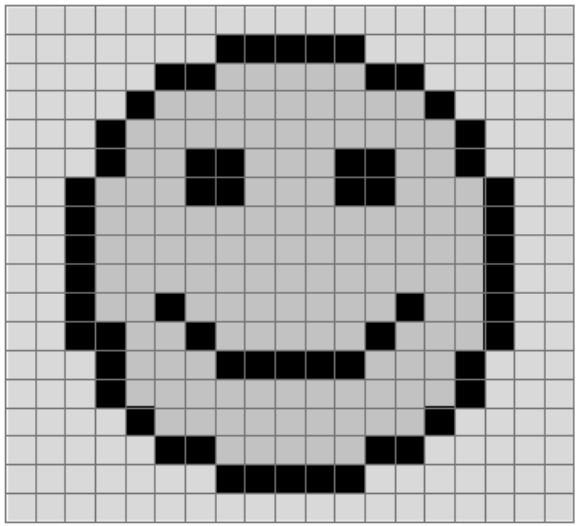
Graphics Display Devices

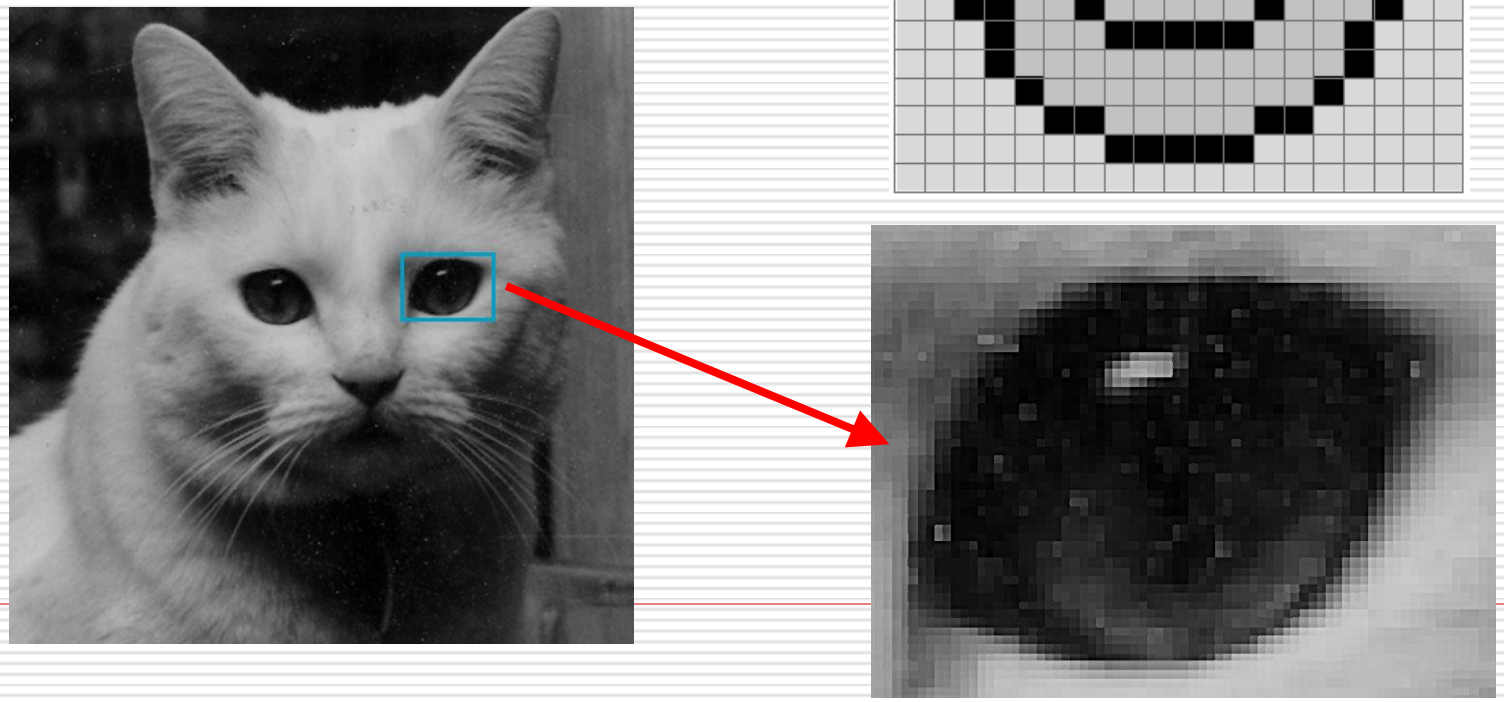
- Raster displays
  - Most computer monitors: moves a beam of electrons across the screen from left to right and top to bottom
  - Printer: does the same thing with ink or toner
  - Coordinate system used



Raster displays

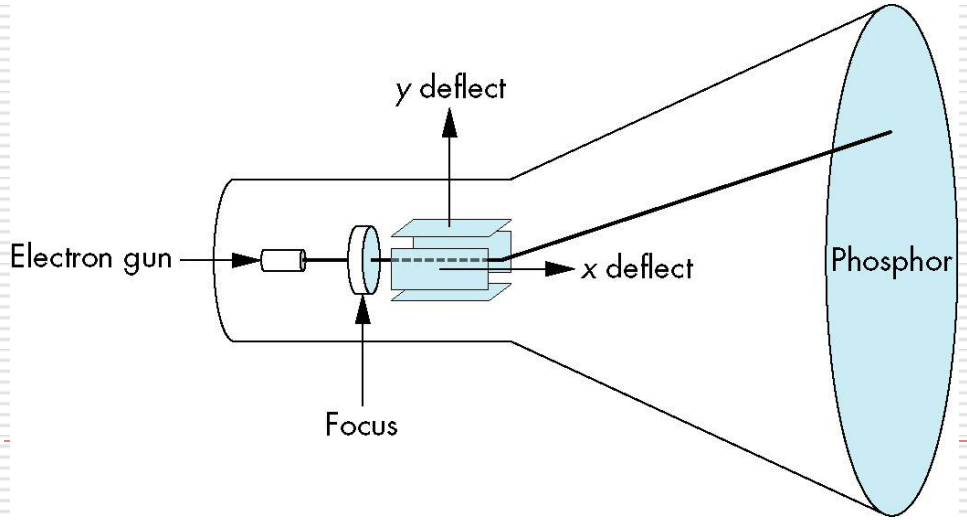
Every shape is drawn in a dot matrix

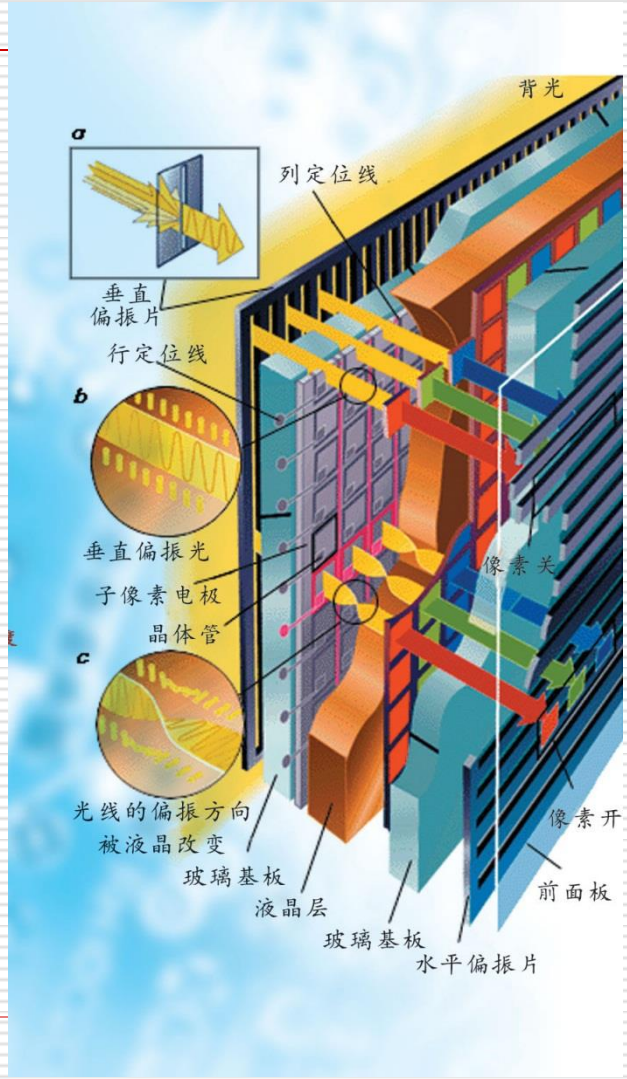




Raster displays

- Computer monitor
  - CRT (Cathode Ray Tube)
  - LCD (Liquid Crystal Display)
  - LED (Light Emitting Diode)

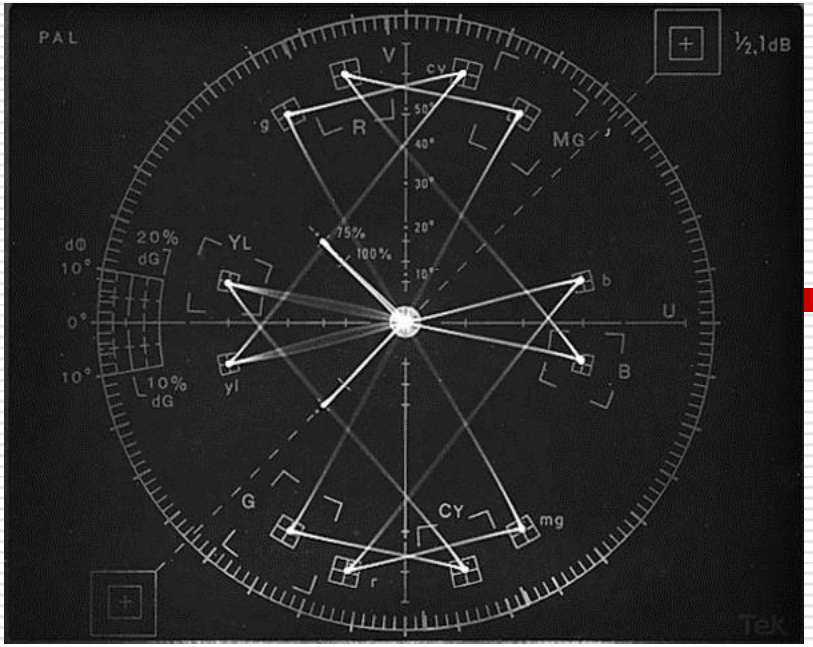




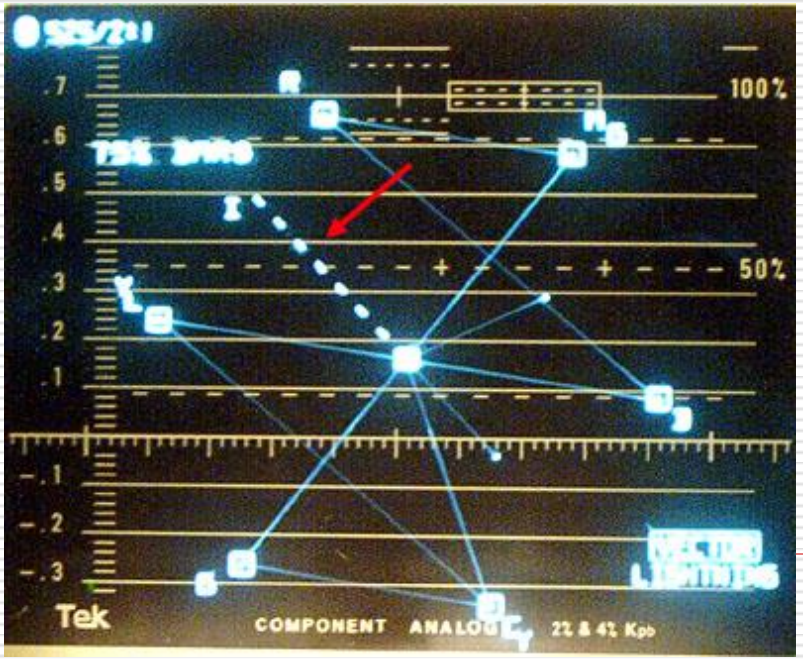
Graphics Display Devices

Graphics displays are either line-drawing devices or raster displays

- Line-drawing devices
  - Pen plotter, which moves an ink pen across a (large) sheet of paper. (E.g., seismic wave plotters.)
  - Vector video device, which moves a beam of electrons across the screen from any one point to any other point, leaving a glowing trail

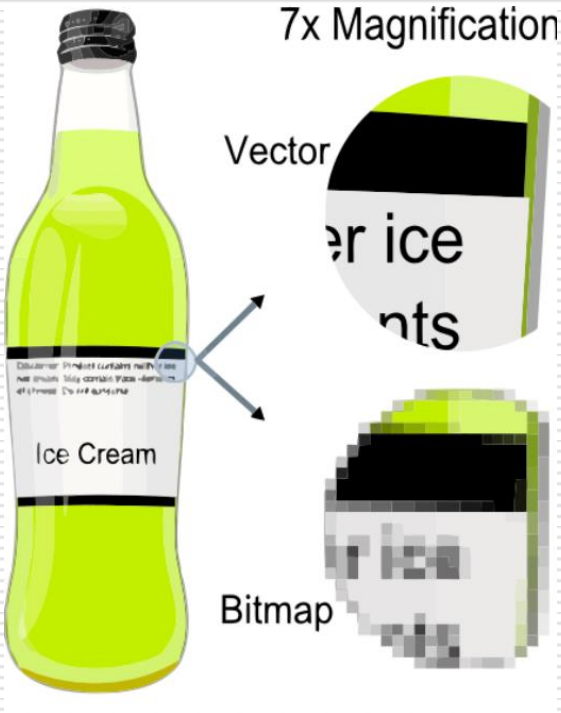

PAL
½,1dB
V
cy
R
MG
YL
b
U
B
yl
G
CY
mg
r
20%
dG
10%
dG
75%
100%
Tek

R
Mg
Yl
B
G
Cy
GAIN
X1
X5


100%
50%
Tek
COMPONENT ANALOG
2% & 4% Kpb

Vector graphics

Vector graphics is the use of geometrical primitives such as points, lines, curves, and other analytic shapes—all of which are based on mathematical expressions—to represent images in computer graphics. Vector graphics are based on vectors (also called paths), which lead through locations called control points or nodes. Each of these points has a definite position on the x and y axes of the work plane.



