Bài 2:
Các giải thuật sinh các thực thể cơ sở

Le Tan Hung
hunglt@it-hut.edu.vn

Giải thuật xây dựng các thực thể cơ sở

- Giải thuật sinh đường thẳng – Line
- Giải thuật sinh đường tròn - Circle
- Giải thuật VanAken sinh Ellipse
- Giải thuật sinh đa giác
- Giải thuật sinh ký tự

Rời rạc hoá điểm ảnh
(Scan Conversion rasterization)

- Scan Conversion rasterization
- Tính chất các đối tượng cần đảm bảo :
  - smooth
  - continuous
  - pass through specified points
  - uniform brightness
  - efficient

Biểu diễn đoạn thẳng

- **Biểu diễn tường minh**

(y-y1)/(x-x1) = (y2-y1)/(x2-x1)1

y = kx + m

- **Biểu diễn không tường minh**

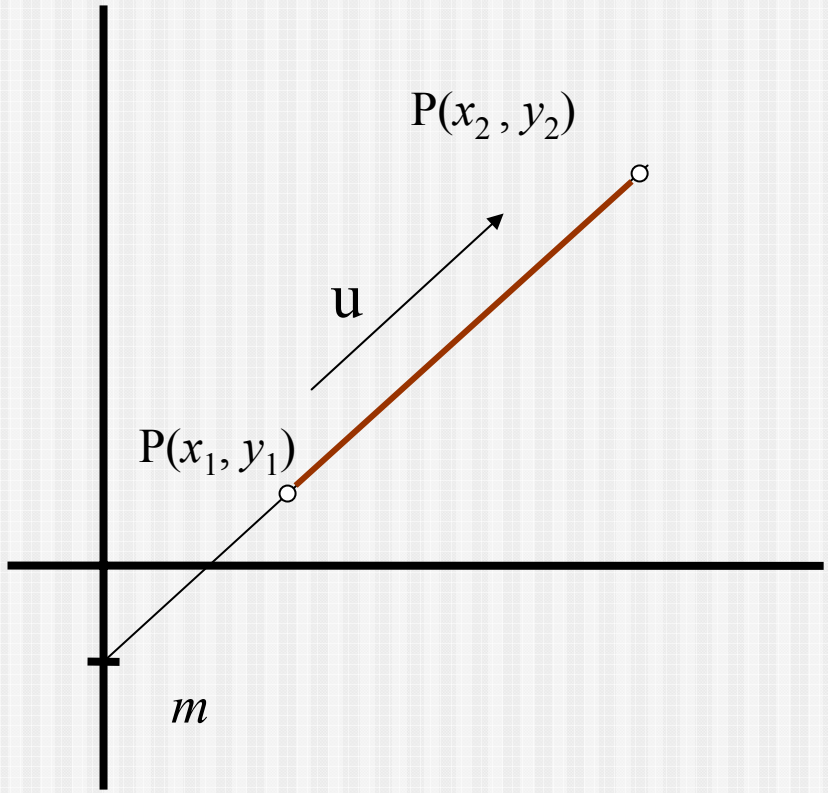
(y2-y1)x - (x2-x1)y + x2y1 - x1y2 = 0

hay rx + sy + t = 0

- **Biểu diễn tham biến**

P(u) = P1 + u(P2 - P1)

u [0,1]



Thuật toán DDA (Digital Differential Analizer)

Giải thuật thông thường

```
DrawLine(int x1,int y1, int x2,int y2,
int color)
{
    float y;
    int x;
    for (x=x1; x<=x2; x++)
    {
        y = y1 + (x-x1)*(y2-y1)/(x2-x1)
         WritePixel(x, Round(y), color );
    }}
```

Giải thuật DDA

- Với $0 < k < 1$

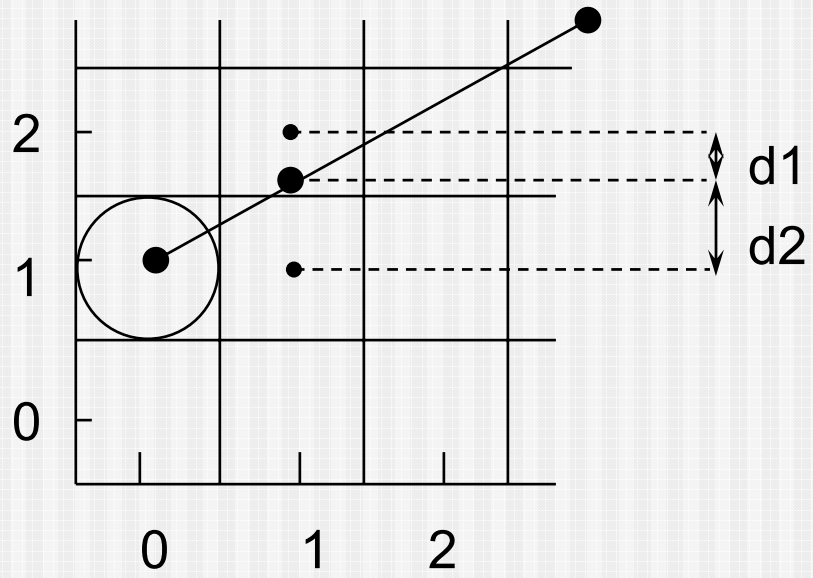
$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + k$$

