

Giới thiệu các thuật toán vẽ và tô các đường cơ bản

TÀI LIỆU

Thích 7

Chia sẻ 7

Tweet

0

Tổng quan

Mục tiêu của chương 1

Học xong chương này, sinh viên phải nắm bắt được các vấn đề sau:

- Thế nào là hệ đồ họa
- Thiết kế và cài đặt được các thủ tục vẽ và tô các đường cơ bản như đường thẳng, đường tròn, elip, và các đường cong khác.

Kiến thức cơ bản cần thiết

Các kiến thức cơ bản cần thiết để học chương này bao gồm :

- Các khái niệm toán học về đường thẳng như : đường thẳng là gì : dạng tổng quát phương trình đường thẳng, hệ số góc, tung độ dốc.
- Hiểu rõ hình dáng của đường thẳng phụ thuộc vào hệ số góc như thế nào.
- Phương trình tổng quát của đường tròn, ellipse (không có tham số và có tham số).
- Kỹ thuật lập trình: thiết lập thủ tục, hàm (lưu ý truyền qui chiếu và truyền giá trị).

Tài liệu tham khảo

Donald Hearn, M. Pauline Baker. **Computer Graphics** . Prentice-Hall, Inc., Englewood Cliffs, New Jersey , 1986 (chapters 3, 55-76).

Nội dung cốt lõi

Thiết lập thủ tục vẽ :

- Đường thẳng bằng giải thuật DDA
- Đường thẳng bằng giải thuật Bresenham
- Đường tròn bằng giải thuật đối xứng
- Đường tròn bằng giải thuật Bresenham
- Đường tròn bằng giải thuật MidPoint
- Ellipse
- Đa giác

Hệ tọa độ thế giới thực, hệ tọa độ thiết bị và hệ tọa độ chuẩn

Một hệ mềm đồ họa được mô tả bao gồm 3 miền như sau :

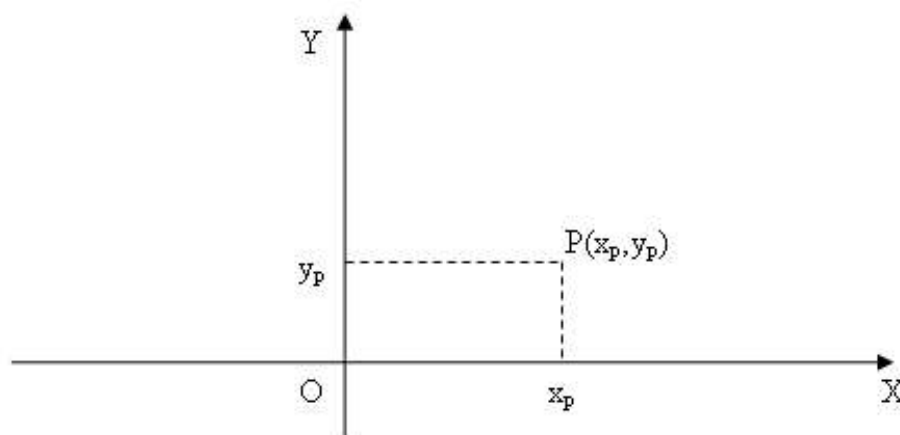
- Miền điều khiển : bao bọc toàn bộ hệ thống.
- Miền thực : nằm trong miền điều khiển. Khi một số nào đó thâm nhập vào miền thực, nó sẽ được chuyển thành số thực dấu phẩy động, và khi có một số rời khỏi miền này thì nó sẽ được chuyển thành số nguyên có dấu 16 bits.
- Miền hiển thị : nằm trong miền điều khiển nhưng phân biệt với miền thực. Chỉ có số nguyên 16 bits mới nằm trong miền hiển thị.

