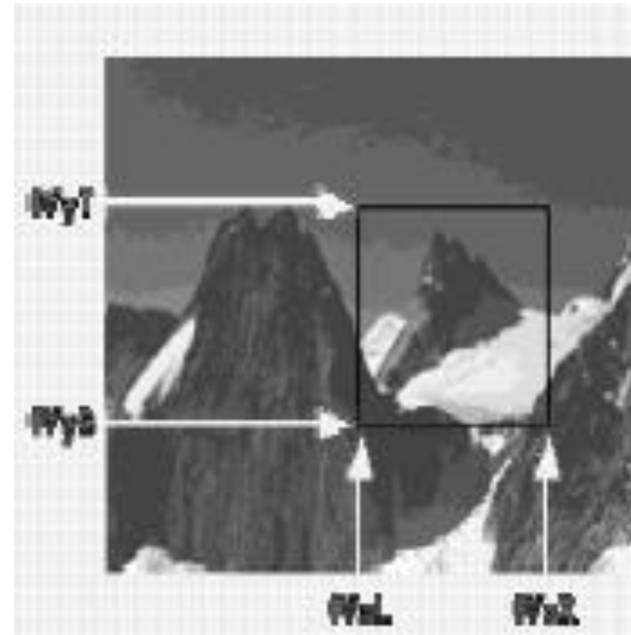
2D Graphics

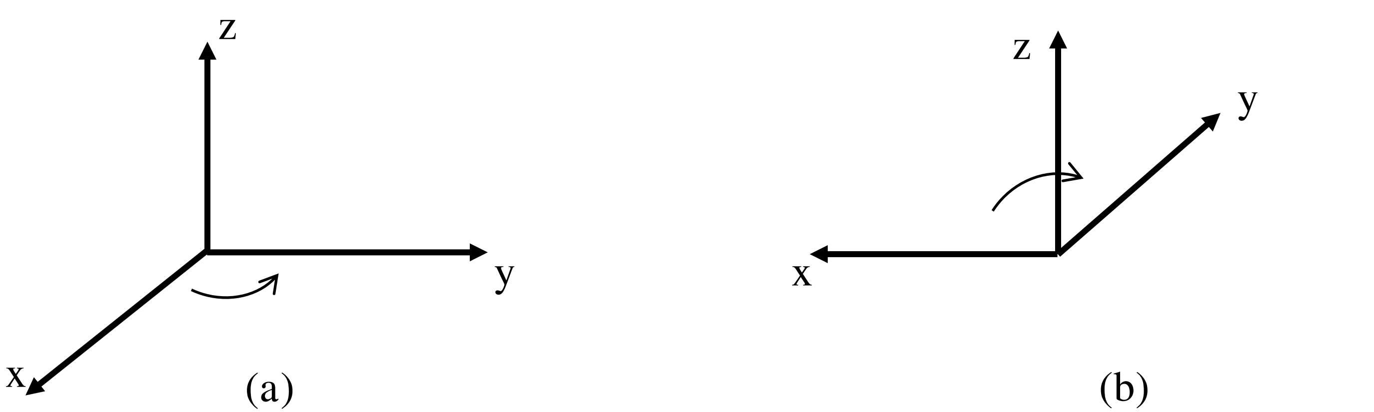
Contents

- Hệ tọa độ và chuyển đổi hệ tọa độ
- Thuật toán vẽ đối tượng đồ họa cơ sở
- Thuật toán xén hình
- Thuật toán tô màu
- Vẽ chữ và dựng Font
- Các phép biến đổi 2D

Hệ tọa độ

- Hệ toạ độ thế giới thực (WCS-World Coordinate System)
  - mô tả các đối tượng thế giới thực.
  - Đơn vị đo phụ thuộc vào không gian, kích thước của đối tượng được mô tả:
    từ nm, mm... → m, km ...

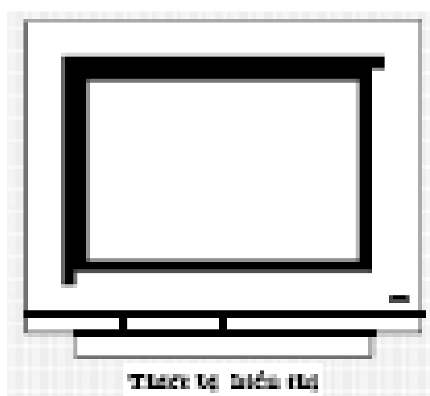


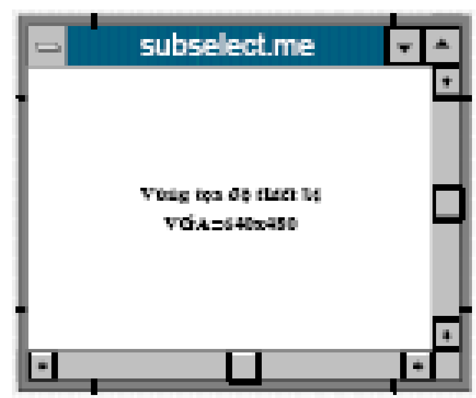


Hệ toạ độ theo quy ước bàn tay phải (a) và bàn tay trái (b)

Hệ tọa độ

- Hệ toạ độ thiết bị (DCS-Device Coordinate System)
  - Dùng trong thiết bị xuất cụ thể: máy in, màn hình …
  - Đặc điểm:
    - Toạ độ điểm (x,y) trong đó x,y $\in$ N.
    - Toạ độ (x,y) giới hạn, phụ thuộc vào từng loại thiết bị
    - Gốc toạ độ O ở góc trên trái màn hình



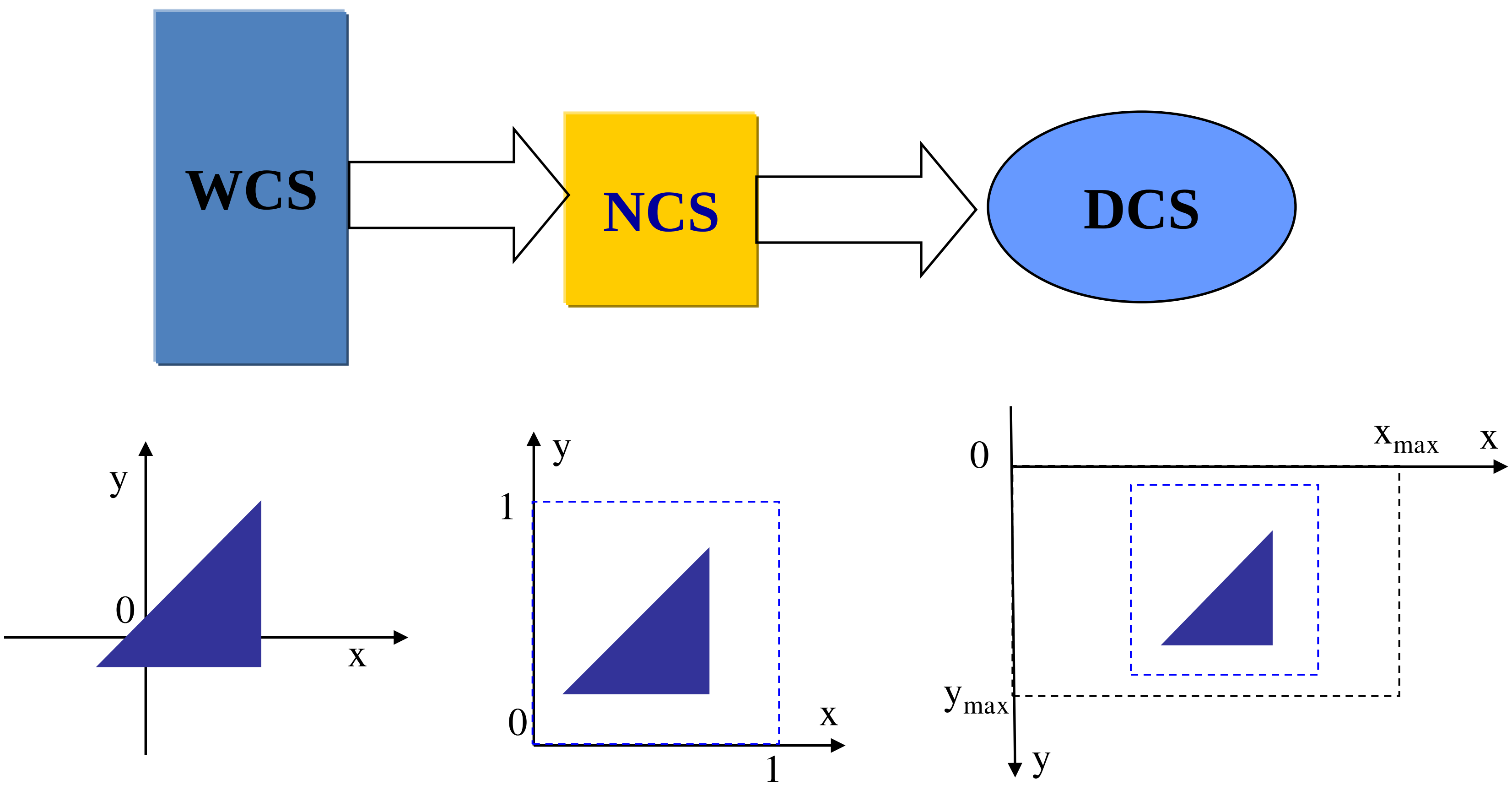


Hệ tọa độ

- Hệ toạ độ chuẩn (NCS - Normalized Coordinate System)
  - Giải quyết vấn đề ứng dụng chạy trên thiết bị khác nhau
  - $x,y \in [0,1]$.
- Các bước mô tả đối tượng thực:
  - Ảnh được định nghĩa theo các toạ độ thế giới thực
  - Chuyển từ toạ độ thế giới thực sang toạ độ chuẩn.
  - Chuyển từ toạ độ chuẩn sang toạ độ thiết bị ứng với từng thiết bị cụ thể

Hệ tọa độ

- Các bước mô tả đối tượng thực:



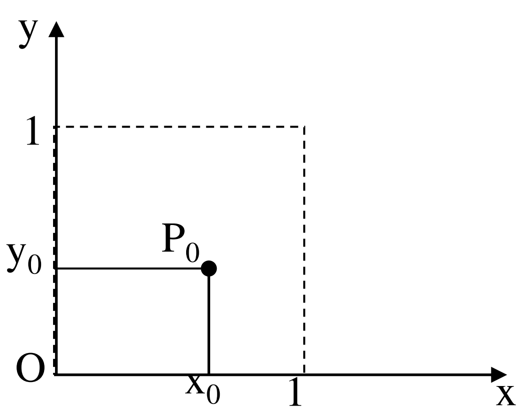
…Chuyển đổi hệ tọa độ

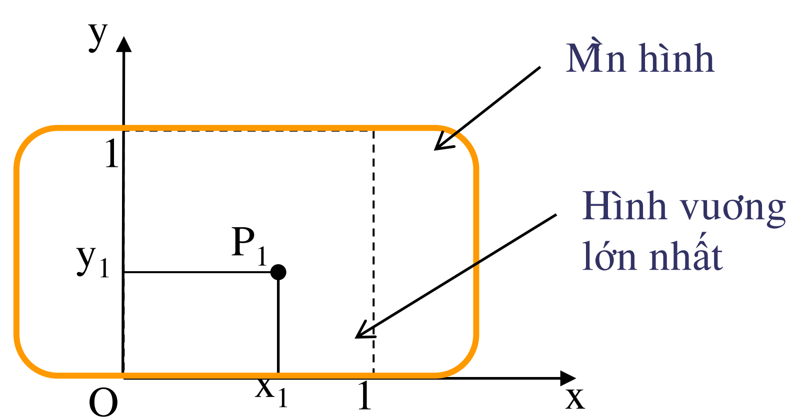
- Chuyển từ hệ toạ độ thực sang hệ toạ độ chuẩn:
  - Gọi c là cạnh hình vuông không gian lớn nhất trong hệ toạ độ thực chứa đối tượng cần hiển thị. P(x,y) ở thế giới thực được ánh xạ thành $P_0(x_0,y_0)$ trong hệ toạ độ chuẩn:

$$x_0 = x/c \qquad y_0 = y/c \qquad (x_0,y_0 \in [0,1])$$

- Chuyển từ hệ toạ độ chuẩn sang hệ toạ độ thiết bị:
  - $P_0(x_0,y_0)$ trong hệ toạ độ chuẩn được ánh xạ thành điểm $P_1(x_1,y_1)$ của hệ toạ độ thiết bị theo công thức:

$$x_1 = y_{max}x_0+(x_{max} - y_{max})/2 \qquad y_1 = y_{max}y_0$$





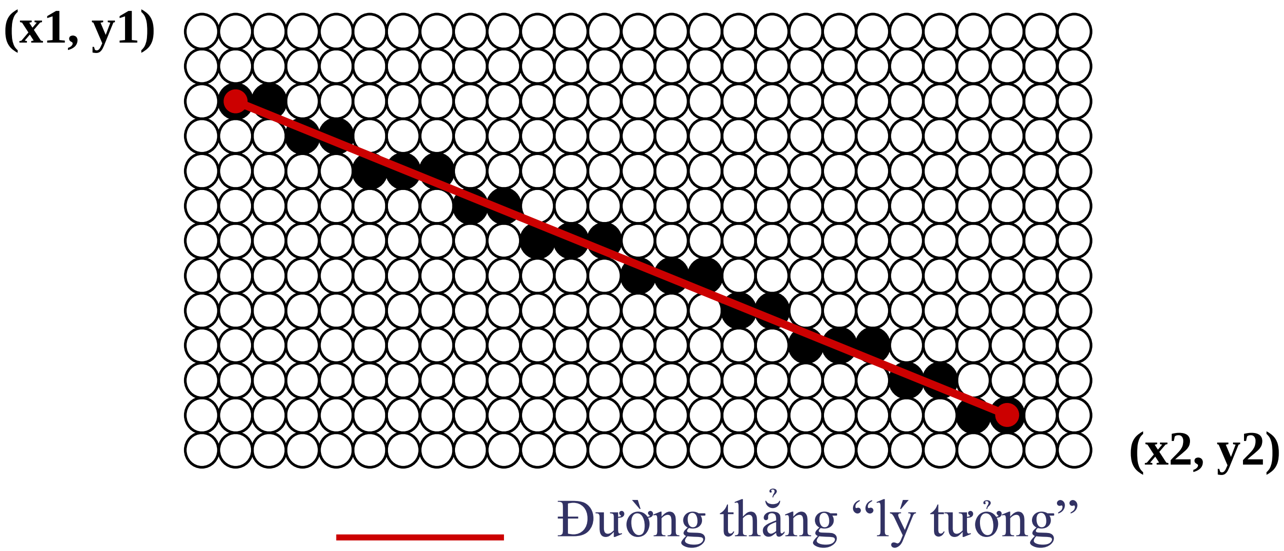
Vẽ đối tượng đồ họa cơ sở

- Thuật toán vẽ đoạn thẳng
- Thuật toán vẽ đường tròn
- Thuật toán vẽ ellipse
- Thuật toán vẽ các đường cong y=f(x)

…Vẽ đối tượng đồ họa cơ sở

Bài toán vẽ đường

- Biến đổi đường liên tục thành rời rạc (Sampling)



Yêu cầu

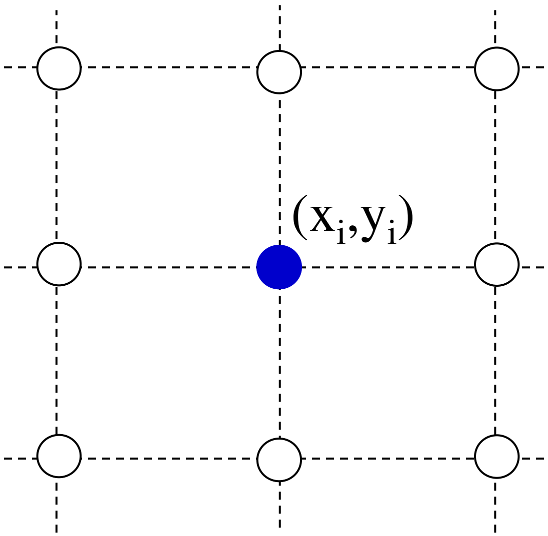
- Hình dạng liên tục, độ dày và độ sáng đều
- Các pixel gần đường "lý tưởng" được hiển thị
- Vẽ nhanh

…Vẽ đối tượng đồ họa cơ sở

Bài toán: Bước thứ i xác định được toạ độ nguyên (x_i,y_i). Tại bước i+1, điểm nguyên (x_i+1,y_i+1) xác định thế nào?

Để đối tượng hiển thị trên lưới nguyên được liền nét, các điểm (x_{i+1},y_{i+1}) có thể chọn là 1 trong 8 điểm quanh (x_i,y_i)

hay: $(x_{i+1},y_{i+1}) = (x_i \pm 1,\ y_i \pm 1)$.



Vẽ đoạn thẳng

- **Biểu diễn dạng tường minh**

  (y-y1)/(x-x1) = (y2-y1)/(x2-x1)

  hay **y = mx + b**

  - m = (y2-y1)/(x2-x1)
  - b = y1- m.x1
  - Δy = m Δx

- **Biểu diễn dạng không tường minh**

  (y2-y1)x - (x2-x1)y + x2y1 - x1y2 = 0

  hay **Ax + By + C = 0**

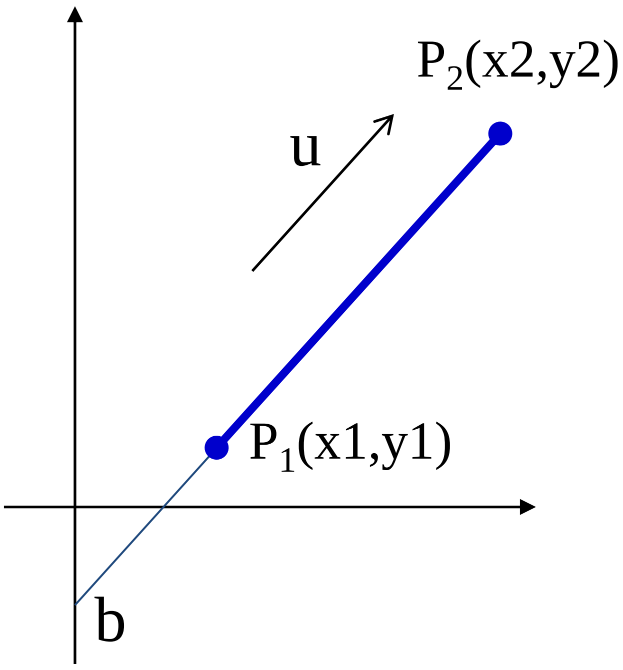
  - A = (y2-y1)
  - B = -(x2-x1)
  - C = x2y1 - x1y2

- **Biểu diễn dạng tham số**

  P(u) = P1 + u(P2 - P1) với u $\in$[0,1]

  hay X(u) = x1 + u(x2 - x1)

  Y(u) = y1 + u(y2 - y1)



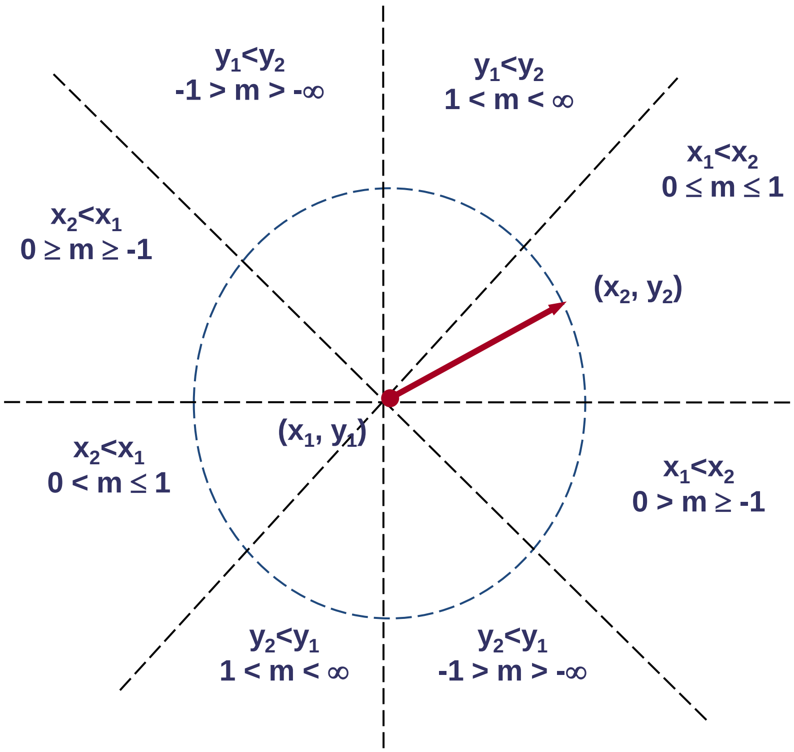
Vẽ đoạn thẳng

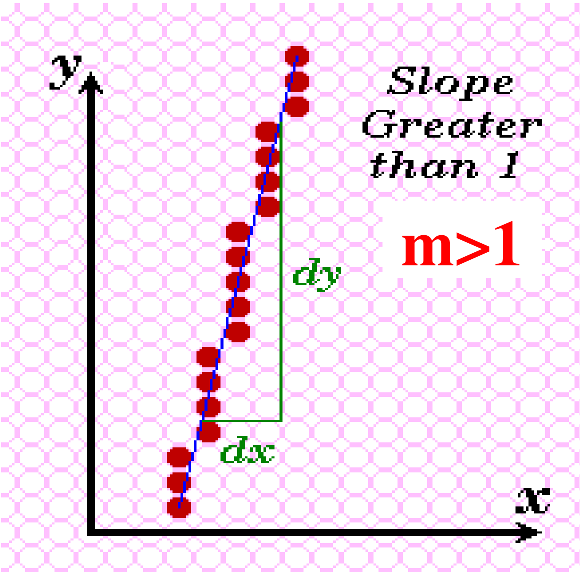
- Bài toán: Vẽ đoạn thẳng qua 2 điểm (x_1,y_1) và (x_2,y_2)
- Giải pháp
  - Cho tọa độ x (hoặc y) biến đổi mỗi lần 1 đơn vị:

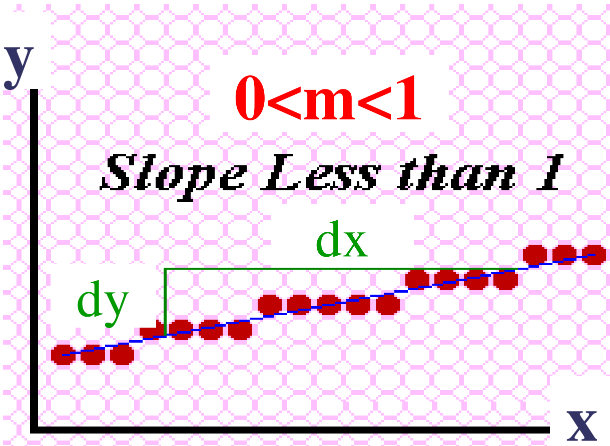
$$x_{i+1}=x_i+1 \quad (\text{hoặc } y_{i+1}=y_i+1)$$

  - Tính tọa độ nguyên y (hoặc x) sao cho gần với toạ độ thực nhất.
- Việc quyết định chọn x hay y biến đổi phụ thuộc vào độ dốc (hệ số góc *m*) của đường thẳng

Vẽ đoạn thẳng







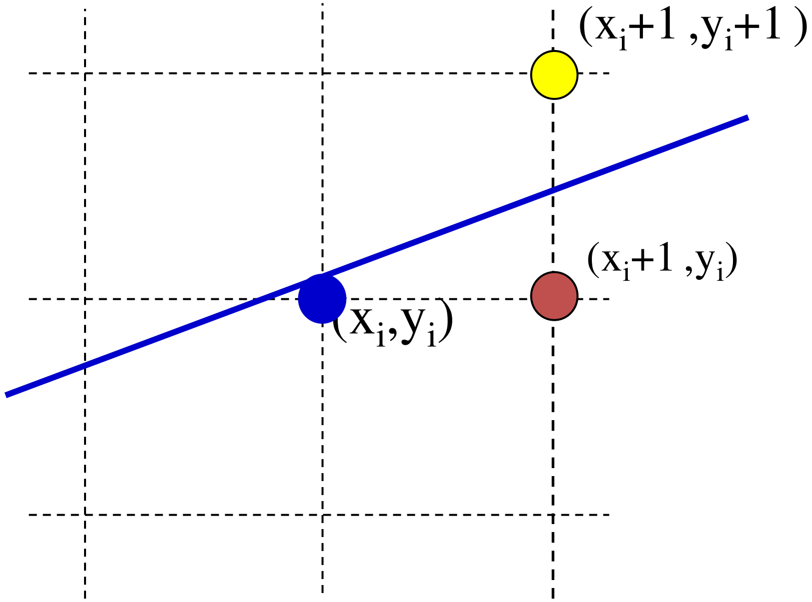
Nếu |dx| > |dy| : y=f(x)