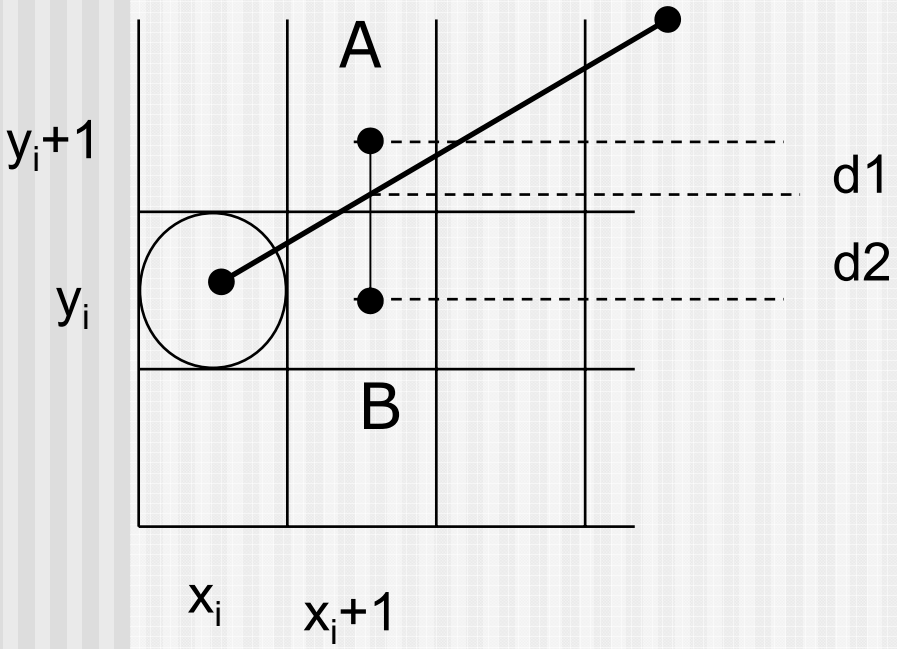
với i=1,2,3....

Giải thuật Bresenham

- 1960 Bresenham thuộc IBM
- điểm gần với đường thẳng dựa trên độ phân giai hưu hạn
- loại bỏ được các phép toán chia và phép toán làm tròn như ta đã thấy trong giải thuật DDA
- Xét đoạn thẳng với $0 < k < 1$