Trong lĩnh vực kỹ thuật đồ họa, chúng ta phải hiểu được rằng thực chất của đồ họa là làm thế nào để có thể mô tả và biến đổi được các đối tượng trong thế giới thực trên máy tính. Bởi vì, các đối tượng trong thế giới thực được mô tả bằng tọa độ thực. Trong khi đó, hệ tọa độ thiết bị lại sử dụng hệ tọa độ nguyên để hiển thị các hình ảnh. Đây chính là vấn đề cơ bản cần giải quyết. Ngoài ra, còn có một khó khăn khác nữa là với các thiết bị khác nhau thì có các định nghĩa khác nhau. Do đó, cần có một phương pháp chuyển đổi tương ứng giữa các hệ tọa độ và đối tượng phải được định nghĩa bởi các thành phần đơn giản như thế nào để có thể mô tả gần đúng với hình ảnh thực bên ngoài.

Hai mô hình cơ bản của ứng dụng đồ họa là dựa trên mẫu số hóa và dựa trên đặc trưng hình học. Trong ứng dụng đồ họa dựa trên mẫu số hóa thì các đối tượng đồ họa được tạo ra bởi lưới các pixel rời rạc. Các pixel này có thể được tạo ra bằng các chương trình vẽ, máy quét, ... Các pixel này mô tả tọa độ xác định vị trí và giá trị mẫu. Thuận lợi của ứng dụng này là dễ dàng thay đổi ảnh bằng cách thay đổi màu sắc hay vị trí của các pixel, hoặc di chuyển vùng ảnh từ nơi này sang nơi khác. Tuy nhiên, điều bất lợi là không thể xem xét đối tượng từ các góc nhìn khác nhau. Ứng dụng đồ họa dựa trên đặc trưng hình học bao gồm các đối tượng đồ họa cơ sở như đoạn thẳng, đa giác,.... Chúng được lưu trữ bằng các mô hình và các thuộc tính. Ví dụ : đoạn thẳng được mô hình bằng hai điểm đầu và cuối, có thuộc tính như màu sắc, độ dày. Người sử dụng không thao tác trực tiếp trên các pixel mà thao tác trên các thành phần hình học của đối tượng.

Hệ tọa độ thế giới thực:

Một trong những hệ tọa độ thực thường được dùng để mô tả các đối tượng trong thế giới thực là hệ tọa độ Descartes. Với hệ tọa độ này, mỗi điểm P được biểu diễn bằng một cặp tọa độ (x_p, y_p) với $x_p, y_p \in \mathbb{R}$ (xem hình 1.1).



Hình 1.1 : Hệ tọa độ thực.

- . O_x : gọi là trục hoành.
- . O_y : gọi là trục tung.
- . x_p : hoành độ điểm P.
- . y_p : tung độ điểm P.

Hệ tọa độ thiết bị

Hệ tọa độ thiết bị (device coordinates) được dùng cho một thiết bị xuất cụ thể nào đó, ví dụ như máy in, màn hình,...

Trong hệ tọa độ thiết bị thì các điểm cũng được mô tả bởi cặp tọa độ (x,y) . Tuy nhiên, khác với hệ tọa độ thực là $x, y \in \mathbb{N}$. Điều này có nghĩa là các điểm trong hệ tọa độ thực được định nghĩa liên tục, còn các điểm trong hệ tọa độ thiết bị là rời rạc. Ngoài ra, các tọa độ x, y của hệ tọa độ thiết bị chỉ biểu diễn được trong một giới hạn nào đó của \mathbb{N} .

Ví dụ : Độ phân giải của màn hình trong chế độ đồ họa là 640×480 . Khi đó, $x \in (0,640)$ và $y \in (0,480)$ (xem hình 1.2).



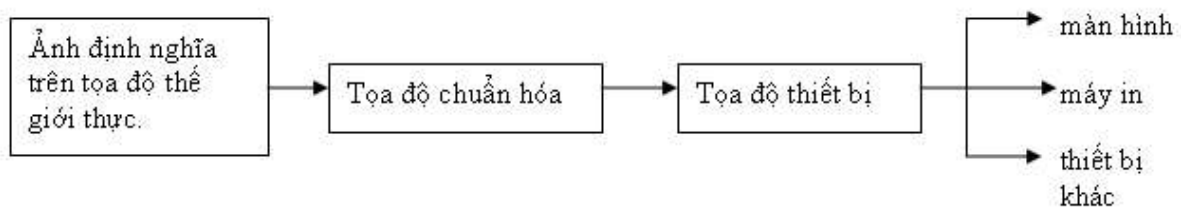
Hình 1.2 : Hệ tọa độ trên màn hình.