www.shutterstock.com · 24753970

Raster image

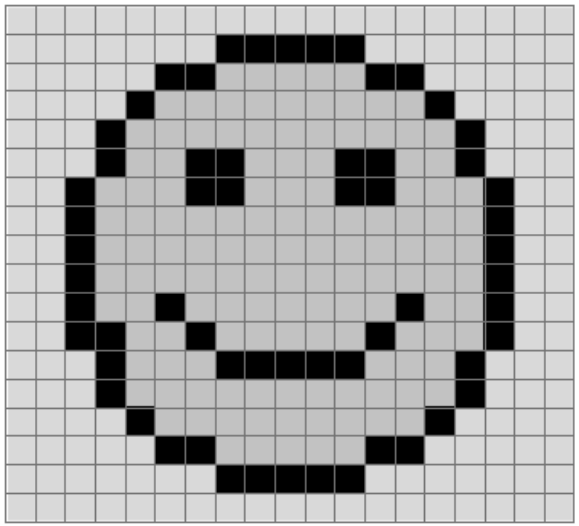
High color resolution

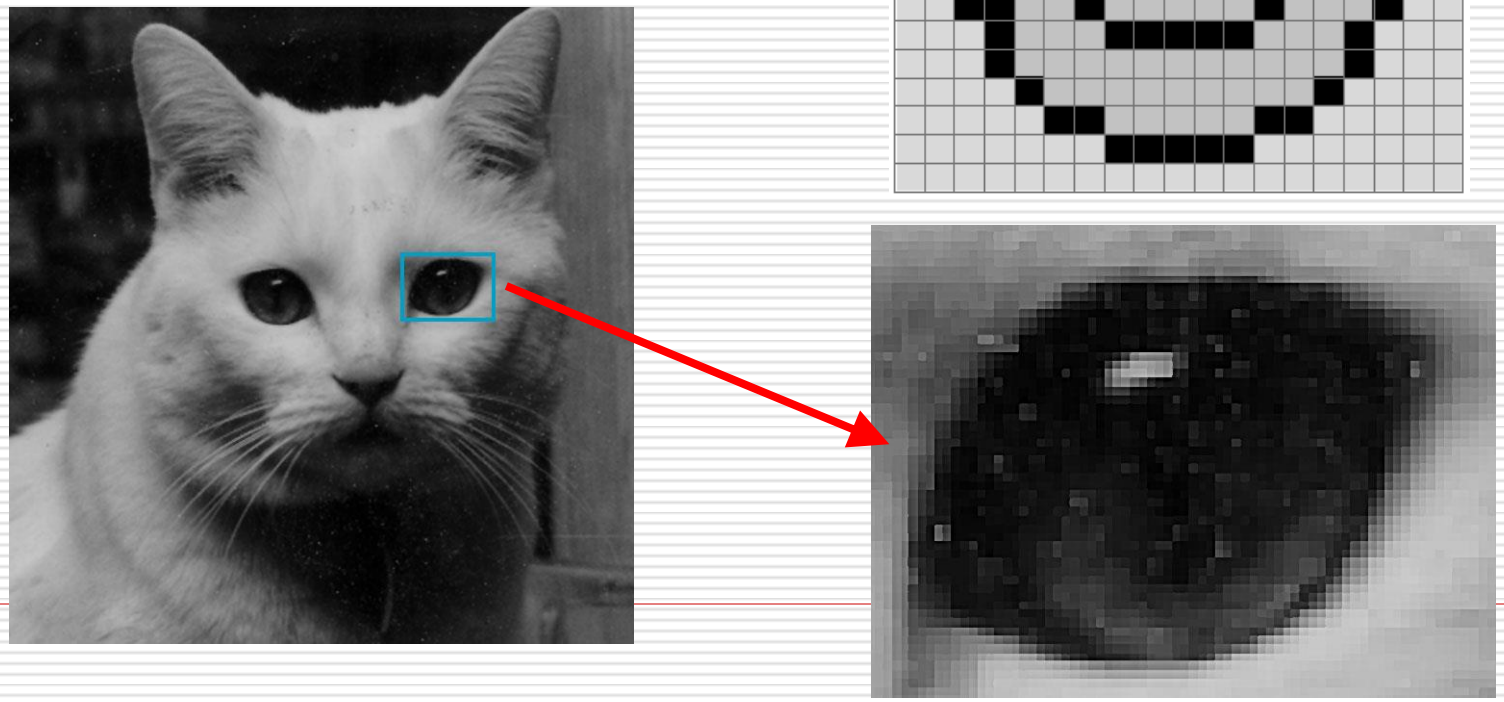
Vector image

Low color resolution

Raster displays

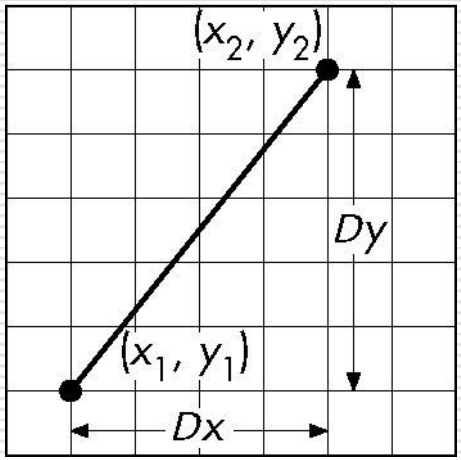
Every shape is drawn in a dot matrix

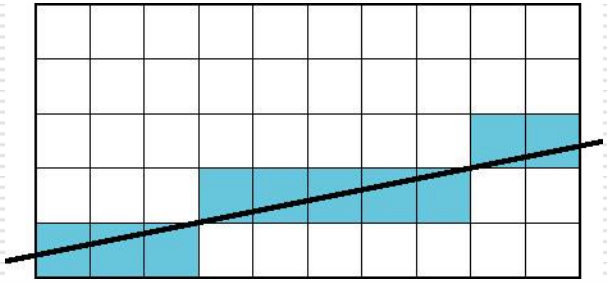




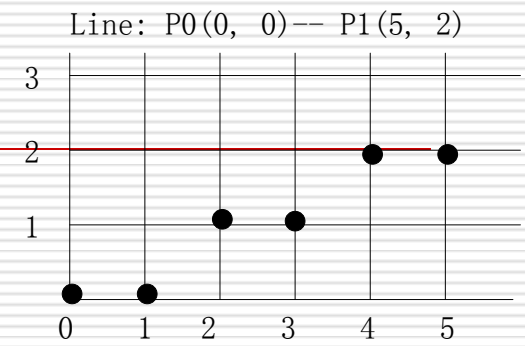
Raster displays

- Scan conversion of line





Scan conversion of line



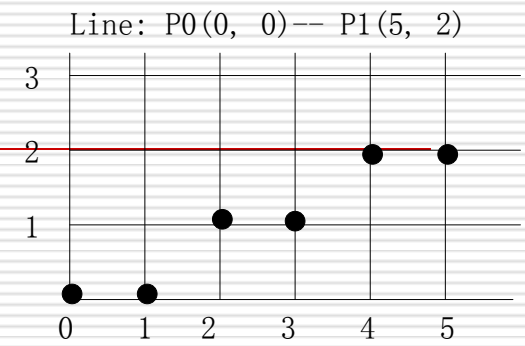
Solution

A line passing $P_0(x_0, y_0), P_1(x_1, y_1)$

$$y = kx + b \qquad k = \frac{y_1 - y_0}{x_1 - x_0}$$

```
for (x=x0; x<=x1; x++) {
      y = kx+b;
      draw_pixel(x, round(y), line_color);
}
```

Scan conversion of line

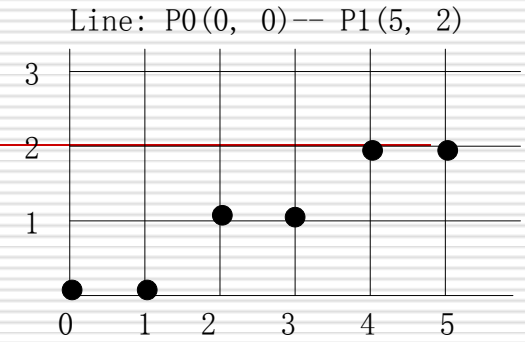


Can we do better?

- Digital differential analyser
- dy/dx = k
- $(x_1, y_1), (x_2, y_2)$
- $y = kx + b \;\Rightarrow\; \Delta y = k\Delta x \;\Rightarrow\; \Delta y = k \quad (\Delta x = 1)$

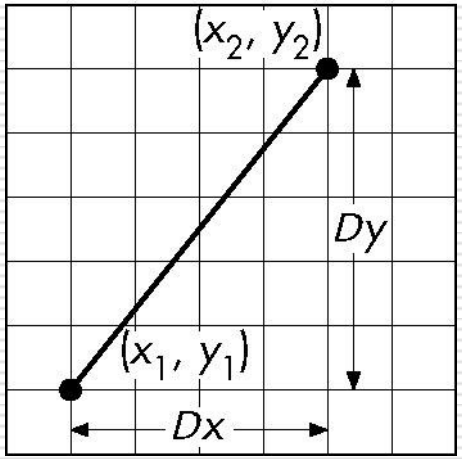
```
for (x=x0; x<=x1; x++) {
        y+=k;
        draw_pixel(x, round(y), line_color);
}
```

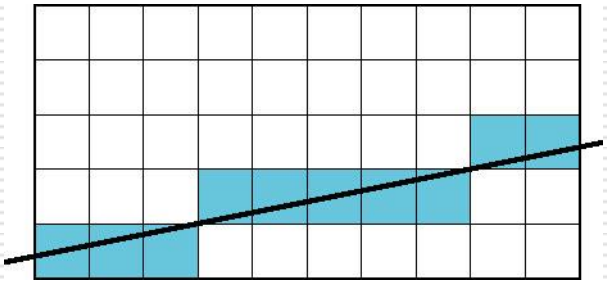
Scan conversion of line

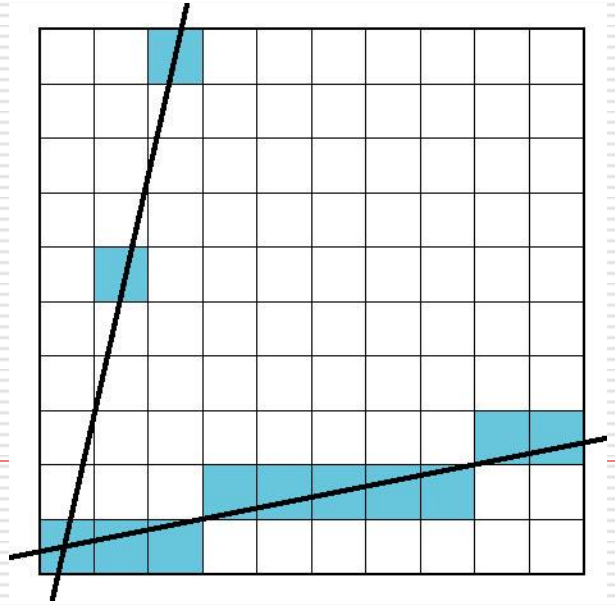


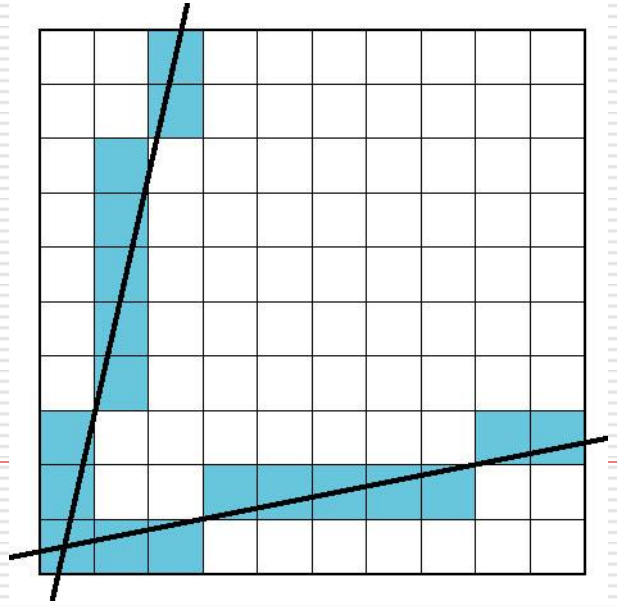
- Can we do better?
  - Digital differential analyser (DDA)
- Again: Can we do better?
- Or: is there any not perfect in DDA

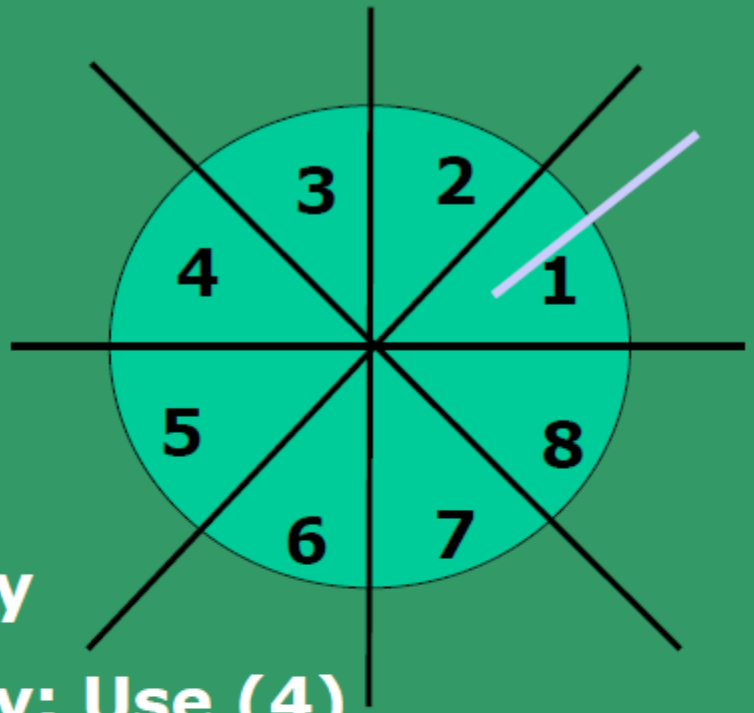
Scan conversion of line











Octant	Change
1	None
2	Switch roles of x & y
3	Switch roles of x & y; Use (4)
4	Draw from P_1 to P_0; Use (8)
5	Draw from P_1 to P_0
6	Draw from P_1 to P_0; Use (2)
7	Switch roles of x & y; Use (8)
8	Use y = y - 1.

MIDPOINT LINE ALGORITHM

Incremental Algorithm (Assume first octant)

Given the choice of the current pixel, which one do we choose next : E or NE?

Equations:

1. $y = (dy/dx) * x + B$

Rewrite as:

2. $F(x,y) = a*x + b*y + c = 0$

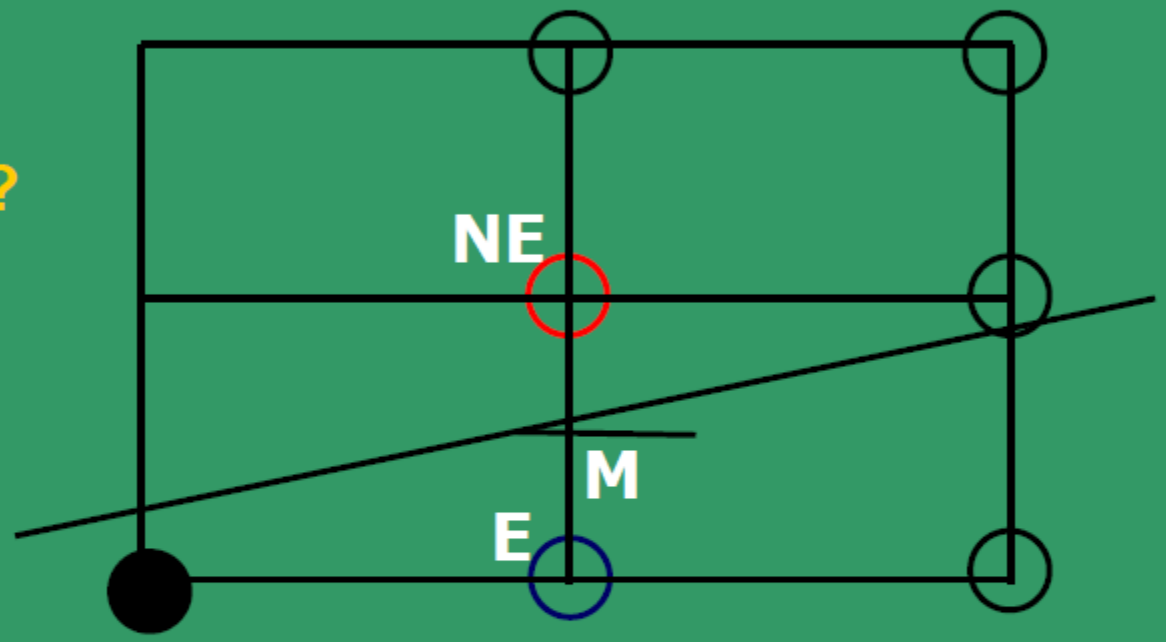
Gives: $F(x,y) = dy*x - dx*y + B*dx = 0$

$\Rightarrow a = dy,\ b = -dx,\ c = B*dx$

Criteria:

Evaluate the mid-point, M,
w.r.t. the equation of the line.

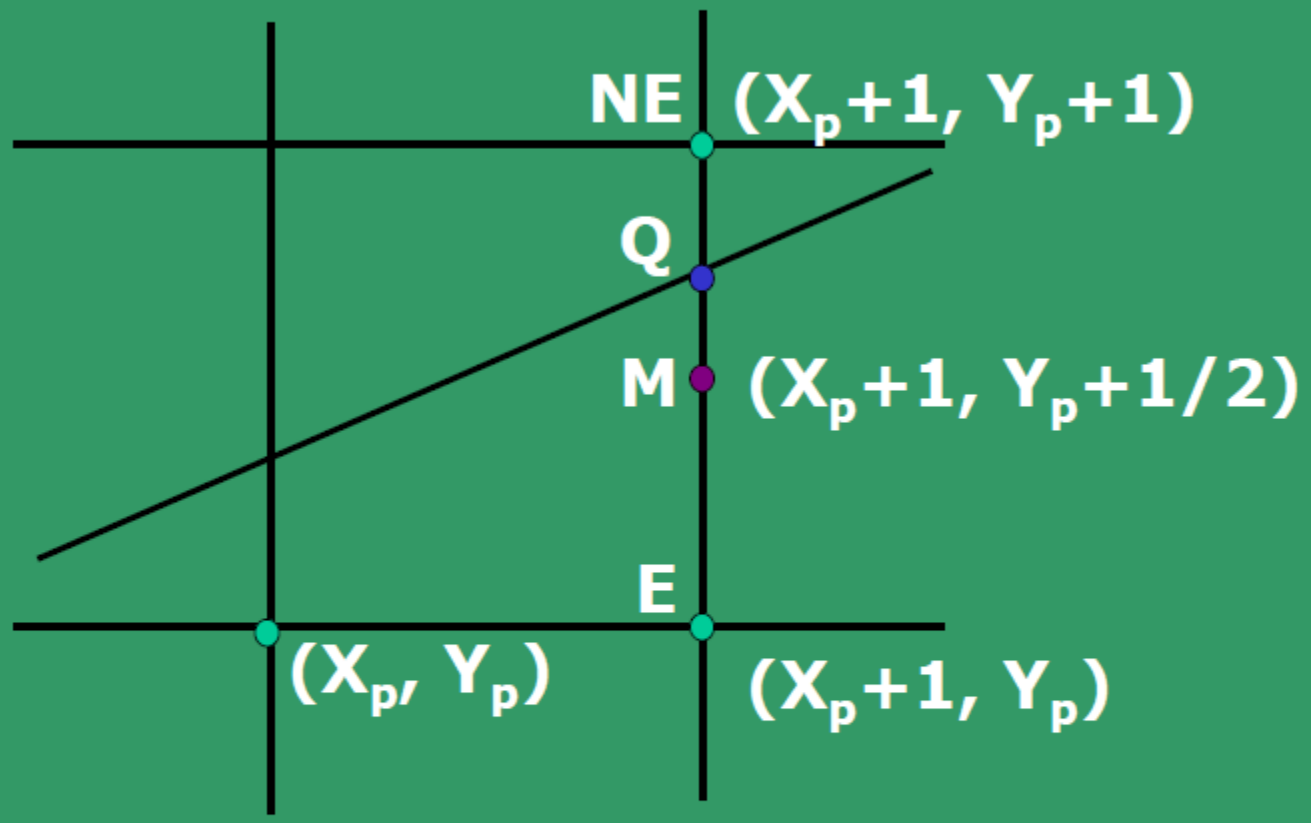
Choice: E or NE?