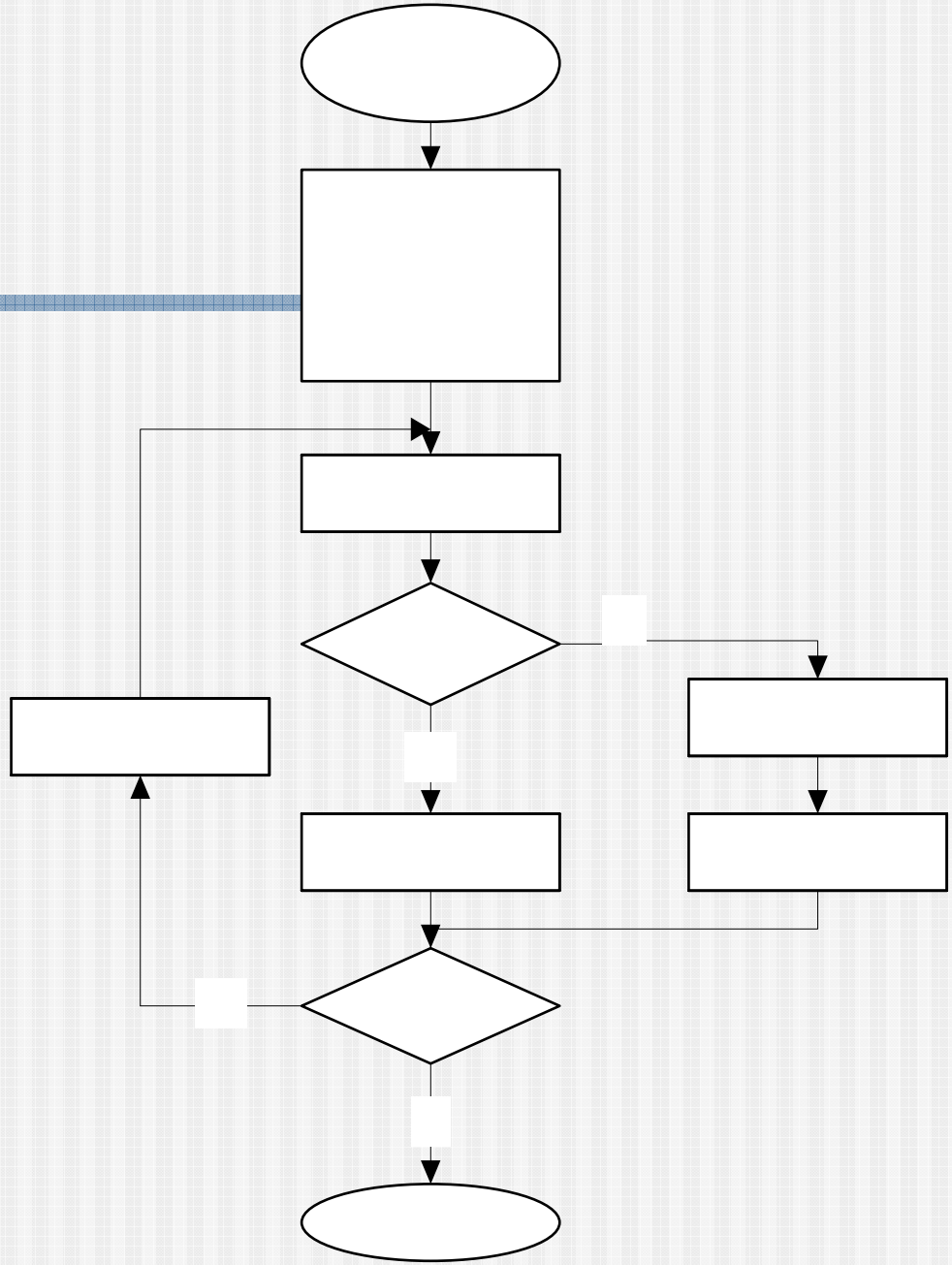
Giải thuật Bresenham



$$d_2 = y - yi = k(xi + 1) + b - yi$$

$$d_1 = yi+1 - y = yi + 1 - k(xi + 1) - b$$

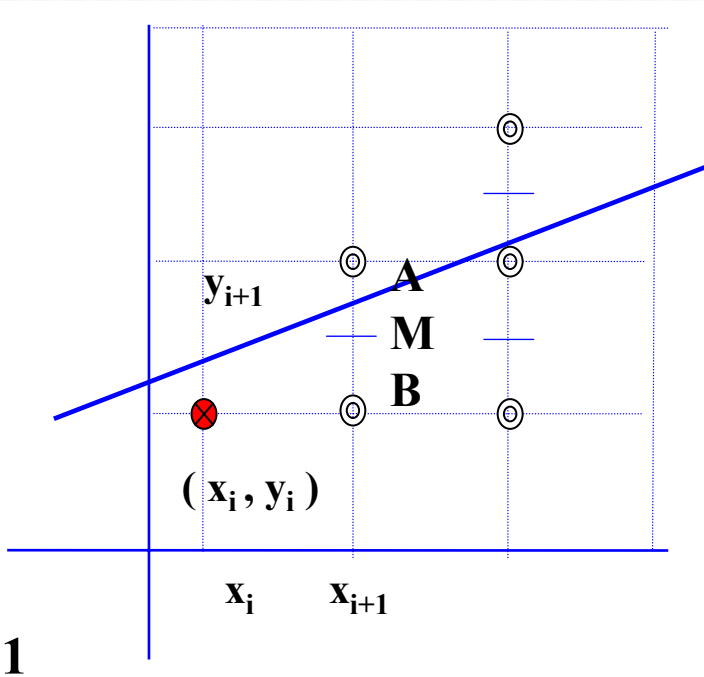
Giải thuật Bresenham



Giải thuật trung điểm-Midpoint

- Jack Bresenham 1965 / Pitteway 1967
- VanAken áp dụng cho việc sinh các đường thẳng và đường tròn 1985
- Các công thức đơn giản hơn, tạo được các điểm tương tự như với Bresenham

- d = F (xi + 1, yi + 1/2) là trung điểm của đoạn AB
- Việc so sánh, hay kiểm tra M sẽ được thay bằng việc xét giá trị d.
  - Nếu d > 0 điểm B được chọn, $y_{i+1} = y_i$
  - nếu d < 0 điểm A được chọn. $y_{i+1} = y_i + 1$
  - Trong trường hợp d = 0 chúng ta có thể chọn điểm bất kỳ hoặc A, hoặc B.

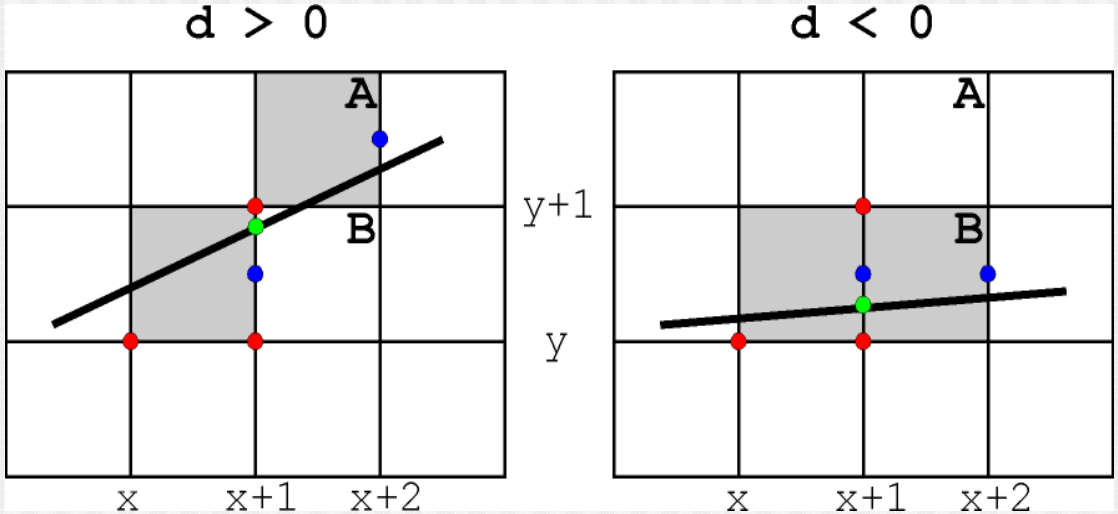


Bresenham's Algorithm: Midpoint Algorithm

- If $d_i > 0$ then chọn điểm **A** $\Rightarrow$ trung điểm tiếp theo sẽ có dạng:

$$\left(x_i + 2, y_i + \frac{3}{2}\right) \Rightarrow d_{i+1} = a(x_i + 2) + b\left(y_i + \frac{3}{2}\right) + c$$

