TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa Công nghệ Thông tin
Bộ môn KHMT

ĐỒ HỌA MÁY TÍNH
(Computer Graphics)

Ngô Trường Giang
E-mail: giangnt@tlu.edu.vn

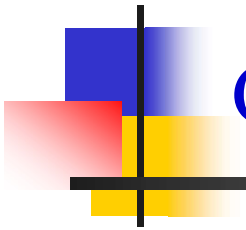
Nội dung

- Tổng quan đồ họa máy tính
- Màu và phối màu
- **Thuật toán cơ sở vẽ đồ họa**
- Các kỹ thuật trong đồ họa 2D
- Phép biến đổi đồ họa 2D
- Phép biến đổi đồ họa 3D
- Quan sát đồ họa 3D
- Mô hình hóa bề mặt

CÁC THUẬT TOÁN CƠ SỞ VẼ ĐỒ HỌA

- Các đối tượng đồ họa cơ sở
- Thuật toán vẽ đoạn thẳng
- Thuật toán vẽ đường tròn, elip

Tham khảo slides của PGS.TS Đặng Văn Đức – Viện CNTT, Viện HLKH&CN VN

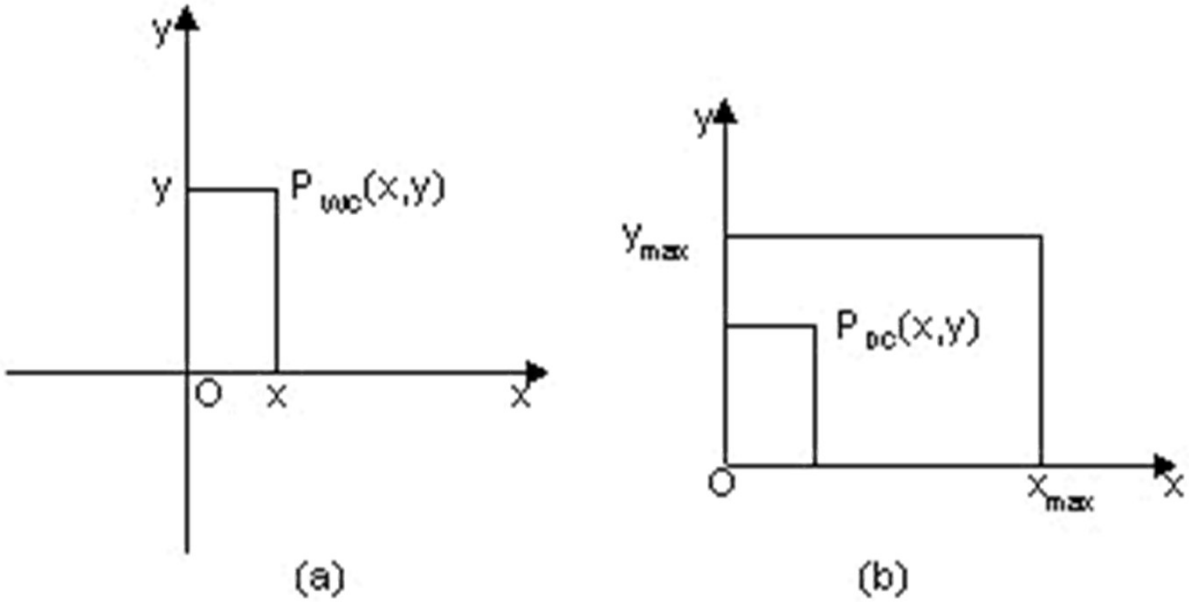


Các đối tượng đồ họa cơ sở

- Hệ tọa độ
- Điểm – Đoạn thẳng
- Đường gấp khúc
- Thuộc tính đường vẽ
- Vùng tô

Hệ tọa độ

- Hệ tọa độ thế giới thực
- Hệ tọa độ thiết bị



Điểm – Đoạn thẳng

- Điểm
  - P(x,y) => p(x,y,color)
  - P(x,y,z) => p(x,y,z,color)
- Đoạn thẳng
  - Phương trình đường thẳng đi qua hai điểm (x1, y1) và (x2, y2) có dạng sau :

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

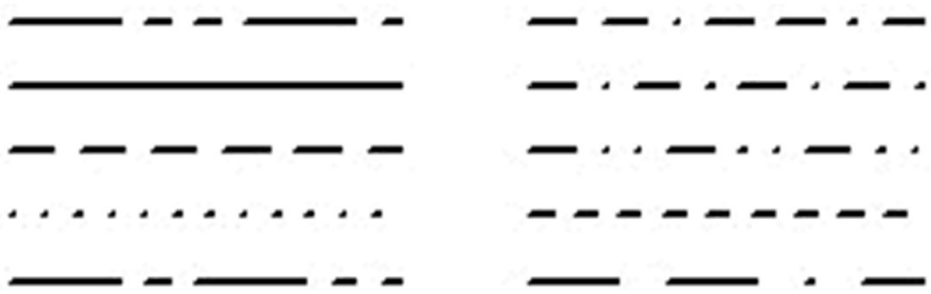
  - Một đoạn thẳng là một đường thẳng bị giới hạn bởi hai điểm đầu, cuối.

Đường gấp khúc

- Đường gấp khúc là tập các đoạn thẳng nối với nhau một cách tuần tự
- Một đa giác là một đường gấp khúc có điểm đầu và điểm cuối trùng nhau.