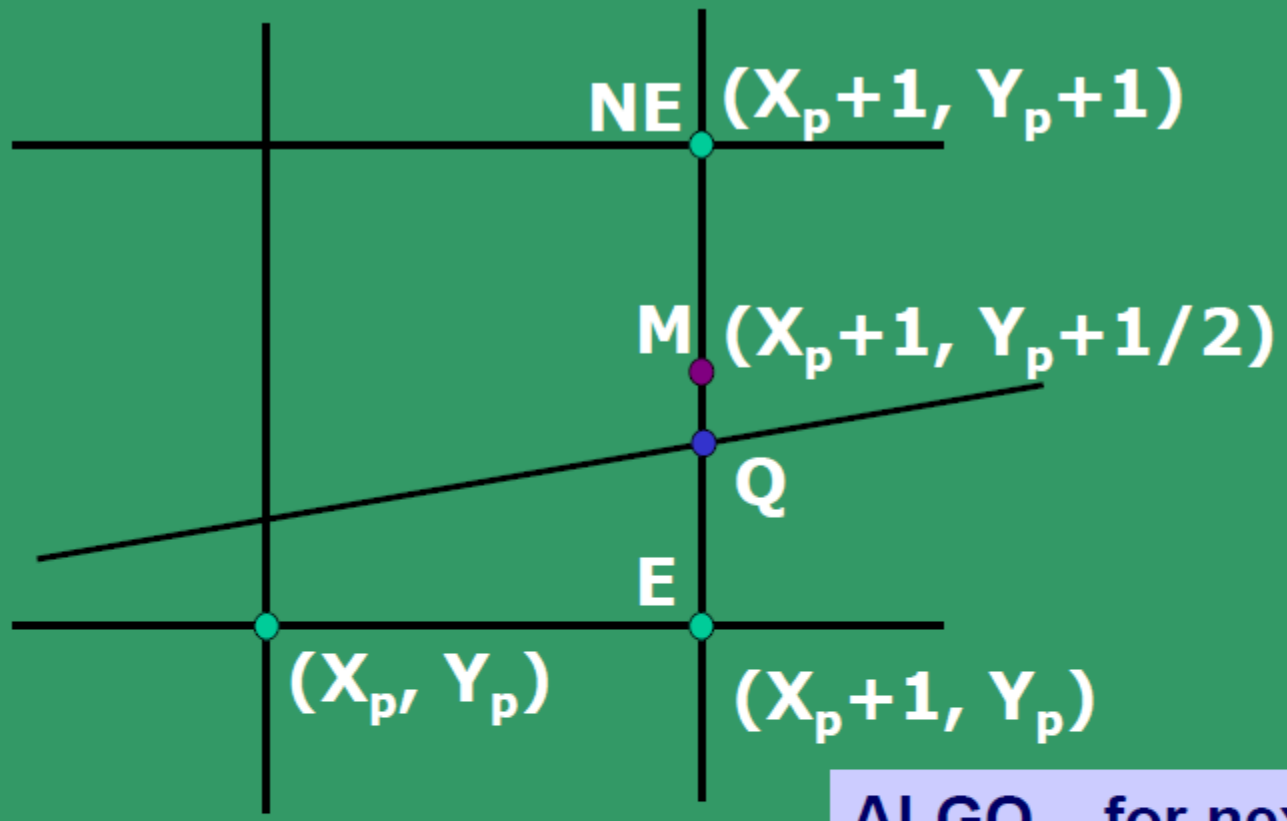
$$F(x,y) = dy*x - dx*y + B*dx = 0$$

$F(x,y) > 0$; if point below the line

$F(x,y) < 0$; if point above the line



Q is above M,
hence select NE pixel as your next choice



Q is below M, hence select E pixel as your next choice

ALGO – for next choice:

If F(M) > 0 ***/*Q is above M */***

then Select NE

/*M is below the line*/

else Select E ;

/* also with F(M) = 0 */

Evaluate mid-point M using a decision variable d = F(X,Y);

$d = F(X_p+1, Y_p+1/2) = a(X_p+1)+b(Y_p+1/2)+c;$
at M,

Set $d_{old} = d$;

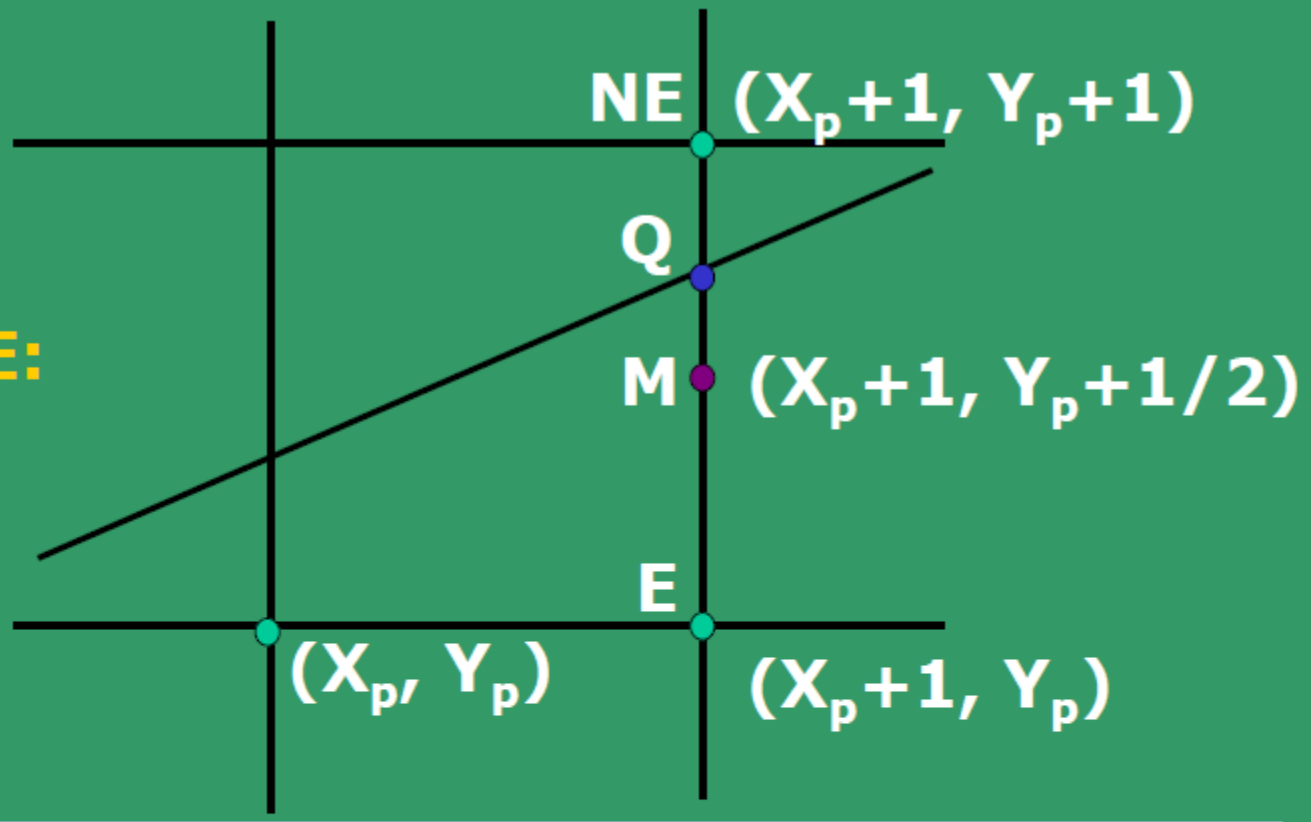
Based on the sign of d, you choose E or NE.

Case I. Chosen E:

$$d_{new} = F(X_p+2, Y_p+1/2)$$
$$= a(X_p+2) + b(Y_p+1/2) + c$$

$$(\Delta d)_E = d_{new} - d_{old} = a \quad /* = dy */$$

Case II.
Chosen NE:



$$d_{new} = F(X_p+2, Y_p+3/2)$$
$$= a(X_p+2) + b(Y_p+3/2) + c$$

$$(\Delta d)_{NE} = d_{new} - d_{old} = a + b \quad /^*= dy - dx \; ^*/$$

Update using $d_{new} = d_{old} + \Delta d$

Midpoint criteria

$d = F(M) = F(X_p+1, Y_p+1/2);$

if d > 0 choose NE

else /* *if d <= 0* */ choose E ;

Case EAST :

increment M by 1 in x

$d_{new} = F(M_{new}) = F(X_p + 2, Y + 1/2)$

$(\Delta d)_E = d_{new} - d_{old} = a = dy$

$(\Delta d)_E = dy$

Case NORTH-EAST:

increment M by 1 in both x and y

$d_{new} = F(M_{new}) = F(X_p + 2, Y_p + 3/2)$

$(\Delta d)_{NE} = d_{new} - d_{old} = a + b = dy - dx$

$(\Delta d)_{NE} = dy - dx$

What is d_{start}?

$$d_{start} = F(x_0 + 1, y_0 + 1/2)$$
$$= ax_0 + a + by_0 + b/2 + c$$
$$= F(x_0, y_0) + a + b/2$$
$$= dy - dx/2$$

Let's get rid of the fraction and see what we end up with for all the variables:

$$d_{start} = 2dy - dx ;$$

$$(\Delta d)_E = 2dy ;$$

$$(\Delta d)_{NE} = 2(dy - dx) ;$$

The Midpoint Line Algorithm

$x = x_0;$ $\quad y = y_0;$

$dy = y_1 - y_0;$ $\quad dx = x_1 - x_0;$

$d = 2dy - dx;$

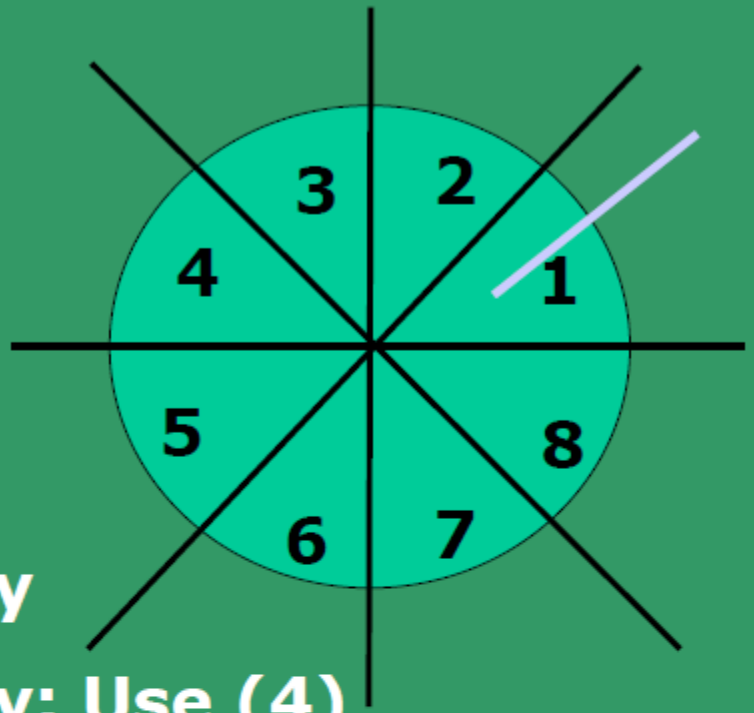
$(\Delta d)_E = 2dy;$

$(\Delta d)_{NE} = 2(dy - dx);$

Plot_Point(x,y)

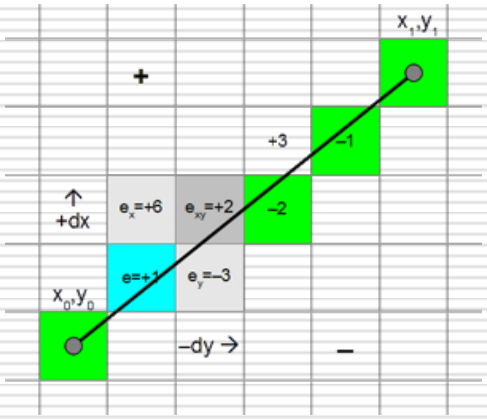
The Midpoint Line Algorithm (Contd.)

```
while (x < x1)
        if (d <= 0)    /* Choose E */
                d = d + (Δd)E ;

        else           /* Choose NE */
                d = d + (Δd)NE ;
                y = y + 1

        endif

        x = x + 1 ;

        Plot_Point(x, y) ;

end while
```

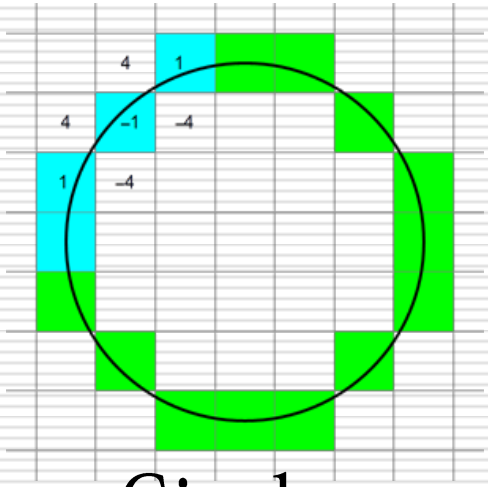


Octant	Change
1	None
2	Switch roles of x & y
3	Switch roles of x & y; Use (4)
4	Draw from P_1 to P_0; Use (8)
5	Draw from P_1 to P_0
6	Draw from P_1 to P_0; Use (2)
7	Switch roles of x & y; Use (8)
8	Use y = y - 1.