Nếu |dx| < |dy| : $x=f^{-1}(y)$

Vẽ đoạn thẳng

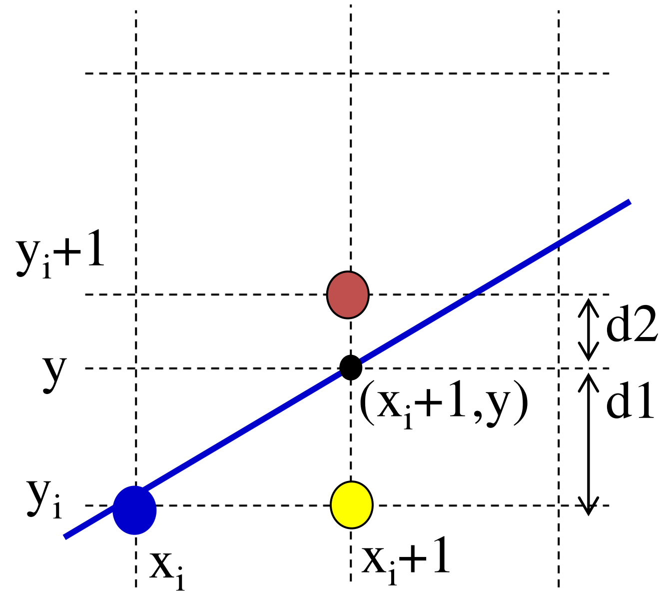
- Xét đoạn thẳng có hệ số góc 0<m<1.
  - Giả sử (x_i,y_i) là điểm đã xác định được ở bước thứ i
  - Điểm cần chọn (x_{i+1},y_{i+1}) ở bước thứ i+1 sẽ là:

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \{y_i , y_i + 1\} \end{cases}$$



Vẽ đoạn thẳng - Thuật toán Bresenham

- Gọi $(x_i+1,y) = (x_i+1,m(x_i+1)+b)$ là điểm thuộc đường thẳng
- Bước i+1, chọn (x_{i+1},y_{i+1}) phụ thuộc vào d1 va d2 hay dấu d1-d2.
  - Nếu d1< d2 , chọn $y_{i+1}= y_i$
  - Nếu d1>= d2 , chọn $y_{i+1} = y_i + 1$



- Có: $d1=y - y_i \quad =m(x_i + 1) + b - y_i$

  $d2=(y_i+1)-y =y_i +1-m(x_i + 1)-b$

- Gọi $D = d1-d2 = 2mx_i-2y_i+2m+2b-1$
- Trong biểu thức D, m là số thực nên việc tính toán với biểu thức này khá chậm. Do đó, thay vì xét dấu D ta xét dấu p_i

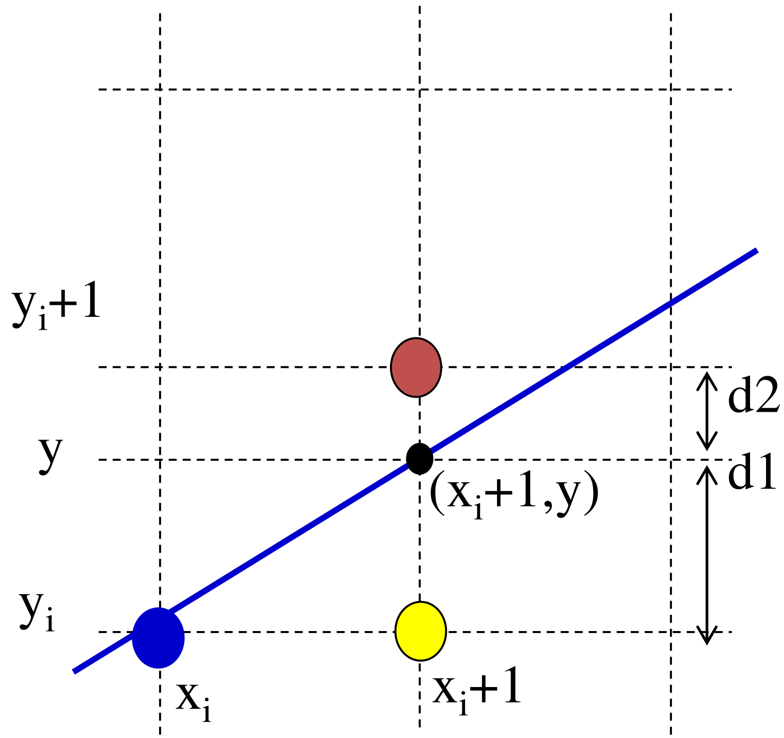
Vẽ đoạn thẳng - Thuật toán Bresenham

- p_i = Dx(d1-d2)   (vì Dx >0).

- Thay m=Dy/Dx, ta có:

  $p_i = 2Dyx_i - 2Dxy_i + c$

  với c = 2Dy+(2b-1)Dx



- Vậy:
  - Nếu p_i<0   hay d1<d2,

    chọn $y_{i+1} = y_i$
  - Neáu p_i>=0 hay d1>=d2,

    chọn $y_{i+1} = y_i + 1$

Vẽ đoạn thẳng - Thuật toán Bresenham

- Xét $p_{i+1} - p_i = (2Dyx_{i+1} - 2Dxy_{i+1} + c) - (2Dyx_i - 2Dxy_i + c)$

  $= 2Dy - 2Dx(y_{i+1} - y_i)$ (do $x_{i+1} = x_i + 1$)
- Vậy $p_{i+1} = p_i + 2Dy - 2Dx(y_{i+1} - y_i)$
- Suy ra cách tính p_{i+1} từ p_i như sau:
  - Nếu $p_i < 0$, $p_{i+1} = p_i + 2Dy$, do $y_{i+1} = y_i$
  - Nếu $p_i >= 0$, $p_{i+1} = p_i + 2(Dy - Dx)$, do $y_{i+1} = y_i + 1$
- Giá trị p_1 tại điểm vẽ đầu tiên (x_1, y_1):

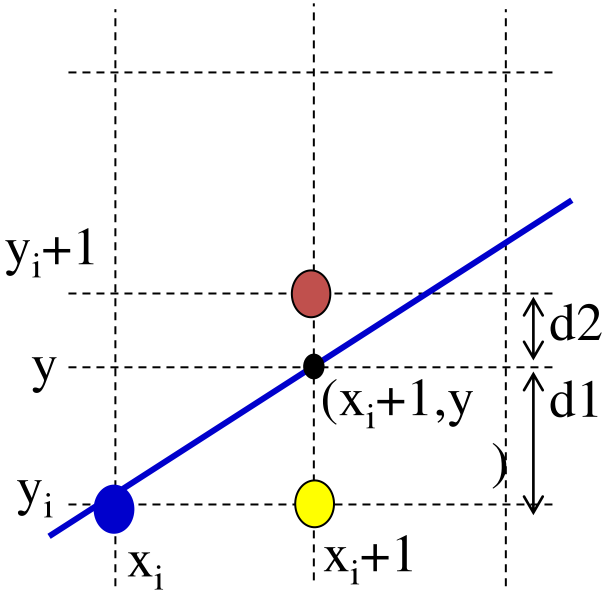
  $p_1 = 2Dyx_1 - 2Dxy_1 + c = 2(Dyx_1 - Dx(y_1 - b)) + 2Dy - Dx$

  $= 2Dy - Dx$. (Do $y_1 = mx_1 + b = (Dy/Dx)x_1 + b \Rightarrow Dyx_1 = Dx(y_1 - b)$)

…Vẽ đối tượng đồ họa cơ sở

Vẽ đoạn thẳng - Thuật toán Bresenham

```
void        LineBres(int x1,int y1,int x2,int y2){
            int   Dx,Dy,P,x,y,const1,const2;
            Dx=x2-x1;          Dy=y2-y1;
            const1=2*Dy;       const2=2*(Dy-Dx);
            P=2*Dy-Dx;
            x=x1;              y=y1;
            while (x<=x2){
                    putpixel(x,y,MAGENTA);
                    if (P<0)  P+=const1;
                    else{
                            P+=const2;
                            y++;
                    }
                    x++;
            }
}
```



Vẽ đoạn thẳng - Thuật toán MidPoint

- Phương trình tổng quát cuả đường thẳng:

Ax+By+C=0

Với $A=y_2-y_1=Dy$; $B=-(x_2-x_1)=-Dx$; $C=x_2y_1-x_1y_2$

- Với F(x,y)=Ax+By+C, ta có:

$$F(x,y)\begin{cases} <0 & \text{nếu } (x,y) \text{ nằm phía trên đường thẳng} \\ =0 & \text{nếu } (x,y) \text{ thuộc về đường thẳng} \\ >0 & \text{nếu } (x,y) \text{ nằm phía dưới đường thẳng} \end{cases}$$

Vẽ đoạn thẳng - Thuật toán MidPoint

- Chọn y_{i+1} là y_i hay y_i+1 bằng cách so sánh điểm thực $Q(x_i+1,y)$ với điểm MidPoint $M(x_i+1, y_i+1/2)$ là trung điểm của $S(x_i+1,y_i)$ và $P(x_i+1, y_i+1)$.

- Nếu Q nằm dưới điểm M, chọn S.

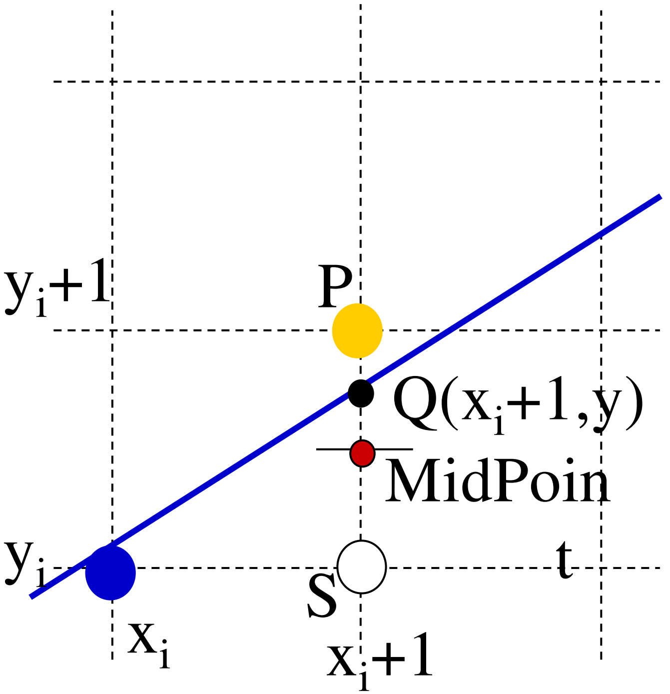
  Ngược lại, chọn P

- Chọn S hay P dựa vào dấu p_i:

  $p_i = 2F(\text{MidPoint}) = 2F(x_i+1, y_i+1/2)$

  Nếu $p_i<0$ ⇨Chọn S, tức là $y_{i+1}=y_i$

  Nếu $p_i \geq 0$ ⇨Chọn P, tức là $y_{i+1}=y_i+1$



Vẽ đoạn thẳng - Thuật toán MidPoint

- Mặt khác:

$$p_{i+1} - p_i = 2F(x_{i+1}+1, y_{i+1}+1/2) - 2F(x_i+1, y_i+1/2)$$
$$= 2[A(x_{i+1}+1)+B(y_{i+1}+1/2)+C] - 2[A(x_i+1)+B(y_i+1/2)+C]$$
$$= 2A+2B(y_{i+1}-y_i) = 2Dy-2Dx(y_{i+1}-y_i)$$

⇨ $p_{i+1} = p_i + 2Dy-2Dx(y_{i+1}-y_i)$

- Vậy:
  - Nếu $p_i<0$, $y_{i+1}=y_i$ thì $p_{i+1} = p_i + 2Dy$
  - Nếu $p_i \geq 0$, $y_{i+1}=y_i+1$ thì $p_{i+1} = p_i + 2Dy-2Dx$
- Ta tính giá trị p_1 ứng với điểm ban đầu (x_1,y_1):

$$p_1 = 2F(x_1+1, y_1+1/2) = 2(Ax_1+By_1+C)+2A+B$$
$$=2A+B \quad \text{(do } Ax_1+By_1+C=0)$$
$$=2Dy-Dx$$

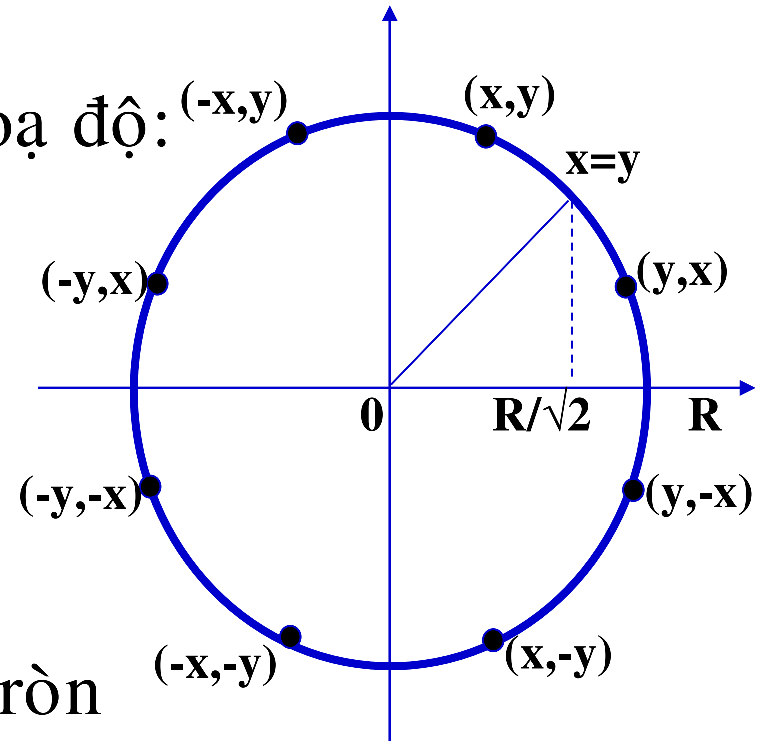
Vẽ đường tròn

- Phương trình đường tròn có dạng:

$$(x-x_c)^2 + (y-y_c)^2 = r^2$$

- Pt đường tròn có tâm ở gốc toạ độ:

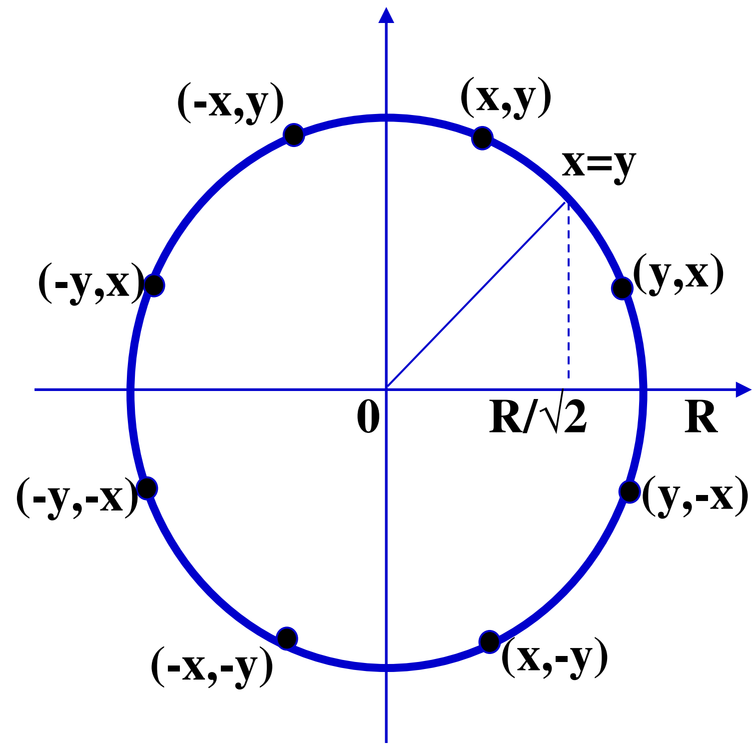
$$x^2 + y^2 = r^2$$



- Do tính đối xứng của đường tròn nên ta chỉ cần vẽ cung **1/4** hoặc **1/8**

Vẽ đường tròn

```
void put8pixel(int xc,int yc, int x, int y)
{
        putpixel( x+xc, y+yc,color);
        putpixel( y+xc, x+yc,color);
        putpixel( y+xc,-x+yc,color);
        putpixel( x+xc,-y+yc,color);
        putpixel(-x+xc,-y+yc,color);
        putpixel(-y+xc,-x+yc,color);
        putpixel(-y+xc, x+yc,color);
        putpixel(-x+xc, y+yc,color);
}
```



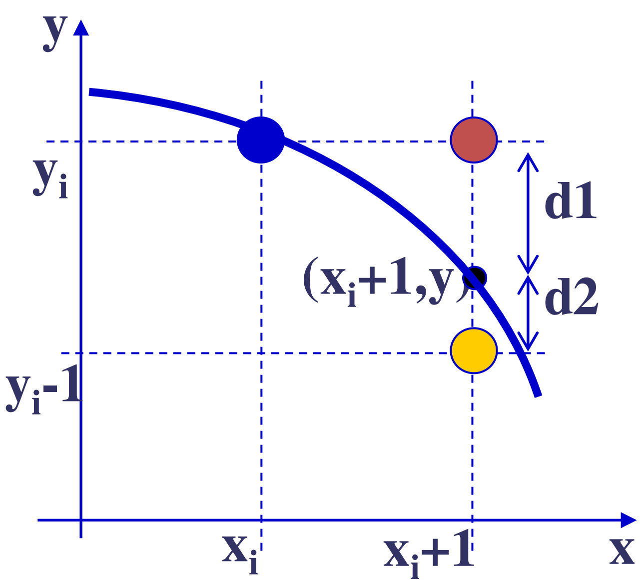
Vẽ đường tròn -Thuật toán Bresenham

- Giả sử tại bước i đã vẽ được điểm (x_i,y_i)
- Điểm cần vẽ kế tiếp (x_{i+1},y_{i+1}) là:

$$(x_i+1,y_i) \text{ hay } (x_i+1,y_i-1).$$

- Giá trị y thực sự thuộc đường tròn ứng với x_i là:

$$y^2 = r^2 - (x_i+1)^2$$



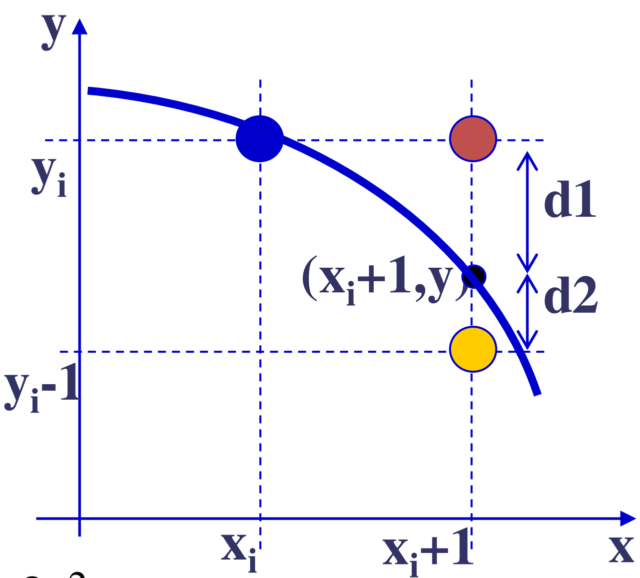
- Gọi $d1 = y_i^2 - y^2$

$$= y_i^2 - r^2 + (x_i+1)^2$$

$$d2 = y^2 - (y_i-1)^2$$

$$= r^2 - (x_i+1)^2 - (y_i-1)^2$$

Vẽ đường tròn -Thuật toán Bresenham



- $p_i = d1 - d2$

  $= y_i^2 - r^2 + (x_i+1)^2 - r^2 + (x_i+1)^2 + (y_i-1)^2$

  $= 2(x_i+1)^2 + y_i^2 + (y_i-1)^2 - 2r^2$

- $p_{i+1} - p_i = 2(x_{i+1}+1)^2 + y_{i+1}{}^2 + (y_{i+1}-1)^2 - 2r^2$

  $-2(x_i+1)^2 - y_i{}^2 - (y_i-1)^2 + 2r^2$

  $= 4x_i + 6 + 2(y_{i+1}{}^2 - y_i{}^2) - 2(y_{i+1} - y_i)$

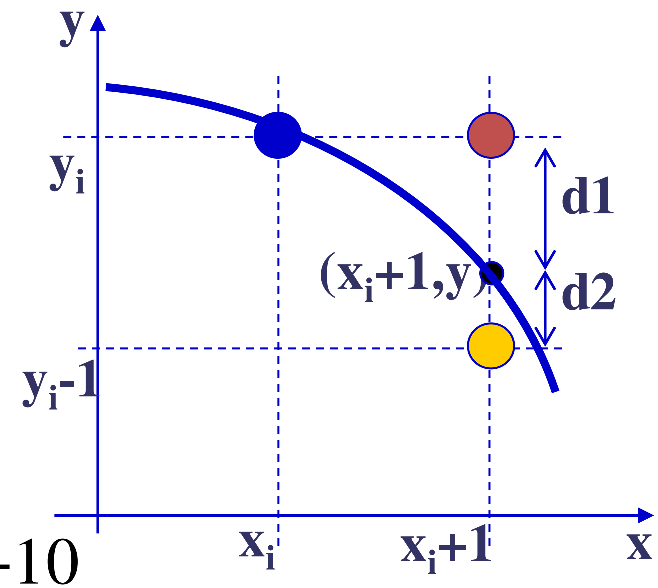
- $\Rightarrow p_{i+1} = p_i + 4x_i + 6 + 2(y_{i+1}{}^2 - y_i{}^2) - 2(y_{i+1} - y_i)$

Vẽ đường tròn -Thuật toán Bresenham

- $p_{i+1} = p_i + 4x_i + 6 + 2(y_{i+1}^2 - y_i^2) - 2(y_{i+1} - y_i)$
- Vậy:
  - Nếu $p_i < 0$ thì $y_{i+1} = y_i$

    khi đó $p_{i+1} = p_i + 4x_i + 6$
  - Nếu $p_i \geq 0$ thì $y_{i+1} = y_i - 1$

    khi đó $p_{i+1} = p_i + 4(x_i - y_i) + 10$

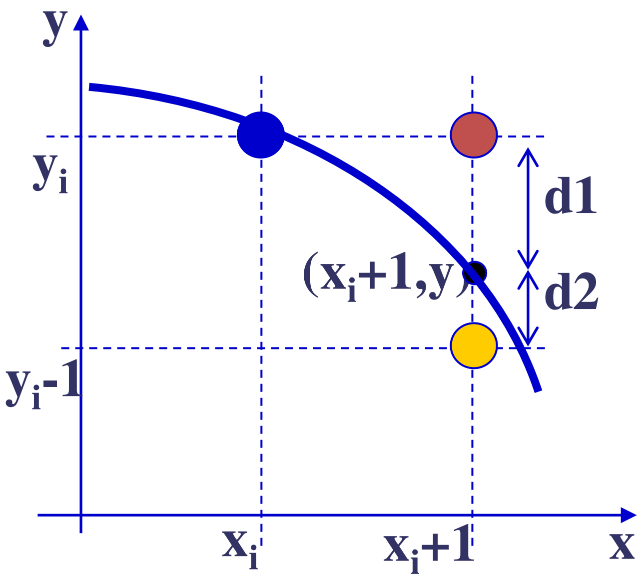


- Giá trị p_i tại điểm đầu tiên $(x_1,y_1)=(0,r)$ là:

  $P_1 = 2 + r^2 + (r-1)^2 - 2r^2 = 3 - 2r$

Vẽ đường tròn -Thuật toán Bresenham

```
void CircleBres(int xc,int yc,int r){
        int x,y,p;
        x=0;    y=r;    p=3-2*r;
        while (x<=y)            {
                put8pixel(xc,yc,x,y);
                if (p<0)
                        p+=4*x+6;
                else{
                        p+=4*(x-y)+10;
                        y--;
                }
                x++;
        }
}
```



Vẽ đường tròn -Thuật toán Midpoint

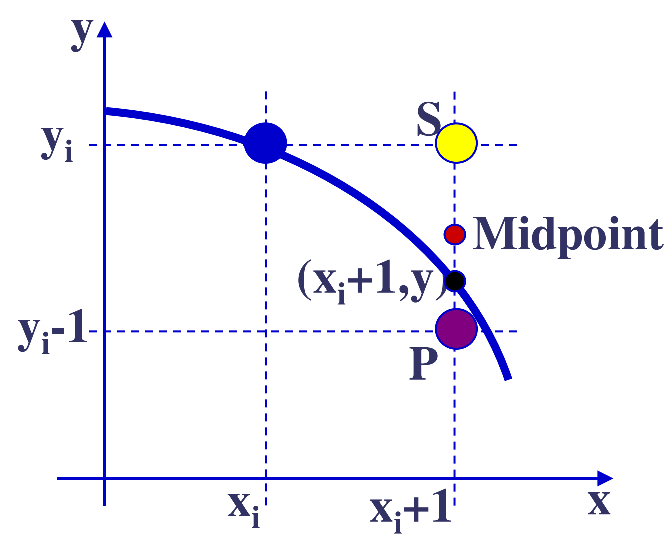
- Gọi $F(x,y)=x^2+y^2-r^2$, ta có:

$$F(x,y)\begin{cases} <0 & \text{nếu } (x,y) \text{ nằm trong đường tròn} \\ =0 & \text{nếu } (x,y) \text{ thuộc đường tròn} \\ >0 & \text{nếu } (x,y) \text{ nằm ngoài đường tròn} \end{cases}$$

- Chọn điểm bắt đầu vẽ là (0,r).