$$= d_i + a + b$$



Midpoint Line Algorithm

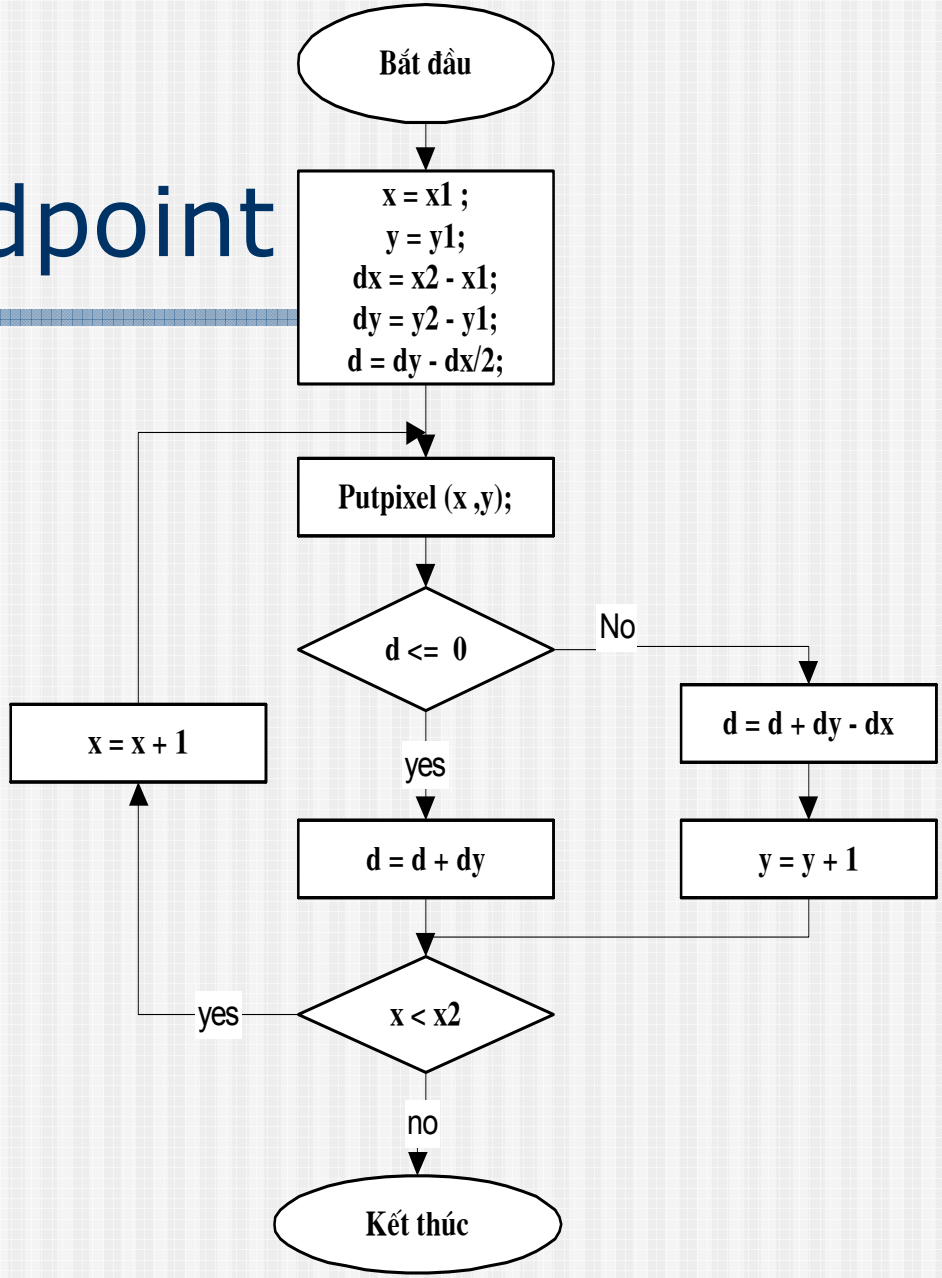
```
dx = x_end-x_start
dy = y_end-y_start
d = 2*dy-dx
x = x_start
y = y_start
while x < x_end
    if d <= 0 then
        d = d+(2*dy)
        x = x+1
    else
        d = d+2*(dy-dx)
        x = x+1
        y = y+1
    endif
    SetPixel(x,y)
endwhile
```

initialisation

choose **B**

choose **A**

Giải thuật Bresenham's Midpoint

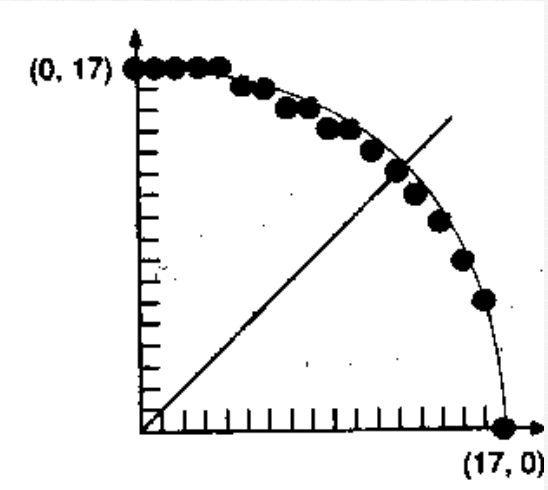


Sinh đường tròn
Scan Converting Circles

- Explicit: y = f(x)

$$y = \pm\sqrt{R^2 - x^2}$$

Usually, we draw a quarter circle by incrementing x from 0 to R in unit steps and solving for +y for each step.



- Parametric:

$$x = R\cos\theta$$

$$y = R\sin\theta$$

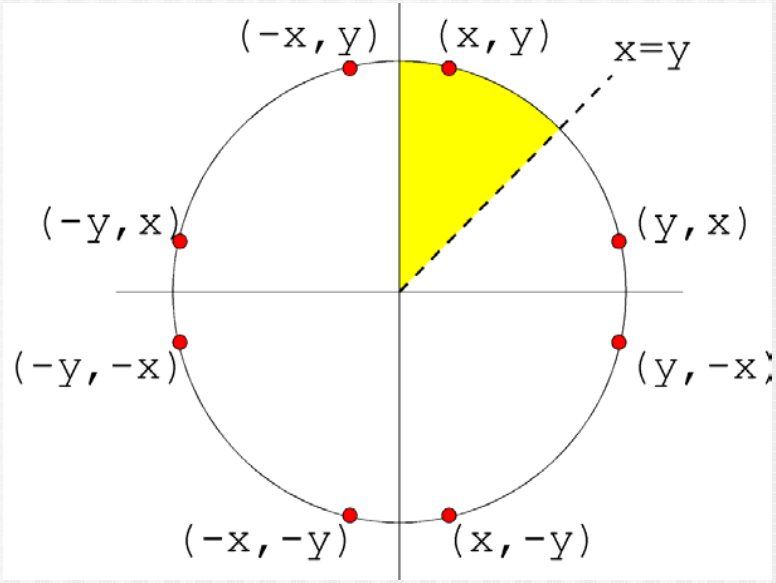
- by stepping the angle from 0 to 90
- avoids large gaps but still insufficient.