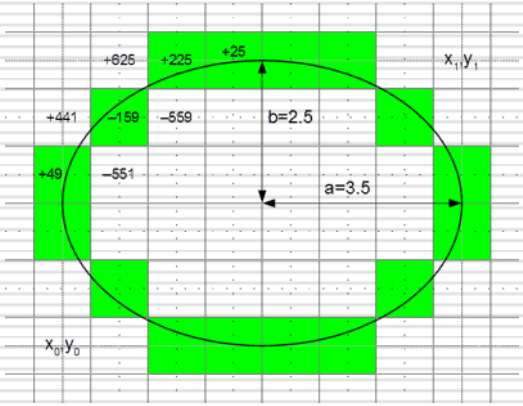
Scan conversion of … …



Line



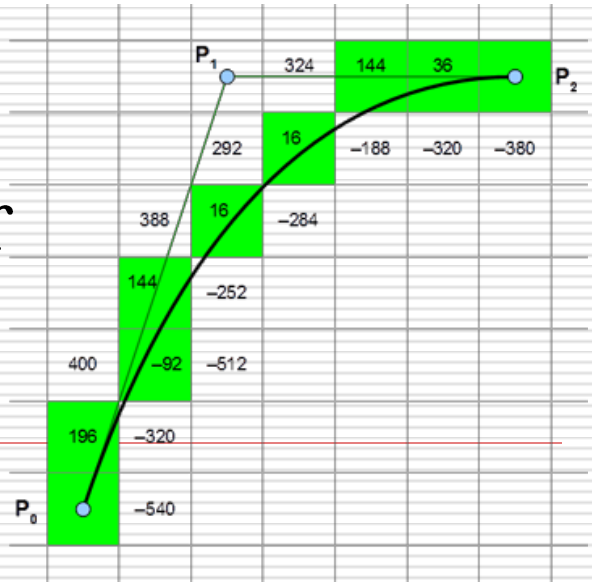
Circle

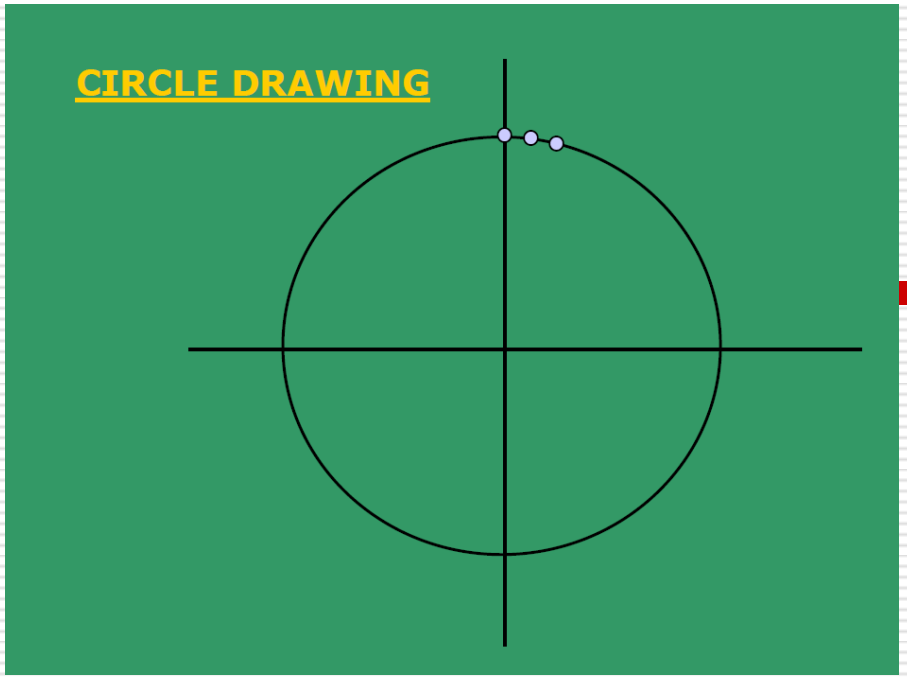


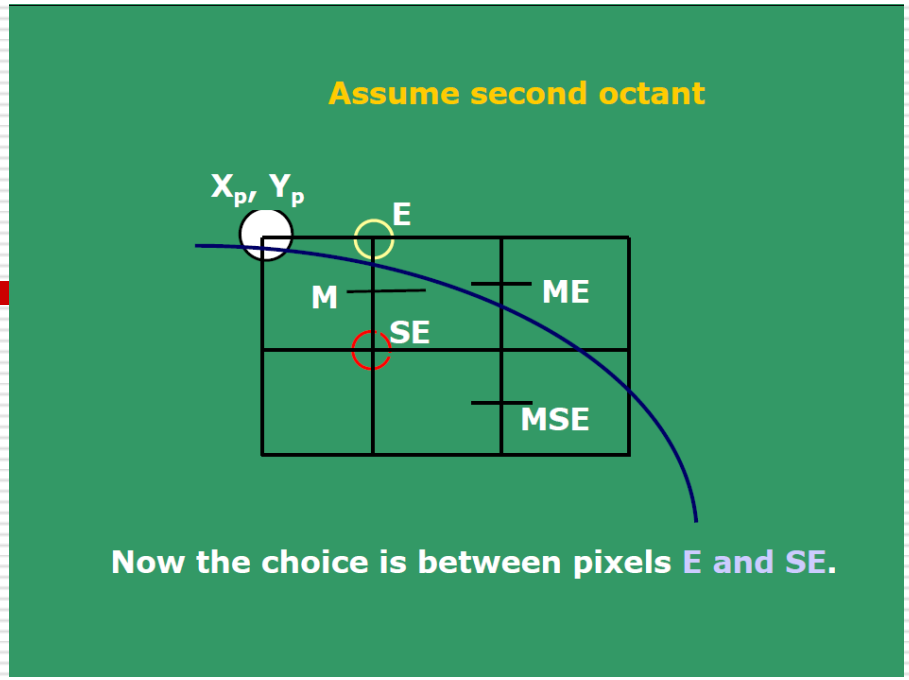
Ellipse

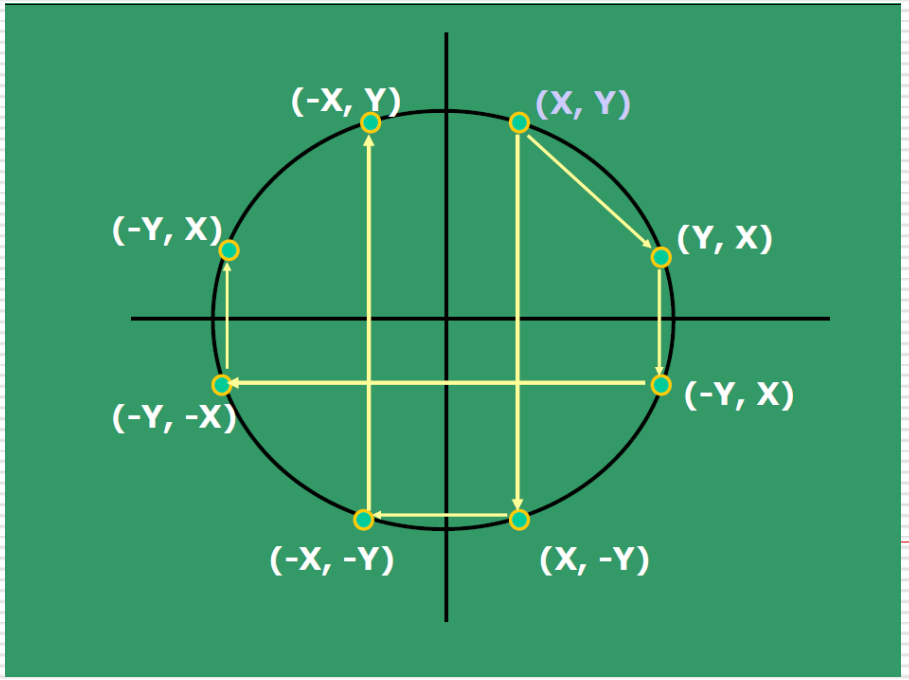
Shape with analytic forms

Bezier curve



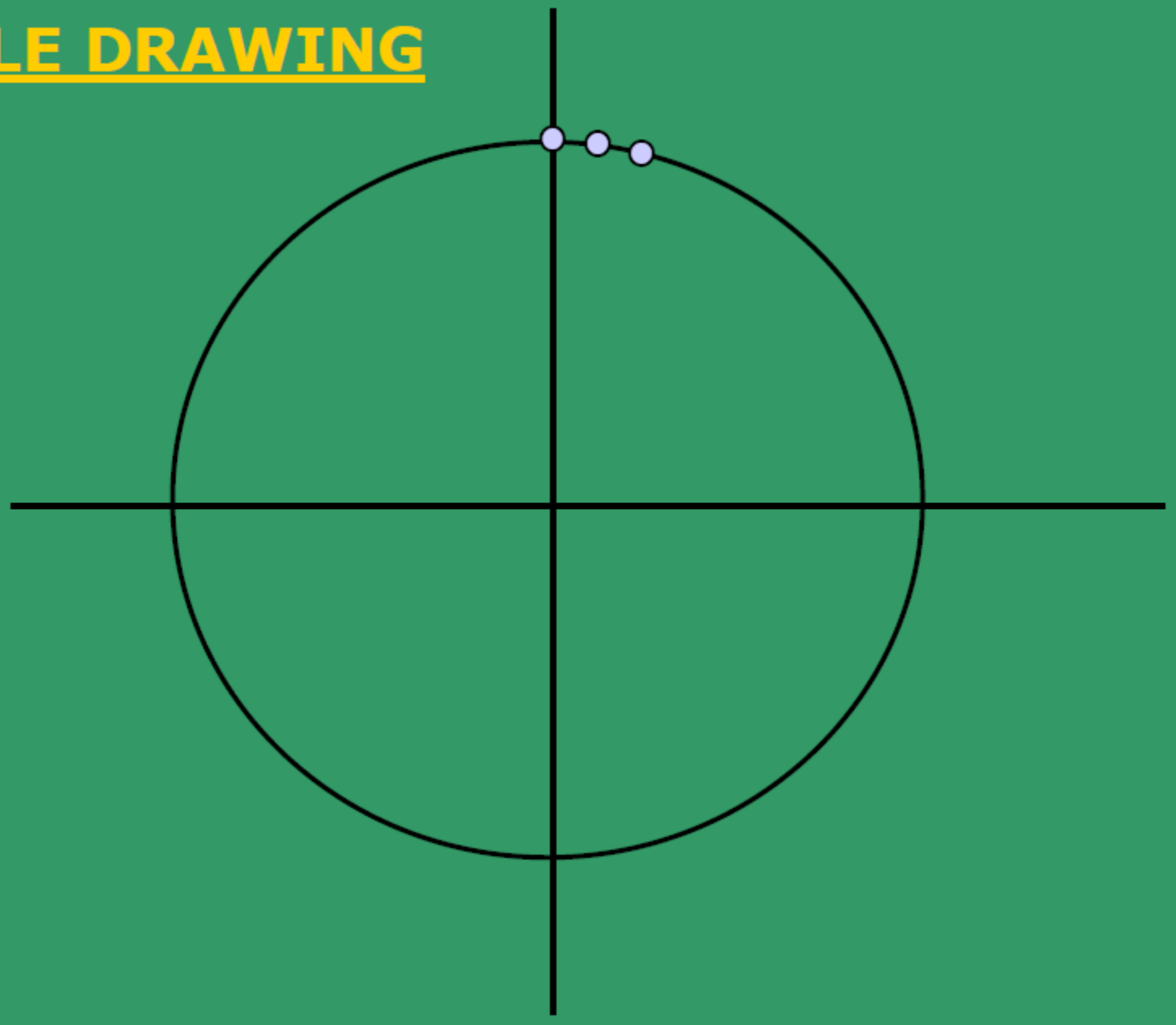




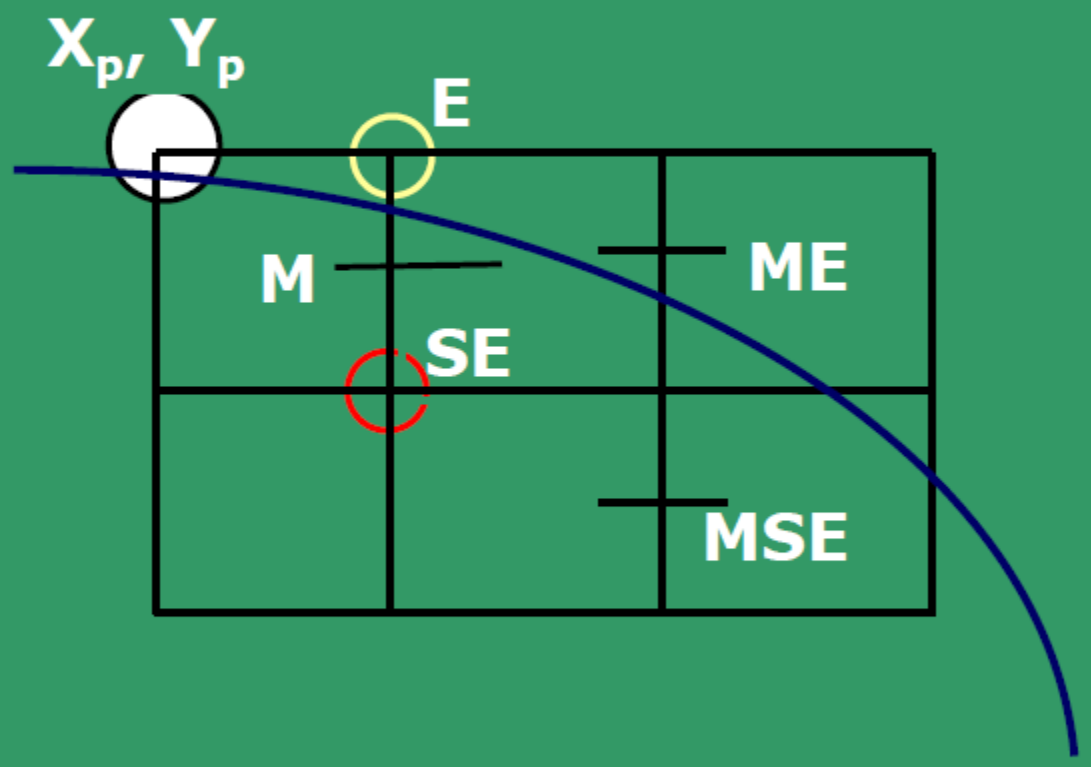


Scan conversion of an arc

CIRCLE DRAWING



Assume second octant



Now the choice is between pixels E and SE.

CIRCLE DRAWING

Only considers circles centered at the origin with integer radii.

Can apply translations to get non-origin centered circles.

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

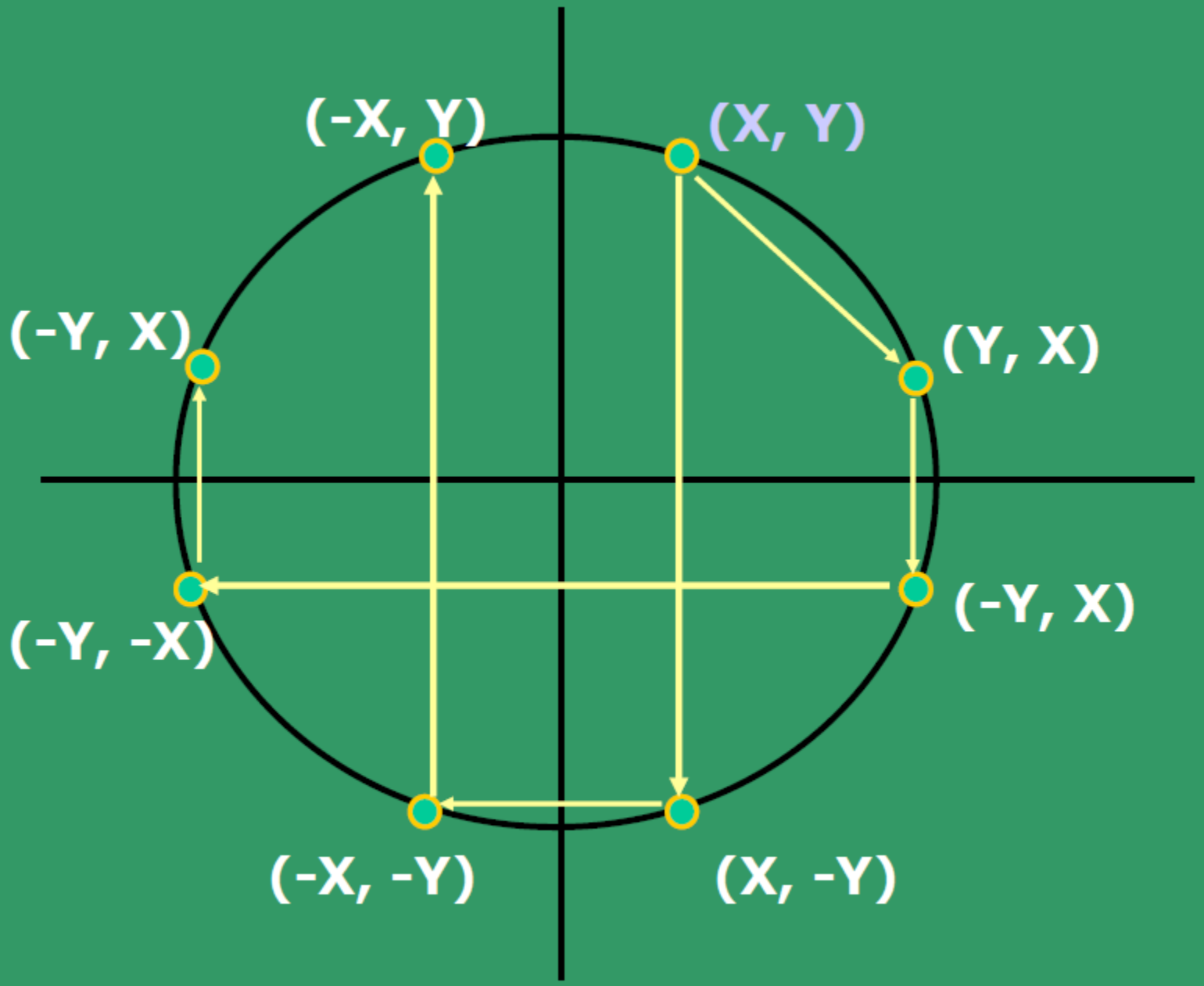
Implicit equation: $F(x,y)= x^2 + y^2 - R^2 =0$

Note: Implicit equations used extensively for advanced modeling

(e.g., liquid metal creature from "Terminator 2")