- Giả sử đã vẽ được điểm (x_i,y_i),

  Điểm cần vẽ kế tiếp (x_{i+1},y_{i+1}) là S hay P.

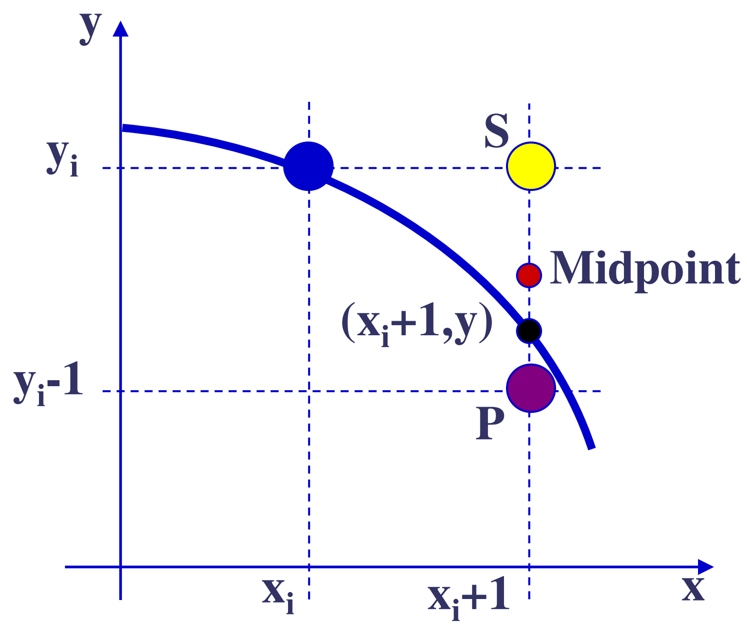


Vẽ đường tròn -Thuật toán Midpoint

- Việc chọn điểm S hay P dựa trên dấu của:

$$p_i = F(MidPoint) = F(x_i+1, y_i-1/2)$$

  - Nếu $p_i<0$ ⇨chọn S
  - Nếu $p_i \geq 0$ ⇨chọn P



Vẽ đường tròn -Thuật toán Midpoint

- Mặt khác:

$$p_{i+1} - p_i = F(x_{i+1}+1, y_{i+1}-1/2) - F(x_i+1, y_i-1/2)$$

$$= [(x_{i+1}+1)^2+(y_{i+1}-1/2)^2-r^2] - [(x_i+1)^2+(y_i-1/2)^2-r^2]$$

$$= 2x_i + 3 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

- Vậy: + Nếu $p_i < 0$ thì $y_{i+1}=y_i$, khi đó $p_{i+1} = p_i + 2x_i+3$

  + Nếu $p_i \geq 0$ thì $y_{i+1}=y_i-1$, khi đó $p_{i+1} = p_i + 2(x_i-y_i)+5$

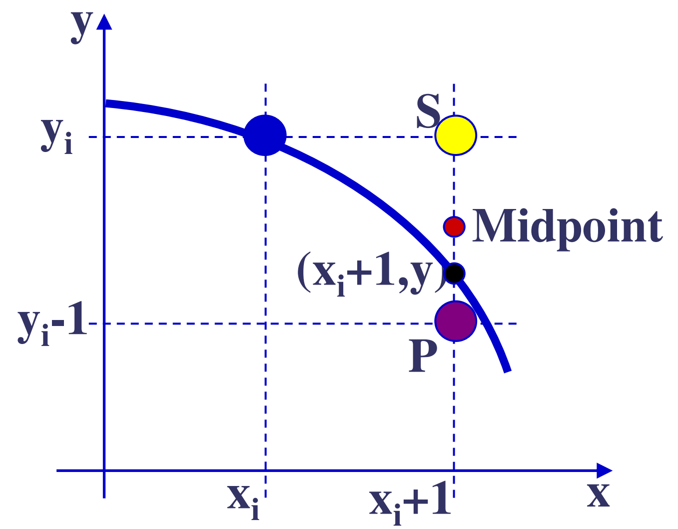
- Giá trị p_i tại điểm đầu tiên $(x_1,y_1)=(0,r)$ là:

$$p_1 = F(x_1+1, y_1-1/2)= F(1, r-1/2)$$

$$= 5/4 – r.$$

Vẽ đường tròn -Thuật toán Midpoint

```
void CircleMidpoint(int xc,int yc,int r){
        int x,y,p;
        x=0;    y=r;     p=5/4-r;
        while (x<=y) {
                put8pixel(xc,yc,x,y);
                if (p<0)   p+=2*x+3;
                else{
                        p+=2*(x-y)+5;
                        y--;
                }
                x++;
        }
}
```



…Vẽ đối tượng đồ họa cơ sở

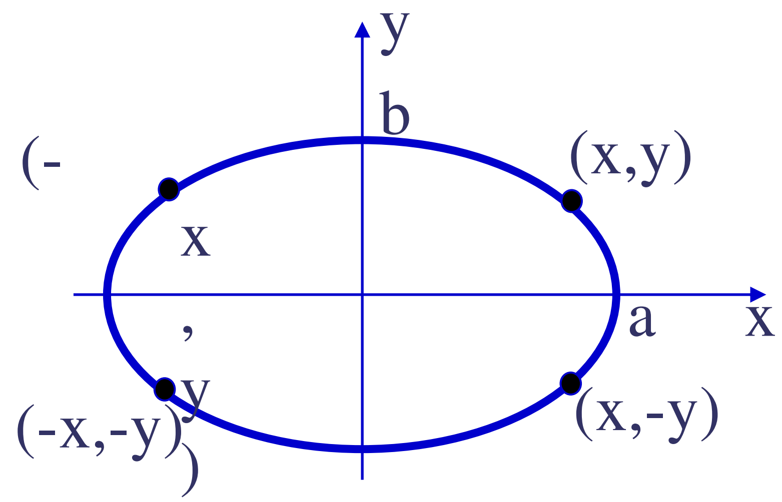
Vẽ vẽ Elip

- Phương trình của Ellipse có tâm ở gốc tọa độ có dạng:

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

- Ta có thể viết lại:

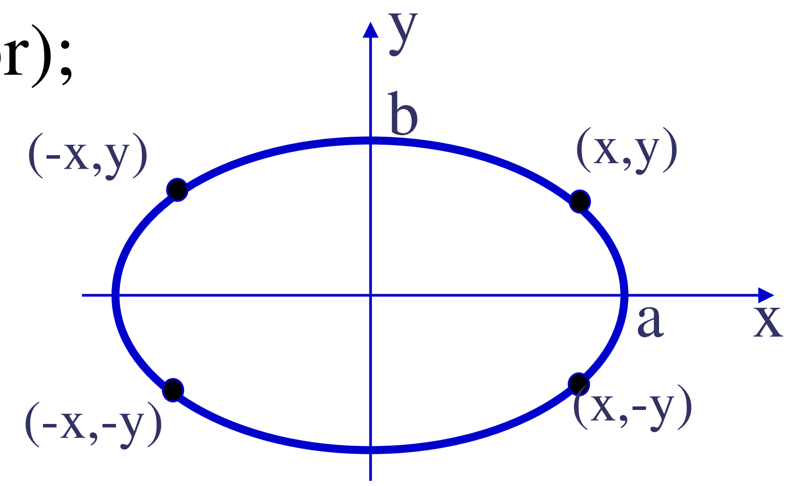
$$y^2 = -\frac{b^2}{a^2}x^2 + b^2$$



- Do tính đối xứng của ellipse nên ta chỉ cần vẽ ¼ ellipse.

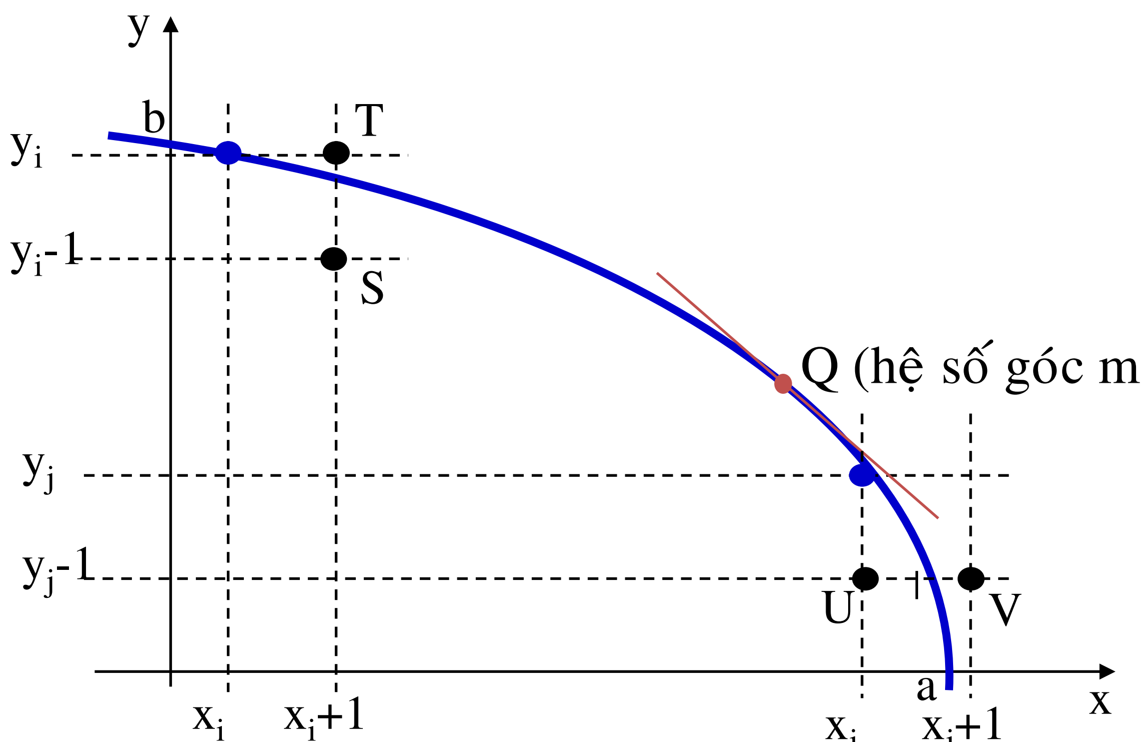
Vẽ vẽ Elip

```
void put4pixel(int xc,int yc, int x, int y, int color){
        putpixel(x+xc, y+yc,color);
        putpixel(x+xc,-y+yc,color);
        putpixel(xc-x,  yc-y,color);
        putpixel(xc-x, yc+y,color);
}
```



Vẽ vẽ Elip

- Việc vẽ ellipse được chia làm 2 nhánh:
  - Nhánh 1: từ điểm (0,b) đến điểm Q (hệ số góc tại Q là –1)
  - Nhánh 2: từ điểm Q đến điểm (a,0)



Vẽ vẽ Elip

- Để xác định hệ số góc m của ellipse tại điểm (x,y) bất kỳ, ta có: $m = \frac{dy}{dx} = -\frac{fx}{fy} = -\frac{2b^2x}{2a^2y}$

Trong đó: fx, fy là đạo hàm riêng của hàm:

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

- Điều kiện chuyển từ nhánh 1 sang nhánh 2 là:

$m<-1$ hay $2b^2x > 2a^2y$

(vì góc ¼ thứ I của hệ tọa độ thiết bị, $a,b,x,y \geq 1$)

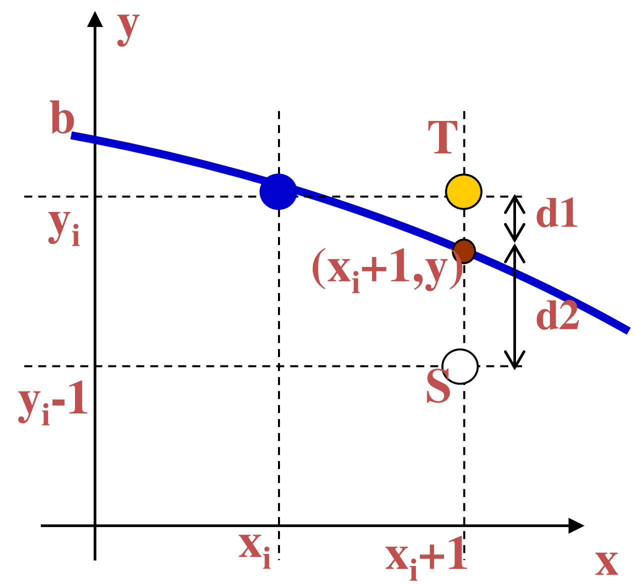
Vẽ vẽ Elip

- Tiếp tuyến tại Q(x_0,y_0) có dạng: $$\frac{xx_0}{a^2}+\frac{yy_0}{b^2}=1$$
- Vì m = $-\frac{x_0 b^2}{y_0 a^2}=-1$ => $$y_0^2=\frac{b^4}{a^4}x_0^2 \quad \text{(a)}$$
- Mặt khác do Q thuộc ellipse nên: $$\frac{x_0^2}{a^2}+\frac{y_0^2}{b^2}=1 \quad \text{(b)}$$
- Từ (a) và (b) ta suy ra:

$$x_0=\frac{a^2}{\sqrt{a^2+b^2}}$$ và $$y_0=\frac{b^2}{\sqrt{a^2+b^2}}$$

Vẽ vẽ Elip - Thuật toán Bresenham

Nhánh 1 (từ điểm (0,b) đến điểm Q)



- Giả sử tại bước i, điểm (x_i, y_i) được vẽ.
- Xét tại bước i+1, ta có: $\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \{y_i, y_i - 1\} \end{cases}$
- Đặt:

$$d_1 = y_i^2 - y^2 = y_i^2 + \frac{b^2}{a^2}.(x_i+1)^2 - b^2$$

$$d_2 = y^2 - (y_i - 1)^2 = -\frac{b^2}{a^2}(x_i+1)^2 + b^2 - (y_i - 1)^2$$

$$p_i = d_1 - d_2 = 2.[\frac{b^2}{a^2}.(x_i+1)^2 - b^2] + 2.(y_i^2 - y_i) + 1$$

$$p_{i+1} = 2.[\frac{b^2}{a^2}.(x_{i+1}+1)^2 - b^2] + 2.(y_{i+1}^2 - y_{i+1}) + 1$$

$$p_{i+1} - p_i = 2.\frac{b^2}{a^2}.[(x_{i+1}+1)^2 - (x_i+1)^2] + 2.(y_{i+1}^2 - y_i^2 - y_{i+1} + y_i)$$

$$\Rightarrow \quad p_{i+1} = p_i + 2.\frac{b^2}{a^2}.[(x_{i+1}+1)^2 - (x_i+1)^2] + 2.(y_{i+1}^2 - y_i^2 - y_{i+1} + y_i)$$

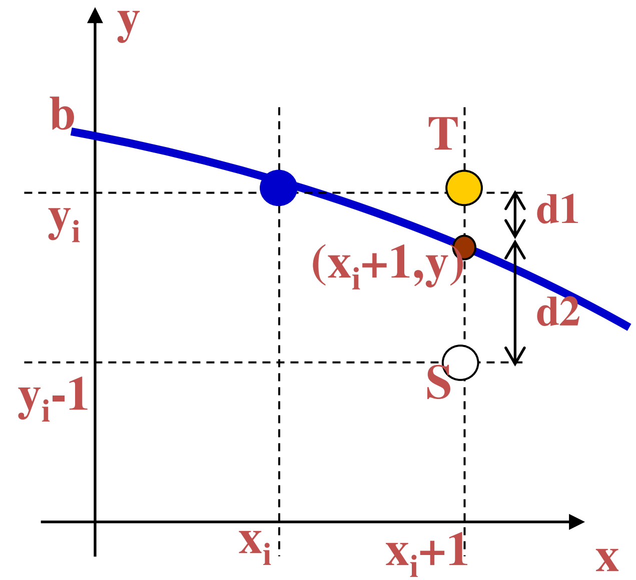
Vậy:

$p_i < 0$: Chọn $y_{i+1} = y_i$

$$\Rightarrow p_{i+1} = p_i + 2.\frac{b^2}{a^2}.(2x_i + 3)$$

$p_i \geq 0$: Chọn $y_{i+1} = y_i - 1$

$$\Rightarrow p_{i+1} = p_i + 2.\frac{b^2}{a^2}.(2x_i + 3) - 4(1-y_i)$$



Điểm đầu tiên (0,b):

$$p_1 = 2\frac{b^2}{a^2} - 2b + 1$$

Nhánh 2 (từ điểm (a,0) đến điểm Q:

Tương tự, ta có:

$$p_{i+1} = p_i + 2.\frac{a^2}{b^2}.[(y_{i+1}+1)^2 - (y_i+1)^2] + 2.(x_{i+1}^2 - x_i^2 - x_{i+1} + x_i)$$

Vậy:

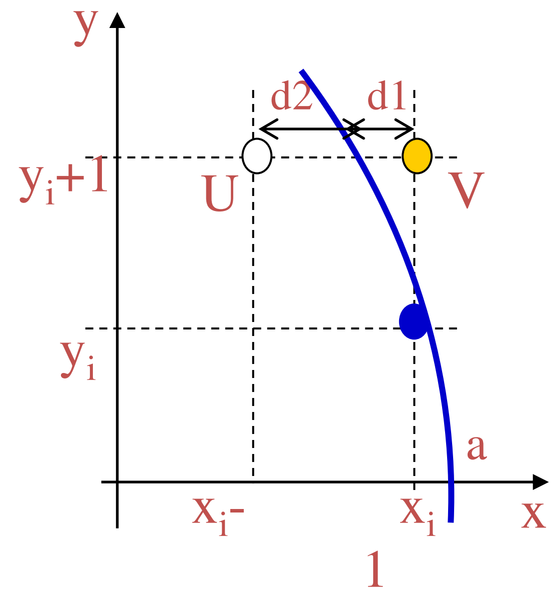
$p_i < 0$: Chọn $x_{i+1} = x_i$

$$\Rightarrow p_{i+1} = p_i + 2.\frac{a^2}{b^2}.(2y+3)$$

$p_i \geq 0$: Chọn $x_{i+1} = x_i - 1$

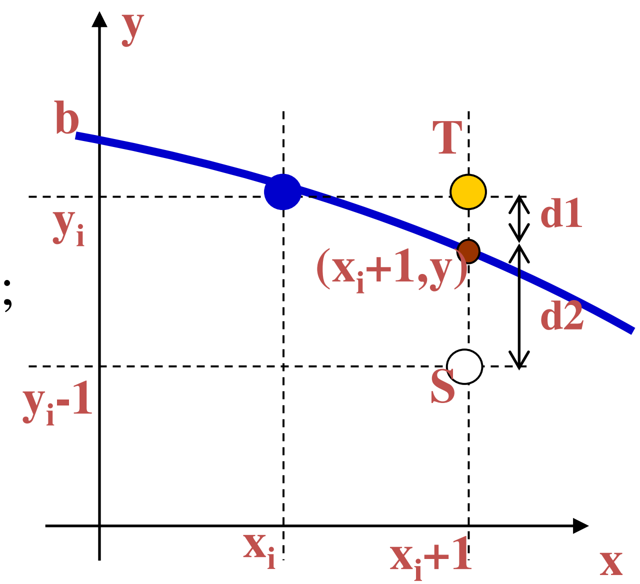
$$\Rightarrow p_{i+1} = p_i + 2.\frac{a^2}{b^2}.(2y+3) - 4(1-x_i)$$

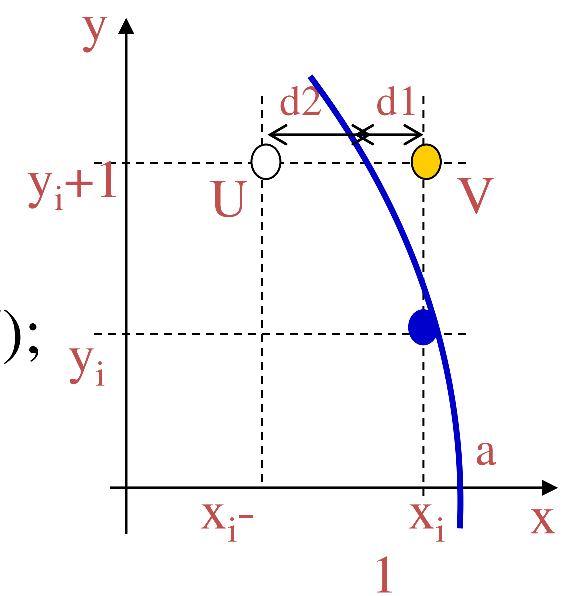
Điểm đầu tiên (a,0), ta có: $p_1 = 2\frac{a^2}{b^2} - 2a + 1$



…Vẽ đối tượng đồ họa cơ sở

```
void ElipBres(int xc,int yc, int a, int b){
        double x,y,p,x0, y0,a2,b2;
        a2=a*a; b2=b*b;
        x=0;      y=b;      p=-2*b+1+2*b2/(a2);
        x0=a2/(sqrt(a2+b2));  y0=b2/(sqrt(a2+b2));
        while (x<=x0){
                put4pixel(xc,yc,YELLOW);
                if (p<0)
                        p+=2*b2*(2*x+3)/a2;
                else{
                        p+=4*(1-y)+ 2*b2 * (2*x+3)/a2;
                        y--;
                }
                x++;
        }
```



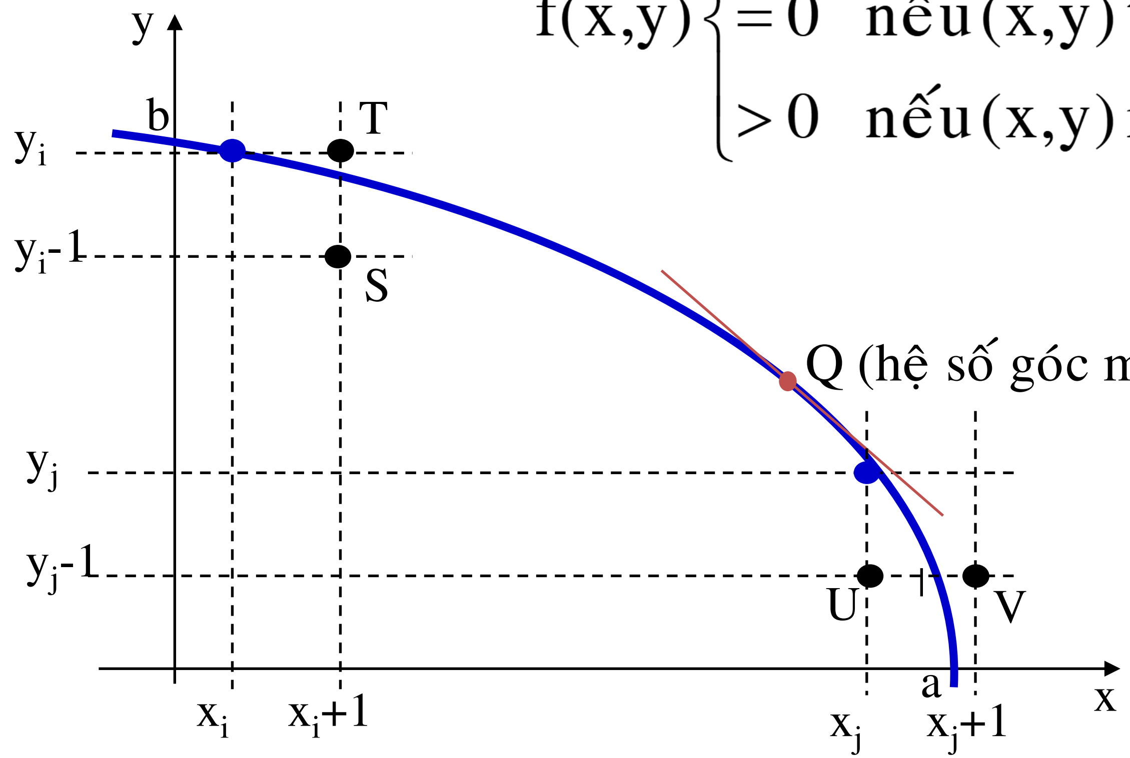



```
x=a;    y=0;    p=2*a2/b2 - 2*a+1;
while (y<=y0)
{
        put4pixel(xc,yc,x,y,YELLOW);
        if (p<0 )
                p+=2*a2*(2*y+3)/b2;
        else
        {       p+=4*(1-x) + 2*a2*(2*y+3)/b2;
                x--;
        }
        y++;
}
}
```

Vẽ vẽ Elip - Thuật toán Midpoint

Đặt $f(x,y) = b^2x^2 + a^2y^2 - a^2b^2$, ta có:

$$f(x,y)\begin{cases} < 0 & \text{nếu } (x,y) \text{ nằm trong ellipse} \\ = 0 & \text{nếu } (x,y) \text{ thuộc ellipse} \\ > 0 & \text{nếu } (x,y) \text{ nằm ngoài ellipse} \end{cases}$$