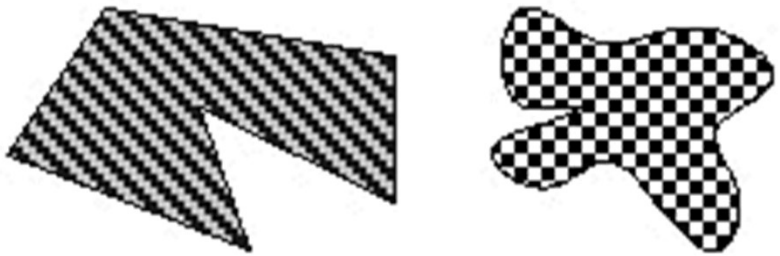
Thuộc tính đường vẽ

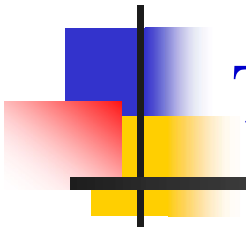
- Các thuộc tính của đoạn thẳng bao gồm :
  - Màu sắc
  - Độ rộng của nét vẽ.
  - Kiểu nét vẽ của đoạn thẳng



Vùng tô

- Một vùng tô bao gồm đường biên và vùng bên trong. Đường biên là một đường khép kín. Các thuộc tính của vùng tô bao gồm:
  - Thuộc tính của đường biên : chính là các thuộc tính như thuộc tính của đoạn thẳng.
  - Thuộc tính của vùng bên trong : bao gồm màu tô và mẫu tô.





Thuật toán vẽ đoạn thẳng

- Thuật toán DDA (Digital Defferencial Analyzer)
  hay thuật toán tăng dần (Basic Incremental Algorithm)
- Thuật toán Bresenham
- Thuật toán trung điểm (Midpoint)

Bài toán

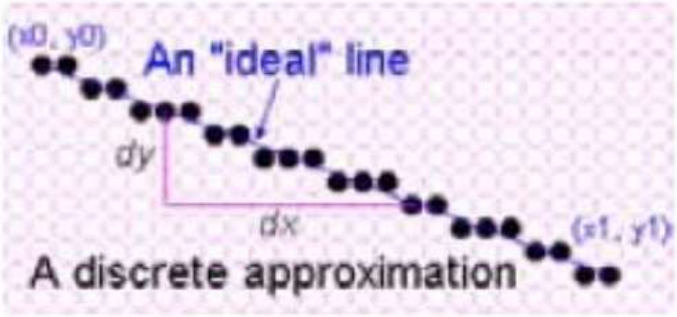
- Input: Điểm đầu(x1,y1), điểm cuối(x2,y2), màu tô C
- Output: Đoạn thẳng nối 2 điểm (x1,y1) (x2,y2) với màu C.
- Phương trình đoạn thẳng đi qua 2 điểm

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 \qquad a = \frac{y_2 - y_1}{x_2 - x_1} \qquad b = y_1 - a x_1$$

$$y = ax + b$$

Bài toán

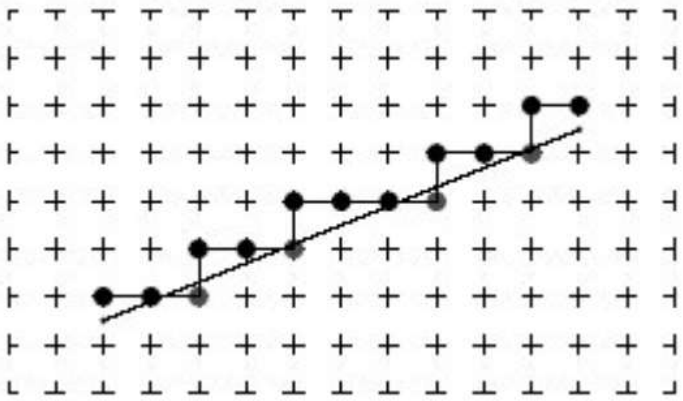
- Chuyển đổi đường quét (Rasterization)
- Biến đổi đường liên tục thành rời rạc (Sampling)
  - Scan conversion = Sampling



- Yêu cầu chất lượng đường vẽ
  - Hình dạng liên tục
  - Độ dày và độ sáng đều
  - Các pixel gần đường "lý tưởng" được hiển thị
  - Vẽ nhanh

Thuật toán DDA

- DDA- Digital Defferencial Analyzer = Finite defferences
  - Cho giá trị bước nhảy trên một trục tính giá trị bước nhảy trên trục kia theo phương trình y=ax+b
  - Với hệ số góc a trong khoảng [0, 1]:
    - dy=a.dx
    - Nếu dx=1 thì $y_{i+1}=y_i+a$
    - Làm tròn số vì a bất kỳ
  - Ý tưởng thuật toán: Với mỗi bước hãy tính số gia trên cơ sở bước trước đó.



Thuật toán DDA (tiếp)

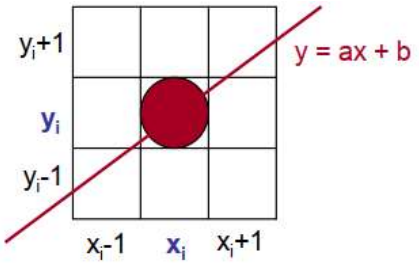
```
void DDALine(int x0, y0, x1, y1, color)
{
    Float dx, dy, y, m;
    dx = x1 – x0;
    dy = y1 – y0;
    m = dy/dx;
    y = y0;
    For( int  x=x0; x<= x1 ;x++)
    {
        WritePixel(x, int(y+0.5), color);
        y = y+m;
    }
}
```