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
```


(-X, Y)
(X, Y)
(-Y, X)
(Y, X)
(-Y, -X)
(-Y, X)
(-X, -Y)
(X, -Y)

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.

Again, use $d_{old} = F(M)$;

$$F(M) = F(X_p + 1, Y_p - 1/2)$$
$$= (X_p + 1)^2 + (Y_p - 1/2)^2 - R^2$$

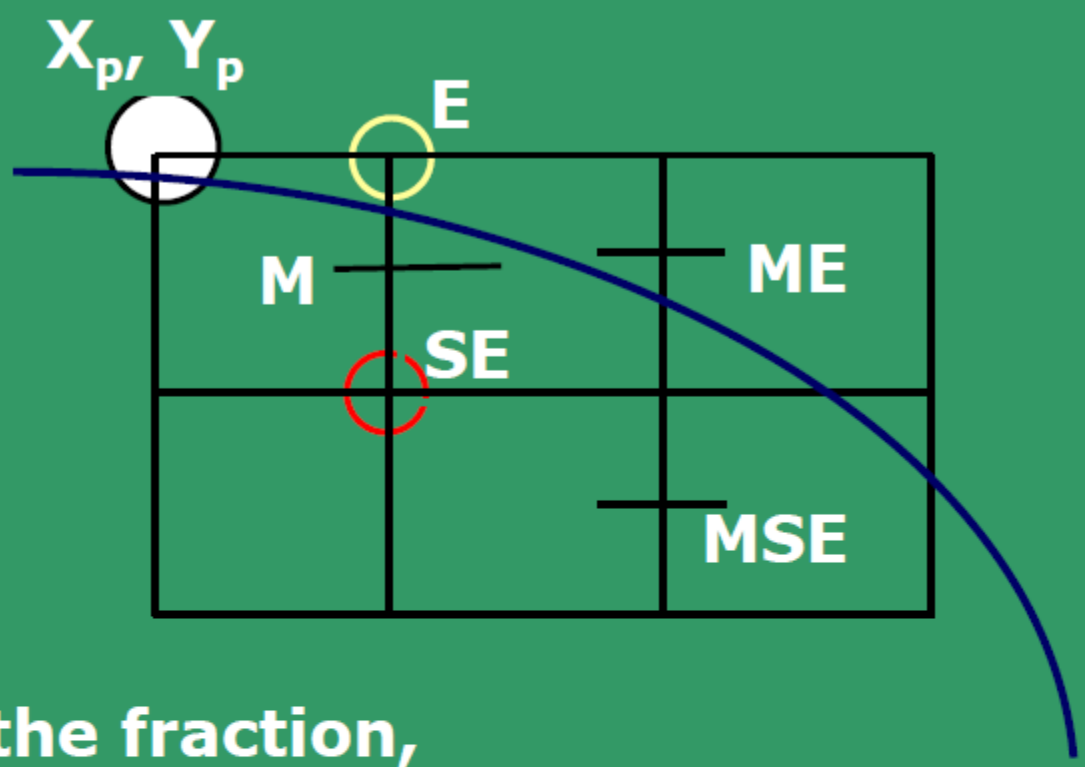
$d >= 0$ choose SE ; next midpoint: M_{new};
Increment + 1 in X, -1 in y; which gives d_{new}.

$d < 0$ choose E ; next midpoint: M_{new};
Increment + 1 in X; which gives = d_{new}.

$$(\Delta d)_{SE} = d_{new} - d_{old}$$
$$= F(X_p + 2, Y_p - 3/2) - F(X_p + 1, Y_p - 1/2)$$
$$= 2X_p - 2Y_p + 5 ;$$

$$(\Delta d)_E = d_{new} - d_{old}$$
$$= F(X_p + 2, Y_p - 1/2) - F(X_p + 1, Y_p - 1/2)$$
$$= 2X_p + 3;$$

$$d_{start} = F(X_0 + 1, Y_0 - 1/2) = F(1, R - 1/2)$$
$$= 1 + (R - 1/2)^2 - R^2 = 1 + R^2 - R + 1/4 - R^2$$
$$= 5/4 - R$$



To get rid of the fraction,

Let $h = d - 1/4$ => $h_{start} = 1 - R$

Comparison is: $h < -1/4$.

Since h is initialized to and incremented by integers, so we can just do with: $h < 0$.

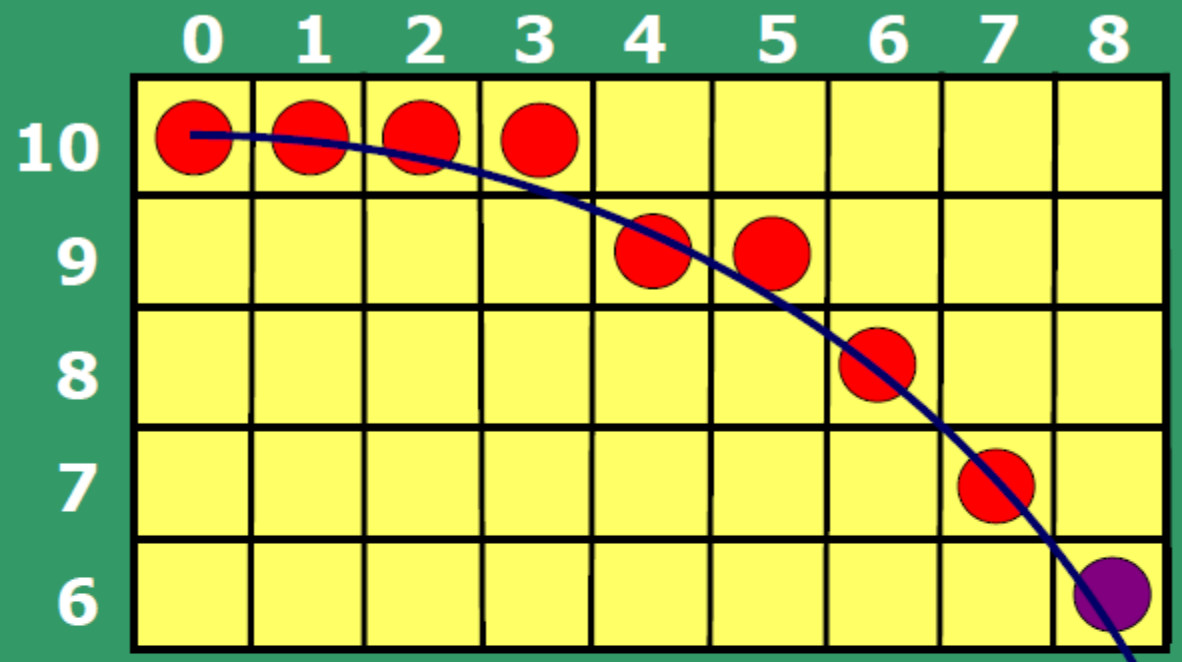
The Midpoint Circle algorithm: (Version 1)

```
x = 0;
y = R;
h = 1 – R;

DrawCircle(x, y);

while (y > x)
    if h < 0   /* select E */
        h = h + 2x + 3;
```

```
        else   /* select SE */
               h = h + 2(x - y) + 5;
               y = y - 1;
    endif

        x = x + 1;
        DrawCircle(x, y);

end_while
```

Example:
R = 10;

Initial Values:
h = 1 – R = -9;
X = 0; Y = 10;
2X = 0;
2Y = 20.



K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
2X	0	2	4	6	8	10	12
2Y	20	20	20	20	18	18	16
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

Problems with this?

Requires at least 1 multiplication and 3 additions per pixel.

Why? Because $(\Delta d)_E$ and $(\Delta d)_{SE}$ are linear functions and not constants.

Solution?

All we have to do is calculate the differences for: $(\Delta d)_E$ and $(\Delta d)_{SE}$ (check if these will be constants). Say, $(\Delta d^2)_E$ and $(\Delta d^2)_{SE}$.

If we chose E, then we calculate $(\Delta d^2)_{E/E}$ and $(\Delta d^2)_{E/SE}$, based on this. Same if we choose SE, then calculate $(\Delta d^2)_{SE/E}$ and $(\Delta d^2)_{SE/SE}$.

If we chose E, go from (X_p, Y_p) to $(X_p + 1, Y_p)$

$$(\Delta d)_{E\text{-old}} = 2X_p + 3, \; (\Delta d)_{E\text{-new}} = 2X_p + 5.$$

Thus $(\Delta d^2)_{E/E} = 2.$

$$(\Delta d)_{SE\text{-old}} = 2X_p - 2Y_p + 5,$$

$$(\Delta d)_{SE\text{-new}} = 2(X_p + 1) - 2Y_p + 5$$

Thus $(\Delta d^2)_{E/SE} = 2.$

If we chose SE,
go from (X_p, Y_p) to $(X_p + 1, Y_p - 1)$

$$(\Delta d)_{E\text{-old}} = 2X_p + 3, \ (\Delta d)_{E\text{-new}} = 2X_p + 5.$$

Thus $(\Delta d^2)_{SE/E} = 2.$

$$(\Delta d)_{SE\text{-old}} = 2X_p - 2Y_p + 5,$$
$$(\Delta d)_{SE\text{-new}} = 2(X_p + 1) - 2(Y_p - 1) + 5$$

Thus $(\Delta d^2)_{SE/SE} = 4.$

So, at each step, we not only increment h, but we also increment $(\Delta d)_E$ and $(\Delta d)_{SE}$.

What are $(\Delta d)_{E\text{-start}}$ and $(\Delta d)_{SE\text{-start}}$?

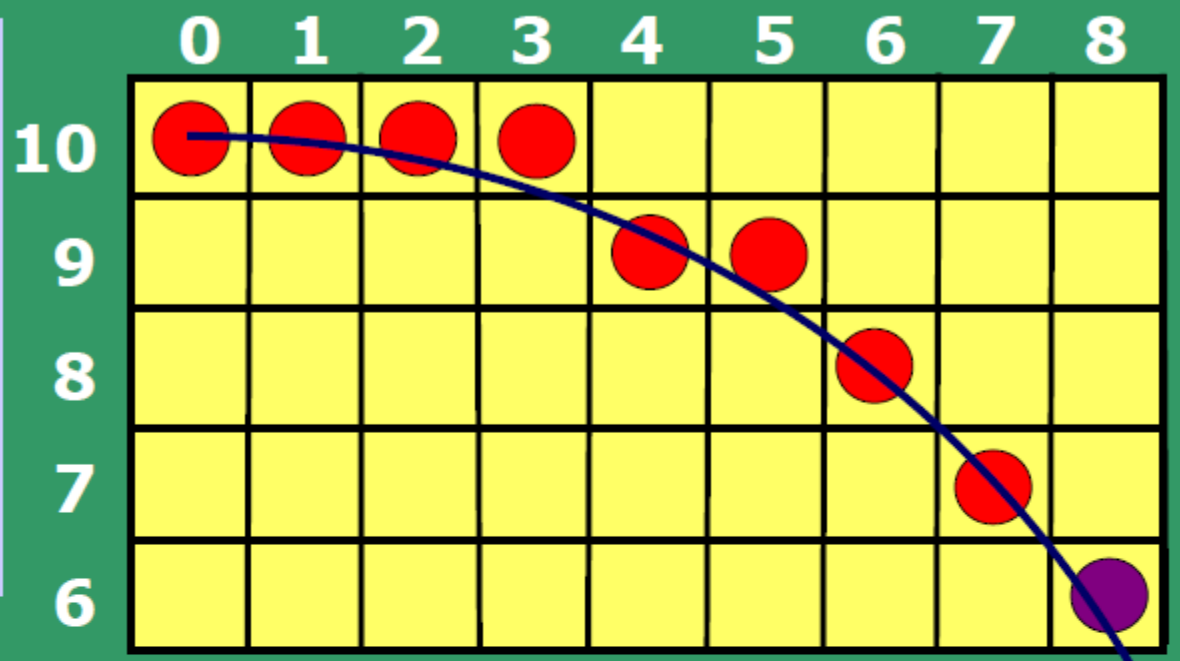
$$(\Delta d)_{E\text{-start}} = 2*(0) + 3 = 3 \ ;$$
$$(\Delta d)_{SE\text{-start}} = 2*(0) - 2*(R) + 5$$

The MidPoint Circle Algorithm
(Version 2):

```
x = 0;                y = radius;
h = 1 – R ;
deltaE = 3 ;      deltaSE = -2*R + 5 ;

DrawCircle(x, y);
  while (y > x)
                if h < 0      /* select E */
                      h = h + deltaE ;
                      deltaE  = deltaE + 2;
                      deltaSE= deltaSE + 2
```

```
else        /* select SE */

        h = h + deltaSE ;

        deltaE   = deltaE + 2 ;
        deltaSE = deltaSE + 4
        y = y – 1 ;

    endif
    x = x + 1 ;