Vẽ vẽ Elip - Thuật toán Midpoint

<u>Nhánh 1 (từ điểm (0,b) đến điểm Q)</u>

- Giả sử tại bước i, điểm (x_i, y_i) được vẽ.
- Xét tại bước i+1, ta có: $\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \{y_i, y_i - 1\} \end{cases}$
- Ta có: $p_i = f(\text{Midpoint}) = f(x_i+1, y_i-1/2)$

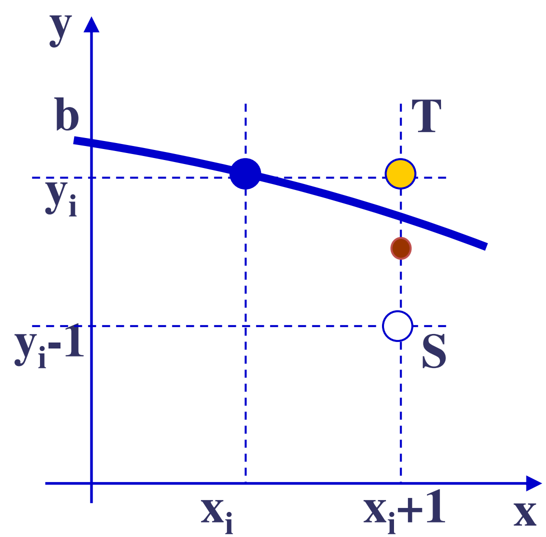
  $= b^2(x_i+1)^2 + a^2(y_i-1/2)^2 - a^2b^2$

Nếu $p_i<0$ thì T

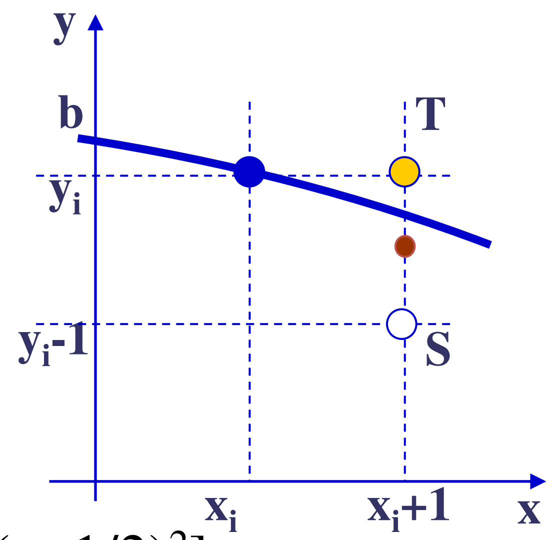
hay $(x_{i+1},y_{i+1}) = (x_i+1,y_i)$

Nếu $p_i \geq 0$ thì vẽ S

hay $(x_{i+1},y_{i+1}) = (x_i+1,y_i-1)$



Vẽ vẽ Elip - Thuật toán Midpoint



- Tính $p_{i+1} = f(x_{i+1}+1,\ y_{i+1}-1/2)$

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

  Vì $x_{i+1} = x_i+1$ nên:

  $p_{i+1} - p_i = b^2\,[(x_{i+1}+1)^2-(x_{i+1})^2\,] + a^2\,[(y_{i+1}-1/2)^2-(y_i-1/2)^2]$

  $\Rightarrow p_{i+1} = p_i + (2x_i+3)*\ b^2 + a^2\,[(y_{i+1}-1/2)^2-(y_i-1/2)^2]$

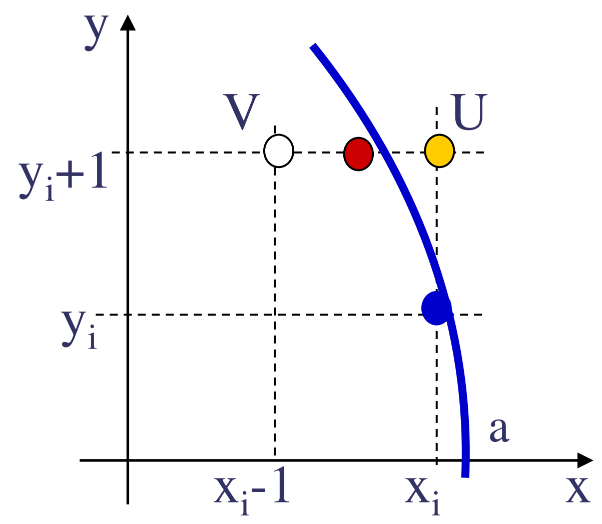
- Vậy: $p_{i+1} = p_i + (2x_i+3)*\ b^2$ nếu $p_i<0$

  $p_{i+1} = p_i + (2x_i+3)*\ b^2 - 2\,a^2(y_i-1)$ nếu $p_i \geq 0$

- Tính p_1 tại (0,b):

  $P_1 = b^2 + a^2(b-1/2)^2 - a^2b^2 = b^2 - a^2b + a^2/4$

Vẽ vẽ Elip - Thuật toán Midpoint



Nhánh 2 (từ điểm Q đến điểm (a,0):

Tương tự, ta có:

$$q_i = f(\text{Midpoint}) = f(x_i - 1/2,\ y_i + 1)$$
$$= b^2(x_i - 1/2)^2 + a^2(y_i + 1)^2 - a^2b^2$$

$$q_{i+1} = q_i + (2y_i + 3) * a^2 \qquad \text{nếu } q_i < 0$$
$$q_{i+1} = q_i + (2y_i + 3) * a^2 - 2b^2(x_i - 1) \qquad \text{nếu } q_i \geq 0$$

Tính q_1 tại (a,0): $q_1 = a^2 - ab^2 + b^2/4$

```
void  ElipMidpoint(int xc,int yc,int a,int b,int color){
        int x,y;              float x0,y0,a2,b2,p;
        a2=a*a; b2=b*b;
        x0=(int)(a2/sqrt(a2+b2));
        y0=(int)(b2/sqrt(a2+b2));
        p=b2-a2*b+(1/4)*a2;               x=0;       y=b;
        while (x<=x0){
                put4pixel(xc,yc,x,y,color);
                if (p<0) p+=(2*x+3)*b2;
                else{
                        p+=(2*x+3)*b2-2*a2*(y-1);
                        y--;
                }
                x++;
        }
```

…Vẽ đối tượng đồ họa cơ sở

```
x=a;    y=0;    p=a2-a*b2+(1/4)*b2;
while (y<=y0)
{
        put4pixel(xc,yc,x,y,color);
        if (p<0)
                p+=a2*(2*y+3);
        else
        {
                p+=(2*y+3)*a2-2*b2*(x-1);
                x--;
        }
        y++;
}
}
```

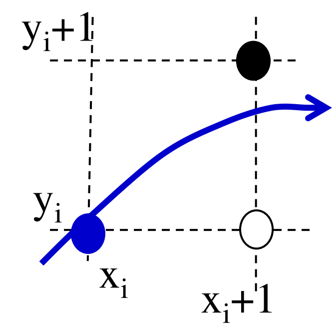
…Vẽ đối tượng đồ họa cơ sở

Thuật toán vẽ đường cong y=f(x)

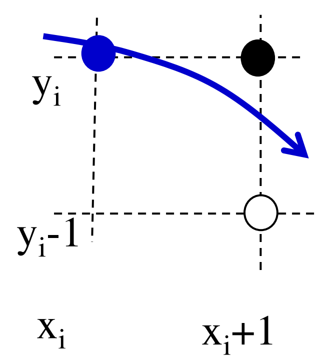
Áp dụng thuật toán MidPoint hay Bresenham để vẽ các đường cong theo các bước sau:

- Bước 1: Dựa vào dáng điệu và phương trình đường cong (tính đối xúng, chẵn lẻ...) để rút gọn một phần đường cong cần vẽ.
- Bước 2: Tính đạo hàm để từ đó phân thành các vùng vẽ:

+ Nếu $0 \le f'(x) \le 1$ thì $\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} \in \{y_i, y_i + 1\} \end{cases}$

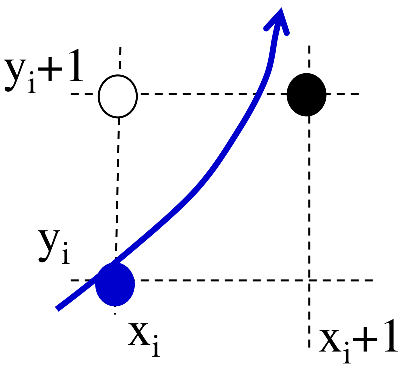


+ Nếu $-1 \le f'(x) \le 0$ thì $\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} \in \{y_i, y_i - 1\} \end{cases}$

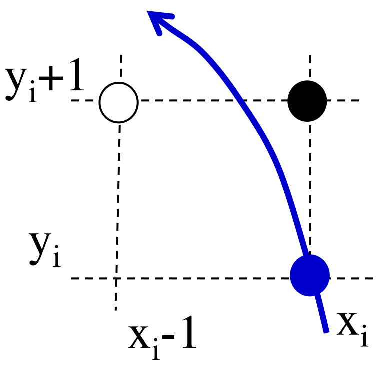


Thuật toán vẽ đường cong y=f(x)

+ Nếu $f'(x) > 1$ thì $\begin{cases} y_{i+1} = y_i + 1 \\ x_{i+1} \in \{x_i, x_i + 1\} \end{cases}$



+ Nếu $f'(x) < -1$ thì $\begin{cases} y_{i+1} = y_i + 1 \\ x_{i+1} \in \{x_i, x_i - 1\} \end{cases}$



Thuật toán vẽ đường cong y=f(x)

Áp dụng thuật toán MidPoint hay Bresenham để vẽ các đường cong theo các bước sau:

- Bước 3: Xác định công thức và dấu của p_i cho từng trường hợp để tìm ra toạ độ của điểm cần vẽ.

  (P_i thường là hàm được xây dựng từ phương trình đường cong để cho $p_i=0$ nếu (x_i,y_i) thuộc về đường cong. Việc chọn p_i cần chú y sao cho thao tác tính p_i hạn chế các phép toán trên số thực)
- Bước 4: Tìm mối liên quan của p_{i+1} và p_i bằng cách xét hiệu $p_{i+1}-p_i$
- Bước 5: Tính p_1 và hoàn chỉnh thuật toán.

Xén hình

Thao tác loại bỏ các phần hình ảnh nằm ngoài một vùng cho trước được gọi là xén hình (Clipping).

❖Bài toán: Cho miền $D \subset R^n$ và $F \subset R^n$. Gọi $F \cap D$ là hình có được từ hình F bằng cách cắt vào trong D, ký hiệu: $Clip_D(F)$.

Bài toán đặt ra là tìm 1 thuật toán để xác định $Clip_D(F)$.

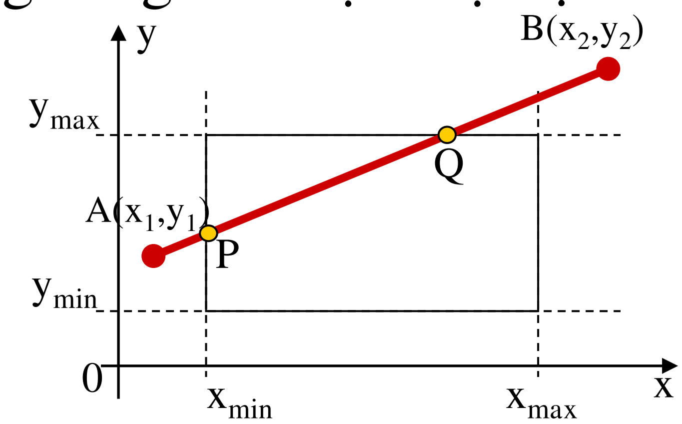
…Xén hình

- F là đoạn thẳng và D là hình chữ nhật
- F là đoạn thẳng và D là đường tròn
- F là đa giác và D là hình chữ nhật

F là đoạn thẳng và D là hình chữ nhật

☝Giả sử cạnh của hình chữ nhật song song với trục toạ độ

Ta có:



$$D = \left\{ (x,y) \in R^2 \middle| \begin{matrix} x_{min} \le x \le x_{max} \\ y_{min} \le y \le y_{max} \end{matrix} \right\}$$

$$F = \left\{ (x,y) \in R^2 \middle| \begin{matrix} x = x_1 + (x_2 - x_1)t = x_1 + tDx \\ y = y_1 + (y_2 - y_1)t = y_1 + tDy \\ 0 \le t \le 1 \end{matrix} \right\}$$

Khi đó, giao của $F \cap D$ chính là

nghiệm của bất phương trình (theo t): $D \cap F = \left\{ \begin{matrix} x_{min} \le x_1 + tDx \le x_{max} \\ y_{min} \le y_1 + tDy \le y_{max} \\ 0 \le t \le 1 \end{matrix} \right.$

F là đoạn thẳng và D là hình chữ nhật

Gọi N là tập nghiệm của hệ: $D \cap F = \begin{cases} x_{min} \le x_1 + tDx \le x_{max} \\ y_{min} \le y_1 + tDy \le y_{max} \\ 0 \le t \le 1 \end{cases}$

Khi đó xảy ra các trường hợp:

-Nếu N=∅

⇨Bất phương trình vô nghiệm

⇨$Clip_D(F)=\varnothing$

-Nếu N≠∅

⇨$N=[t_1, t_2]$, với $t_1 \le t_2$.

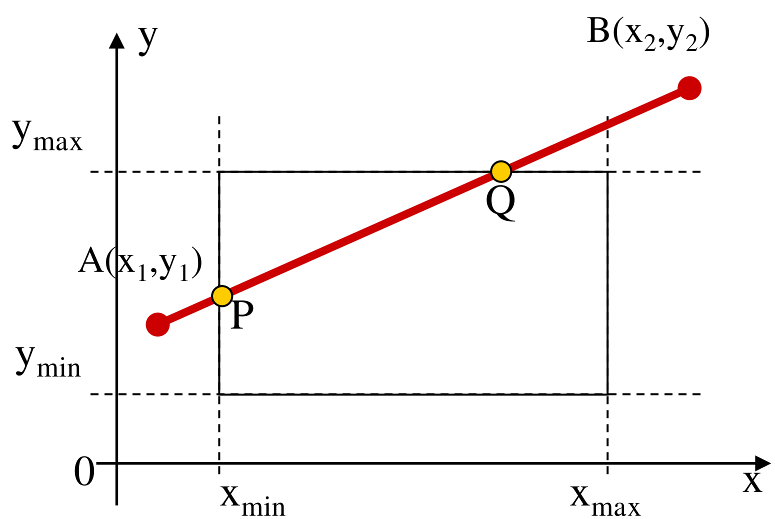
F là đoạn thẳng và D là hình chữ nhật

Nếu N≠∅ , gọi P và Q là 2 giao điểm có toạ độ như sau:

$$\begin{cases} P_x = x_1 + (x_2 - x_1)t_1 \\ P_y = y_1 + (y_2 - y_1)t_1 \end{cases} \quad \text{và} \quad \begin{cases} Q_x = x_1 + (x_2 - x_1)t_2 \\ Q_y = y_1 + (y_2 - y_1)t_2 \end{cases}$$

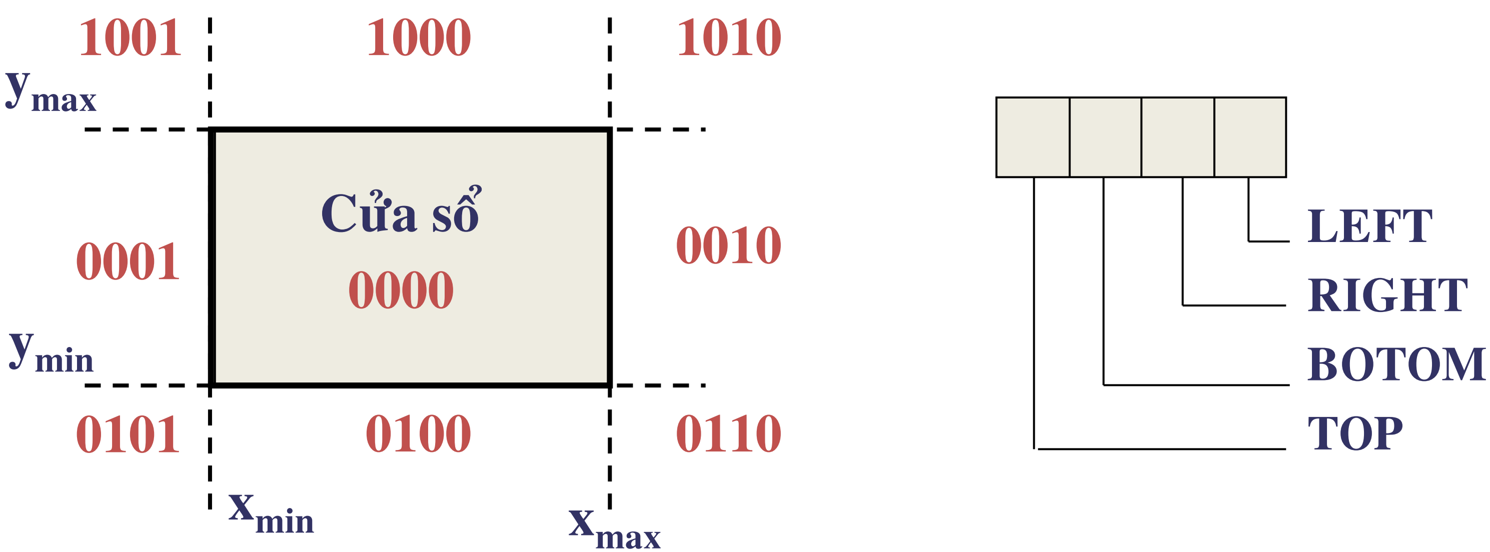
Vậy $Clip_D(F)=PQ$

☞ tính toán nhiều trên số thực



Thuật toán Cohen-Sutherland

Chia mặt phẳng thành 9 vùng: cửa sổ và 8 vùng xung quanh nó. Mỗi vùng được gán bởi một mã nhị phân 4 bít.



Thuật toán Cohen-Sutherland

Giả sử có điểm P(x,y), lúc đó gán mã cho điểm P:

$$P_{left} = \begin{cases} 1, \text{ nếu } P_x < x_{min} \\ 0, \text{ ngược lại} \end{cases} \qquad P_{right} = \begin{cases} 1, \text{ nếu } P_x > x_{max} \\ 0, \text{ ngược lại} \end{cases}$$

$$P_{bottom} = \begin{cases} 1, \text{ nếu } P_y < y_{min} \\ 0, \text{ ngược lại} \end{cases} \qquad P_{top} = \begin{cases} 1, \text{ nếu } P_y > y_{max} \\ 0, \text{ ngược lại} \end{cases}$$

Thuật toán Cohen-Sutherland

Hàm xác định mã

```
int    ma(point M){
       int m=0;
       if (M.x<min.x)      m|= 1;
       if (M.x>max.x)      m|= 2;
       if (M.y<min.y)      m|= 4;
        if (M.y>max.y)     m|= 8;
       return m;
}
```

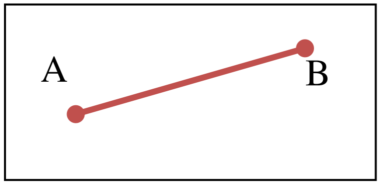
Thuật toán Cohen-Sutherland

Xét đoạn thẳng AB, ta có các trường hợp sau:

1.Nếu (Ma(A)=0000) và (Ma(B)=0000)

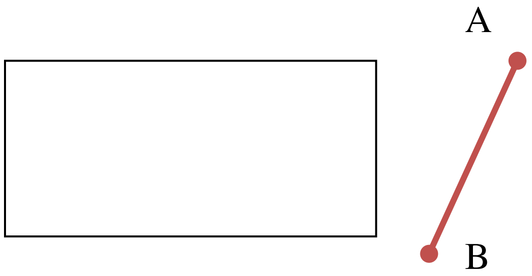
{hay (Ma(A) or Ma(B))=0000} thì

⇨ $Clip_D(F)=AB$



2.Nếu (Ma(A) and Ma(B))≠0000 thì

⇨ $Clip_D(F)=\varnothing$



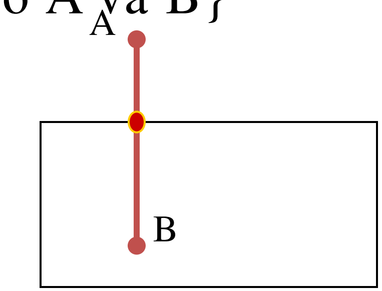
Thuật toán Cohen-Sutherland

3.Nếu ((Ma(A) and Ma(B))=0000)

và (Ma(A)≠0000 hoặc Ma(B)≠0000) thì

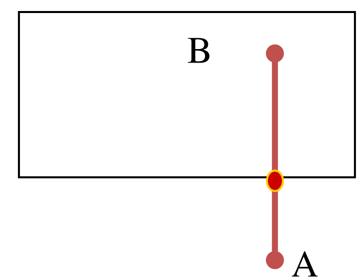
Giả sử Ma(A)≠0000 {nếu ma(A)=0 ta đổi vai trò A và B}

-Nếu $A_x=B_x$ (AB thẳng đứng) thì



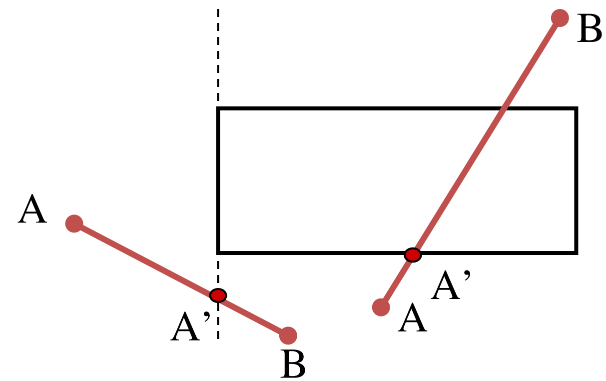
+ Nếu $A_y>y_{max}$ (A ở trên) thì $A_y=y_{max}$,

ngược lại (A ở dưới) $A_y=y_{min}$



⇨ $Clip_D(F)=AB$

Thuật toán Cohen-Sutherland



-Ngược lại (trường hợp $A_x \neq B_x$):

+Tính hsố góc m=(By-Ay)/(Bx-Ax)

{để tính giao của AB với hcn}

Vì A nằm ngoài hình chữ nhật nên:

+Nếu $A_x < x_{min}$ thì

Thay A bởi điểm giao của AB với cạnh trái (nối dài) của HCN.

+Nếu $A_x > x_{max}$ thì

Thay A bởi điểm giao của AB với cạnh phải (nối dài) của HCN.

+Nếu $A_y < y_{min}$ thì

Thay A bởi điểm giao của AB với cạnh dưới (nối dài) của HCN.