- Implicit: $f(x) = x^2+y^2-R^2$

If f(x,y) = 0 then it is on the circle.
f(x,y) > 0 then it is outside the circle.
f(x,y) < 0 then it is inside the circle.

Midpoint Circle Algorithm

- Sử dụng phương pháp biểu diễn không tường minh trong giải thuật
- Thực hiện giải thuật trên 1/8 đường tròn và lấy đối xứng xho các góc còn lại.
- Với d_i là giá trị của đường tròn tại một điểm bất kỳ ta có

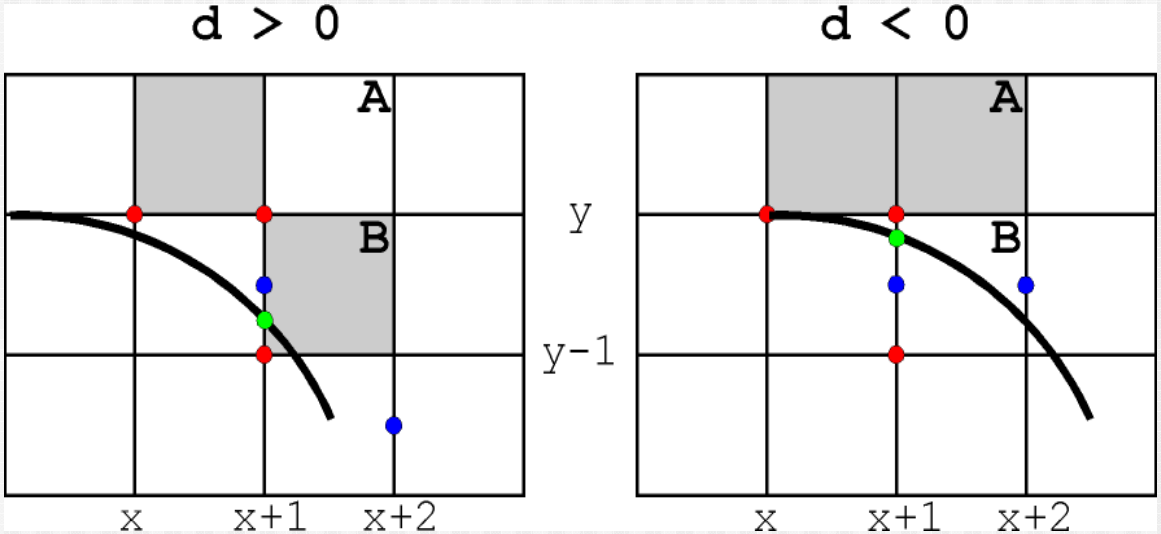


Midpoint Circle Algorithm

- As with the line, we determine the value of the decision variable by substituting the mid-point of the next pixel into the implicit form of the circle:

$$d_i = (x_i + 1)^2 + \left(y_i - \frac{1}{2}\right)^2 - r^2$$

- If $d_i < 0$ we choose pixel **A** otherwise we choose pixel **B**
  - Note: we currently assume the circle is centered at the origin



Midpoint Circle Algorithm

```
d = 1-r
x = 0
y = r
while y < x
    if d < 0 then
        d = d+2*x+3
        x = x+1
    else
        d = d+2*(x-y)+5
        x = x+1
        y = y-1
    endif
    SetPixel(cx+x,cy+y)
endwhile
```