DrawCircle(x, y) ;
end_while
```

Example:
R = 10;
Initial Values:
X = 0; Y = 10;
h = 1 – R = -9;
$\Delta_E = 3$;
$\Delta_{SE} = -15$;



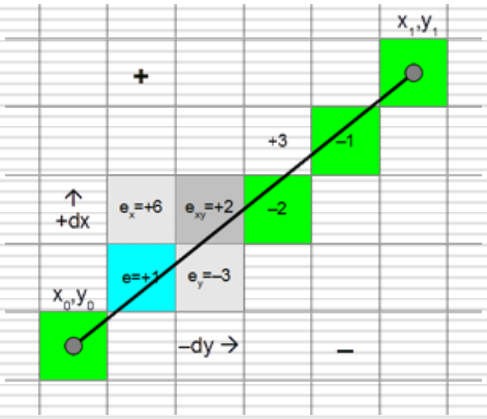
K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
Δ_E	5	7	9	11	13	15	17
Δ_{SE}	-13	-11	-9	-5	-3	1	5
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

Comparison of the solutions with two different methods

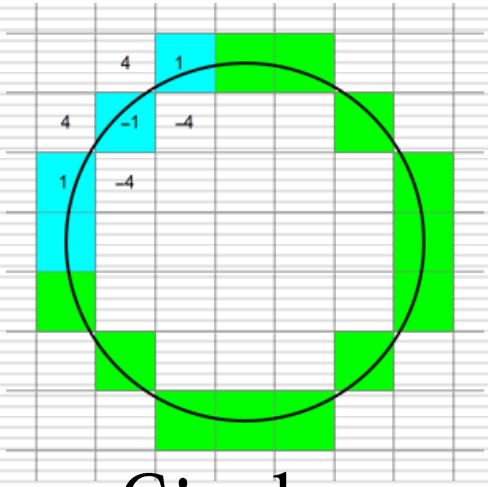
K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
Δ_E	5	7	9	11	13	15	17
Δ_{SE}	-13	-11	-9	-5	-3	1	5
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

K	1	2	3	4	5	6	7
h	-6	-1	6	-3	8	5	6
2X	0	2	4	6	8	10	12
2Y	20	20	20	20	18	18	16
X, Y	(1, 10)	(2, 10)	(3, 10)	(4, 9)	(5, 9)	(6, 8)	(7, 7)

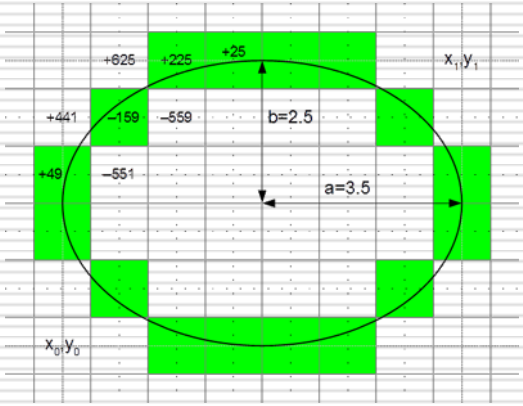
Scan conversion of … …



Line



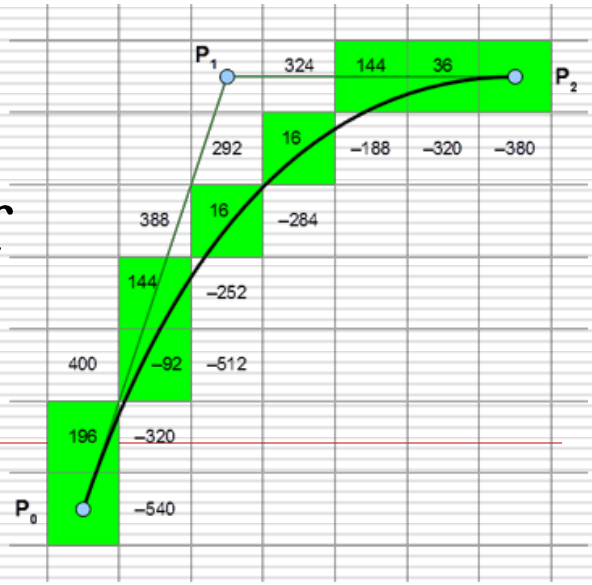
Circle



Ellipse

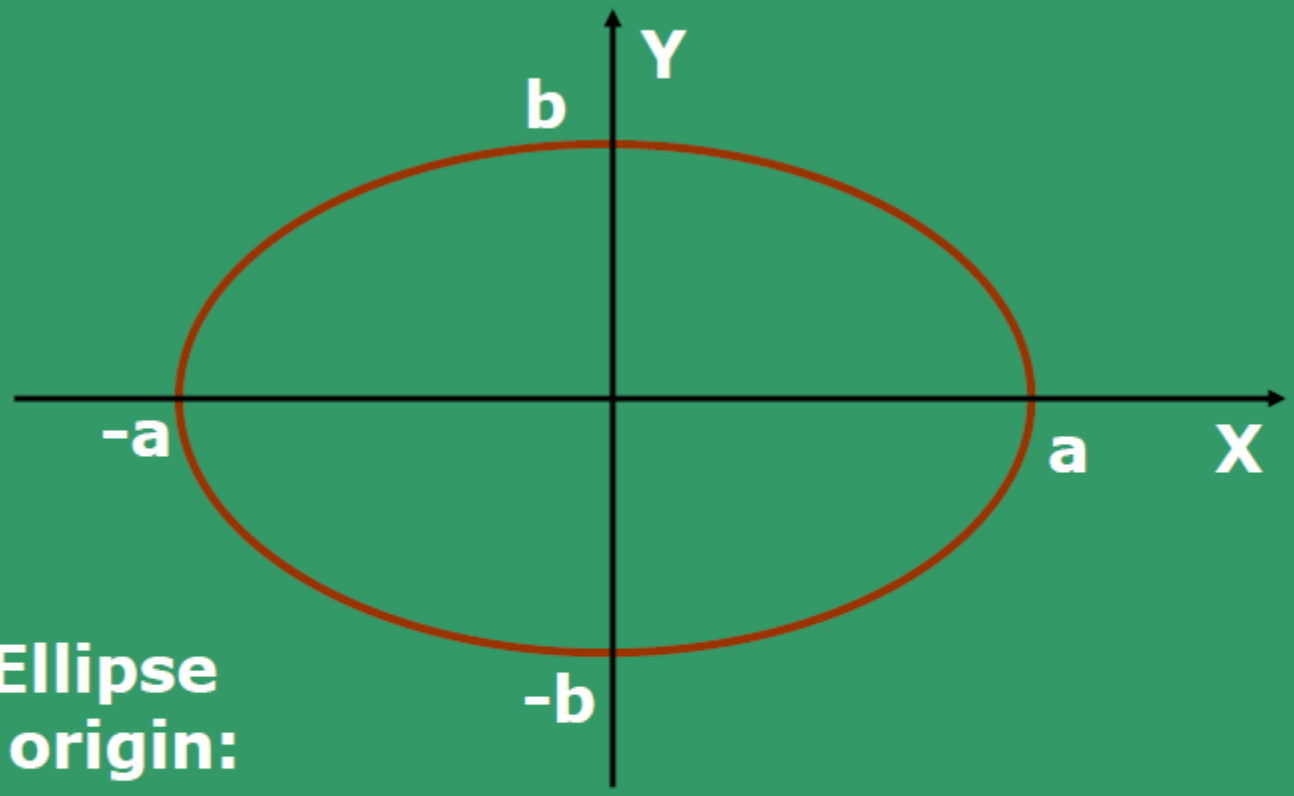
Shape with analytic forms

Bezier curve



ELLIPSE DRAWING

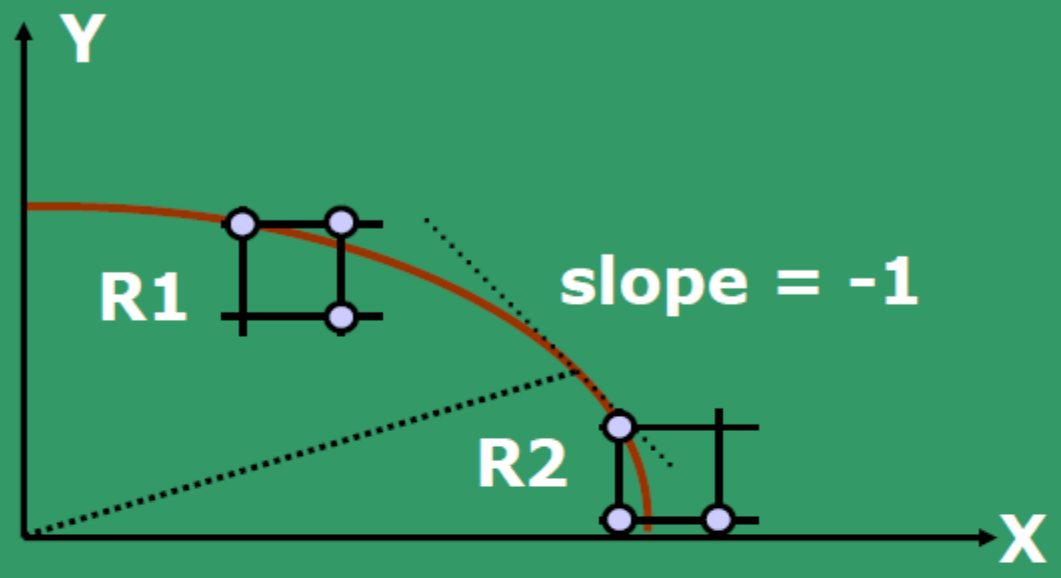
SCAN CONVERTING ELLIPSES



Equation of Ellipse centered at origin:

$$F(X,Y) = b^2X^2 + a^2Y^2 - a^2b^2 = 0$$

Length of the major axis: 2a;
and minor axis: 2b.



Draw pixels in two regions R1 and R2, to fill up the first Quadrant.

Points in other quadrants are obtained using symmetry.

We need to obtain the point on the contour where the slope of the curve is -1.

This helps to demarcate regions R1 and R2.

The choice of pixels in R1 is between E and SE, whereas in R2, it is S and SE.

$$F(X,Y) = b^2X^2 + a^2Y^2 - a^2b^2 = 0$$

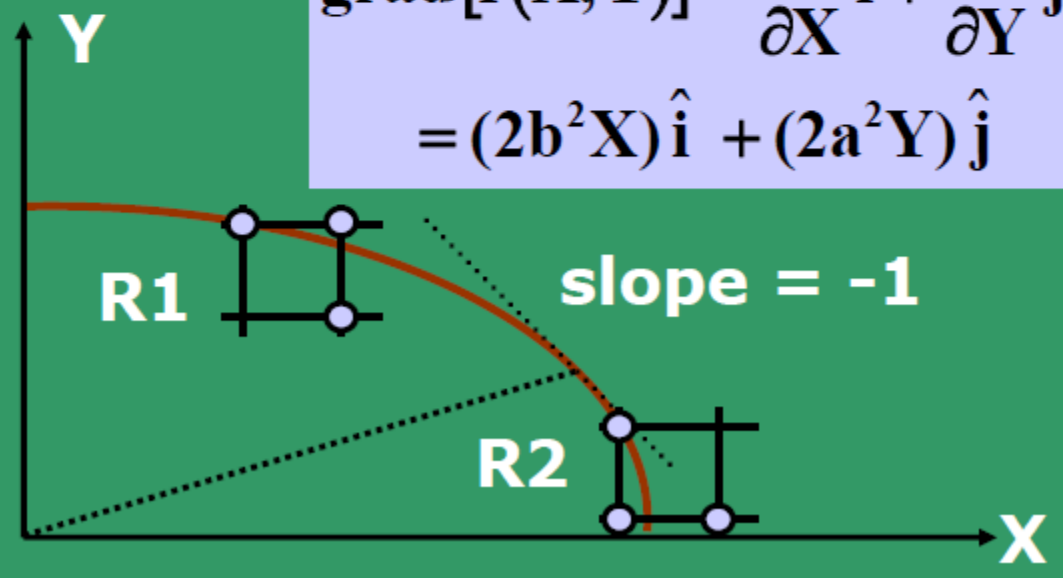
$$\text{grad}[f(X,Y)] = \frac{\partial f}{\partial X}\hat{i} + \frac{\partial f}{\partial Y}\hat{j}$$

$$= (2b^2X)\hat{i} + (2a^2Y)\hat{j}$$

$$\text{In R1}: \left|\frac{\partial f}{\partial Y}\right| > \left|\frac{\partial f}{\partial X}\right|$$

and

$$\text{in R2}: \left|\frac{\partial f}{\partial X}\right| > \left|\frac{\partial f}{\partial Y}\right|$$



At the region boundary point on the ellipse:

$$\left|\frac{\partial f}{\partial Y}\right| = \left|\frac{\partial f}{\partial X}\right|$$

Based on this condition,
we obtain the criteria when the next mid-point moves from R1 to R2 :

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

When the above condition occurs,
we switch from R1 to R2.

Analysis in region R1:

Let the current pixel be (X_p, Y_p); $d_{old} = F(M_1)$;

$$F(M_1) = d_{old} = F(X_p + 1, Y_p - 1/2)$$
$$= b^2(X_p + 1)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$

For choice E ($d<0$):

$$d_{new} = F(X_p + 2, Y_p - 1/2)$$
$$= b^2(X_p + 2)^2 + a^2(Y_p - 1/2)^2 - a^2b^2$$
$$= d_{old} + b^2(2X_p + 3);$$

Thus, $(\Delta d)_{E1} = b^2(2X_p + 3);$

For choice SE ($d \geq 0$):

$$d_{new} = F(X_p + 2, Y_p - 3/2)$$
$$= b^2(X_p + 2)^2 + a^2(Y_p - 3/2)^2 - a^2b^2$$
$$= d_{old} + b^2(2X_p + 3) + a^2(-2Y_p + 2);$$

Thus, $(\Delta d)_{SE1} = b^2(2X_p + 3) + a^2(-2Y_p + 2);$

Initial Condition:
In region R1, first point is (0, b).

$$(d_{init})_{R1} = F(1, b - 1/2) = b^2 + a^2(1/4 - b) ;$$

Problem with a fractional (floating point) value for $(d_{init})_{R1}$?

Switch to Region R2, when:

$$b^2(X_p + 1) \geq a^2(Y_p - 1/2)$$

Let the last point in R1 be (X_k, Y_k).

$$F(M_2) = F(X_k + 1/2, Y_k - 1)$$

$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$$

$$= (d_{init})_{R2}$$

$$F(M_2) = d_{old} = F(X_k + 1/2, Y_k - 1)$$

$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 1)^2 - a^2b^2$$

For choice SE ($d<0$):

$$d_{new} = F(X_K + 3/2, Y_k - 2)$$
$$= b^2(X_k + 3/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$$
$$= d_{old} + b^2(2X_k + 2) + a^2(-2Y_k + 3);$$

Thus, $(\Delta d)_{SE2} = b^2(2X_k + 2) + a^2(-2Y_k + 3);$

For choice S ($d \geq 0$):

$$d_{new} = F(X_K + 1/2, Y_k - 2)$$
$$= b^2(X_k + 1/2)^2 + a^2(Y_k - 2)^2 - a^2b^2$$
$$= d_{old} + a^2(-2Y_k + 3);$$

Thus, $(\Delta d)_{S2} = a^2(-2Y_k + 3);$

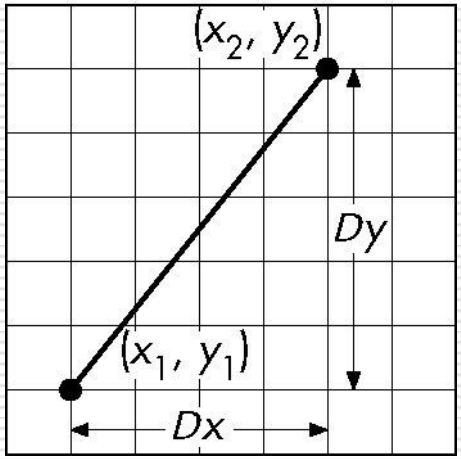
Stop iteration, when $Y_k = 0$;

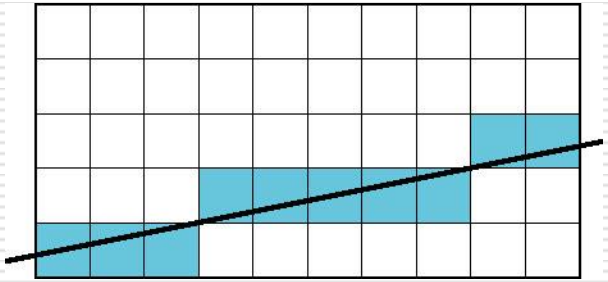
```
void MidPointEllipse (int a, int b, int value);
{
        double d2; int X = 0; int Y = 0;
        sa = sqr(a); sb = sqr(b);
        double d1 = sb – sa*b + 0.25*sa;
                EllipsePoints(X, Y, value);
        /* 4-way symmetrical pixel plotting */

        while  ( sa*(Y - 0.5) > sb*(X + 1))
                                        /*Region R1 */
        {       if (d1 < 0)             /*Select E */
                        d1 += sb*((X<<1) + 3);
                  else                  /*Select SE */
                        { d1 += sb*((X<<1) + 3) + sa*
                                (-(Y<<1) + 2); Y-- ; }
                X++ ; EllipsePoints(X, Y, value);
        }
```

```
        double d2 = sb*sqr(X + 0.5) +
                        sa*sqr(Y - 1) - sa*sb;

        while  ( Y > 0)   /*Region R2 */

        {       if (d2 < 0)          /*Select SE */
                    { d2 += sb*((X<<1) + 2) +
                              sa*(-(Y<<1) + 3);
                              X++; }

                else                 /*Select S */
                    d2 += sa*(-(Y<<1) + 3);