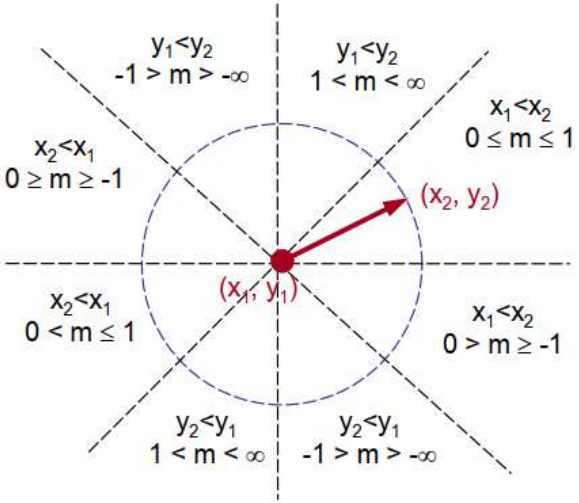
Thuật toán DDA

- Nhận xét thuật toán DDA
  - Không có phép nhân
  - Có phép chia và làm tròn số -> chậm
- Quy tắc tổng quát khi vẽ đồ họa:
  - Cộng và trừ nhanh hơn nhân
  - Nhân nhanh hơn chia
  - Sử dụng bảng để đánh giá hàm rời rạc nhanh hơn tính toán
  - Tính toán số nguyên nhanh hơn số thực
  - Tránh các tính toán không cần thiết nhờ nhận ra các trường hợp đặc biệt của đường vẽ

Thuật toán Bresenham vẽ line

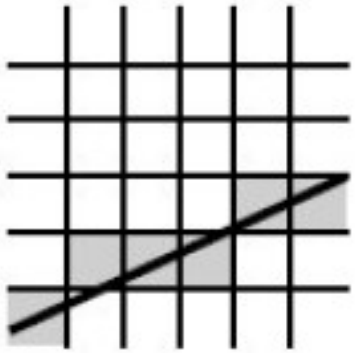
- Giả sử vừa vẽ điểm tại (x_i, y_i), bây giờ phải xác định điểm sẽ vẽ thuộc một trong 8 pixel liền kề: (x_i+1, y_i), (x_i-1, y_i), (x_i, y_i-1), (x_i, y_i+1)...
- Hình dạng đoạn thẳng phụ thuộc vào các giá trị dx và dy
  - dx=0 -> đ/thẳng song song trục y
  - dy=0 -> đ/thẳng song song trục x
  - dx>0 -> tọa độ x biến thiên tăng dần
  - dx<0 -> tọa độ x biến thiên giảm dần
  - Xét tương tự với dy
  - Nếu abs(dx)>abs(dy): y=f(x)
  - Nếu abs(dx)<abs(dy): x=f(y)





Thuật toán Bresenham vẽ line (tt)

slope > 1, cannot step along x



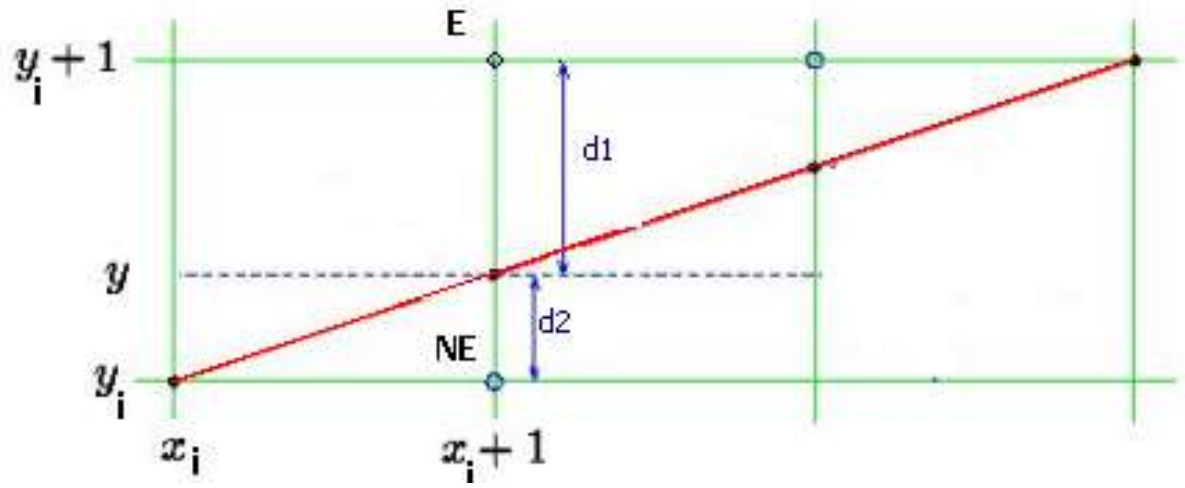
slope < 1, can step along x

To handle slope > 1, swap x and y

Thuật toán Bresenham vẽ line(tt)

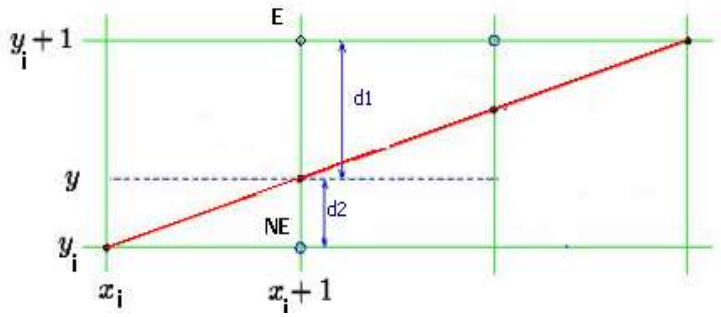
- Xét đoạn thẳng có hệ số góc $0 \le m \le 1$.
- Điểm vừa chọn là (x,y) -> điểm tiếp theo sẽ vẽ là (x+1,y) hay (x+1, y+1). Việc lựa chọn sẽ phụ thuộc vào khoảng cách d:

d=d1-d2



Thuật toán Bresenham vẽ line(tt)

- Nếu d>0 ⇔ d1>d2 chọn NE x_i+1,y_i
- Nếu d<0 ⇔ d1<d2 chọn E x_i+1,y_i+1
  - $d1=y_i+1-y=y_i+1-a(x_i+1)-b$
  - $d2=y-y_i=a(x_i+1)+b-y_i$
  - $d=d1-d2=-2a(x_i+1)+2y_i-2b+1$