initialisation

stop at diagonal ⇒ end of octant

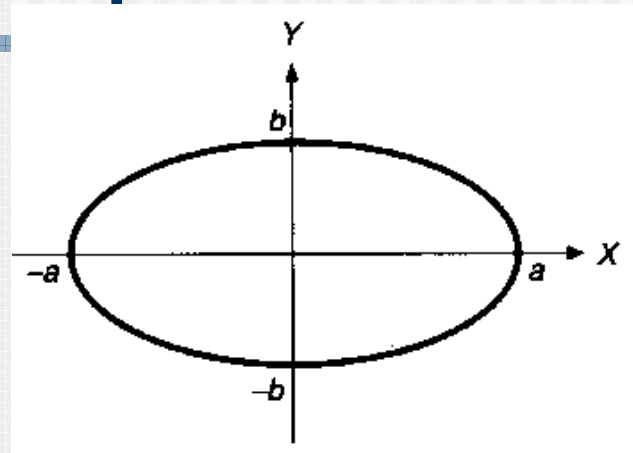
choose **A**

choose **B**

Translate to the circle center

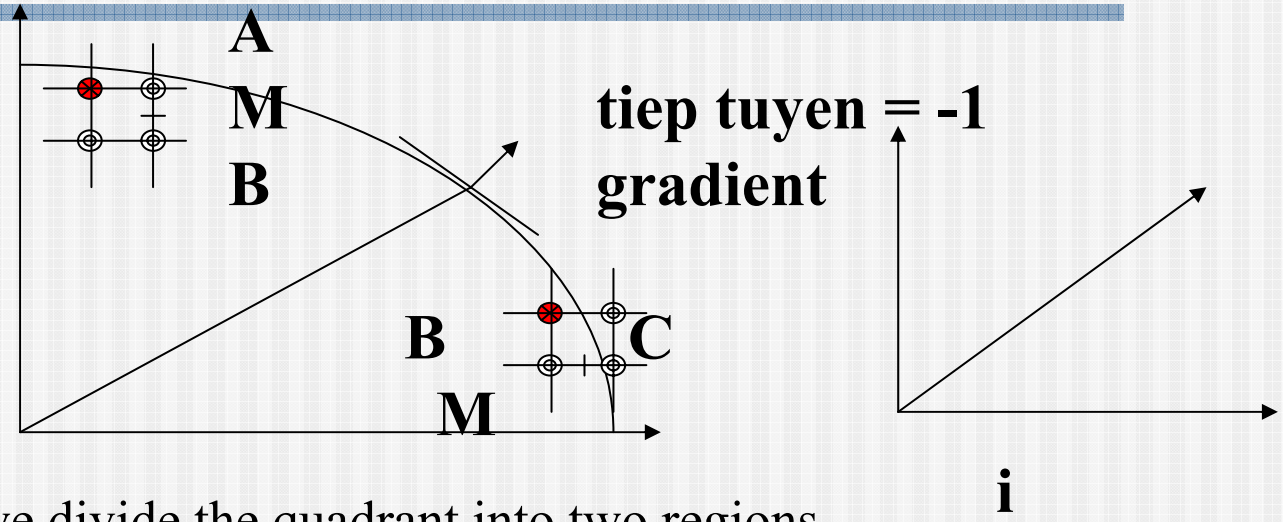
Scan Converting Ellipses

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



- 2a is the length of the major axis along the x axis.
- 2b is the length of the minor axis along the y axis.
- The midpoint can also be applied to ellipses.
- For simplicity, we draw only the arc of the ellipse that lies in the first quadrant, the other three quadrants can be drawn by symmetry

Scan Converting Ellipses: Algorithm



- Firstly we divide the quadrant into two regions
- Boundary between the two regions is
  - the point at which the curve has a slope of -1
  - the point at which the gradient vector has the i and j components of equal magnitude

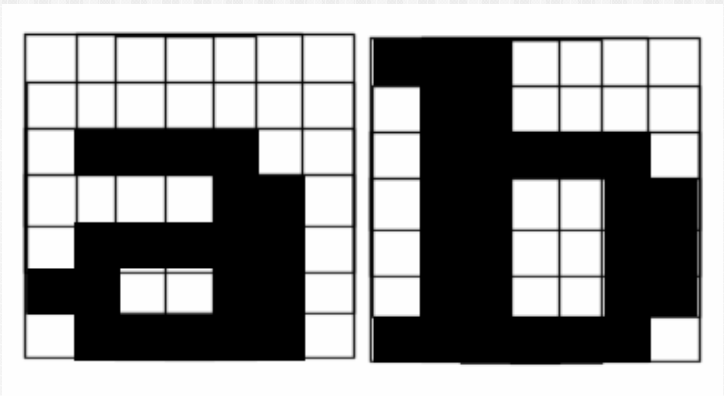
$$grad\, F(x,y) = \partial F/\partial x\,\mathbf{i} + \partial F/\partial y\,\mathbf{j} = 2b^2 x\,\mathbf{i} + 2a^2 y\,\mathbf{j}$$

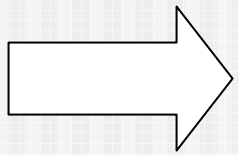
Ellipses: Algorithm (cont.)

- At the next midpoint, if $a^2(y_p-0.5)<=b^2(x_p+1)$, we switch region 1=>2
- In region 1, choices are E and SE
  - Initial condition: $d_{init} = b^2+a^2(-b+0.25)$
  - For a move to E, $d_{new} = d_{old}+Delta_E$ with $Delta_E = b^2(2x_p+3)$
  - For a move to SE, $d_{new} = d_{old}+Delta_{SE}$ with $Delta_{SE} = b^2(2x_p+3)+a^2(-2y_p+2)$
- In region 2, choices are S and SE
  - Initial condition: $d_{init} = b^2(x_p+0.5)^2+a^2((y-1)^2-b^2)$
  - For a move to S, $d_{new} = d_{old}+Delta_s$ with $Delta_s = a^2(-2y_p+3)$
  - For a move to SE, $d_{new} = d_{old}+Delta_{SE}$ with $Delta_{SE} = b^2(2x_p+2)+a^2(-2y_p+3)$
- Stop in region 2 when the y value is zero.