                Y-- ; EllipsePoints(X, Y, value);
        }
}
```

Raster displays

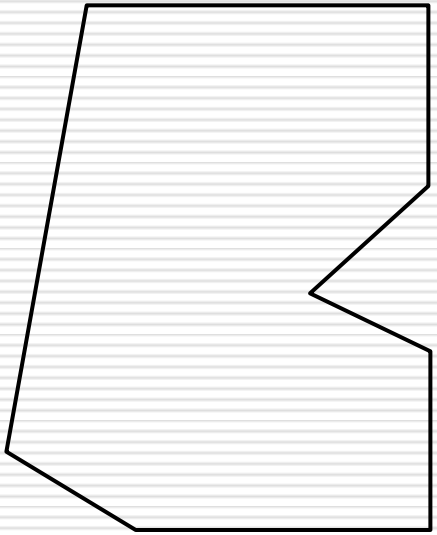
- 1D: scan conversion of line
- 2D ?

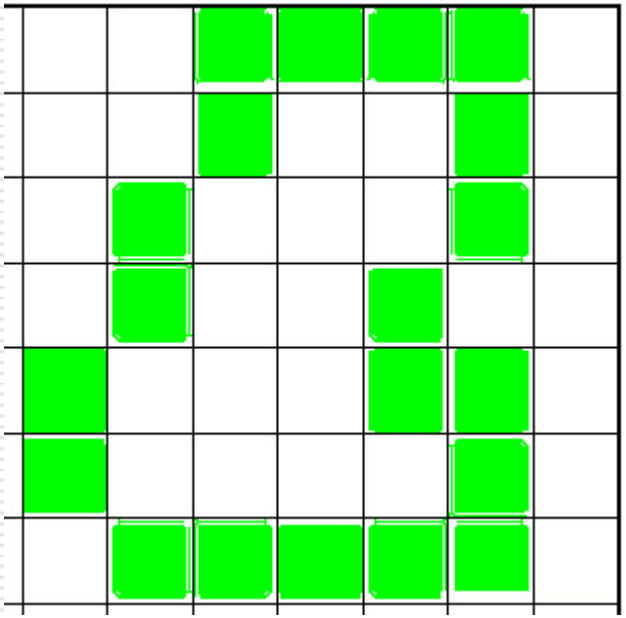




Raster displays

- 2D: Scan conversion of polygonal region



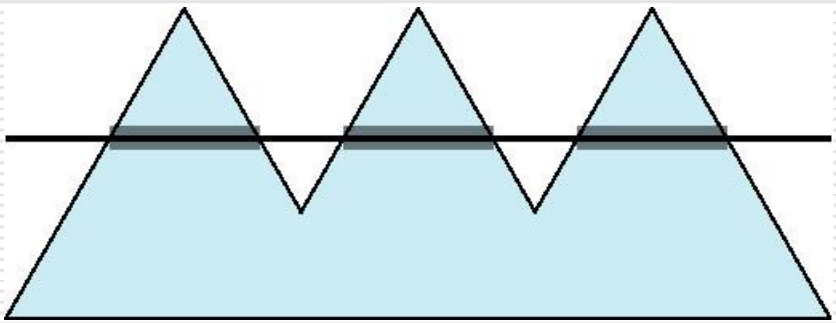


2D -- Scan line algorithm

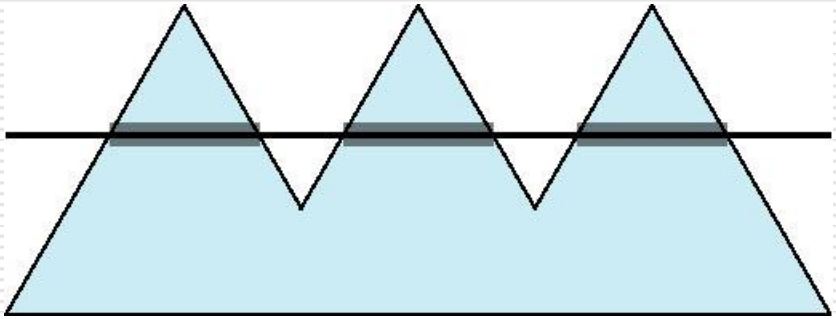
- Basic idea

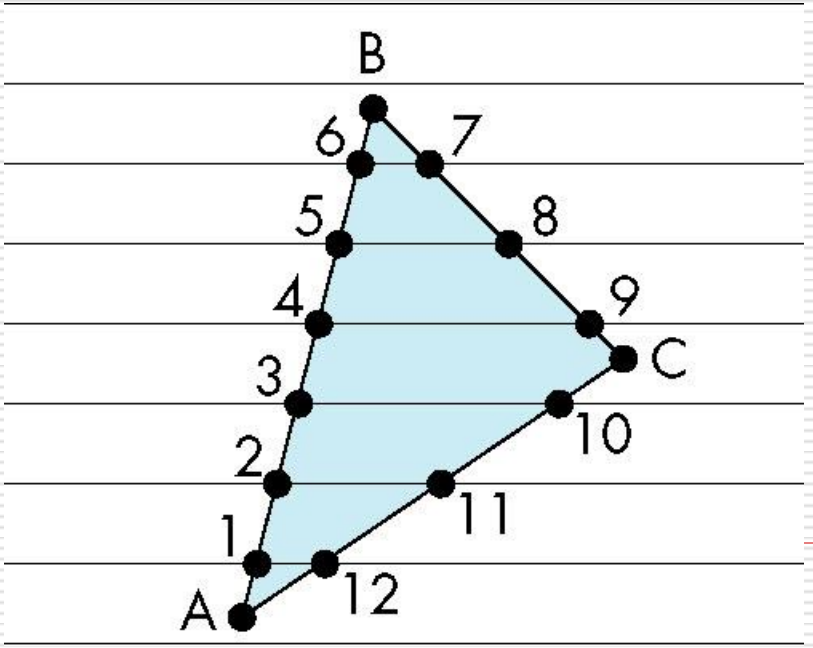
  According scan line order, calculate the intersection region of scan line and polygon, then display these region using the required color, then the work is completed.

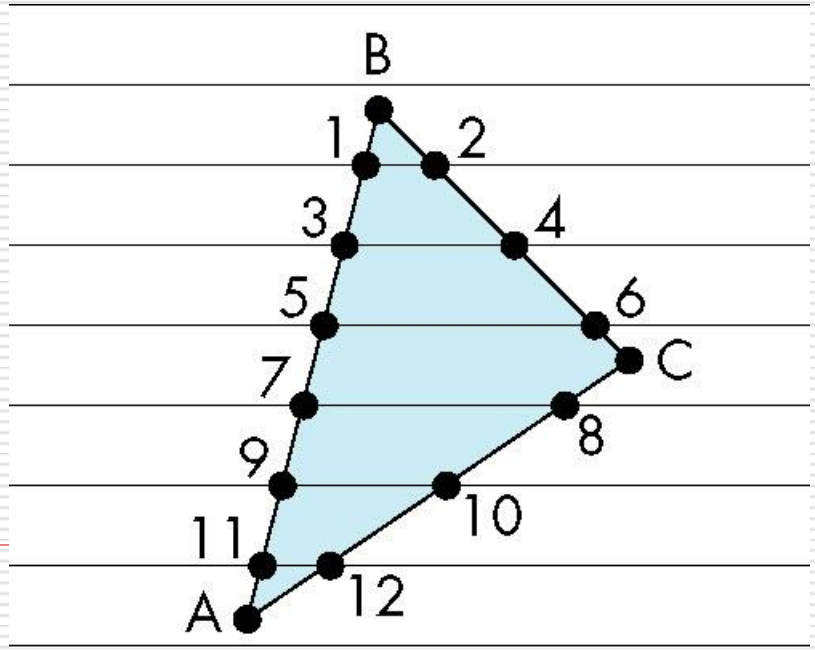
- The Scan line algorithm can be divided into four steps
  - Intersection
  - Sort
  - Pair
  - Filling



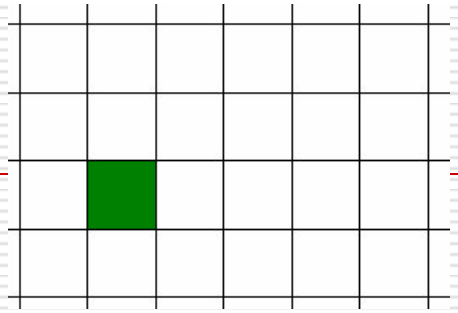
2D -- Scan line algorithm





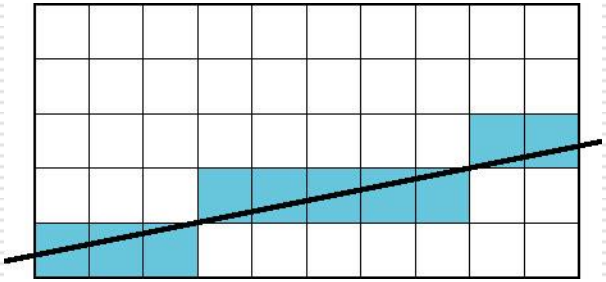


Begin OpenGL

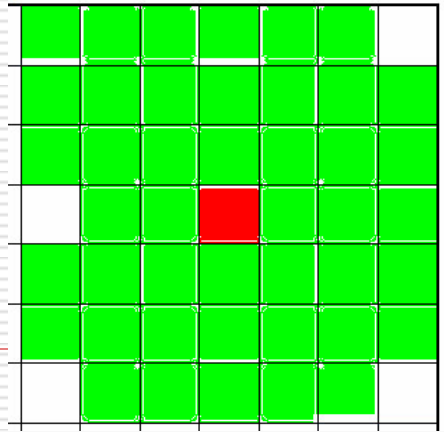


Draw point, line, face:

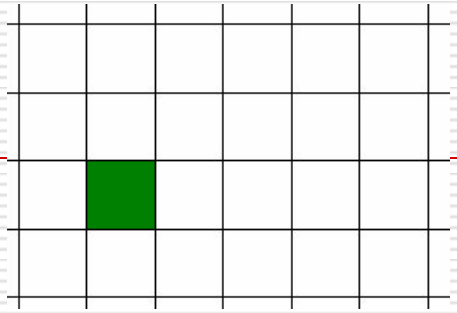
```
glBegin(parameter);

  … …

glEnd();
```



parameter: GL_POINTS, GL_LINES, GL_POLYGON, GL_TRIANGLES



Begin OpenGL

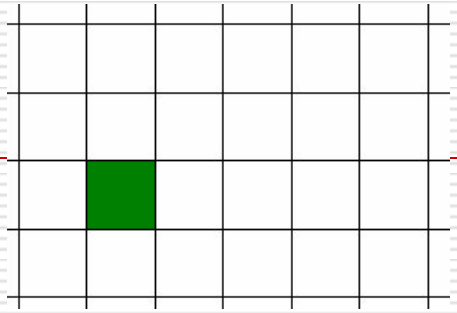


Draw points:

```
glBegin(GL_POINTS);
  glVertex3f(-0.5,-0.5,0.0);
glEnd();
glBegin(GL_POINTS);
  glVertex3f(0.0,0.5,0.0);
glEnd();
……
```

Low efficiency!

Begin OpenGL

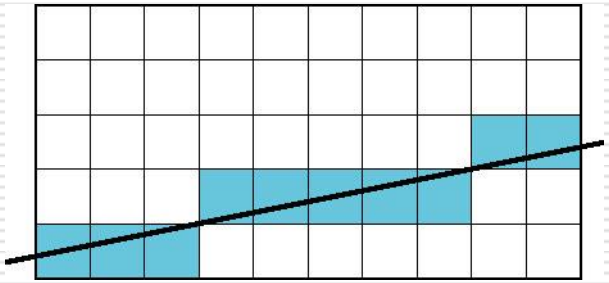


Draw points:

```
glBegin(GL_POINTS);
  glVertex3f(-0.5,-0.5,0.0);
  glVertex3f(0.5,0.0,0.0);
  glVertex3f(0.0,0.5,0.0);
  … …
glEnd();
```

Begin OpenGL

Draw lines:



Low efficiency!

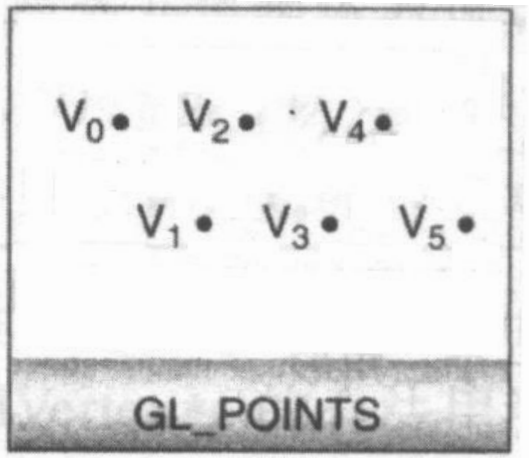
```
glBegin(GL_LINES);
  glVertex3f(-0.5,-0.5,0.0);
  glVertex3f(0.5,0.0,0.0);
glEnd();
glBegin(GL_LINES);
  glVertex3f(0.0,0.5,0.0);
  glVertex3f(0.0,0.0,0.5);
glEnd();
… …
```

Begin OpenGL

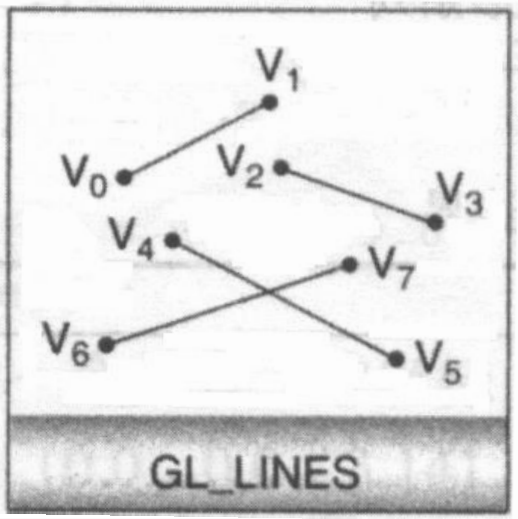
Draw all lines at one time:

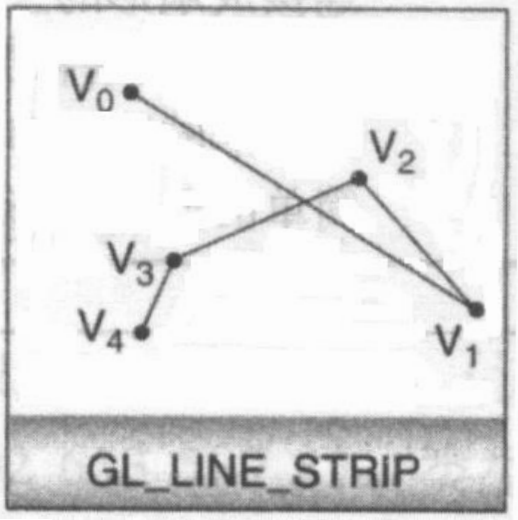
```
glBegin(GL_LINES);
  glVertex3f(-0.5,-0.5,0.0);
  glVertex3f(0.5,0.0,0.0);
  glVertex3f(0.0,0.5,0.0);
  glVertex3f(0.0,0.0,0.5);
  … …
glEnd();
```

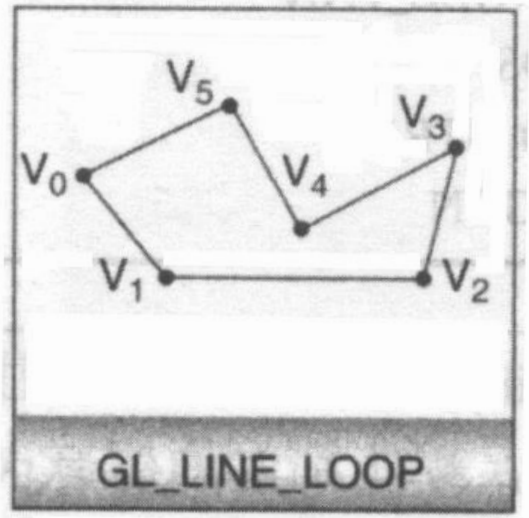
Begin OpenGL



OpenGL programming guide, 8th edtion

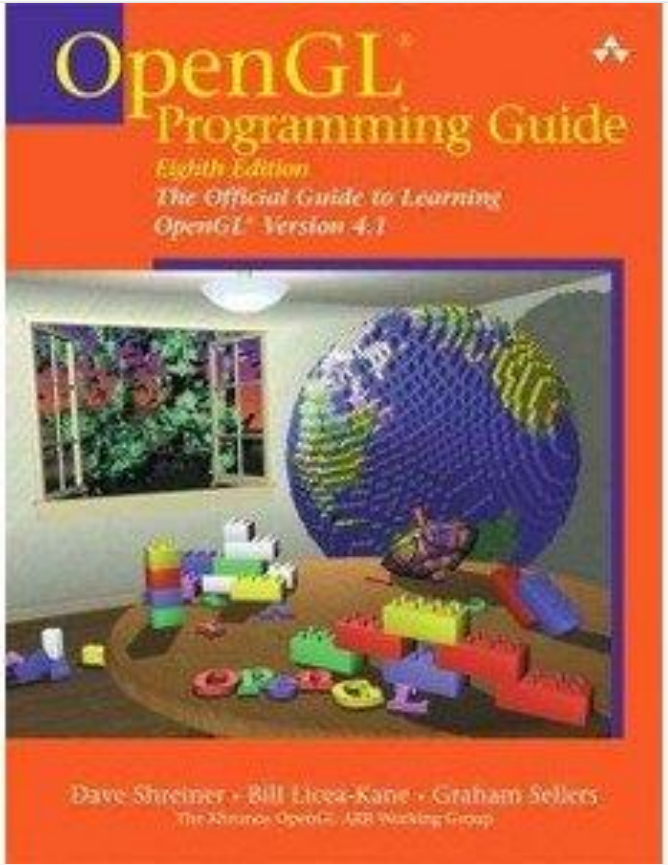






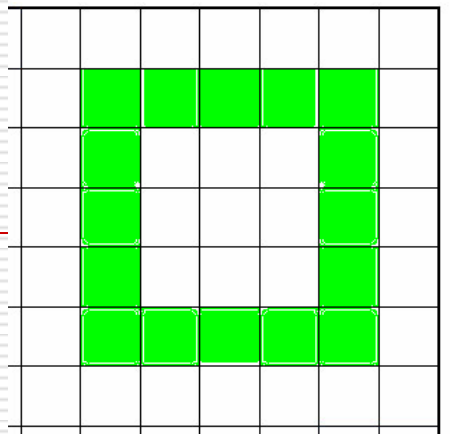
The red book

- Possibly the best resource for OpenGL is this book
- Need it for most class projects
- But do not need to buy it. There is an online version



http://www.glprogramming.com/red/

Begin OpenGL



Draw polygon:

```
glBegin(GL_POLYGON);
  glVertex3f(-0.5,-0.5,0.0);
  glVertex3f(0.5,0.0,0.0);
  glVertex3f(0.0,0.5,0.0);
  glVertex3f(0.0,0.0,0.5);
  … …
glEnd();
```

Must be convex polygon

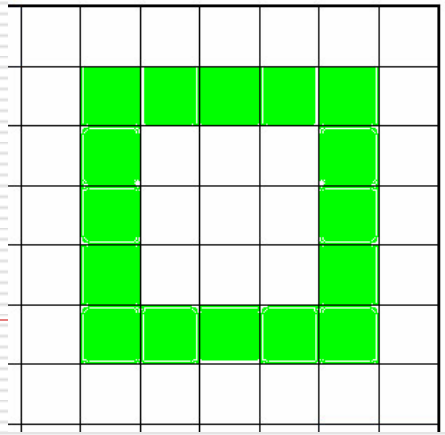
Begin OpenGL

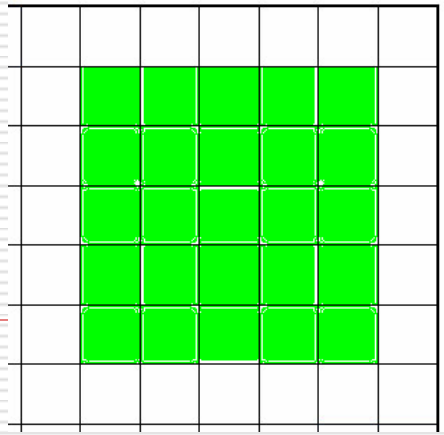
Draw polygon - Region filling:

```
glPolygonMode(parameter1, parameter2);
```

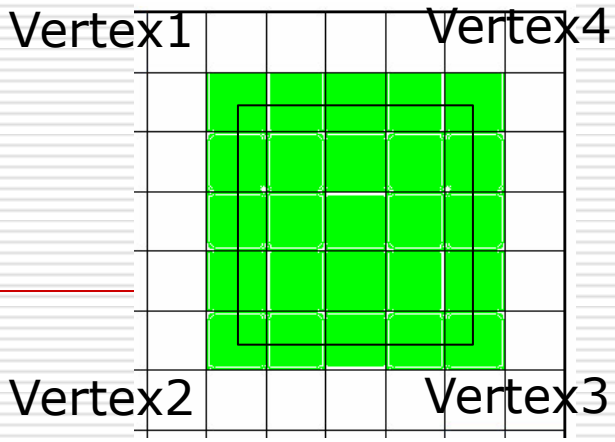
parameter2: GL_LINE, GL_FILL

parameter1: GL_FRONT, GL_BACK





Begin OpenGL



Draw polygon:

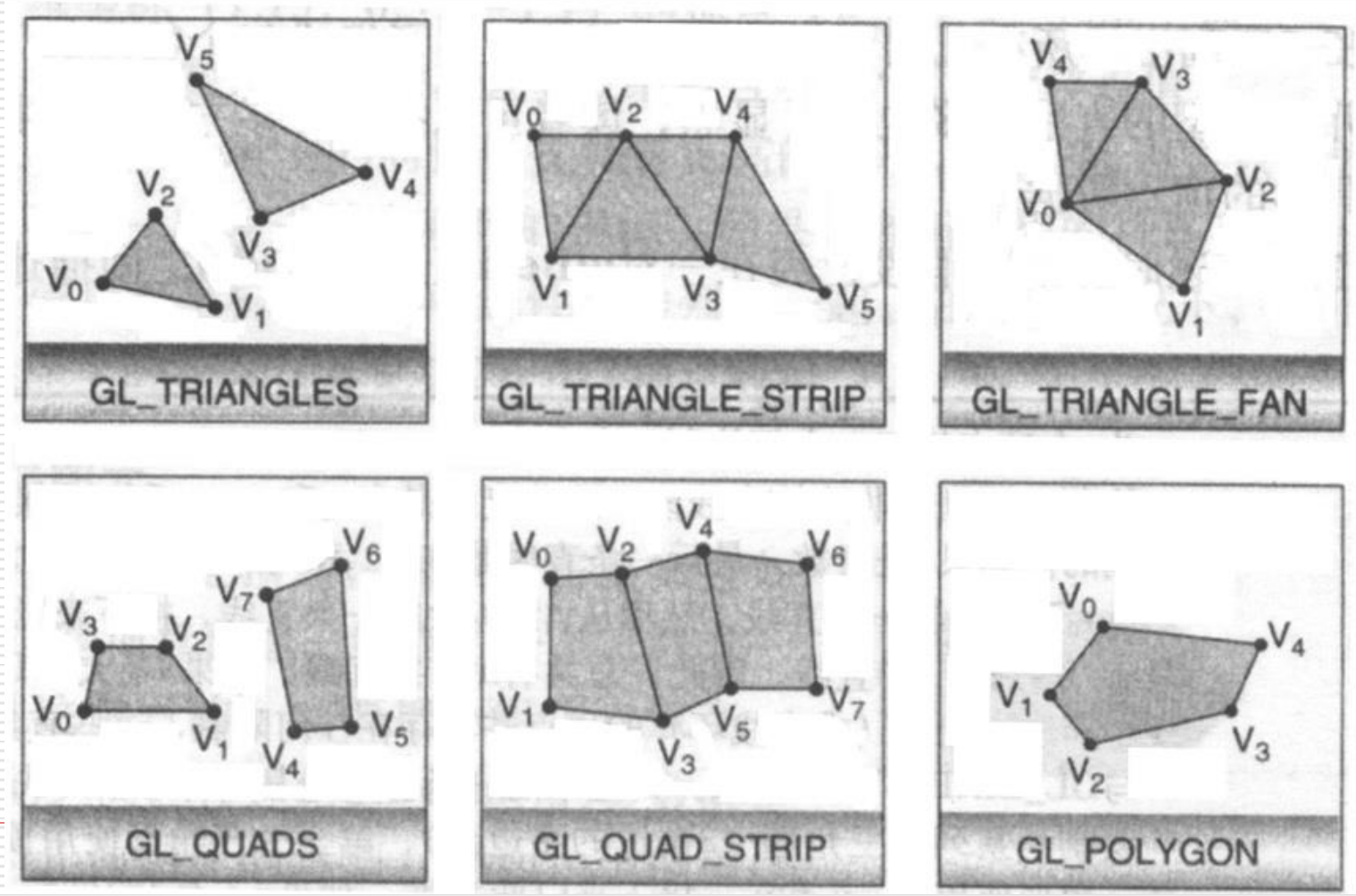
```
glPolygonMode(GL_FRONT, GL_FILL);
glBegin(GL_POLYGON);
  glVertex3f(coordinate of vertex1);
  glVertex3f(coordinate of vertex2);
  glVertex3f(coordinate of vertex3);
  glVertex3f(coordinate of vertex4);
  … …
glEnd();
```

Begin OpenGL

Draw triangle:

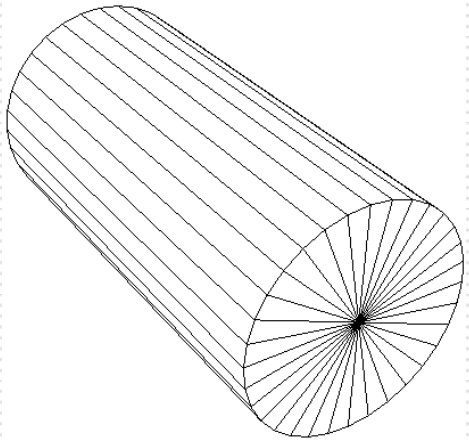
```
glPolygonMode(GL_FRONT, GL_FILL);
glBegin(GL_TRIANGLES);
  glVertex3f(coordinate of vertex1);
  glVertex3f(coordinate of vertex2);
  glVertex3f(coordinate of vertex3);
  … …
  … …
glEnd();
```

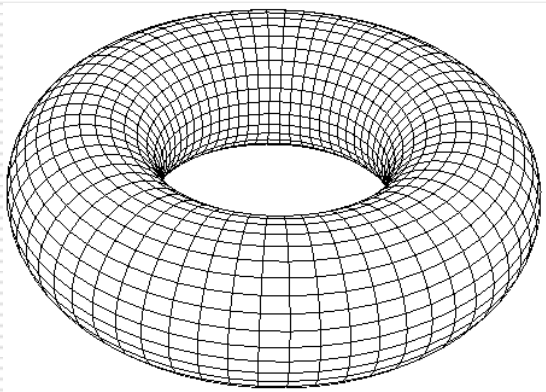
Begin OpenGL

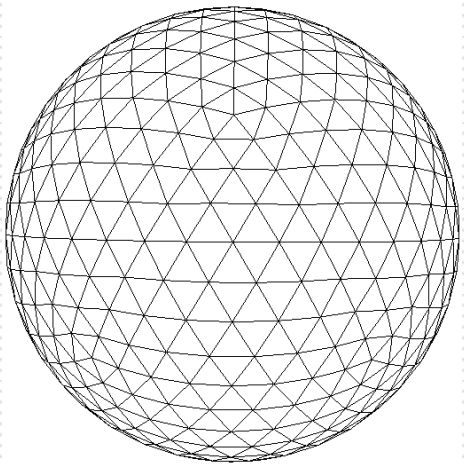


Begin OpenGL

Representing 3D shape using triangle/quad meshes

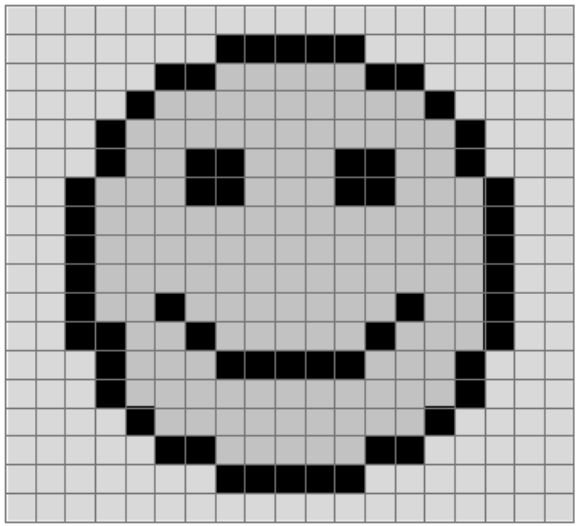


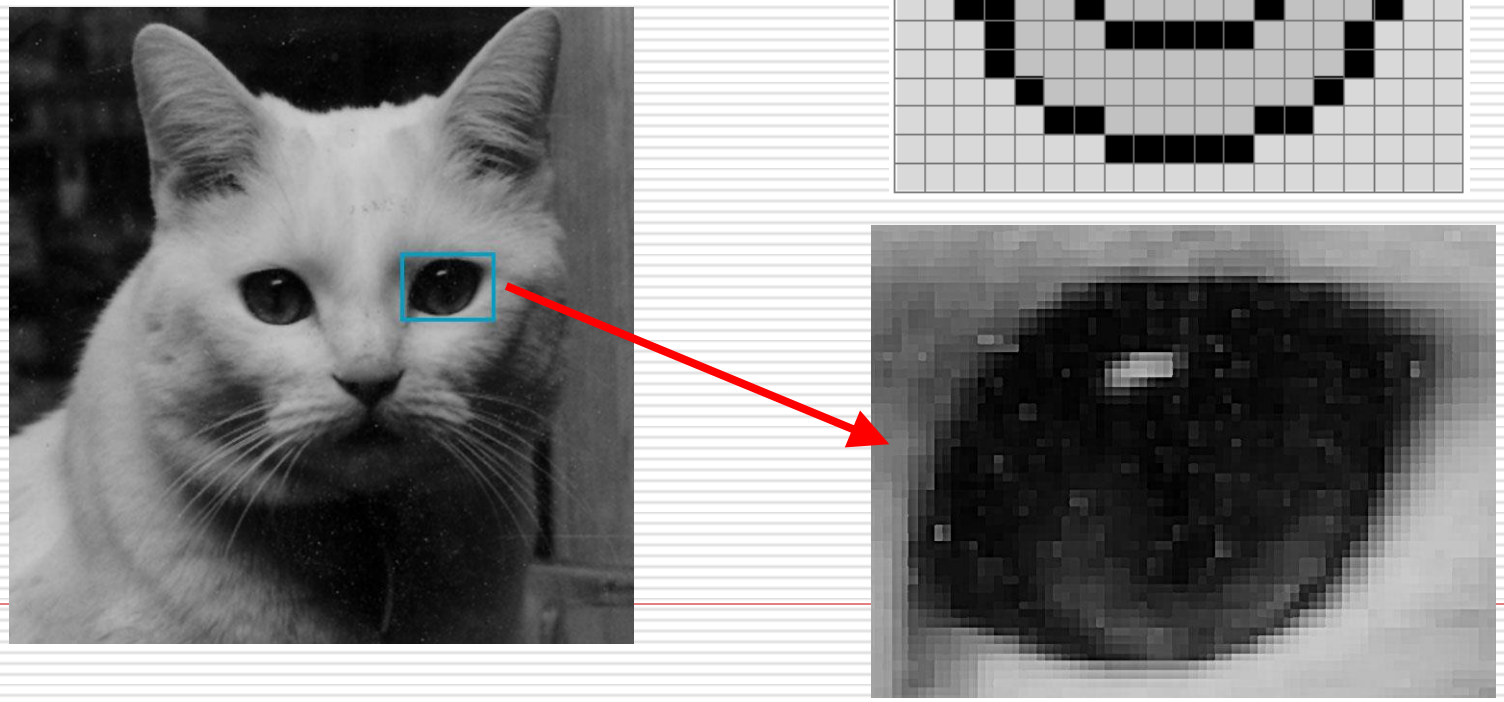




Raster displays

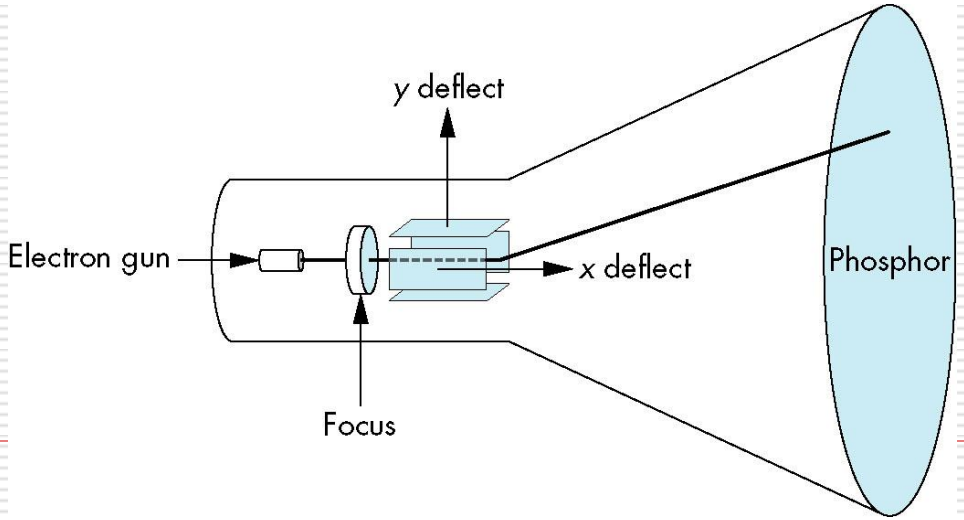
Every shape is drawn in a dot matrix

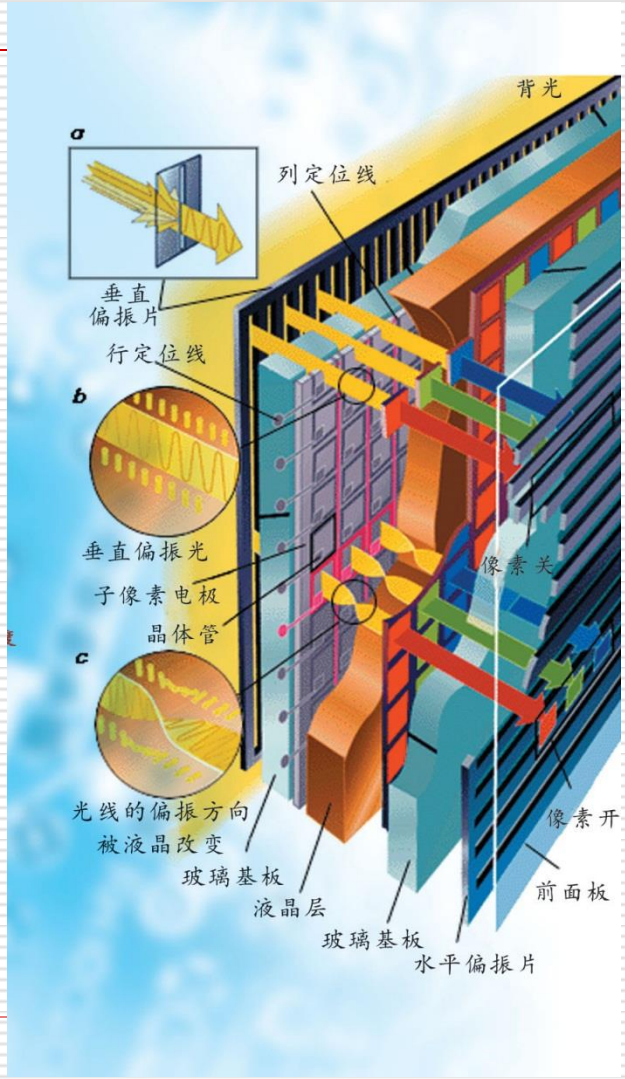




Raster displays

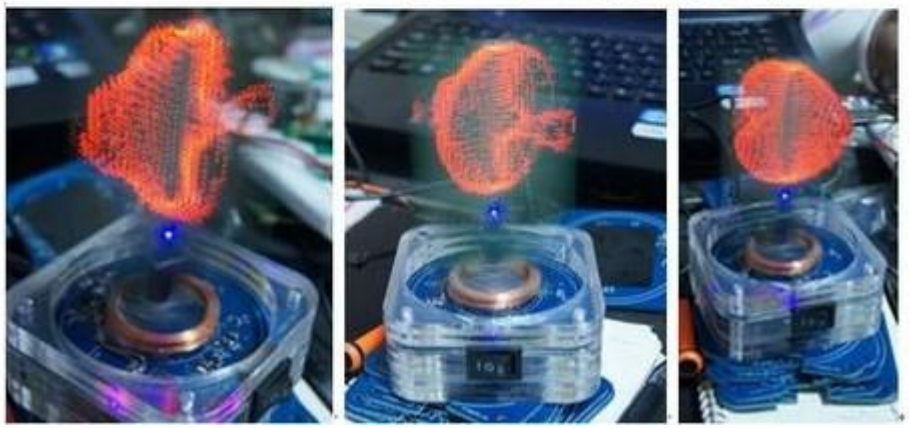
- Computer monitor
  - CRT (Cathode Ray Tube)
  - LCD (Liquid Crystal Display)
  - LED (Light Emitting Diode)

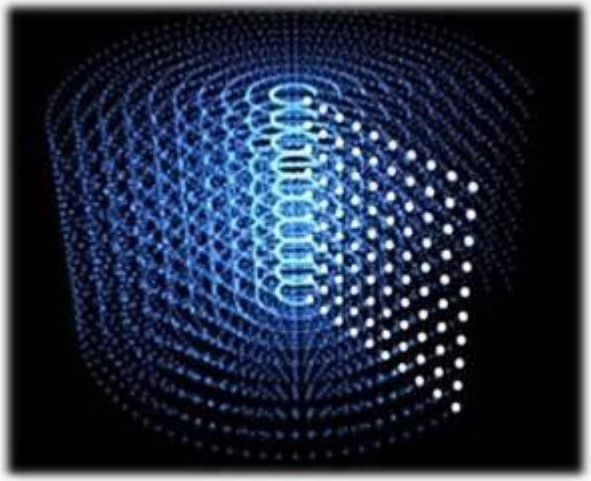




3D Raster displays

- 3D display
  - Three undergraduates
  - Department of Precision Instrument





3D Raster displays

Fundamentals of Computer Graphics

End.

Thanks