- Nếu chọn E => $d_{newE}= d(x_i+1,y_i+1)$
  - $=-2a(x_i+1)+2y_i-2b+1 +2-2a= d+2-2a=d+ \Delta_E$
- Nếu chọn NE => $d_{newNE}= d(x_i+1,y_i)$
  - $=-2a(x_i+1)+2y_i-2b+1 -2a= d-2a=d+ \Delta_{NE}$

Thuật toán Bresenham vẽ line(tt)

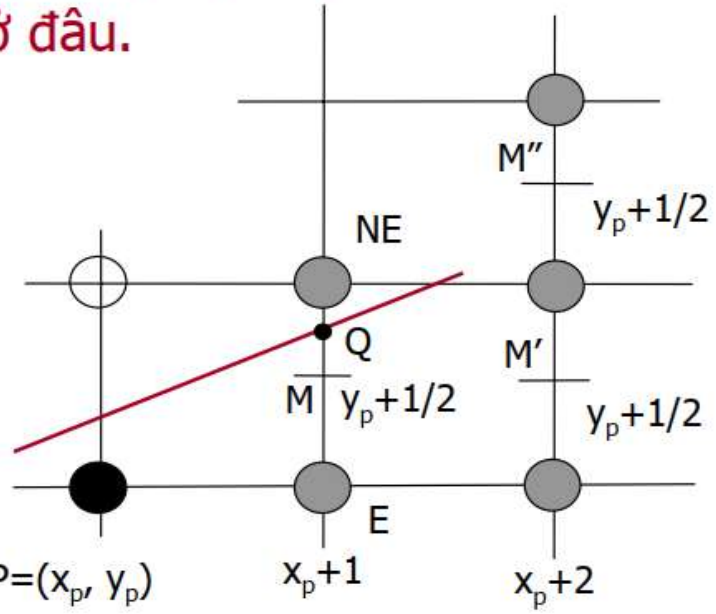
- Tìm d_{start}
- Giả sử điểm khởi đầu là xi,yi
  - $d_{start}= -2a(x_i+1)+2y_i-2b+1$
  - Mà ta có $y_i=ax_i+b \Rightarrow ax_i=y_i-b$
  - $d_{start} =-2a+1$
- Ta có a=dy/dx. Nhân các vế với dx ta có
  - $d_{start}=-2dy+dx$
  - $\Delta_E= -2dy+2dx$
  - $\Delta_{NE}= -2dy$

Thuật toán Bresenham vẽ line(tt)

1. Chọn vị trí vẽ đầu tiên là x1,y1
2. Tính tham số thứ nhất d=-2dy+dx
   - Nếu d >0 vị trí vẽ tiếp theo là x_i+1,y_i
   - Ngược lại vị trí vẽ tiếp theo là x_i+1,y_i+1
3. Tiếp tục tăng x để tính d tiếp theo từ d trước đó
   - Nếu trước đó d_i >0 thì $d_{i+1}=d_i-2dy$
   - Ngược lại thì $d_{i+1}=d_i-2dy+2dx$
   - Nếu d_{i+1} vị trí vẽ tiếp theo là x_i+2,y_i
   - Ngược lại vị trí vẽ tiếp theo là x_i+2,y_i+2
4. Lặp lại bước 3 cho đến khi x= x2

Thuật toán trung điểm vẽ line

- Pitteway công bố 1967, Van Aken cải tiến 1984
- Giả sử ta đã chọn P để vẽ, xác định pixel tiếp theo tại N hay NE
  - Giao của đường thẳng với Xp+1 tại Q, M là trung điểm của NE và E
- Ý tưởng: M nằm phía nào của đường thẳng, nếu M phía trên đường thẳng thì chọn E, ngược lại chọn NE.
- Nhiệm vụ: Xác định M ở đâu.



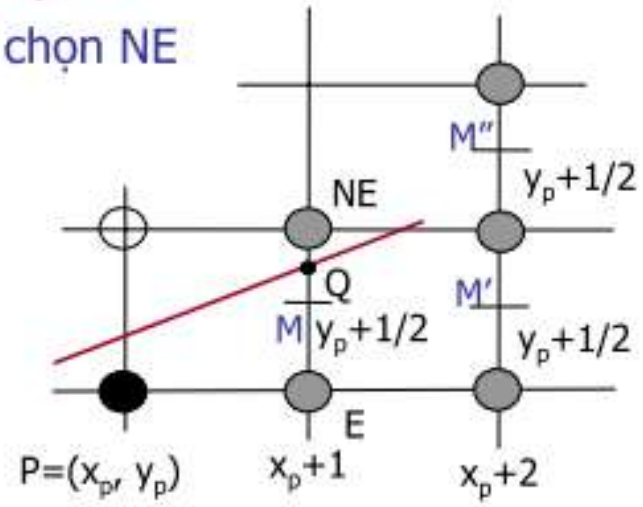
Thuật toán trung điểm vẽ line(tt)

- Phương trình đường thẳng: F(x,y)=ax+by+c

$$y = \frac{dy}{dx}x + B \qquad F(x,y) = \frac{dy}{dx}x + B - y = 0$$

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

  - a=dy, b=-dx, c=B.dx
- Giá trị hàm tại M: F(M)=F(xp+1, yp+1/2)=d
  - Nếu d>0, M nằm dưới đường thẳng -> chọn NE
  - Nếu d<0, M nằm phía trên -> chọn E
  - Nếu d=0, chọn E hay NE tùy ý



Thuật toán trung điểm vẽ line(tt)

- Giá trị của hàm tại M của điểm tiếp theo sẽ vẽ
  - Gọi giá trị d vừa tính là

$$d_{old} = a(x_p + 1) + b(y_p + \tfrac{1}{2}) + c$$

  - Giả sử vừa chọn E:

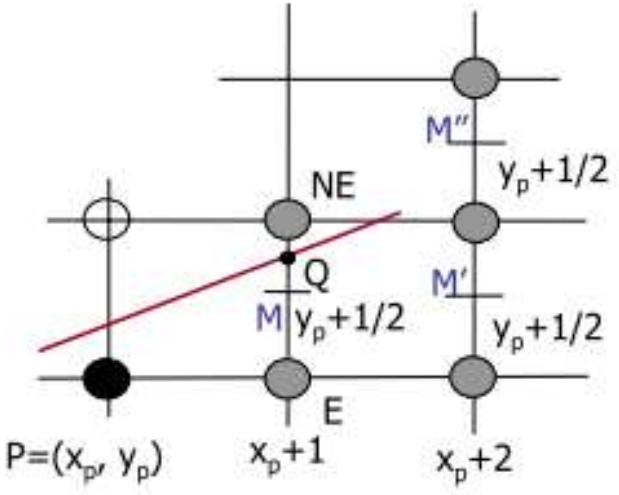
$$d_{new} = F(x_p + 2, y_p + \tfrac{1}{2}) = a(x_p + 2) + b(y_p + \tfrac{1}{2}) + c$$

$d_{new}=d_{old}+a=d_{old}+dy$ -> dy là số gia của điểm tiếp theo

  - Giả sử vừa chọn NE

$$d_{new} = F(x_p + 2, y_p + \tfrac{3}{2})$$
$$= a(x_p + 2) + b(y_p + \tfrac{3}{2}) + c$$

    - $d_{new}=d_{old}+a+b=d_{old}+dy-dx$
    - **dy-dx** là số gia của điểm tiếp theo



Thuật toán trung điểm vẽ line(tt)

- Tính giá trị khởi đầu của d
  - Giả sử vẽ đoạn thẳng từ (x0, y0) đến (x1, y1) -> trung điểm thứ nhất có tọa độ (x0+1, y0+1/2)

$$F(x_0+1, y_0+\tfrac{1}{2}) = a(x_0+1)+b(y_0+\tfrac{1}{2})+c =$$

$$a.x_0+b.y_0+c+a+\frac{b}{2} = F(x_0,y_0)+a+\frac{b}{2}$$

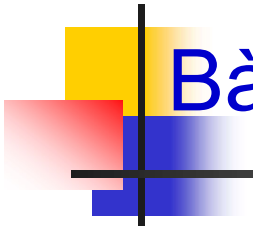
  - F(x0, y0) = 0 -> d_{start}=a+b/2=dy-dx/2
  - Tránh số thập phân của d_{start}, định nghĩa lại hàm như sau

    F(x,y)=2(ax+by+c)
  - Do vậy, ta có

    d_{start}=2dy -dx; Δ_E=2dy; Δ_{NE}=2(dy-dx)

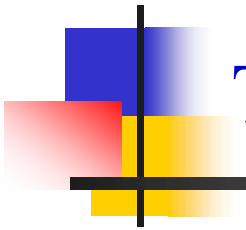
Thuộc tính đường vẽ

- Thuật toán vẽ đoạn thẳng nói trên đều vẽ đoạn thẳng có độ rộng 1 pixel, nét liên tục
- Hai thuộc tính quan trọng của đường vẽ
  - Độ rộng: vẽ đoạn thẳng từ (x0, y0) đến (x1, y1)
    - Nếu dy>dx: các pixel được vẽ thêm tại tọa độ bên trái và bên phải điểm vẽ (x-1 và x+1)
    - Nếu dx>dy: vẽ thêm các pixel phía trên và dưới điểm vừa vẽ
  - Đường nét đứt:
    - Sử dụng các pattern, mặt nạ với bit cao nhất bằng 1
    - Dựa trên kết quả phép AND mặt nạ với mẫu để quyết định có vẽ điểm ảnh tại vị trí hiện hành hay không.



Bài tập

- Xây dựng thuật toán vẽ đoạn thẳng với hệ số góc và hướng bất kỳ, có thuộc tính.
- Cài đặt các thuật toán vẽ đoạn thẳng với hệ số góc bất kỳ

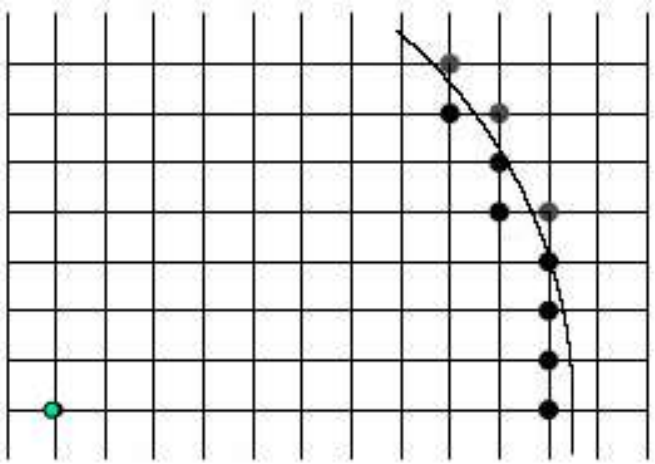


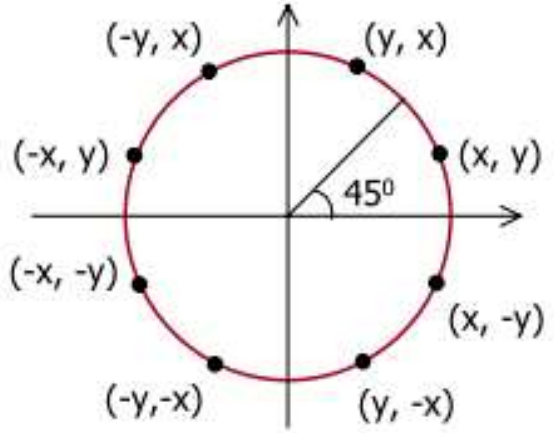
Thuật toán vẽ đường tròn, elip

- Một số tính chất của đường tròn
- Thuật toán Bresenhamvẽ đường tròn
- Thuật toán midpoint vẽ đường tròn
- Thuật toán Bresenhamvẽ elip
- Thuật toán midpoint vẽ elip