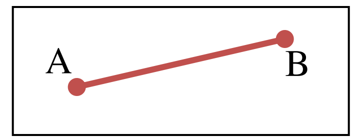
+Nếu $A_y > y_{max}$ thì

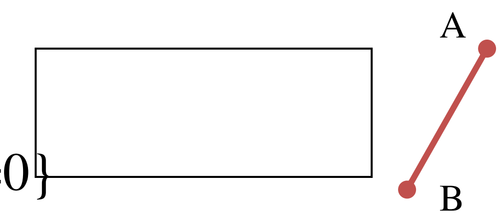
Thay A bởi điểm giao của AB với cạnh trên (nối dài) của HCN

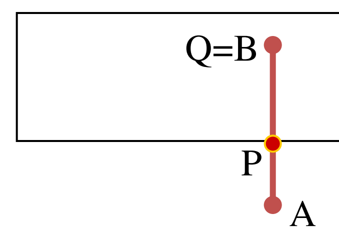
Quá trình lặp lại này dừng khi xảy ra một trong hai trường hợp 1 hay 2

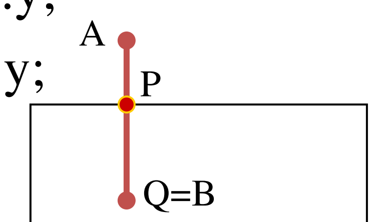
…Xén hình

```
void      ClipCohen(Point  A, Point B, Point  wmin, Point  wmax)
 { int     thoat, ve;              double m;
   thoat=0; ve=1;
   while  (thoat==0)
   {
     if((ma(A) | ma(B))==0)          thoat=1
     else
        if((ma(A) & ma(B)) !=0)        {thoat=1;  ve=0}
        else
          { if (m(A)==0)       hoanvi(&A,&B);
            if(A.x==B.x)
                 {  if (A.y>wmax.y)       A.y=wmax.y;
                 else                     A.y=wmin.y;
                 }
          else
```

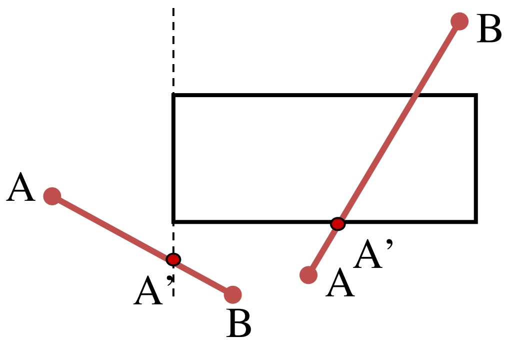






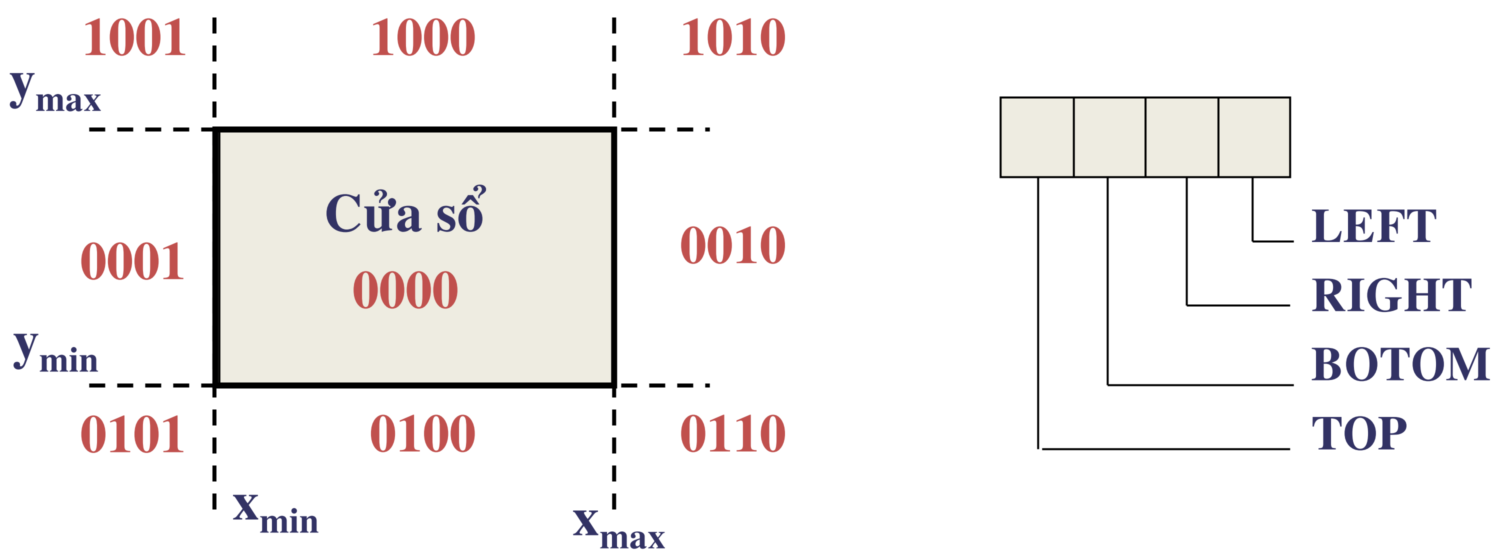


```
{     m=(double)(B.y-A.y)/(B.x-A.x);
      if (A.x<wmin.x)
       {    A.y=A.y+ (wmin.x-A.x)*m;        A.x=wmin.x;  }
      else
          if (A.x>wmax.x)
           {      A.y=A.y+ (wmax.x-A.x)*m;      A.x=wmax.x;    }
         else
            if (A.y<wmin.y)
              {A.x=A.x+ (wmin.y-A.y)/m;    A.y=wmin.y;          }
            else
                 if (A.y>wmax.y)
                 {  A.x=A.x+ (wmax.y-A.y)/m;   A.y=wmax.y; }
}
        }
      }//end while
 if   (ve)  veduongthang(A,B);
}
```



Thuật toán chia nhị phân

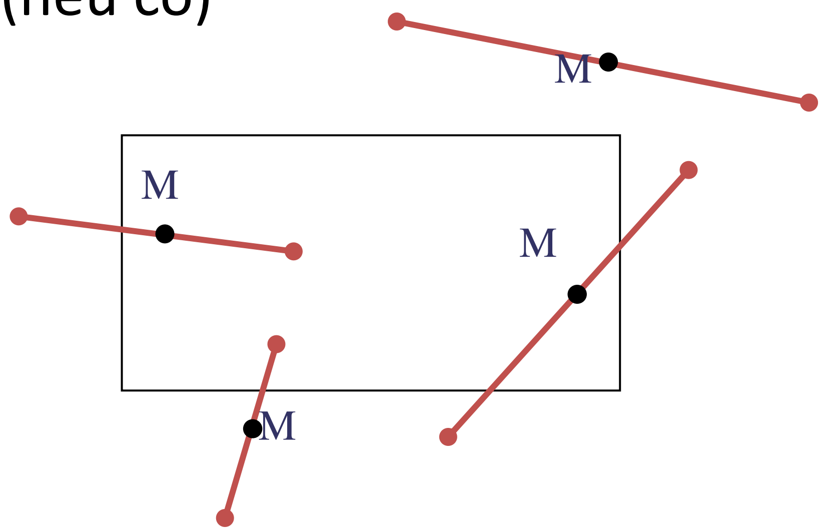
Chia mặt phẳng thành 9 vùng, mỗi vùng được gán bởi một mã nhị phân 4 bít.



Thuật toán chia nhị phân

☞**Tư tưởng của thuật toán như sau**:

Lấy trung điểm của đoạn thẳng và kiểm tra mã của nó để loại dần các đoạn con không chứa giao điểm, và cuối cùng cho điểm giữa hội tụ về giao điểm của đoạn thẳng với hình chữ nhật, kết quả là ta thu được đoạn con nằm trong hình chữ nhật (nếu có)



Thuật toán chia nhị phân

☞ **Mệnh đề**:

Cho M trung điểm của đoạn AB,

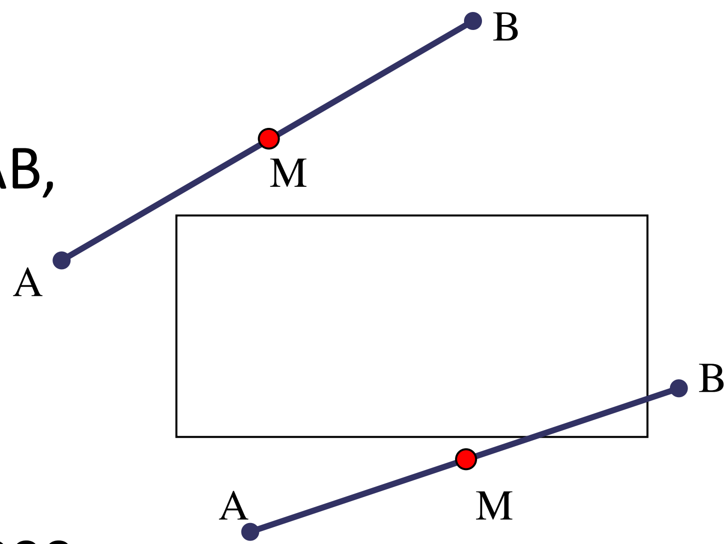
Mã(A) $\neq$ 0000,

Mã(B) $\neq$ 0000,

Mã(M) $\neq$ 0000

thì ta có: [Mã(A) AND Mã(M)] $\neq$ 0000

hoặc [Mã(M) AND Mã(B)] $\neq$ 0000.



☞ **Ý nghĩa hình học của mệnh đề**:

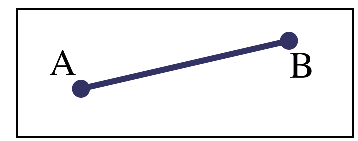
Nếu cả ba điểm A, B, M đều ở ngoài hình chữ nhật thì có ít nhất một đoạn AM hoặc BM hoàn toàn nằm ngoài hình chữ nhật.

Thuật toán chia nhị phân

☞Thuật toán:

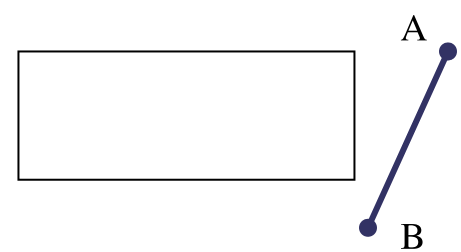
1.Nếu (Mã(A) = 0000) và (Mã(B) = 0000) thì

⇨ $Clip_D(F) = AB$



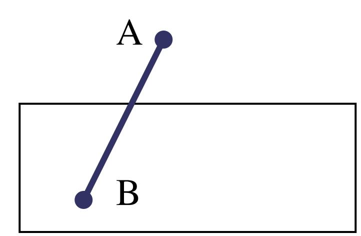
2.Nếu (Mã(A) AND Mã(B)) $\neq$ 0000 thì

⇨ $Clip_D(F) = \emptyset$



3.Nếu (Mã(A) $\neq$ 0000) và (Mã(B) = 0000) thì:

Đổi vai trò của A, B và áp dụng 4



Thuật toán chia nhị phân

☞Thuật toán:

4.Nếu (Mã(A) = 0000) và (Mã(B) ≠ 0000) thì:

P:=A; Q:=B;

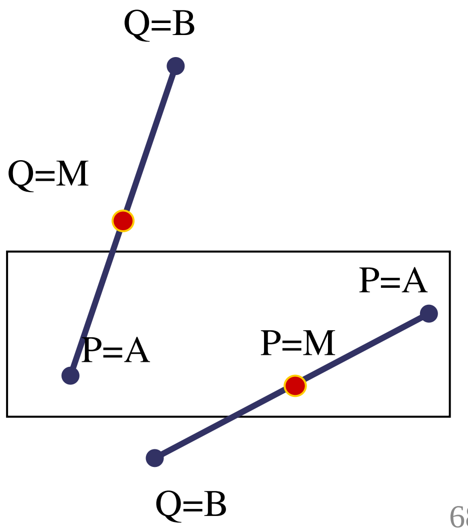
Trong khi $|x_P - x_Q| + |y_P - y_Q| \geq 2$ thì:

Lấy trung điểm M của PQ

Nếu Mã(M) ≠ 0000 thì Q:= M.

Ngược lại: P:= M.

⇨ $Clip_D(F) = AP$

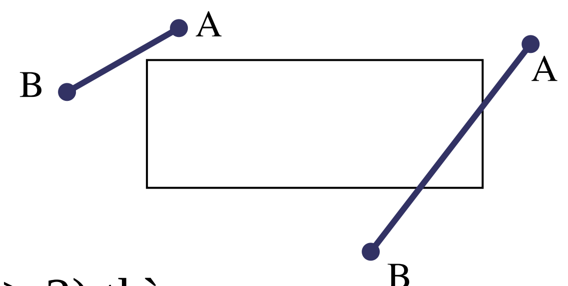


Thuật toán chia nhị phân

☞Thuật toán:

5.Nếu (Mã(A) ≠ 0000 ≠ Mã(B) và [Mã(A) AND Mã(B)]= 0000 thì:

P:=A; Q:=B;



Lấy M: trung điểm PQ;

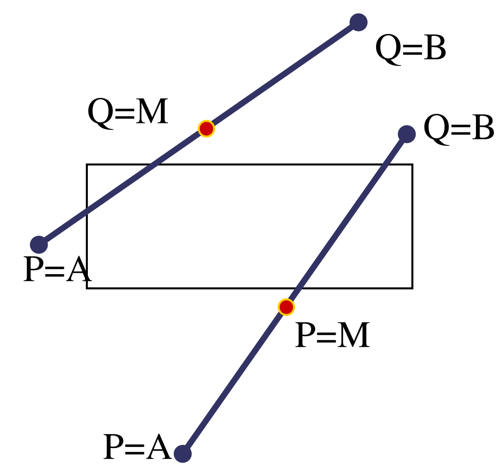
Khi (Mã(M) ≠ 0000) và $(|x_P - x_Q| + |y_P - y_Q| \geq 2)$ thì:

Nếu (Mã(M) AND Mã(Q)) ≠ 0000 thì

Q:=M.

Ngược lại P:=M.

Lấy M: trung điểm PQ.



Thuật toán chia nhị phân

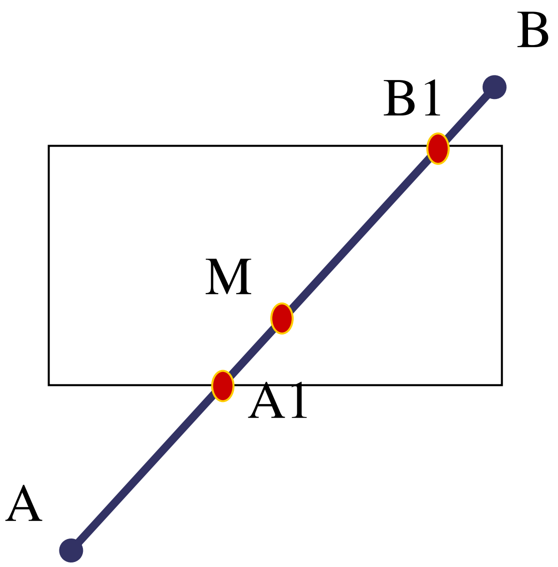
Nếu Mã(M) $\neq$ 0000 thì $Clip_D(F) = \varnothing$

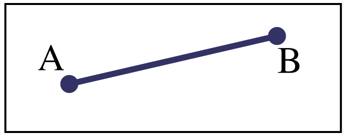
Ngược lại, áp dụng 4 ta có:

$$Clip_D(MP) = MA_1$$

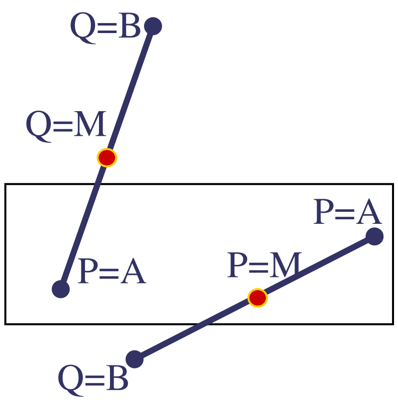
$$Clip_D(MQ) = MB_1$$

⇨ $Clip_D(F) = A_1B_1$



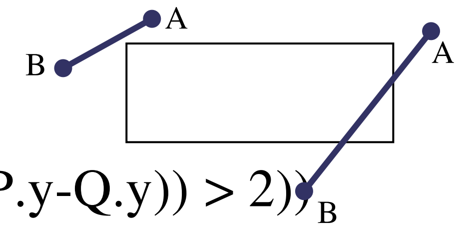


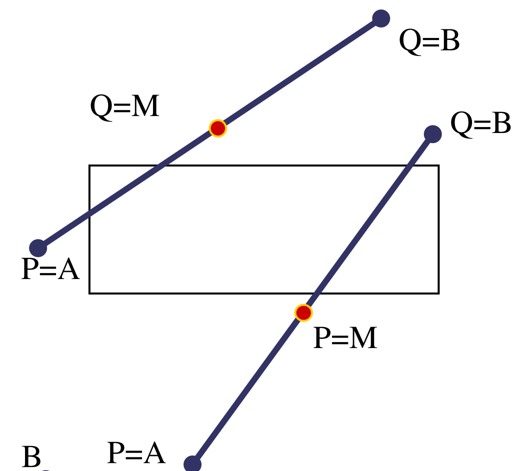

```
void XenHinhNhiPhan(Point A,Point B)
{           Point P,Q,M;
            if ((Ma(A) | Ma(B))==0) VeDuongThang(A,B);
            if ((Ma(A) & Ma(B)) != 0) return;
            if ((Ma(A) != 0) && (Ma(B) == 0)) HoanVi(&A,&B);
            if ((Ma(A) == 0) && (Ma(B) != 0))
            {           P=A; Q=B;
                        while ((abs(P.x-Q.x)+abs(P.y-Q.y)) > 2)
                        {           M.x=(P.x+Q.x)/2;
                                    M.y=(P.y+Q.y)/2;
                                    if (Ma(M)==0)           P=M;
                                    else                    Q=M;
                        }
                        VeDuongThang(A,P);
            }
```

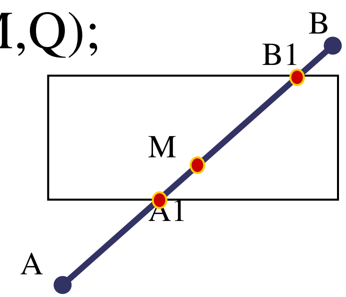


…Xén hình

```
if (((Ma(A) != 0) && (Ma(B) != 0)) && ((Ma(A) & Ma(B))==0))
{          P=A; Q=B;
           M.x=(P.x+Q.x)/2;   M.y=(P.y+Q.y)/2;
           while ((Ma(M)!=0) && ((abs(P.x-Q.x)+abs(P.y-Q.y)) > 2))
           {          if ((Ma(P) & Ma(M))!=0)   P=M;
                      else                      Q=M;
                      M.x=(P.x+Q.x)/2;
                      M.y=(P.y+Q.y)/2;
           }
           if (Ma(M)==0)
           {          XenHinhNhiPhan(P,M);
                      XenHinhNhiPhan(M,Q);
           }
}
}
```

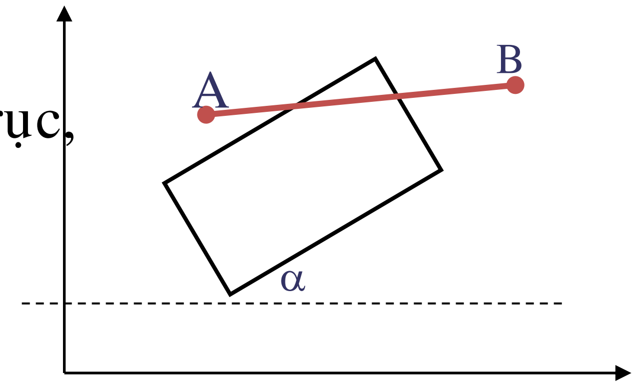






F là đoạn thẳng và D là hình chữ nhật

☝ **Cạnh của hình chữ nhật tạo với trục hoành một góc α**



1, Gọi R là ma trận của phép quay đổi trục,

2,Ta tính: $(X_{max}, Y_{max}) = (x_{max}, y_{max}).R$

$(X_{min}, Y_{min}) = (x_{min}, y_{min}).R$

thì A_1=A.R; B_1=B.R

Đặt F = đoạn A_1B_1

D = hình chữ nhật được tạo bởi 2 điểm (X_{min}, Y_{min}), (X_{max}, Y_{max})

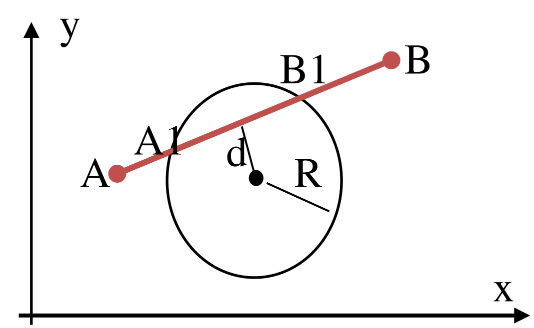
3, Xác định $Clip_D$(F) bằng một trong các thuật toán trên.

4, Nếu $Clip_D(F)=\varnothing$ thì kết quả của phép xén hình là $\varnothing$

Nếu $Clip_D(F)= A_2B_2$, kết quả là A_3B_3 với $A_3=A_2.R^{-1}$, $B_3=B_2.R^{-1}$

F là đoạn thẳng và D là đường tròn

Bài toán được giải quyết bằng cách xét vị trí tương đối giữa đường thẳng và đường tròn.



1. Tính khoảng cách d từ O đến AB
2. Nếu d>R thì

   ⇨ $Clip_D(AB)=\varnothing$

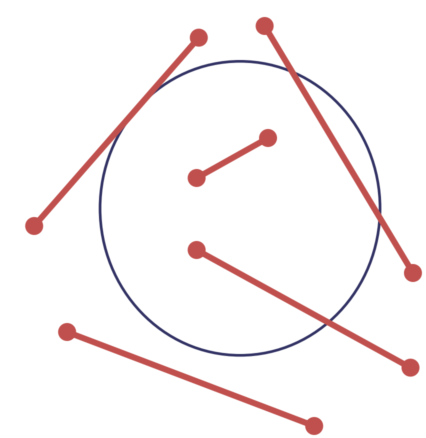
3. Nếu d=R thì

   ⇨ $Clip_D(AB)=\{A_0\}$

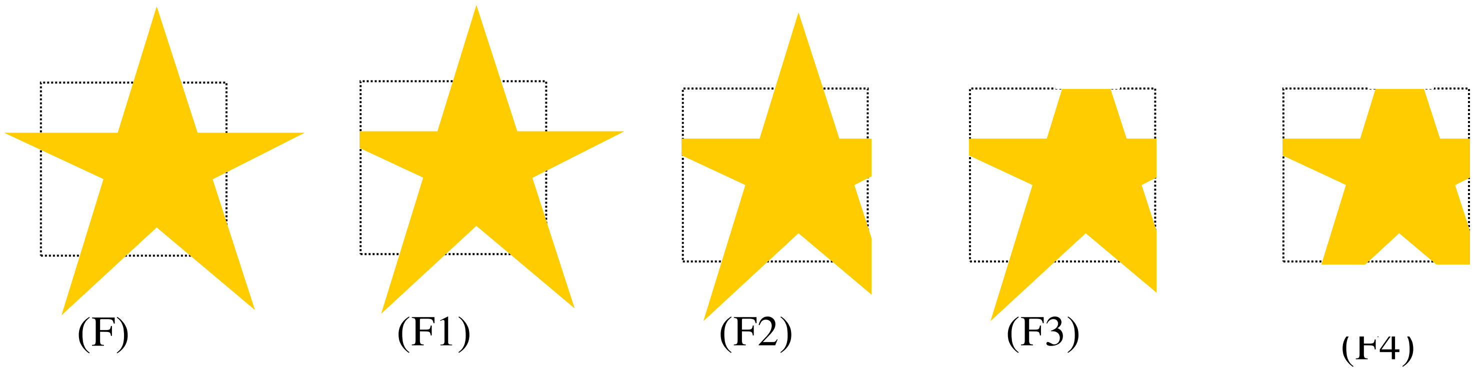
   $\{A_0$: tiếp điểm của đthẳng với đtròn$\}$.

4. Nếu d<R

………..



F là đa giác và D là hình chữ nhật



1. Với đa giác F, cắt bỏ phần bên trái HCN (nghĩa là bên trái của cạnh trái nối dài) ta thu được đa giác mới F_1

2. Với đa giác F_1 cắt bỏ phần bên phải HCN ta thu được đa giác mới F_2

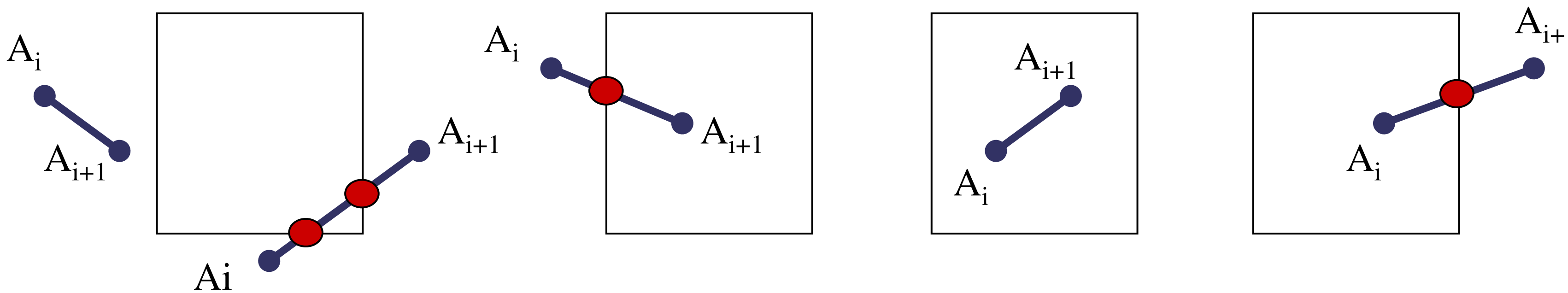
3. Với đa giác F_2 cắt bỏ phần bên trên HCN ta thu được đa giác mới F_3

4. Với đa giác F_3 cắt bỏ phần bên dưới HCN ta thu được đa giác mới F_4

☞**Kết quả:** Nếu $F_4 = \varnothing$ thì $Clip_D(F)=\varnothing$.