Hệ tọa độ thiết bị chuẩn (Normalized device coordinates)

Do cách định nghĩa các hệ tọa độ thiết bị khác nhau nên một hình ảnh hiển thị được trên thiết bị này là chính xác thì chưa chắc hiển thị chính xác trên thiết bị khác. Người ta xây dựng một hệ tọa độ thiết bị chuẩn đại diện chung cho tất cả các thiết bị để có thể mô tả các hình ảnh mà không phụ thuộc vào bất kỳ thiết bị nào.

Trong hệ tọa độ chuẩn, các tọa độ x, y sẽ được gán các giá trị trong đoạn từ $[0,1]$. Như vậy, vùng không gian của hệ tọa độ chuẩn chính là hình vuông đơn vị có góc trái dưới là $(0, 0)$ và góc phải trên là $(1, 1)$.

Quá trình mô tả các đối tượng thực như sau (xem hình 1.3):

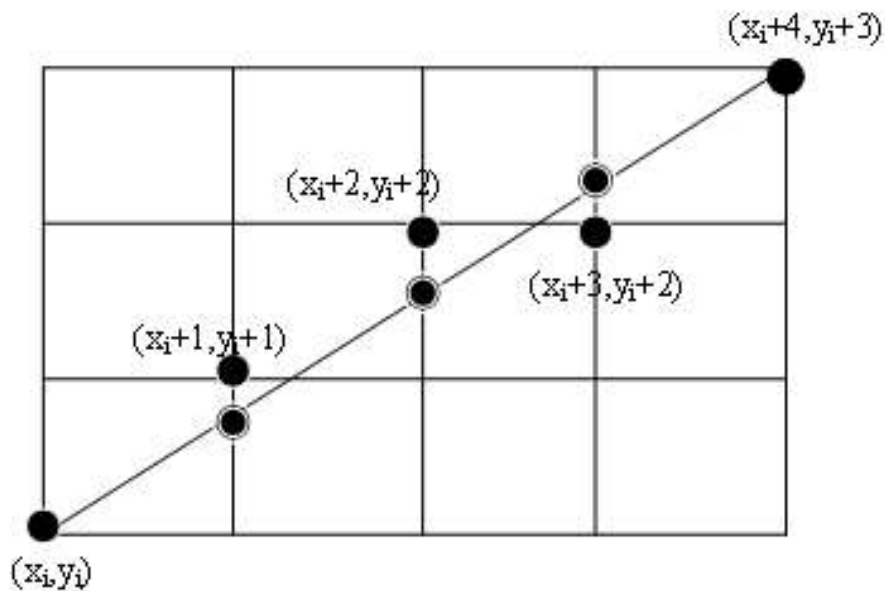


Hình 1.3 : Hệ tọa độ trên màn hình.

Thuật toán vẽ đoạn thẳng

Xét đoạn thẳng có hệ số góc $0 < m \leq 1$ và $\Delta x > 0$. Với các đoạn thẳng dạng này, nếu (x_i, y_i) là điểm đã được xác định ở bước thứ i thì điểm kế tiếp (x_{i+1}, y_{i+1}) ở bước thứ $i+1$ sẽ là một trong hai điểm sau (xem hình vẽ 1.4) :

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



Hình 1.4 : Các điểm vẽ gần với điểm muốn vẽ.

Vấn đề đặt ra là chọn điểm vẽ như thế nào để đường thẳng được vẽ gần với đường thẳng muốn vẽ nhất và đạt được tối ưu hóa về mặt tốc độ ?

Thuật toán DDA (Digital Differential Analyzer)

Là thuật toán tính toán các điểm vẽ dọc theo đường thẳng dựa vào hệ số góc của phương trình đường thẳng $y=mx+b$.