Ký tự Bitmap

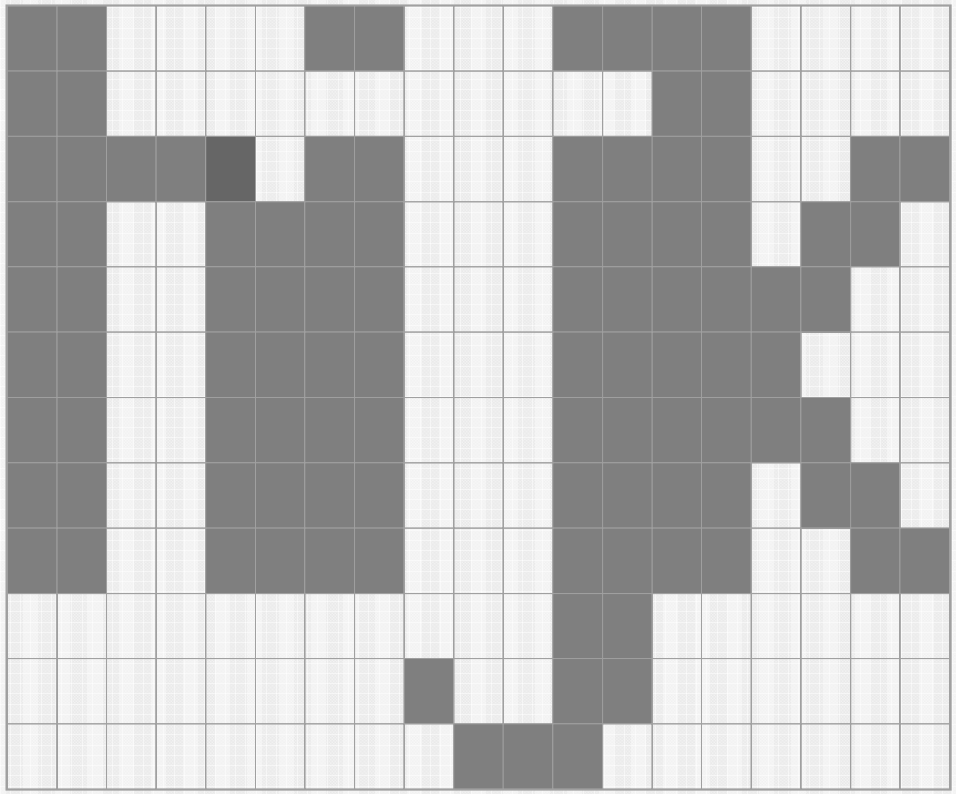
- Trên cơ sở định nghĩa mỗi ký tự với một font chư cho trước là một bitmap chữ nhật nhỏ
- Font/typeface: set of character shapes
- fontcache
  - các ký tự theo chuỗi liên tiếp nhau trong bộ nhớ
- Dạng cơ bản
  - (thường N, nghiêng I, đậm B, nghiêng đậm B+I)
- Thuộc tính
  - Also colour, size, spacing and orientation





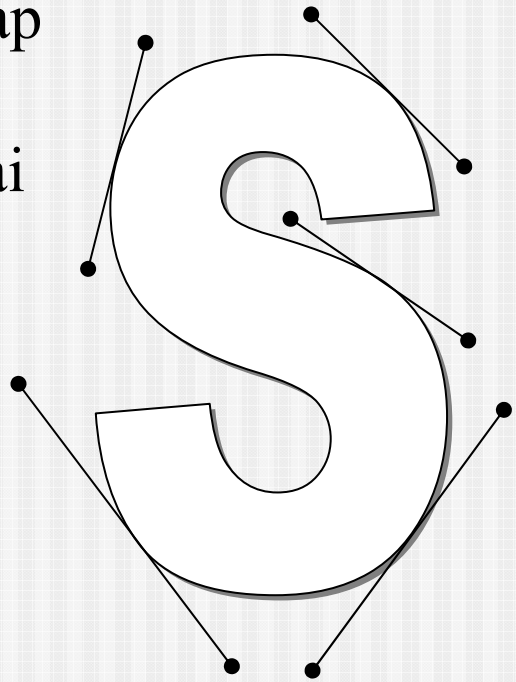
ab

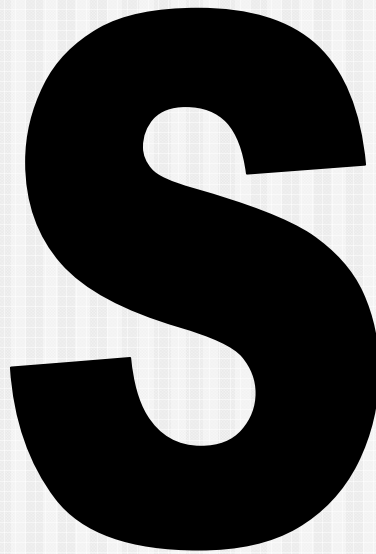
Cấu trúc font chữ



Ký tự vector

- Xây dựng theo phương pháp định nghĩa các ký tự bởi đường cong mềm bao ngoài của chúng.
- Tốn kém nhất về mặt tính toán
- Chất lượngcao





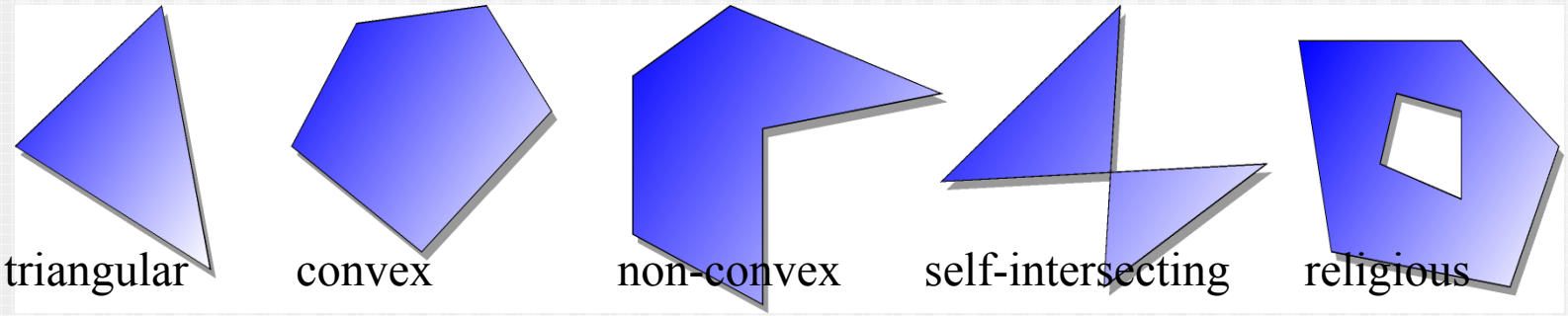
So sánh

- Đơn giản trông việc sinh ký tự (copypixel)
- Lưu trữ lớn
- Các phép biến đổi (I,B, scale) đòi hỏi lưu trữ thêm
- Kích thước không dổi

- Phức tạp (Tính toán phương trình)
- Lưu trữ gọn nhẹ
- Các phép biến đổi dựa vào các công thức biến đổi
- Kích thước phụ thuộc vào môi trường (ko có kích thước cố định)