Ngược lại kết quả xén là đa giác F_4, hay $Clip_D(F)=F_4$

Giải thuật Sutherland - Hodgeman



i.Nếu tất cả các đỉnh đa giác đều nằm trong HCN, hình cần xén chính là đa giác.

ii.Ngược lại: Từ một đỉnh nằm ngoài HCN, chạy theo dọc biên của đa giác. Với mỗi cạnh của đa giác, ta có các trường hợp sau:

- Nếu cả hai đỉnh đều nằm ngoài hình chữ nhật thì:

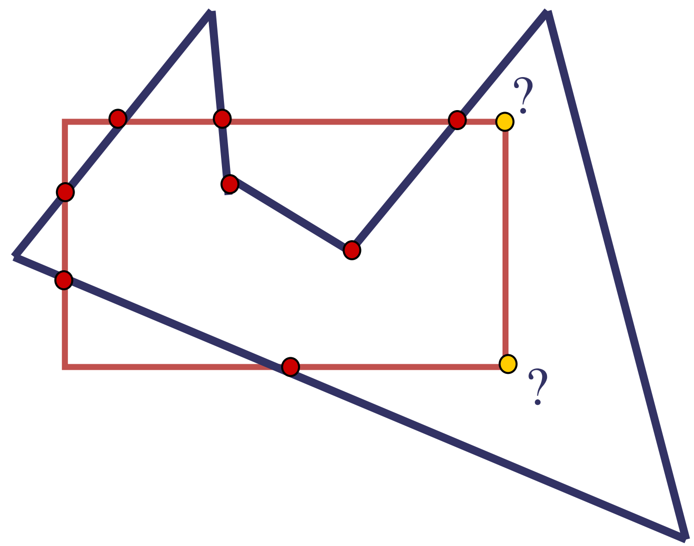
  Nếu $Ma(A_i)$ and $Ma(A_{i+1}) \neq 0000$ thì không lưu đỉnh

  Ngược lại thì lưu hai giao điểm.

- Ai ngoài, A_{i+1} trong: lưu giao điểm P và A_{i+1}.
- Cả hai đỉnh đều nằm trong hình chữ nhật: lưu A_i và A_{i+1}.
- A_i trong, A_{i+1} ngoài: lưu A_i và giao điểm P.

Giải thuật Sutherland - Hodgeman

-Sau khi duyệt qua tất cả các cạnh của đa giác thì ta có được một dãy các đỉnh mới phát sinh: $B_1, B_2, ..., B_n$



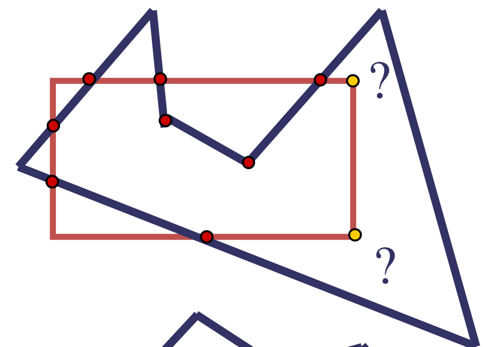
Nếu trong dãy các đỉnh mới này có hai đỉnh liên tiếp không nằm trên cùng một cạnh của hình chữ nhật , giả sử hai đỉnh đó là B_i và B_{i+1} thì ta đi dọc các cạnh của hình chữ nhật từ B_i đến B_{i+1} để tìm tất cả các đỉnh của hình chữ nhật nằm trong đa giác rồi bổ sung chúng vào giữa B_i và B_{i+1}.

…Xén hình

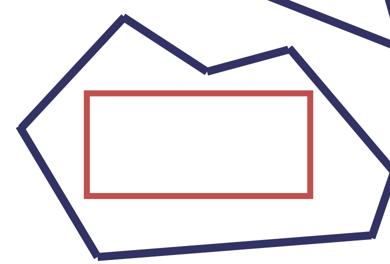
Giải thuật Sutherland - Hodgeman

Tập đỉnh mới tìm được chính là đa giác xén được.

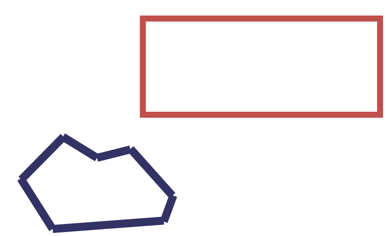
Nếu tập đỉnh mới này là rỗng:



+Nếu có một đỉnh của hình chữ nhật nằm trong đa giác thì hình xén được chính là toàn bộ hình chữ nhật.



+Ngược lại, hình xén được là rỗng.

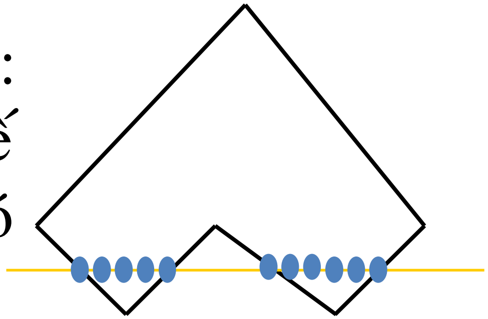


Tô màu

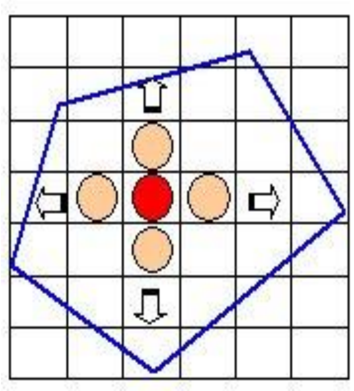
- Thuật toán tô màu theo dòng quét
- Thuật toán tô màu theo đường biên

Có 2 cách tiếp cận chính để tô màu một vùng

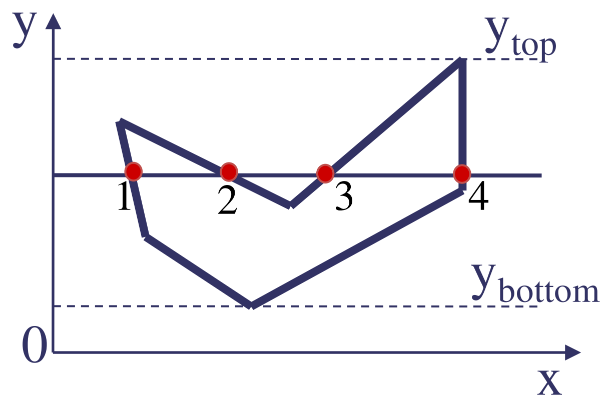
Tô theo dòng quét (tô loạt –scan line fill): xác định các phần giao của các dòng quét kế tiếp nhau với đường biên của vùng tô, sau đó sẽ tô màu các điểm thuộc về phần giao này.



Tô theo đường biên (tô loang-boundary fill): bắt đầu từ một điểm trong vùng tô và từ đó loang dần ra cho tới khi gặp các điểm biên.



Thuật toán tô màu theo dòng quét



Thuật toán chung:

- Giả sử vùng tô là mộ đa giác n đỉnh

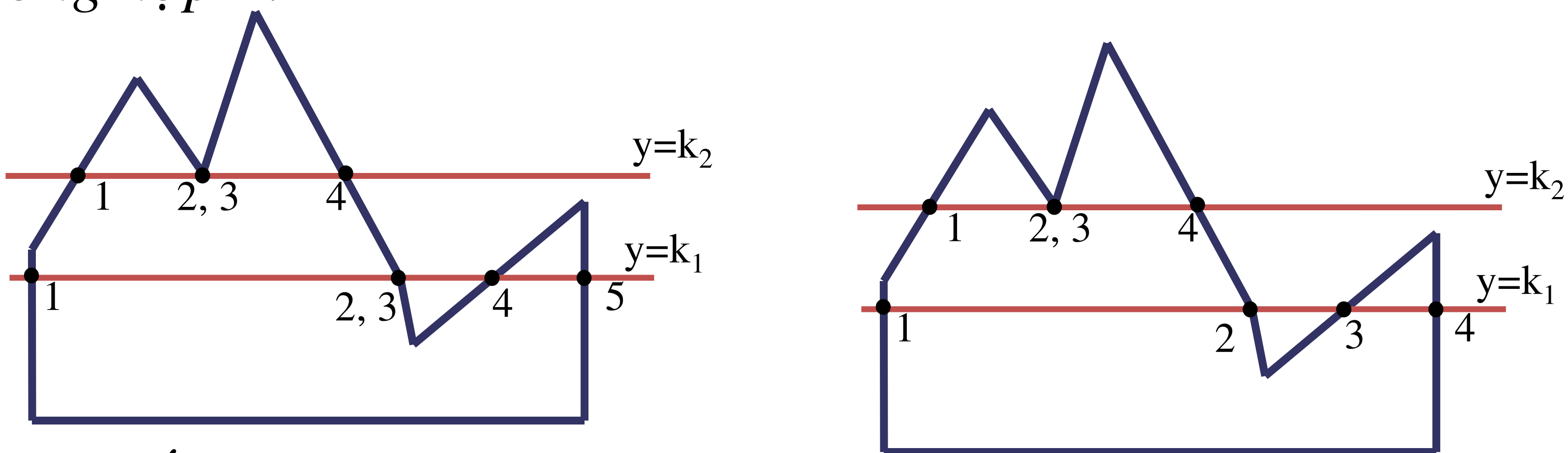
  $P_i(x_i,y_i)$, i=1,2...n.

- Các bước chính của thuật toán như sau:
  - Tìm y_{top} và y_{bottom} là gía trị lớn nhất và nhỏ nhất của tập các tung độ của các đỉnh da giác đã cho.
  - Ứng với mỗi dòng quét y=k ($y_{bottom} \le k \le y_{top}$), lặp:
    - Tìm tất cả hoành độ giao điểm của dòng quét y=k với các cạnh của đa giác.
    - Sắp xếp các hoành độ giao điểm theo thứ tự tăng dần: x_1, x_2,...
    - Tô màu các đoạn thẳng trên đường thẳng y=k lầ lượt được giới hạn bởi các cặp hòanh độ giao điểm (x_1,x_2), (x_3,x_4), (x_5,x_6)... (x_{2m-1},x_{2m})

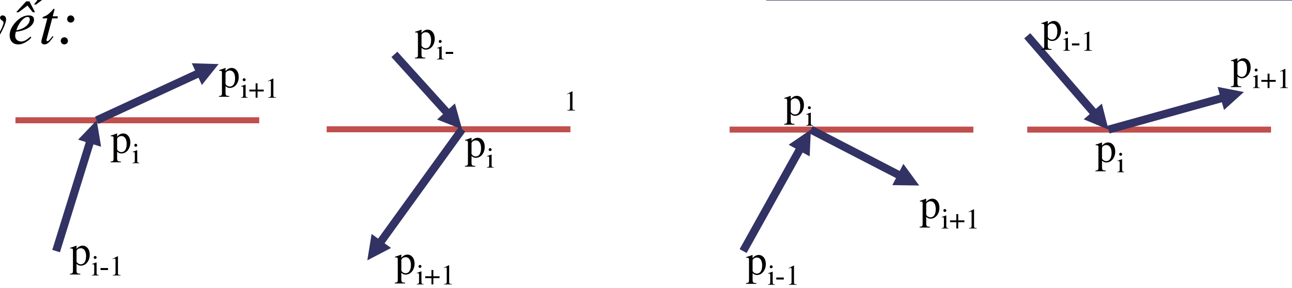
Thuật toán tô màu theo dòng quét

Một số trường hợp:

- *Trường hợp 1*:



- *Giải quyết:*



(a) (b)

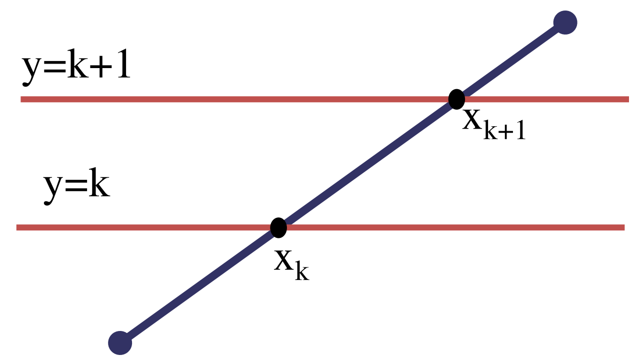
Quy tắc tính 1 giao điểm (a) và 2 giao điểm (b)

Thuật toán tô màu theo dòng quét

- *Trường hợp 2:* Nếu giải hệ phương trình tìm giao điểm của cạnh đa giác với mỗi dòng quét sẽ gặp các phép toán nhân, chia... trên số thực. Điều này sẽ làm giảm tốc độ của thuật toán khi phải lặp đi lặp lại nhiều lần ứng với mỗi dòng quét.
- *Giải quyết:* Nếu gọi x_k và x_{k+1} lần lượt là hoành độ giao điểm của một cạnh nào đó với các dòng quét y=k và y=k+1. Ta có:

$$x_{k+1} - x_k = \frac{y_{k+1} - b}{m} - \frac{y_k - b}{m} = \frac{y_{k+1} - y_k}{m} = \frac{k+1-k}{m} = \frac{1}{m}$$

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

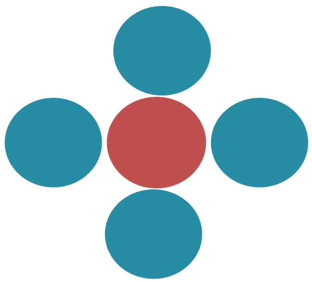


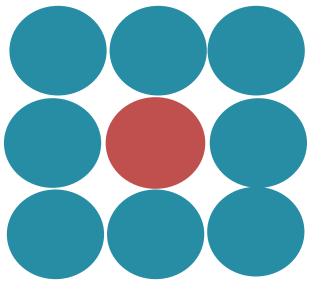
Thuật toán tô màu theo đường biên

- **Ý tưởng**:

Bắt đầu từ điểm P(x,y) nằm bên trong vùng tô, kiểm tra các điểm lân cận của P đã được tô màu hay có phải là điểm biên hay không, nếu không phải là điểm đã tô và và không phải là điểm biên thì ta sẽ tô màu nó. Quá trình này được lặp lại cho đến khi không còn tô được điểm nào nữa thì dừng.

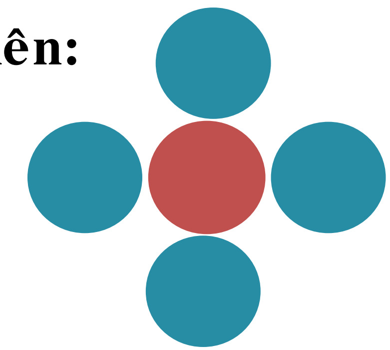
⇨ chọn điểm lân cận: chọn 4 hay 8 lân cận đối với điểm đang xét.



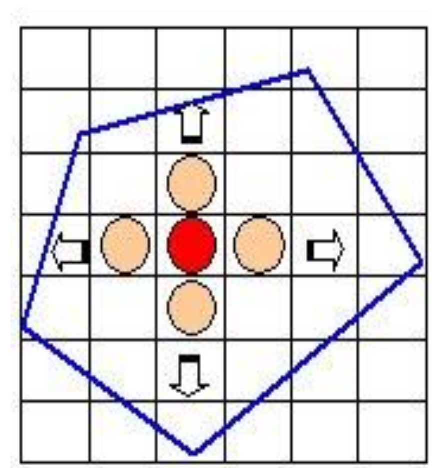


Thuật toán tô màu theo đường biên

- **Thủ tục minh hoạ thuật toán tô màu theo đường biên:**

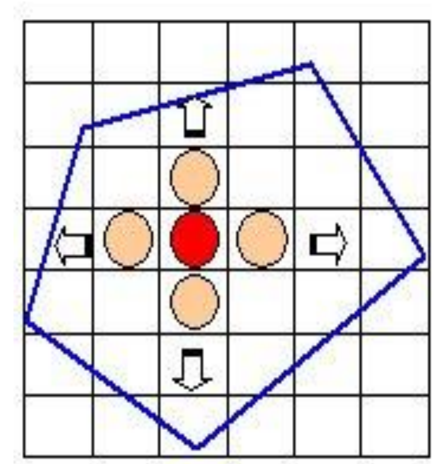


```
void      Toloang(int x,int  y,int  mauto,int  mauvien)
{         int   mau;          mau=getpixel(x,y);
          if (mau != mauvien) & (mau != mauto)
          {     Putpixel(x, y, mauto);
                Toloang (x-1, y, mauto,mauvien);
                Toloang (x, y+1, mauto,mauvien);
                Toloang (x+1, y, mauto,mauvien);
                Toloang (x, y-1, mauto,mauvien);
          }
}
```



☞Nhược điểm của phương pháp đệ quy là không thực hiện được khi vùng loang có diện tích lớn (dẫn đến tràn Stack).

Thuật toán tô màu theo đường biên



Phương pháp không đệ quy:

- **Bước 1**:Khởi tạo hàng đợi (hoặc stack) với phần tử đầu tiên là P(x,y) đã được tô.
- **Bước 2**: Khi hàng đợi (hoặc stack) không rỗng thì:

+ Lấy ra từ hàng đợi (hoặc stack) một điểm Q.

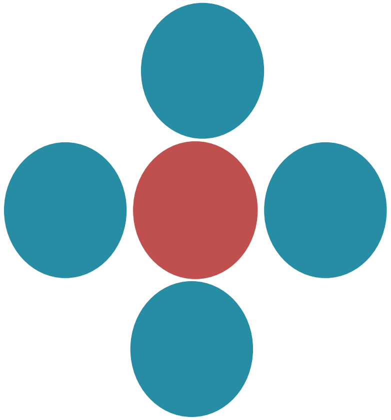
+ Tìm các điểm lân cận của Q chưa tô thì tô chúng và đưa chúng vào hàng đợi (hoặc stack) .

Bước 2 này được lặp đi lặp lại cho đến khi hàng đợi (hoặc stack) rỗng

Thuật toán tô màu theo đường biên

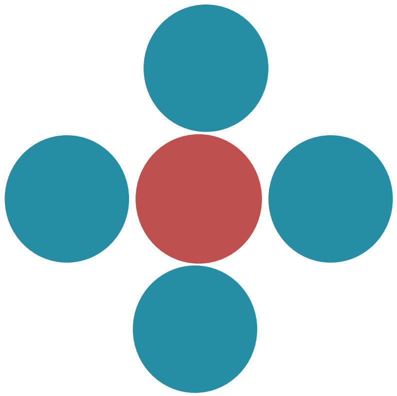
```
struct DS  {  int x,y;
              struct DS *next;       };
struct DS *dau;

void push(int x,int y){
        struct DS *P;
        P=(DS*) calloc(1,sizeof(DS));
        P->x=x;P->y=y;P->next=NULL;
        if(dau!=NULL)     P->next=dau;
        dau=P;
}
void pop(int *x,int *y){
        struct DS *P;
        P=dau;   dau=dau->next;  *x=P->x;   *y=P->y;
        free(P);
        }
```



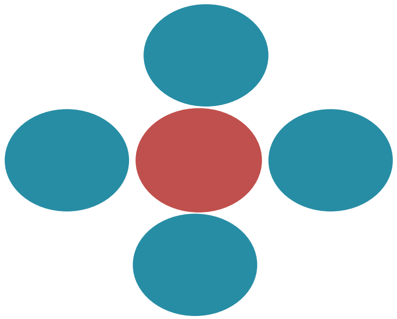
Thuật toán tô màu theo đường biên

```
void tolancan(int x,int y, int mauto, int mauvien)
 {
   int mau;
   mau=getpixel(x,y);
   if ((mau!=mauto)&&(mau!=mauvien))
        {
              putpixel(x,y,mauto);
              push(x,y);
        }
 }
```



Thuật toán tô màu theo đường biên

```
void toloang_stack(int x0,int y0,int mauto,int mauvien)
{       int x,y;
        putpixel(x0,y0,mauto);
        dau=NULL;
        push(x0,y0);
        while(dau!=NULL)
        {       pop(&x,&y);
                tolancan(x-1,y,mauto,mauvien);
                tolancan(x+1,y,mauto,mauvien);
                tolancan(x,y+1,mauto,mauvien);
                tolancan(x,y-1,mauto,mauvien);
        }
}
```



...Tô màu

Flood Fill (Boundary fill)	Scan line fill (Scan Conversion)
Đơn giản	Phức tạp hơn
Thuật toán rời rạc hóa trong không gian màn hình	Thuật toán rời rạc hóa trong đối tượng hoặc/và không gian màn hình
Yêu cầu gọi hệ thống GetPixel/Val	Độc lập với thiết bị
Đòi hỏi điểm seed	Không đòi hỏi điểm seed
Yêu cầu stack rất lớn	Yêu cầu stack nhỏ

Các khái niệm cơ sở về Font

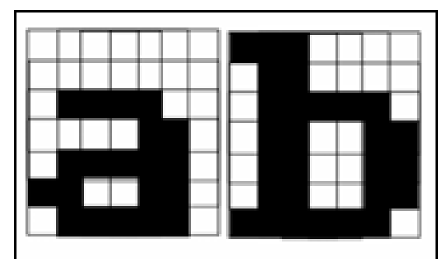
- Phông chữ được *Guttenberg* thiết kế. Được sử dụng từ nhiều thế kỷ. Ngày nay rất phong phú.
- Phông là tập đầy đủ các ký tự có kiểu dáng (style)
  - Weight: light, normal, **bold**
  - Shape: round, oval, straight
  - Posture: Oblique, *Italic*
  - Serif, sans-serif

Các loại Font

- font bitmap (raster)
- font véctơ
- font TrueType

Font raster (bitmap)

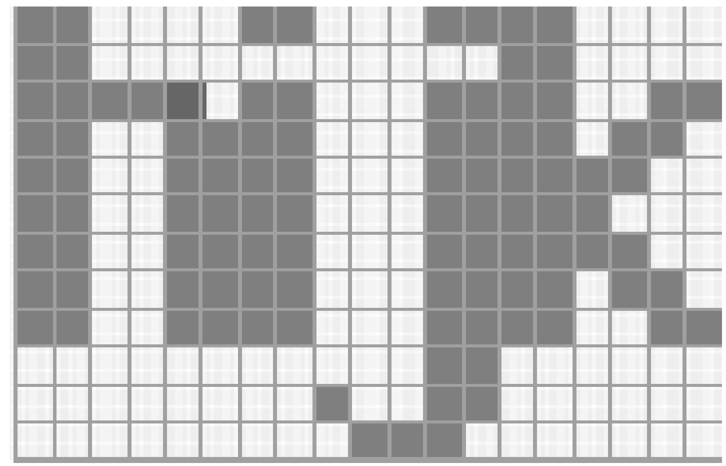
- Là loại phông đầu tiên của màn hình máy tính. Ngày nay vẫn đang sử dụng
- Ban đầu font bitmap được nhúng trong các vỉ điều khiển màn hình, máy in
- Trên cơ sở định nghĩa mỗi ký tự với một font chư cho trước là một bitmap chữ nhật nhỏ
  - Font/typeface: set of character shapes
  - Fontcache: các ký tự theo chuỗi liên tiếp nhau trong bộ nhớ
  - Dạng cơ bản: (thường N, nghiêng I, đậm B, nghiêng đậmB+I)
  - Thuộc tính: colour, size, spacing and orientation



…Vẽ chữ

Font raster (bitmap)

```
Typedef struct   {
        int leftx,
        int width; //độ rộng chữ
  } Charlocation; //Vị trí của text
Typedef struct   {
        CacheId;
        Heiglit; // Độ cao chữ
        CharSpace; // Khoảng cách giữa các ký tự
        Charlocation Table [128];
  } fontcache
```



Nhận xét về font bitmap

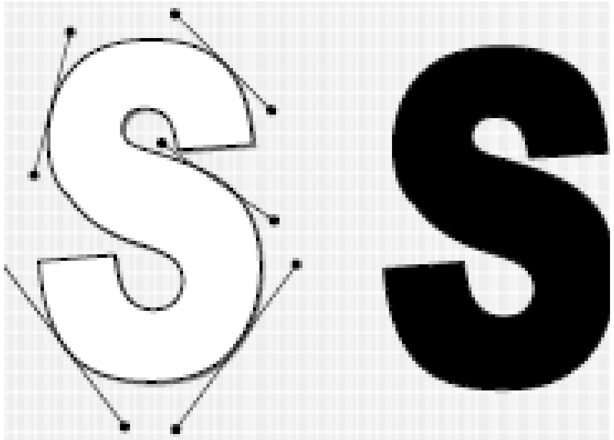
- Có độ rộng và độ cao cố định
- Lưu trữu: tách biệt các ảnh phông. Vị trí byte thứ nhất của khối bitmap trong bộ font:

Offset = (ASCII code) * (Bytes per character)

- Ưu điểm
  - Hiển thị nhanh, đơn gian trong việc sinh ký tự
  - Dễ tạo lập và dễ sửa đổi
- Nhược điểm
  - Kích thước không đổi
  - Co dãn, các phép biến đổi (I, B, scale) đòi hỏi phải lưu trữ thêm
  - Dung lượng lưu trữ lớn

Font vécto

- Không sử dụng Bitmap để mô tả ký tự
- Sử dụng ngôn ngữ mô tả nào đó
  - Ngôn ngữ mô tả bao gồm các lệnh như Line, Curve, Polygon...
  - Tọa độ: tương đối trong chữ nhật chứa ký tự
  - Chương trình con xử lý các lệnh để hiển thị



Font véctơ

- Ưu điểm
  - Chất lượng cao
  - Dễ co giãn, trơn tru, dễ tạo lập hiệu ứng đặc biệt: xoay, gập, cong...
  - Lưu trữ gọn nhẹ
- Nhược điểm:
  - Phức tạp (tính toán phương trình)
  - Khi hiển thị font nhỏ: chậm hơn bitmap font.
  - Vấn đề hiển thị font nét chữ dày.