Trong đó, $m = \frac{\Delta y}{\Delta x}$, $\Delta y = y_{i+1} - y_i$, $\Delta x = x_{i+1} - x_i$

Nhận thấy trong hình vẽ 1.4 thì tọa độ của điểm x sẽ tăng 1 đơn vị trên mỗi điểm vẽ, còn việc quyết định chọn y_{i+1} là $y_i + 1$ hay y_i sẽ phụ thuộc vào giá trị sau khi làm tròn của tung độ y. Tuy nhiên, nếu tính trực tiếp giá trị thực của y ở mỗi bước từ phương trình $y=mx+b$ thì cần một phép toán nhân và một phép toán cộng số thực.

$$y_{i+1} = mx_{i+1} + b = m(x_i + 1) + b = mx_i + b + m$$

Để cải thiện tốc độ, người ta khử phép nhân trên số thực.

$$\text{Ta có : } y_i = mx_i + b$$

$$\text{Suy ra } y_{i+1} = y_i + m \rightarrow \text{int}(y_{i+1})$$

Tóm lại khi $0 < m \leq 1$:

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

$$y_{i+1} = y_i + m \rightarrow \text{int}(y_{i+1})$$

Trường hợp $m > 1$: chọn bước tăng trên trục y một đơn vị.

$$x_{i+1} = x_i + 1/m \rightarrow \text{int}(x_{i+1})$$

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

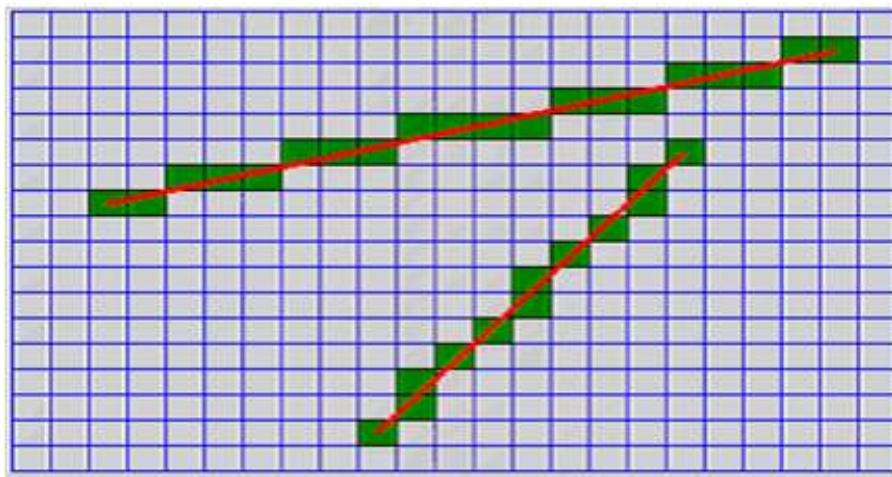
Hai trường hợp này dùng để vẽ một điểm bắt đầu từ bên trái đến điểm cuối cùng bên phải của đường thẳng (xem hình 1.5). Nếu điểm bắt đầu từ bên phải đến điểm cuối cùng bên trái thì xét ngược lại :