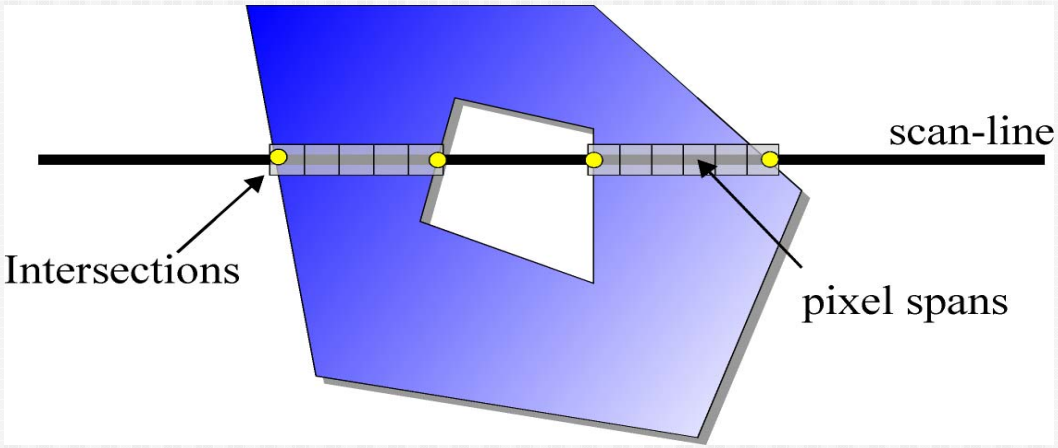
Giải thuật đường quét sinh đa giác
Polygon Scan Conversion

- Tồn tại rất nhiều giải thuật sinh đa giác.
- Mỗi giải thuật phục vụ cho 1 loại đa giác nhất định:
  - some algorithms allow *triangular polygons* only
  - others require that the polygons are *convex* and *non self-intersecting* and have no *holes*



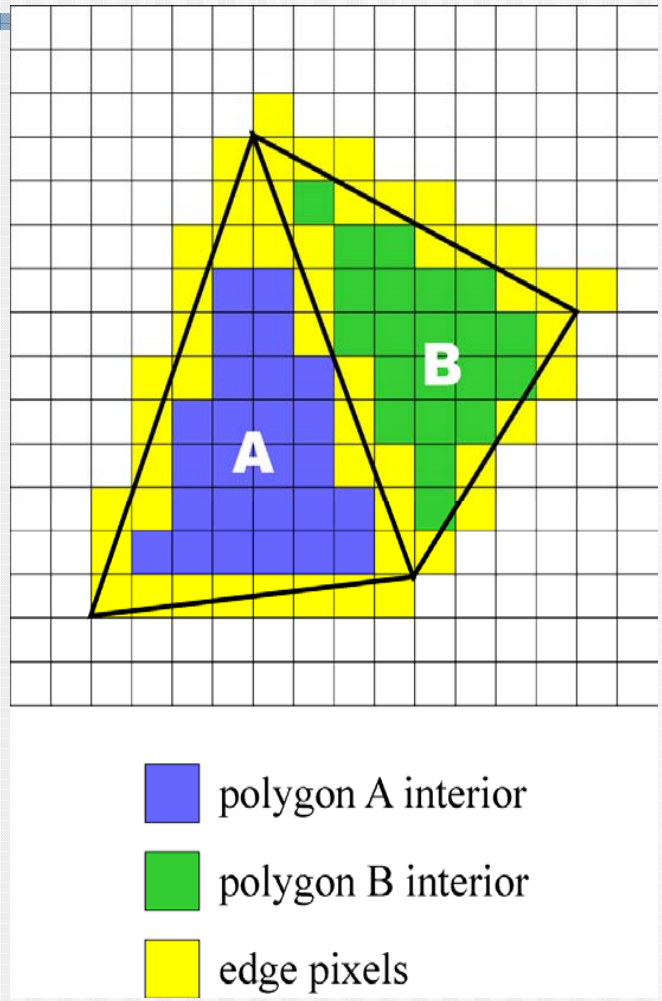
Polygon Scan Conversion

- *Polygon scan conversion* là giải thuật chung kinh điển cho các loại khác nhau
- Cho mỗi đoạn thẳng quét, chúng ta xác định các cạnh của đa giác cắt đoạn thẳng compute *spans* representing the interior portions of the polygons along this *scan-line* and fill the associated pixels.
- This represents the heart of a *scan-line rendering algorithm* used in many commercial products including Renderman and 3D Studio MAX.



Polygon Scan Conversion

- Dùng giải thuật (trung điểm) để xác định các điểm biên cho mỗi đa giác theo thứ tự tăng của x.
- Các diểm phải:
  - Không bị chia sẻ bởi các đa giác lân cận
  - Các đa giác chỉ toàn các điểm cạnh(điểm biên)
- Đảm bảo các đa giác chia sẻ điểm biên mà không chia sẻ các điểm ảnh bên trong của mình.



Polygon Scan Conversion

- Thủ tục chung:
  - Xác định giao của đường thẳng quét với cạnh đa giác
  - Sắp xếp các giao điểm theo mức độ tăng dần của x value
  - Điền các điểm ảnh vào giữa cặp các điểm x
- Need to handle 4 cases to prevent pixel sharing:
  - if intersection has fractional x value, do we round up or down?
    - if inside (on left of span) round up, if outside (on right) round down
  - what happens if intersection is at an integer x value?
    - if on left of span assume its interior otherwise exterior
  - how do we handle shared vertices?
    - ignore pixel associated with y_{max} of an edge
  - how do we handle horizontal edges?
    - handled as a result of previous rule (lower edges not drawn)

Polygon Scan Conversion