Font True Type

- True Type là công nghệ font của Apple Computer Inc. (1987); Tác giả: Kathryn Weisberg, Sampo Kaasila...
- MS bắt đầu sử dụng True Type vào đầu 1992 trên Windows 3.1
- Công nghệ True Type bao gồm:
  - Các tệp chứa True Type Fonts (TTF)
  - Bộ raster hóa True Type của hệ điều hành (MacOS, Windows...) trước khi hiển thị, in trên giấy.
- OpenType: 5-1996 MS và Adobe System kết hợp công nghệ True Type với PostScript.

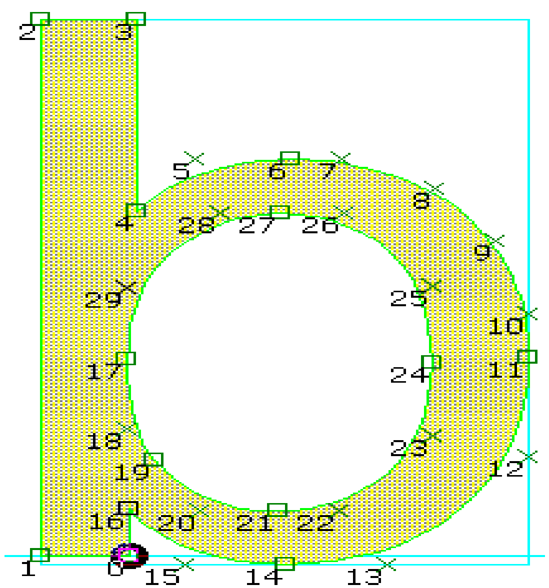
Font True Type

- Tệp TTF chứa
  - Mô tả hình dạng ký tự
  - Các thông tin khác: tên font, bản quyền, hãng sản xuất..
- Mô tả ký tự trong TTF:
  - Mô tả đường viền font -> TTF là font outline
  - Mô tả toán học của ký tự từ dãy các điểm
  - Viền ký tự được tạo bởi các line và B-splines toàn phương
    - Mỗi B-Spline tương ứng với dãy các Bezier bậc 2 được xác định bởi ba điểm điều khiển.
    - Thí dụ: có ba điểm điều khiển (A_x, A_y), (B_x B_y), (C_x, C_y)

    Các điểm P trên đường cong được tính với t trong khoảng [0,1]

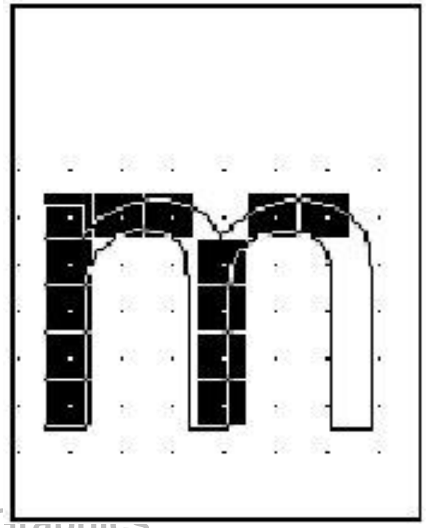
Font True Type

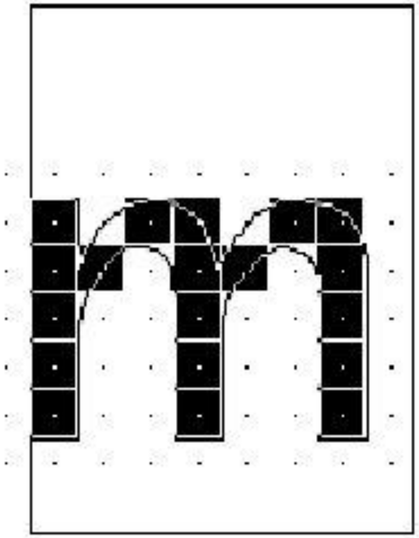
- Thí dụ mô tả outline của chữ **b** Arial:
  - các điểm 4, 5, 6 xác định Bézier
  - các điểm từ 6 đến 11 xác định B-Spline, chứa 4 Béziers
- Đánh số điểm điều khiển
  - Theo thứ tự: tô màu phía phải
  - Đánh số kế tiếp giữa các contour
- Chuyển outline sang bitmap

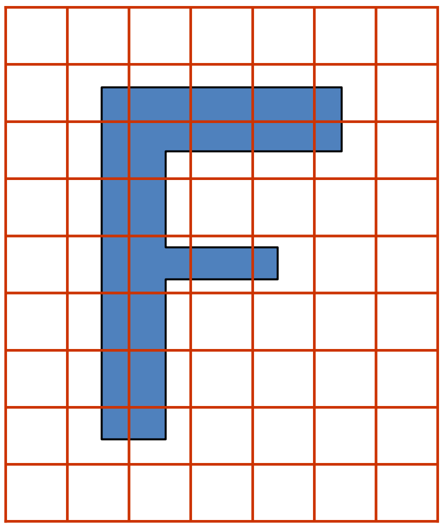


Font True Type

- Hinting
  - Sau khi chuyển outline sang bitmap thường phải hiệu chỉnh nét vẽ để có chất lượng cao
  - Mỗi nét trong phông có chương trình con bằng ngôn ngữ thông dịch (tựa asm) kèm theo để thực hiện hiệu chỉnh.
    - Ngôn ngữ này có khoảng 150 lệnh, kích thước lệnh 1 byte







Font True Type

- MS Windows sử dụng TTF?
  - Nạp font từ tệp
  - Trình Scaler:
    - Ánh xạ tọa độ điểm điều khiển của glyph từ FUnit sang tọa độ thiết bị (pixel/dot). (Co dãn đường viền đến kích thước yêu cầu theo mật độ cho trước trên thiết bị ra).
  - Trình Interpreter:
    - Áp dụng 'hint' cho đường viền: biến đổi đường cong để hình thành đường viền đã hiệu chỉnh (grid-fitting).
  - Trình Scan-converter:
    - Tô trong đường viền đã hiệu chỉnh bằng các pixel để tạo bitmap cho chữ đặc.
  - Hiển thị / in các bitmap ký tự.

Biến đổi 2D

- Các phép biến đổi hình học cơ sở
- Phép biến đổi ngược
- Hệ toạ độ thuần nhất
- Kết hợp các phép biến đổi

Các phép toán cơ sở với ma trận

- Cộng, trừ ma trận

[A(m, n)] + [B(m, n)] = [C(m, n)]

$$\lfloor c_{ij} \rfloor = \lfloor a_{ij} \rfloor + \lfloor b_{ij} \rfloor$$

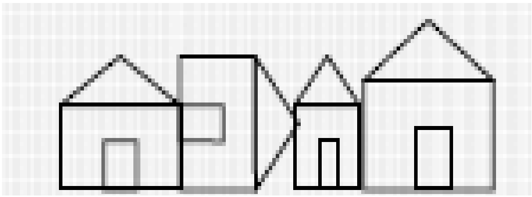
- Nhân hai ma trận

[A(m, n)] [B(n, p)]= [C(m, p)]

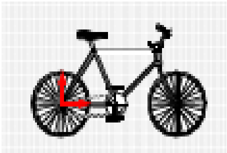
$$c_{jk} = \sum_{i=1}^{n} a_{ji} b_{ik}$$ j=1,...,m và k=1,...,p

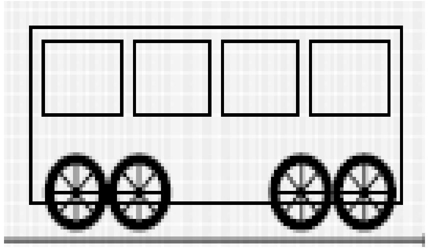
Ứng dụng biến đổi 2D

- Mô hình hóa (modeling)
  - Định vị và thay đổi kích thước các phần của đối tượng phức tạp



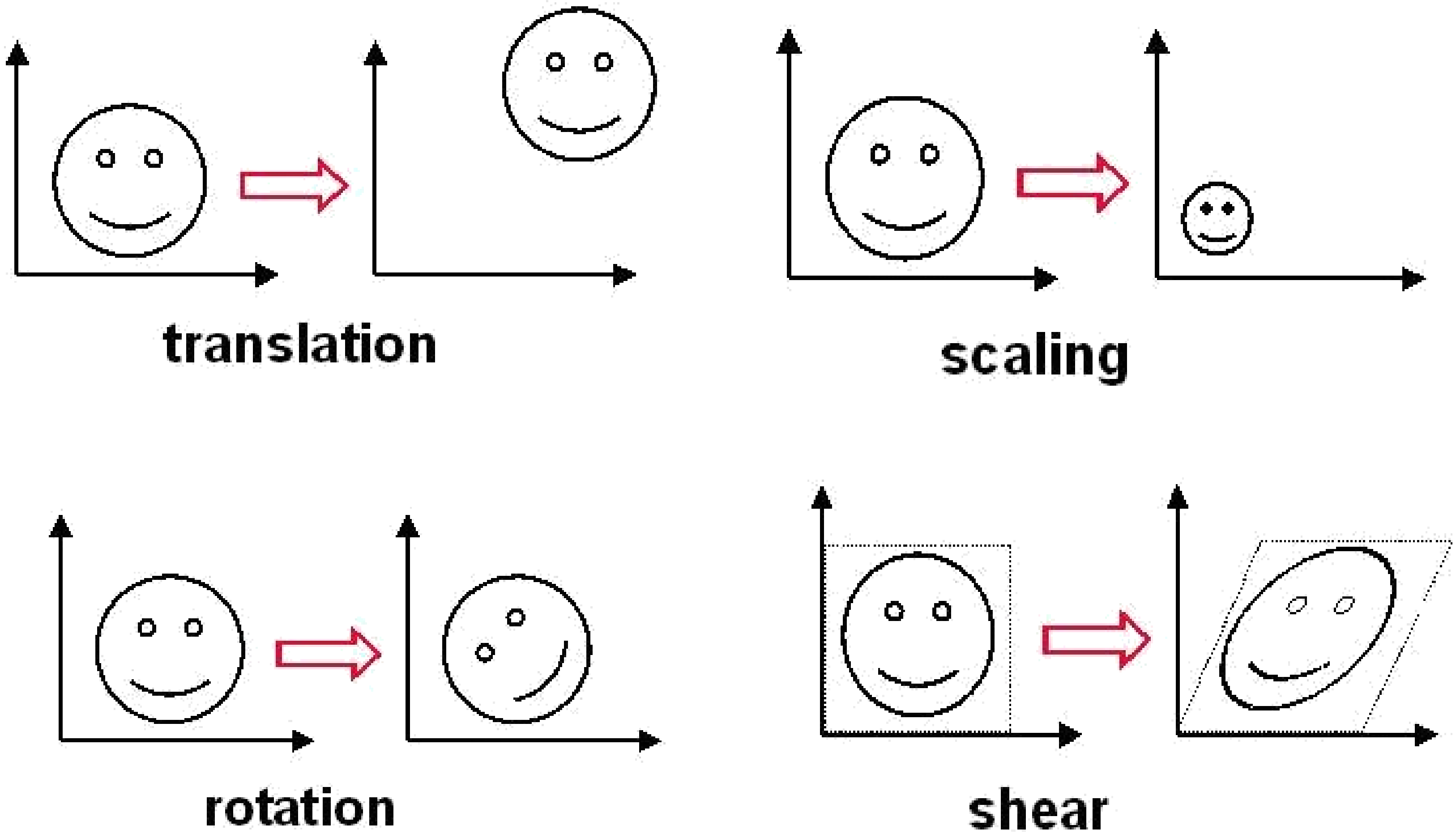
- Quan sát (viewing)
  - Định vị và quan sát camera ảo
- Animation
  - Xác định đối tượng chuyển động và thay đổi theo thời gian như thế nào.

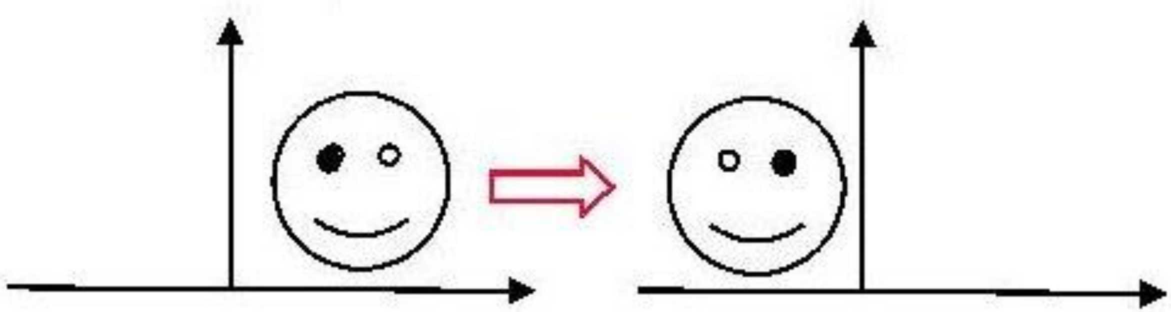




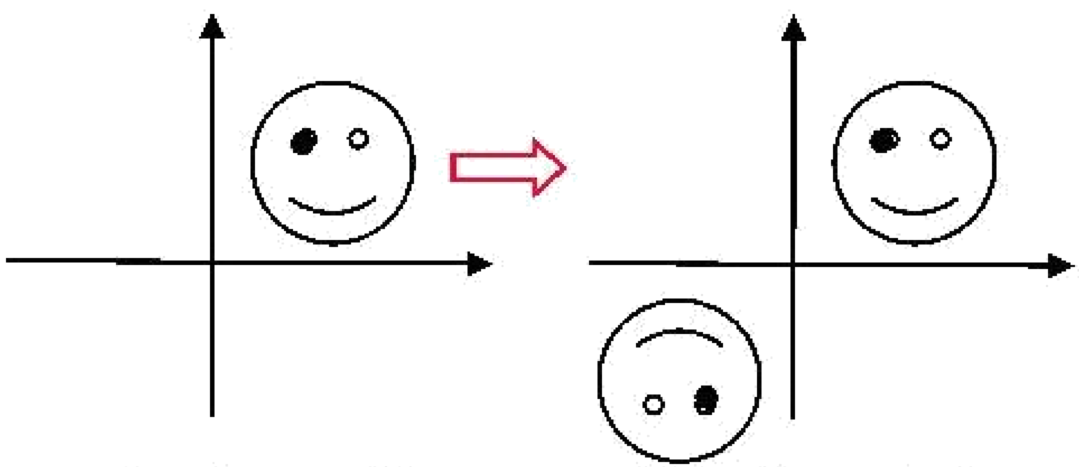
Các phép biến đổi hình học cơ sở bao gồm:

- Tịnh tiến (translation)
- Quay (rotation)
- Tỉ lệ (scaling).
- Đối xứng (reflection)
- Biến dạng (shearing).


translation
scaling
rotation
shear



reflection with respect to the y-axis



reflection with respect to the origin

Một phép biến đổi điểm là một ánh xạ T được định nghĩa:

$$T: R^2 \rightarrow R^2$$

$$P(x,y) \rightarrow Q(x',y')$$

hay T là hàm số T(x,y) theo hai biến x,y: $\begin{cases} x' = f(x,y) \\ y' = g(x,y) \end{cases}$

Phép biến đổi affine là phép biến đổi với f(x,y) và g(x,y) là các hàm tuyến tính có dạng:

$$\begin{cases} x' = f(x,y) = ax + cy + e \\ y' = g(x,y) = bx + dy + f \end{cases}$$

với a, b, c, d, e, f $\in R$ và $ad-bc \neq 0$

$$\begin{cases} x' = f(x,y) = ax + cy + e \\ y' = g(x,y) = bx + dy + f \end{cases}$$

với a, b, c, d, e, f $\in$R và ad-bc$\neq$0

Dưới dạng ma trận, hệ này có dạng:

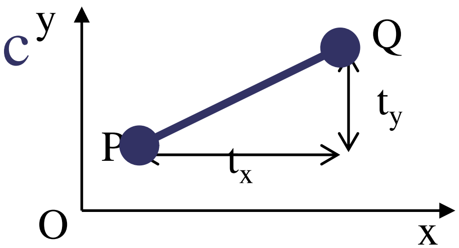
$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \end{bmatrix}$$

Trong đó: $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ là ma trận biến đổi.

$T = \begin{bmatrix} e & f \end{bmatrix}$ là vector offset hay vector tịnh tiến.

Phép tịnh tiến

Nếu gọi t_x và t_y là độ dời theo trục hoành và trục tung thì toạ độ của điểm mới Q(x’, y’) sẽ là:

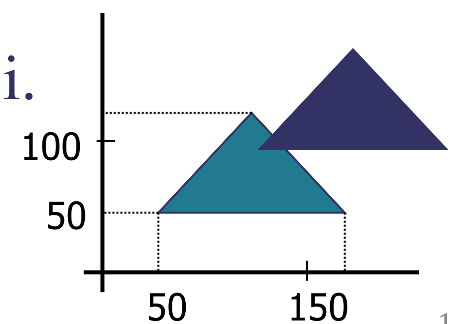


$$\begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

hay $$[x' \quad y'] = [x \quad y] \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + [t_x \quad t_y]$$

hay Q=P+T

Với: $T = [t_x \quad t_y]$ là vector tịnh tiến/vector độ dời.

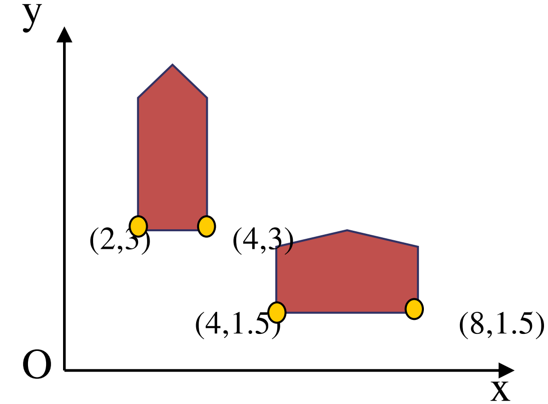


$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ là ma trận đơn vị.

Phép tỉ lệ

Gọi hệ số tỉ lệ: s_x và $s_{y,}$ phép tỉ lệ với tâm tỉ lệ là gốc toạ độ:

$$\begin{cases} x' = x.s_x \\ y' = y.s_y \end{cases} \quad \text{hay} \quad \begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



Biến đổi tỉ lệ với sx=2 và sy=0.5

hay Q=P.M Với $M = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$

Khi: $(s_x,s_y)=(-1,1) \rightarrow$ phép lật (flipping), có ảnh đối xứng qua trục y

$(s_x,s_y)=(1,-1) \rightarrow$ phép lật (flipping), có ảnh đối xứng qua trục x

$|s_x| , |s_y| < 1 \rightarrow$ thu nhỏ đối tượng.

$|s_x| , |s_y| > 1 \rightarrow$ phép phóng to đối tượng.

$s_x=s_y=s \rightarrow$ phép đồng dạng (uniform scaling)

Phép tỉ lệ

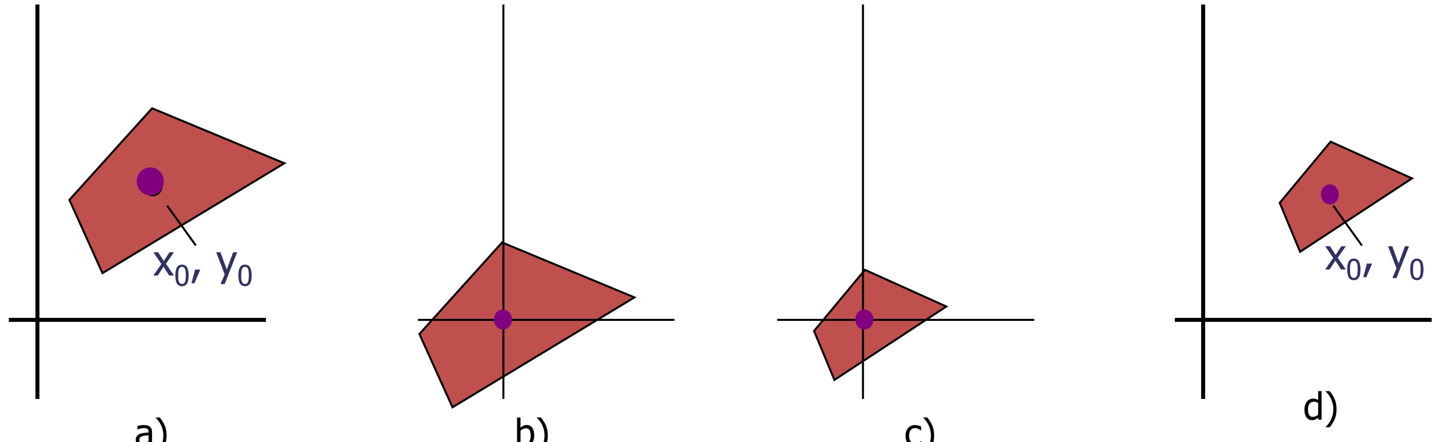
☞ **Phép biến đổi tỉ lệ với tâm tỉ lệ là điểm T(t_x,t_y):**

Xây dựng từ những phép biến đổi cơ bản sau:

+ Tịnh tiến theo vector ($-t_x$,$-t_y$) để đưa về gốc toạ độ O

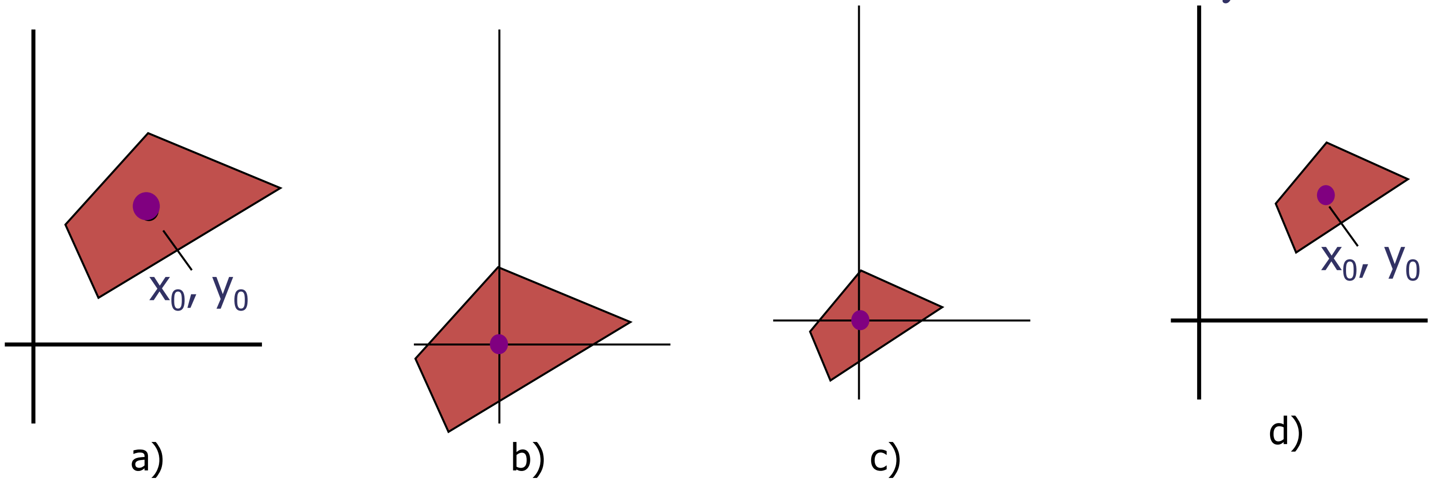
+ Biến đổi tỉ lệ quanh gốc toạ độ O theo hệ số tỉ lệ s_x và s_y

+ Tịnh tiến theo vector (t_x, t_y) để đưa về vị trí ban đầu.