$$0 < m \leq 1: x_{i+1} := x_i - 1$$

$$y_{i+1} := y_i - m @ \text{int}(y_{i+1})$$

$$m > 1: x_{i+1} := x_i - 1/m @ \text{int}(x_{i+1})$$

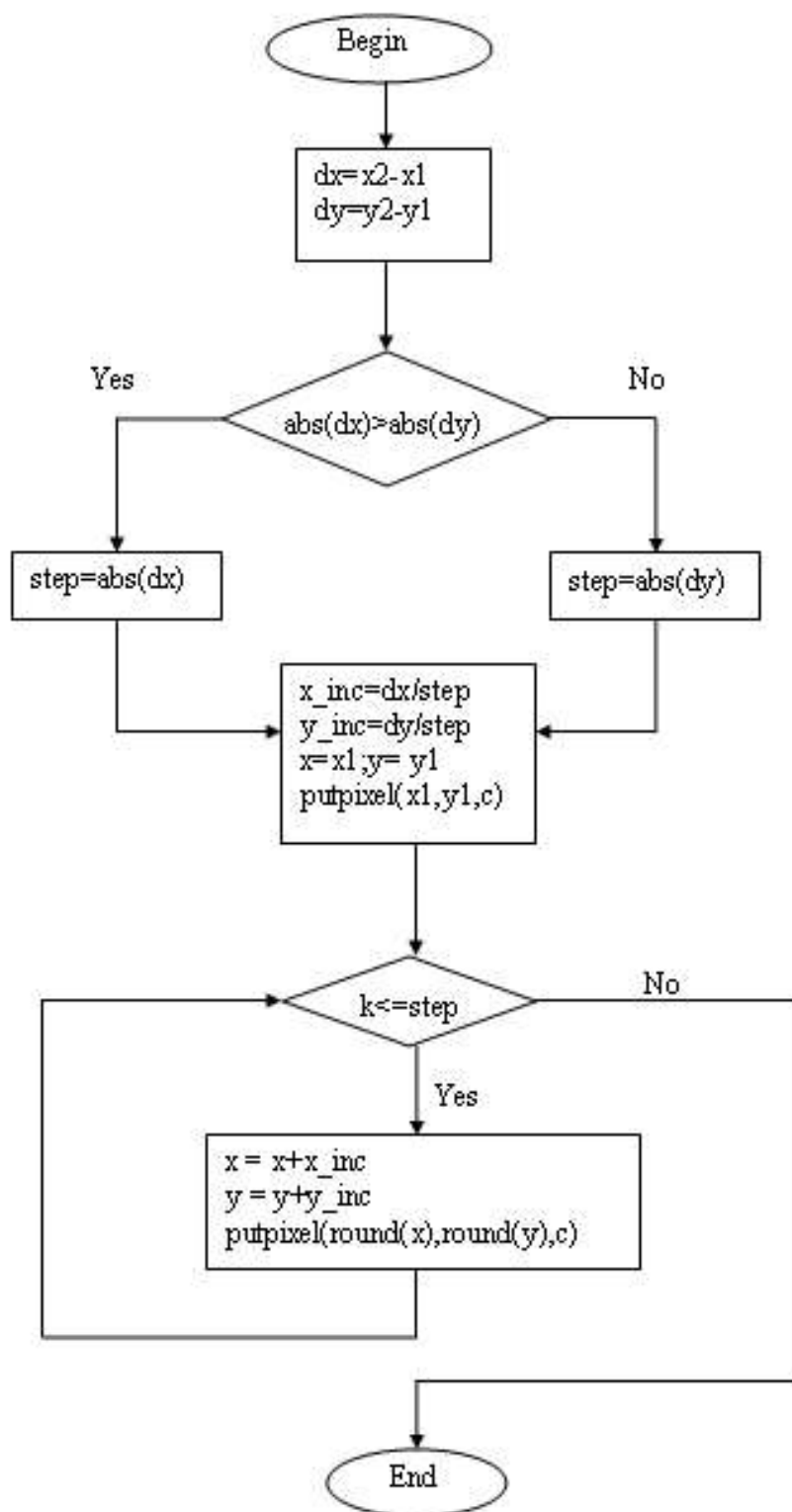
$$y_{i+1} := y_i - 1$$



Hình 1.5 : Hai dạng đường thẳng có $0 < m < 1$ và $m > 1$.

Tương tự, có thể tính toán các điểm vẽ cho trường hợp $m < 0$: khi $|m| \leq 1$ hoặc $|m| > 1$ (sinh viên tự tìm hiểu thêm).