Một số tính chất của đường tròn

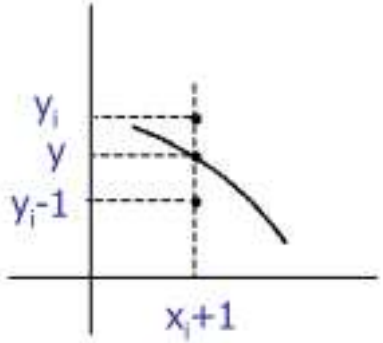
- Tương tự như vẽ đoạn thẳng, đường tròn đồ họa hình thành bởi các pixel gần đường tròn toán học nhất (Rasterization).
- Một vài tính chất cơ bản:
  - Vẽ đường tròn tâm tại gốc tọa độ sau đó dịch chuyển đến vị trí mong muốn
  - Tính đối xứng: khi biết tọa độ 1 điểm dễ dàng suy ra tọa độ của 7 điểm còn lại
  - Sử dụng phương trình để tính tọa độ đường tròn -> dấu phảy di động.
  - Các thuật toán tối ưu khác





Thuật toán Bresenham vẽ Circle(tt)

1. Chọn vị trí thứ nhất để vẽ có tọa độ $(x_1, y_1)=(0, r)$
2. Tính tham số thứ nhất: $p_1=3-2r$
   - Nếu $p_1<0$: vị trí vẽ tiếp theo là (x_1+1, y_1). Ngược lại vẽ tại tọa độ (x_1-1, y_1-1).
3. Tiếp tục tăng x để tính p tiếp theo từ p trước đó
   - Nếu trước đó có $p_i<0$: $p_{i+1}=p_i+4x_i+6$
   - Ngược lại, ta có: $p_{i+1}=p_i+4(x_i-y_i)+10$
   - Nếu kết quả $p_{i+1}<0$: điểm sẽ chọn tiếp theo là (x_i-2, y_{i+1}).
   - Ngược lại, ta chọn: $(x_i+2, y_{i+1}-1)$.
   - Nếu $p_i<0$ thì $y_{i+1}= y_i$, ngược lại $y_{i+1}= y_i-1$
4. Lặp lại bước 3 cho đến khi x=y.

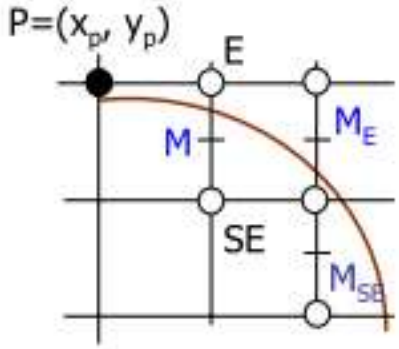


Thuật toán trung điểm vẽ Circle(tt)

- Ý tưởng thuật toán: Khi đã vẽ điểm P tại (x_p, y_p), phải quyết định điểm vẽ tiếp theo là E hay SE
- Phương trình đường tròn

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

  - F(x,y)=0 -> (x,y) trên đường tròn
  - F(x,y)<0 -> (x,y) trong đường tròn
  - F(x,y)>0 -> (x,y) ngoài đường tròn
    - Nếu M trong vòng tròn -> E gần đường tròn
    - Ngược lại -> SE gần đường tròn



Thuật toán trung điểm vẽ Circle(tt)

- Biến quyết định d: giá trị hàm tại điểm giữa M

$$d_{old} = F(x_p + 1, y_p - \tfrac{1}{2}) = (x_p + 1)^2 + (y_p - \tfrac{1}{2})^2 - R^2$$

  - Nếu d_{old}<0 thì chọn E, x_p tăng 1, y_p giữ nguyên.

$$d_{new} = F(x_p + 2, y_p - \tfrac{1}{2}) = (x_p + 2)^2 + (y_p - \tfrac{1}{2})^2 - R^2$$

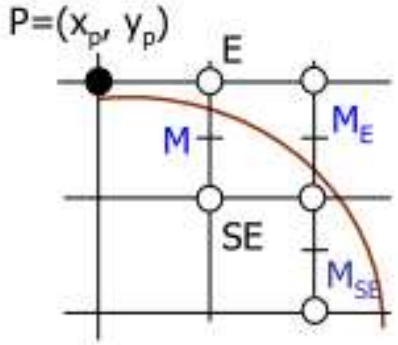
$$d_{new} = d_{old} + (2x_p + 3) = d_{old} + \Delta_E$$

  - Nếu d_{old}>0 thì chọn SE, x_p tăng 1, y_p giảm 1.

$$d_{new} = F(x_p + 2, y_p - \tfrac{3}{2})$$

$$= (x_p + 2)^2 + (y_p - \tfrac{3}{2})^2 - R^2$$

$$d_{new} = d_{old} + (2x_p - 2y_p + 5) = d_{old} + \Delta_{SE}$$



Thuật toán trung điểm vẽ Circle(tt)