Phép tỉ lệ

☞ Phép biến đổi tỉ lệ với tâm tỉ lệ là điểm $T(t_x,t_y)$:



Công thức biến đổi như sau: Q=(P-T).M+T Với $T = \begin{bmatrix} t_x & t_y \end{bmatrix}$

Hay $\begin{bmatrix} x' & y' \end{bmatrix} = \left(\begin{bmatrix} x & y \end{bmatrix} - \begin{bmatrix} t_x & t_y \end{bmatrix}\right) \times \begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix} + \begin{bmatrix} t_x & t_y \end{bmatrix}$

Phép quay

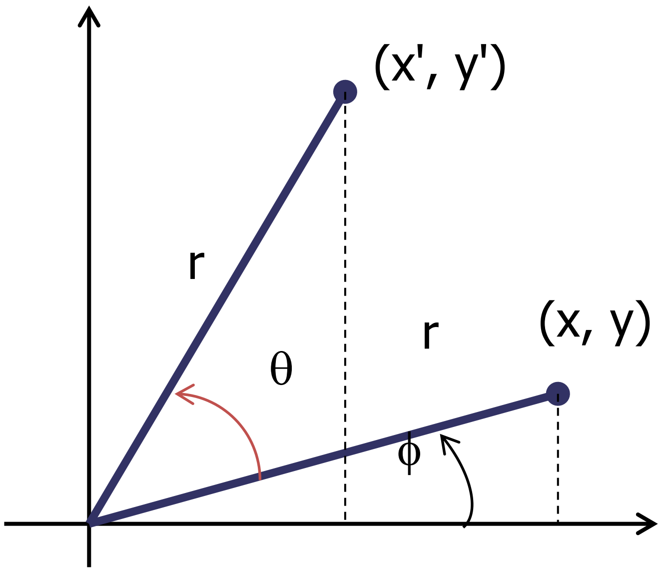
- Phép quay điểm P(x,y) quanh gốc toạ độ một góc θ :

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

$$x = r\cos\phi,\ y = r\sin\phi$$

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

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



Phép quay

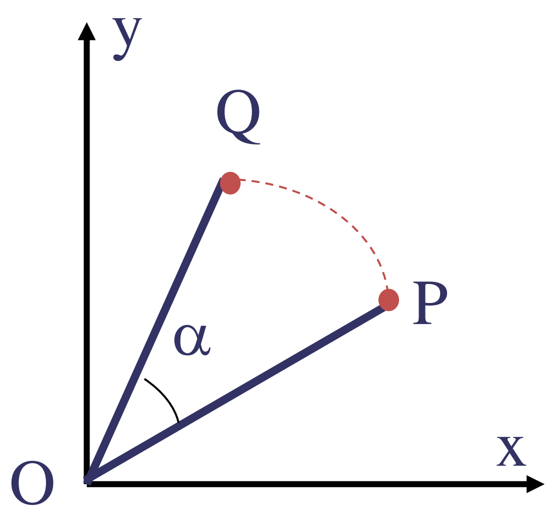
- ☞ Phép quay điểm P(x,y) quanh gốc toạ độ một góc α:

Ta có công thức biến đổi: $\begin{cases} x' = x.\cos\alpha - y.\sin\alpha \\ y' = x.\sin\alpha + y.\cos\alpha \end{cases}$

hay $\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$

hay Q=P.M

Với ma trận M: $M = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$



Phép quay

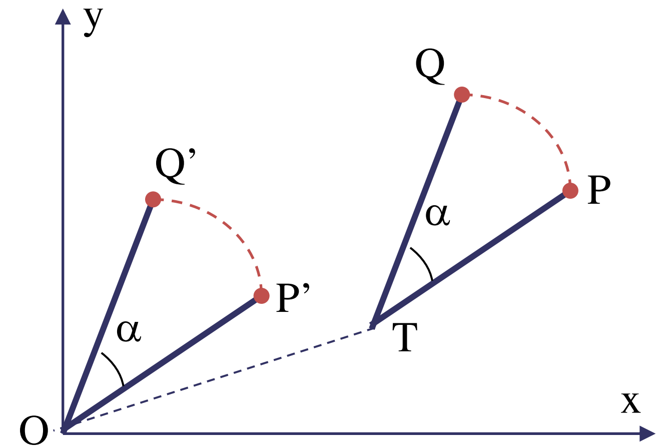
Phép quay quanh điểm $T(t_x,t_y)$ một góc α:

Xây dựng từ những phép biến đổi cơ bản sau:

+Tịnh tiến theo vector $(-t_x,-t_y)$

+Quay quanh gốc toạ độ O một góc α

+Tịnh tiến theo vector (t_x,t_y)



Công thức biến đổi như sau:

Q=(P-T).M+T   Với $T = \begin{bmatrix} t_x & t_y \end{bmatrix}$

hay: $\begin{bmatrix} x' & y' \end{bmatrix} = \left(\begin{bmatrix} x & y \end{bmatrix} - \begin{bmatrix} t_x & t_y \end{bmatrix}\right) \times \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} + \begin{bmatrix} t_x & t_y \end{bmatrix}$

Phép biến đổi ngược

Với phép biến đổi affine ta có:

$$Q = P.M + T \qquad \Rightarrow P = (Q-T).M^{-1}$$

Với $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ (ad-bc≠0) thì $M^{-1} = \dfrac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

☞Phép tịnh tiến: đơn giản là trừ x và y đi một lượng t_x và t_y.

☞ Phép biến đổi tỉ lệ: $M^{-1} = \dfrac{1}{s_x s_y}\begin{bmatrix} s_y & 0 \\ 0 & s_x \end{bmatrix} = \begin{bmatrix} 1/s_x & 0 \\ 0 & 1/s_y \end{bmatrix}$

☞ Phép quay: $M^{-1} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$

Hệ tọa độ thuần nhất (hormogeneous coordinates)

Công thức của phép biến đổi affine 2D là: Q=P.M+T

(với M là ma trận 2x2 và T là vector tịnh tiến)

Dạng ma trận của phép biến đổi affine 2D là:

$$\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \end{bmatrix}$$

Để chỉ còn phép nhân ma trận, ta đưa nó về dạng:

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \times \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$$

P(x,y,1) và Q(x',y',1) được gọi là toạ độ thuần nhất.

Vậy công thức của phép biến đổi theo hệ toạ độ thuần nhất là:

Q=P.M với M là ma trận 3x3.

Ta có : $M = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{pmatrix}$ Khi đó: $M^{-1} = \dfrac{1}{ad - bc} \begin{pmatrix} d & -b & 0 \\ -c & a & 0 \\ cf - de & be - af & 1 \end{pmatrix}$

Ma trận của các phép biến đổi cơ sở ở hệ toạ độ thuần nhất được viết lại:

-Phép tịnh tiến:

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tr_x & tr_y & 1 \end{pmatrix} \Rightarrow M^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -tr_x & -tr_y & 1 \end{pmatrix}$$

-Phép biến đổi tỉ lệ:

$$M = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow M^{-1} = \frac{1}{s_x s_y} \begin{pmatrix} s_y & 0 & 0 \\ 0 & s_x & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

-Phép quay:

$$M = \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow M^{-1} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Phép Đối xứng (reflection)

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

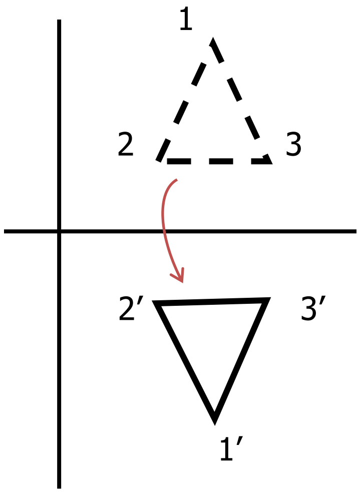
Đối xứng qua trục x

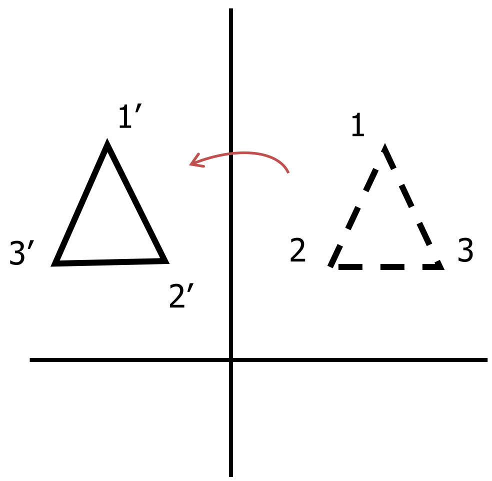
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

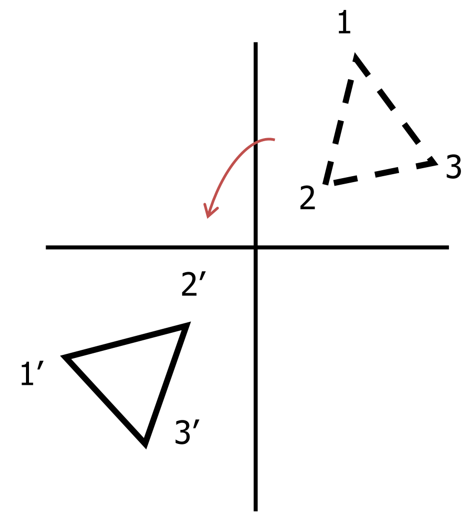
Đối xứng qua trục y

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Đối xứng qua gốc tọa độ







Kết hợp các phép biến đổi

Quá trình áp dụng các phép biến đổi liên tiếp để tạo nên một phép biến đổi tổng thể gọi là sự kết hợp các phép biến đổi.

Giả sử có 2 phép biến đổi: T_1: $P \rightarrow Q$

T_2: $Q \rightarrow W$

Ta tìm phép biến đổi kết hợp: T: $P \rightarrow W$

Ở hệ toạ độ thuần nhất:

+T_1 có ma trận biến đổi M_1 nên: $Q = P.M_1$

+T_2 có ma trận biến đổi M_2 nên: $W = Q.M_2$

$$= (P.M_1).M_2 = P.(M_1.M_2)$$

$$= P.M$$

Vậy kết hợp hai phép biến đổi cũng là một phép biến đổi có ma trận biến đổi : $M = M_1.M_2$

Kết hợp các phép tịnh tiến

Gọi T_1 và T_2 là 2 phép tịnh tiến sao cho:

$$T_1: \; P(x,y) \rightarrow P'$$

$$T_2: \; P' \rightarrow Q(x',y')$$

Ma trận biến đổi từ P sang Q là:

$$M = M_1 M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tr_{x1} & tr_{y1} & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tr_{x2} & tr_{y2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tr_{x1} + tr_{x2} & tr_{y1} + tr_{y2} & 1 \end{pmatrix}$$

Vậy $Q = P.M$ hay $[x' \quad y'] = [x \quad y] \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tr_{x1} + tr_{x2} & tr_{y1} + tr_{y2} & 1 \end{pmatrix}$

Vậy, kết hợp hai hay nhiều phép tịnh tiến là một phép tịnh tiến.

Kết hợp các phép tỉ lệ

☞ Phép biến đổi tỉ lệ với tâm tỉ lệ là gốc toạ độ:

Gọi T_1 và T_2 là 2 phép tỉ lệ sao cho: T_1: $P(x,y) \rightarrow P'$

T_2: $P' \rightarrow Q(x',y')$

Ma trận biến đổi từ P sang Q là:

$$M = M_1M_2 = \begin{pmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_{x1}s_{x2} & 0 & 0 \\ 0 & s_{y1}s_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Vậy Q=P.M hay $[x' \quad y'] = [x \quad y] \times \begin{pmatrix} s_{x1}s_{x2} & 0 & 0 \\ 0 & s_{y1}s_{y2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Vậy, kết hợp hai hay nhiều phép tỉ lệ là một phép tỉ lệ.

Kết hợp các phép tỉ lệ

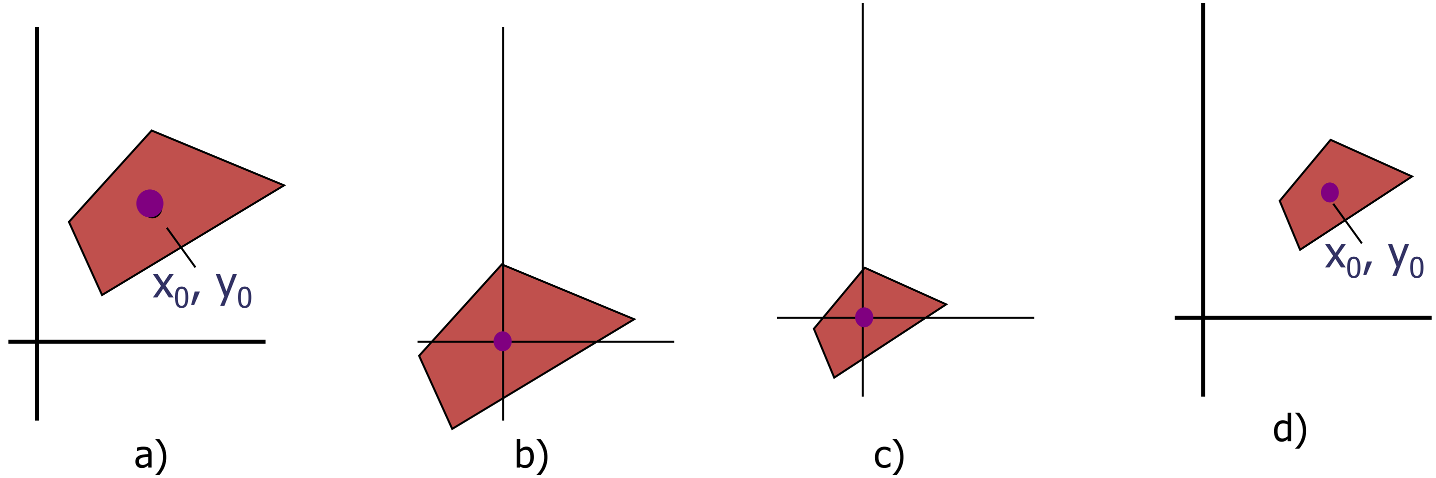
☞Phép biến đổi tỉ lệ với tâm tỉ lệ là điểm I:

Giả sử tâm tỉ lệ có gốc toạ độ I(x0 , y0). Phép tỉ lệ tâm I được kết hợp từ các phép biến đổi cơ sở sau:

-Tịnh tiến theo vector (-x0 ,-y0) để đưa tâm tỉ lệ về gốc toạ độ .(M1)

-Biến đổi tỉ lệ quanh gốc toạ độ O theo tỉ lệ sx và sy.(M2)

-Tịnh tiến theo vector (x0 ,y0) để đưa tâm tỉ lệ về vị trí ban đầu.(M3)



Kết hợp các phép tỉ lệ

☞Phép biến đổi tỉ lệ với tâm tỉ lệ là điểm I:

Ta có ma trận của phép biến đổi kết hợp là:

$$M = M1.M2.M3$$

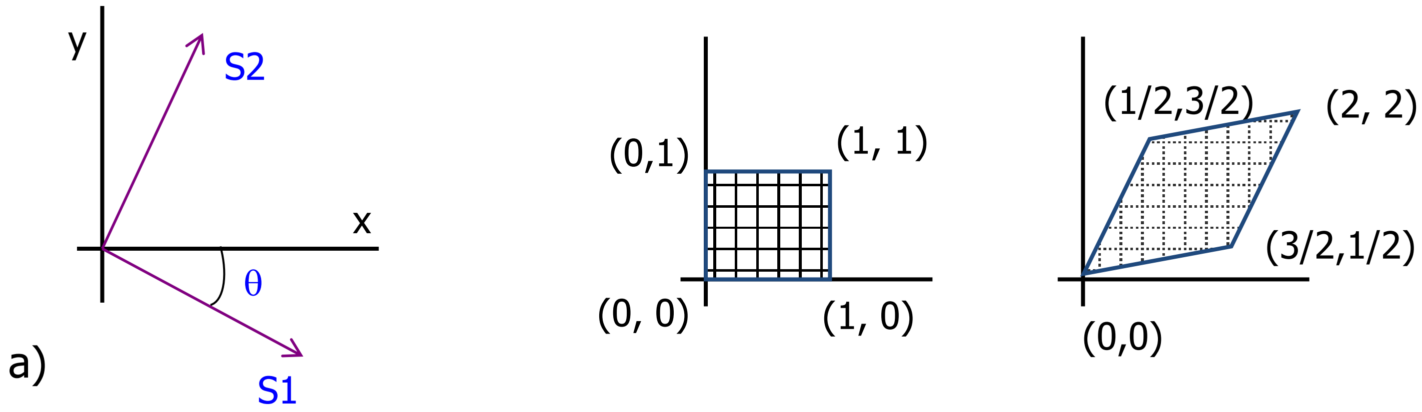
$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_0 & -y_0 & 1 \end{pmatrix} \times \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_0 & y_0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ (1-Sx).x_0 & (1-Sy).y_0 & 1 \end{pmatrix}$$

Kết hợp các phép tỉ lệ

☞Phép biến đổi tỉ lệ theo hướng tùy ý:

- Biến đổi tỉ lệ cơ bản
  - Tỷ lệ Sx và Sy áp dụng theo chiều trục x và y
- Tỉ lệ theo hướng tùy ý
  - Thực hiện chuyển đổi gộp: xoay và tỷ lệ
- Vấn đề
  - Cho trước hình vuông ABCD, hãy biến đổi tỉ lệ nó theo hướng như biểu diễn trên hình a) và theo tỷ lệ S1, S2.



Kết hợp các phép tỉ lệ

☞**Phép biến đổi tỉ lệ theo hướng tùy ý:**

- Giải pháp
  - Xoay hướng S1, S2 sao cho trùng với trục x và y (góc θ)
  - Áp dụng biến đổi theo tỷ lệ S1, S2
  - Xoay trả lại hướng ban đầu
- Ma trận tổ hợp

$$\begin{bmatrix} S1.\cos^2\theta + S2.\sin^2\theta & (-S1+S2)\sin\theta\cos\theta & 0 \\ (-S1+S2)\sin\theta\cos\theta & S1.\sin^2\theta + S2.\cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Kết hợp các phép quay

a.Phép quay quanh gốc toạ độ

Gọi T_1 và T_2 là 2 phép quay quanh gốc toạ độ sao cho:

$$T_1: \ P(x,y) \rightarrow P'$$

$$T_2: \ P' \rightarrow Q(x',y')$$

Ma trận biến đổi từ P sang Q là:

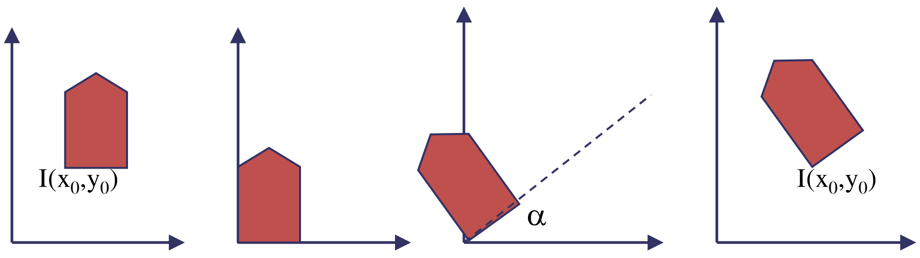
$$M=M_1M_2=\begin{pmatrix} \cos\alpha_1 & \sin\alpha_1 & 0 \\ -\sin\alpha_1 & \cos\alpha_1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\times\begin{pmatrix} \cos\alpha_2 & \sin\alpha_2 & 0 \\ -\sin\alpha_2 & \cos\alpha_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha_1+\alpha_2) & \sin(\alpha_1+\alpha_2) & 0 \\ -\sin(\alpha_1+\alpha_2) & \cos(\alpha_1+\alpha_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Vậy Q=P.M hay $\begin{bmatrix} x' & y' \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} \times \begin{pmatrix} \cos(\alpha_1+\alpha_2) & \sin(\alpha_1+\alpha_2) & 0 \\ -\sin(\alpha_1+\alpha_2) & \cos(\alpha_1+\alpha_2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Vậy, kết hợp hai hay nhiều phép quay quanh gốc toạ độ là một phép quay quanh gốc toạ độ.

Kết hợp các phép quay

b.Phép quay có tâm bất kỳ

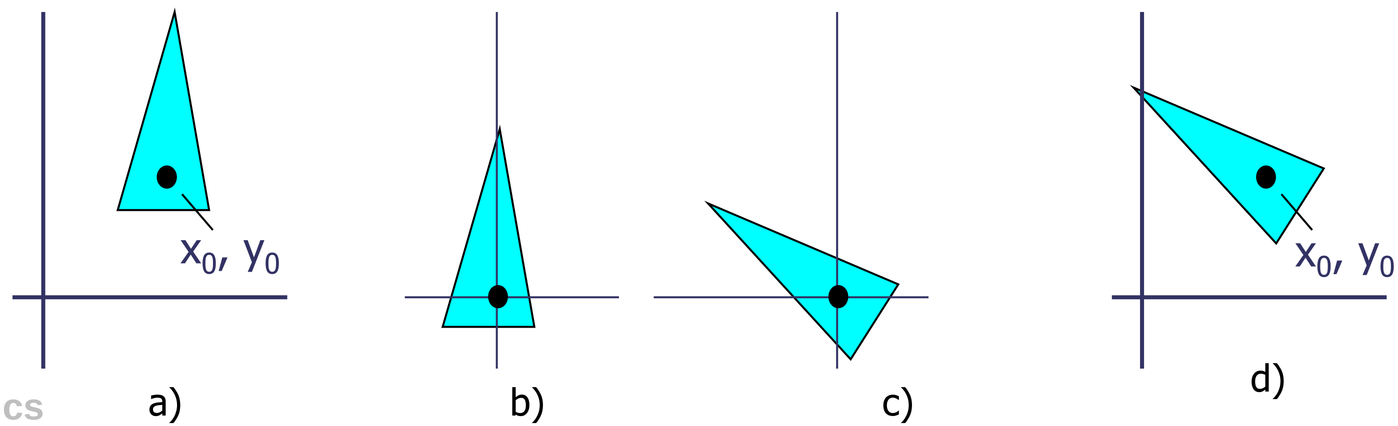


Phép quay quanh tâm I(x0 , y0) một góc α được kết hợp từ các phép biến đổi cơ sở sau:

-Tịnh tiến theo vector ($-x_0$,$-y_0$) để đưa tâm quay về gốc toạ độ (M_1)

-Quay quanh gốc toạ độ một góc α. (M_2)

-Tịnh tiến theo vector (x_0 ,y_0) để đưa tâm quay về vị trí ban đầu.(M_3)



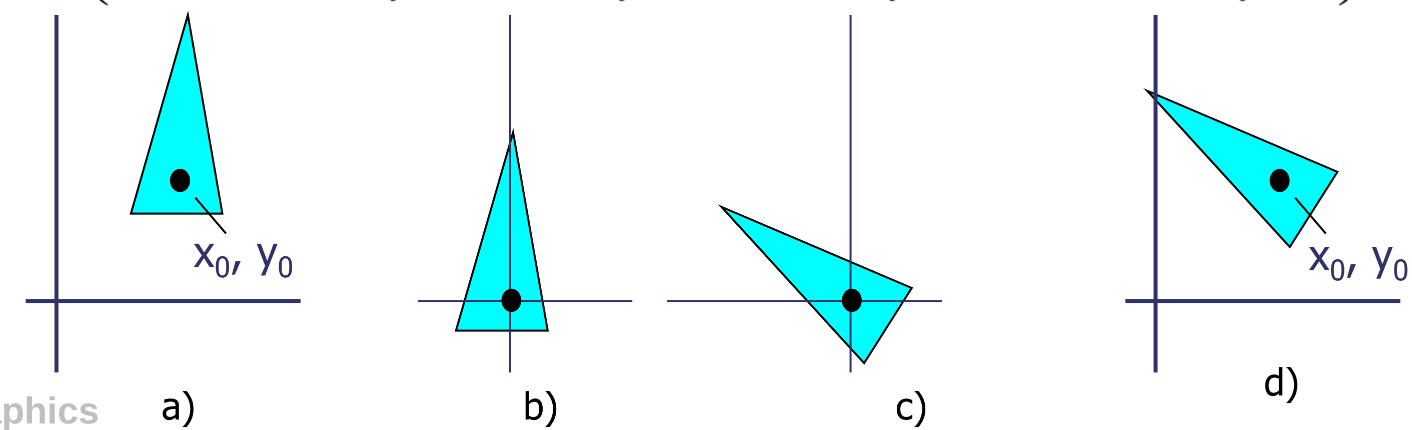
Kết hợp các phép quay

b.Phép quay có tâm bất kỳ

Ta có ma trận của phép biến đổi kết hợp là: M=M1.M2.M3

$$=\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -x_0 & -y_0 & 1\end{pmatrix}\times\begin{pmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{pmatrix}\times\begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ x_0 & y_0 & 1\end{pmatrix}$$

$$=\begin{pmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ (1-\cos\alpha).x_0+\sin\alpha.y_0 & -\sin\alpha.x_0+(1-\cos\alpha).y_0 & 1\end{pmatrix}$$



☞ **<u>Ví dụ1</u>**:

Tìm ma trận biến đổi trong hệ toạ độ thuần nhất của phép tỉ lệ với Sx=Sy=2

a.Tâm tỉ lệ là gốc toạ độ

b.Tâm tỉ lệ là điểm (5,2)

Ấp dụng để tìm ảnh của tam giác ABC với A(0,0), B(1,1) C(5,2)

☞ **<u>Ví du 2</u>**:

Tìm ma trận biến đổi trong hệ toạ độ thuần nhất của phép lấy đối xứng:

a.Qua dường thẳng y=x-1.

b.Qua điểm (1,2)

Áp dụng tìm ảnh của tam giác ABC với A(0,0), B(0,2), C(-2,0)

Bài tập

1. Tìm ma trận biến đổi để đối tượng đối xứng qua y=x và y=-x.
2. Cho tam giác A(3, 1), B(1, 3), C(3,3). Xác định tọa độ mới của các đỉnh tam giác sau khi:
   - Quay một góc 90^0 ngược chiều kim đồng hồ xung quanh điểm P(2, 2).
   - Phóng to tam giác lên hai lần, giữ nguyên vị trí điểm C.
3. Tìm ma trận biến đổi trong phép đối xứng qua đường thẳng nằm nghiêng có độ nghiêng m và đi qua điểm (0, c).

2D Graphics

Thank You…!