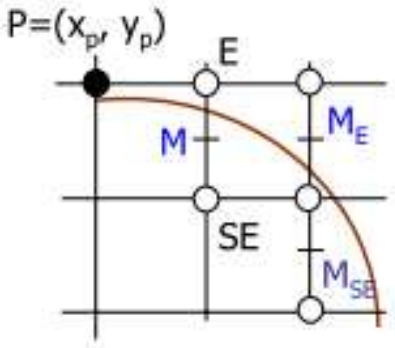
- Vòng lặp của thuật toán
  - Chọn pixel để vẽ dựa trên dấu biến quyết định d của vòng lặp trước
  - Cập nhật biến quyết định d bởi giá trị Δ tương ứng với pixel SE hay E vừa chọn
- Giá trị khởi đầu
  - Điểm vẽ đầu tiên có tọa độ (0, R)
  - Biến quyết định d có giá trị:

$$d = F(1, R - \tfrac{1}{2}) = 1 + (R^2 - R + \tfrac{1}{4}) - R^2 = \tfrac{5}{4} - R$$

  - Đặt biến quyết định mới h=d-1/4, ta có:

$$h + \tfrac{1}{4} = \tfrac{5}{4} - R$$

$$h = 1 - R$$



Thuật toán trung điểm vẽ Circle

```
void CirclePoints(int x, y, color)
{
        PutPixel(x,y, color);
        PutPixel(y,x, color);
        PutPixel(y,-x, color);
        PutPixel(x,-y, color);
        PutPixel(-x,-y, color);
        PutPixel(-y,-x, color);
        PutPixel(-y,x, color);
        PutPixel(-x,y, color);
}
```

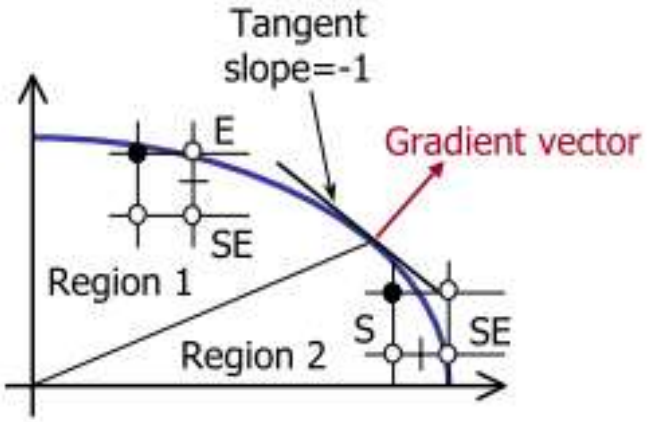
```
void MidpointCircle(int radius, color)
{
        int x, y; float d;
        x=0; y=radius;
        d=1-radius;
        CirclePoints(x, y, color);
        While( y>x)
        {
                if (d<0 ) //Chọn E
                {
                        d=d+2*x+3; x=x+1
                } else          // Chọn SE
                {
                        d=d+2*(x-y)+5; x=x+1; y=y-1
                }
                CirclePoints(x, y, color);
        }
}
```

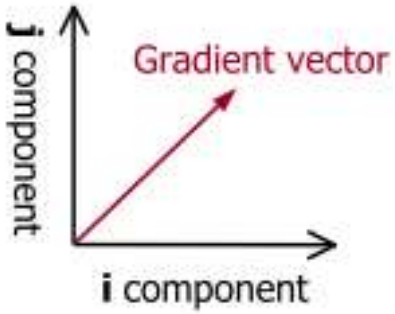
Thuật toán trung điểm vẽ Elip

- Phương trình elíp có tâm tại gốc tọa độ

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

- Áp dụng giải pháp trung điểm vẽ đường tròn để vẽ elíp
  - Tính đối xứng của elíp: khi biết tọa độ 1 điểm có thể dễ dàng suy ra tọa độ ba điểm khác.





Thuật toán trung điểm vẽ Elip(tt)

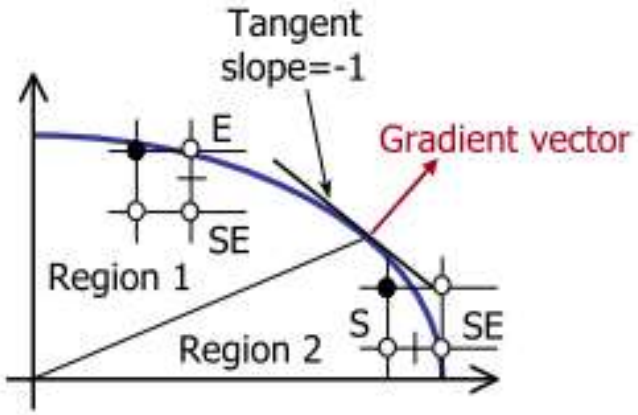
- Tìm ranh giới hai miền trong ¼ elíp
  - Vị trí: Điểm P là tiếp điểm của tiếp tuyến có hệ số góc –1
  - Xác định:
    - Véc tơ vuông góc với tiếp tuyến tại tiếp điểm -> gradient

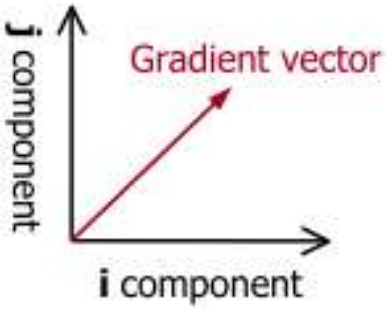
$$gradF(x,y) = \frac{\partial F}{\partial x} i + \frac{\partial F}{\partial y} j = 2b^2 x.i + 2a^2 y.j$$

    - Tại P1 các thành phần i và j của véc tơ gradient có cùng độ lớn

Miền 1: Thành phần **j** lớn hơn thành phần **i**

$$a^2 y > b^2 x$$





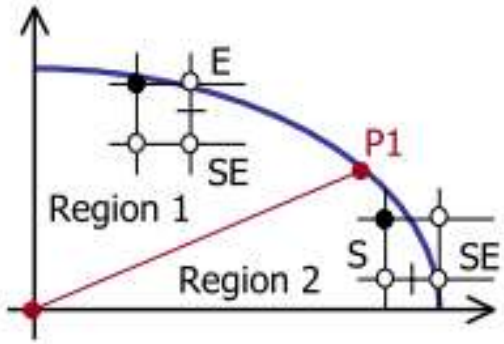
Thuật toán trung điểm vẽ Elip

- Ý tưởng: Đánh giá hàm tại điểm giữa hai tọa độ pixel để chọn vị trí tiếp theo để vẽ. Dấu của nó cho biết điểm giữa nằm trong hay ngoài elíp.
- Với vùng 1:
  - Tính biến quyết định d=F(x,y)=F(x_p+1, y_p-1/2)
  - Nếu d<0: chọn E, x tăng 1, y không thay đổi.

$$d_{old} = F(x_p + 1, y_p - \tfrac{1}{2}) = b^2(x_p + 1)^2 + a^2(y_p - \tfrac{1}{2})^2 - a^2b^2$$

$$d_{new} = F(x_p + 2, y_p - \tfrac{1}{2}) = b^2(x_p + 2)^2 + a^2(y_p - \tfrac{1}{2})^2 - a^2b^2$$

$$d_{new} = d_{old} + b^2(2x_p + 3) = d_{old} + \Delta_E$$



  - Nếu d≥0: chọn SE, x tăng 1, y giảm 1

$$d_{new} = F(x_p + 2, y_p - \tfrac{3}{2}) = b^2(x_p + 2)^2 + a^2(y_p - \tfrac{3}{2})^2 - a^2b^2$$

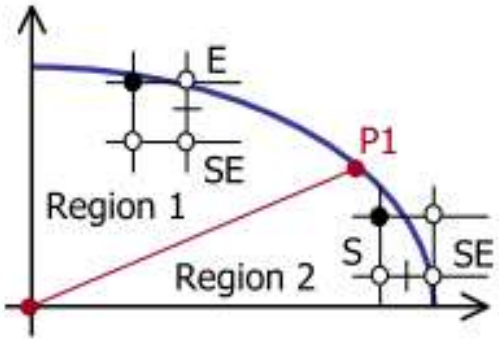
$$d_{new} = d_{old} + b^2(2x_p + 3) + a^2(-2y_p + 2) = d_{old} + \Delta_{SE}$$

Thuật toán trung điểm vẽ Elip(tt)

- Với vùng 2:
  - Tính biến quyết định d =F(x, y)=F(x_p+1/2, y_p-1)
    - Nếu d<0: chọn SE, x tăng 1, y giảm 1.
    - Nếu d≥0: chọn S, x không tăng, y giảm 1
  - Tìm số gia như vùng 1

$$\Delta_S = a^2(-2y_p+3)$$

$$\Delta_{SE} = b^2(2x_p+2)+a^2(-2y+3)$$



Thuật toán trung điểm vẽ Elip(tt)

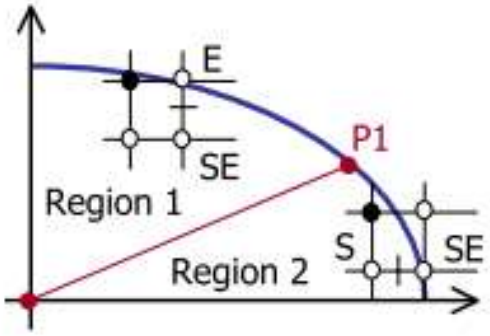
- Tìm giá trị khởi đầu của số gia d
  - Miền 1:
    - Giả sử a, b nguyên; điểm bắt đầu vẽ là (0, b)
    - Điểm giữa thứ nhất: (1, b-1/2)

$$F(1, b-\tfrac{1}{2}) = b^2 + a^2(b-\tfrac{1}{2})^2 - a^2b^2 = b^2 + a^2(-b+\tfrac{1}{4}) = b^2 - a^2b + \frac{a^2}{4}$$

  - Miền 2:
    - Phụ thuộc vào điểm giữa (x_p+1, y_p-1/2) của điểm tiếp theo điểm cuối cùng của miền 1.

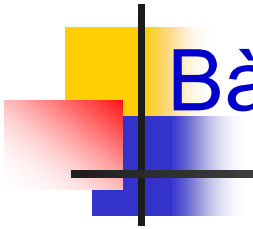
$$F(x_p + \tfrac{1}{2}, y_p - 1) = b^2(x+\tfrac{1}{2})^2 + a^2(y-1)^2 - a^2b^2$$

$$= b^2x^2 + b^2x + \frac{b^2}{4} + a^2(y-1)^2 - a^2b^2$$



Thuật toán trung điểm vẽ Elip(tt)

```
void draw_ellipse(int a, b, color)
{
    int x, y; float d1, d2;
    {
        x=0;   y=b; d1=b²-a²b+a²/4;
        EllipsePoints(x, y, color);
        while (a²(y-1/2)>b²(x+1))
        {
            if (d1<0) {Chọn E}
            {
                d1=d1+b2(2*x+3); x=x+1;
            }
            else    {Chọn SE}
            {
                d1=d1+b2(2*x+3)+a²(-2*y+2);
                x=x+1;y=y-1;
            }
            EllipsePoints (x, y, color);
        }
        d2=b²(x+1/2)²+a²(y-1)² –a²b²;
        while (y>0 do)   {Vùng 2}
        {
            if (d2<0 )  { Chon SE }
            {
                d2=d²+b²(2*x+2)+a²(-2*y+3);
                x=x+1;y=y-1;
            }
            else
            }
                d2=d²+a²(-2*y+3); y=y-1;
            }
            EllipsePoints (x, y, color);
        }
    }
```



Bài tập

- Xây dựng thuật toán vẽ cung tròn, cung elíp.
- Cài đặt các thuật toán vẽ hình tròn có tâm và bán kính bất kỳ.
- Cài đặt các thuật toán vẽ hình elip có tâm và bán kính bất kỳ