Lưu đồ thuật toán DDA



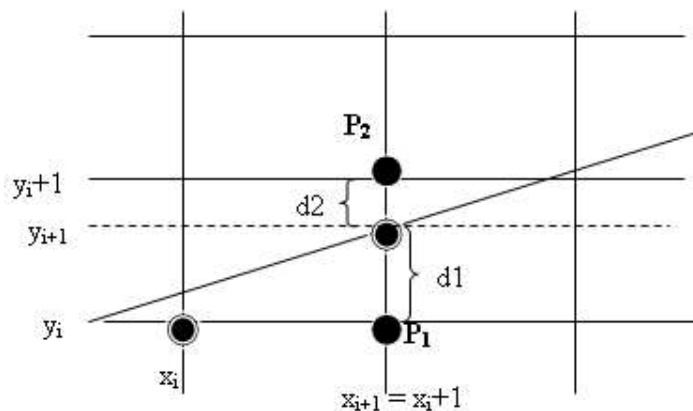
Cài đặt minh họa thuật toán DDA

```

Procedure DDA ( x1, y1, x2, y2, color : integer );
  Var dx, dy, step : integer;
  X_inc, y_inc , x, y : real ;
  Begin
    dx:=x2-x1;
    dy:=y2-y1;
    if abs(dx)>abs(dy) then steps:=abs(dx)
    else steps:=abs(dy);
    x_inc:=dx/steps;
    y_inc:=dy/steps;
    x:=x1; y:=y1;
    putpixel(round(x),round(y), color);
    for k:=1 to steps do
      begin
        x:=x+x_inc;
        y:=y+y_inc;
        putpixel(round(x),round(y), color);
      end;
    end;
  end;

```

Thuật toán Bresenham



Hình 1.6 : Dạng đường thẳng có $0 \leq m \leq 1$.

Gọi $(x_i + 1, y_{i+1})$ là điểm thuộc đoạn thẳng (xem hình 1.6). Ta có $y = m(x_i + 1) + b$.

Đặt $d_1 = y_{i+1} - y_i$

$d_2 = (y_i + 1) - y_{i+1}$

Việc chọn điểm (x_{i+1}, y_{i+1}) là P1 hay P2 phụ thuộc vào việc so sánh d_1 và d_2 hay dấu của $d_1 - d_2$.

- Nếu $d_1 - d_2 < 0$: chọn điểm P1, tức là $y_{i+1} = y_i$

- Nếu $d_1 - d_2 \geq 0$: chọn điểm P2, tức là $y_{i+1} = y_i + 1$

$$\text{Xét } P_i = \Delta x (d_1 - d_2)$$

$$\text{Ta có : } d_1 - d_2 = 2 y_{i+1} - 2y_i - 1$$

$$= 2m(x_i+1) + 2b - 2y_i - 1$$

$$\text{Suy ra } P_i = \Delta x (d_1 - d_2) = \Delta x [2m(x_i+1) + 2b - 2y_i - 1]$$

$$= \Delta x [2 \frac{Dy}{Dx} (x_i+1) + 2b - 2y_i - 1]$$

$$= 2\Delta y(x_i+1) - 2\Delta x.y_i + \Delta x(2b - 1)$$

$$= 2\Delta y.x_i - 2\Delta x.y_i + 2\Delta y + \Delta x(2b - 1)$$

$$\text{Vậy } C = 2\Delta y + \Delta x(2b - 1) = \text{Const}$$

$$\Rightarrow P_i = 2\Delta y.x_i - 2\Delta x.y_i + C$$

Nhận xét rằng nếu tại bước thứ i ta xác định được dấu của P_i thì xem như ta xác định được điểm cần chọn ở bước $(i+1)$. Ta có :

$$P_{i+1} - P_i = (2\Delta y.x_{i+1} - 2\Delta x.y_{i+1} + C) - (2\Delta y.x_i - 2\Delta x.y_i + C)$$

$$\Leftrightarrow P_{i+1} = P_i + 2\Delta y - 2\Delta x (y_{i+1} - y_i)$$

- Nếu $P_i < 0$: chọn điểm $P1$, tức là $y_{i+1} = y_i$ và $P_{i+1} = P_i + 2\Delta y$.

- Nếu $P_i \geq 0$: chọn điểm $P2$, tức là $y_{i+1} = y_i + 1$ và $P_{i+1} = P_i + 2\Delta y - 2\Delta x$

- Giá trị P_0 được tính từ điểm vẽ đầu tiên (x_0, y_0) theo công thức :

$$P_0 = 2\Delta y.x_0 - 2\Delta x.y_0 + C$$

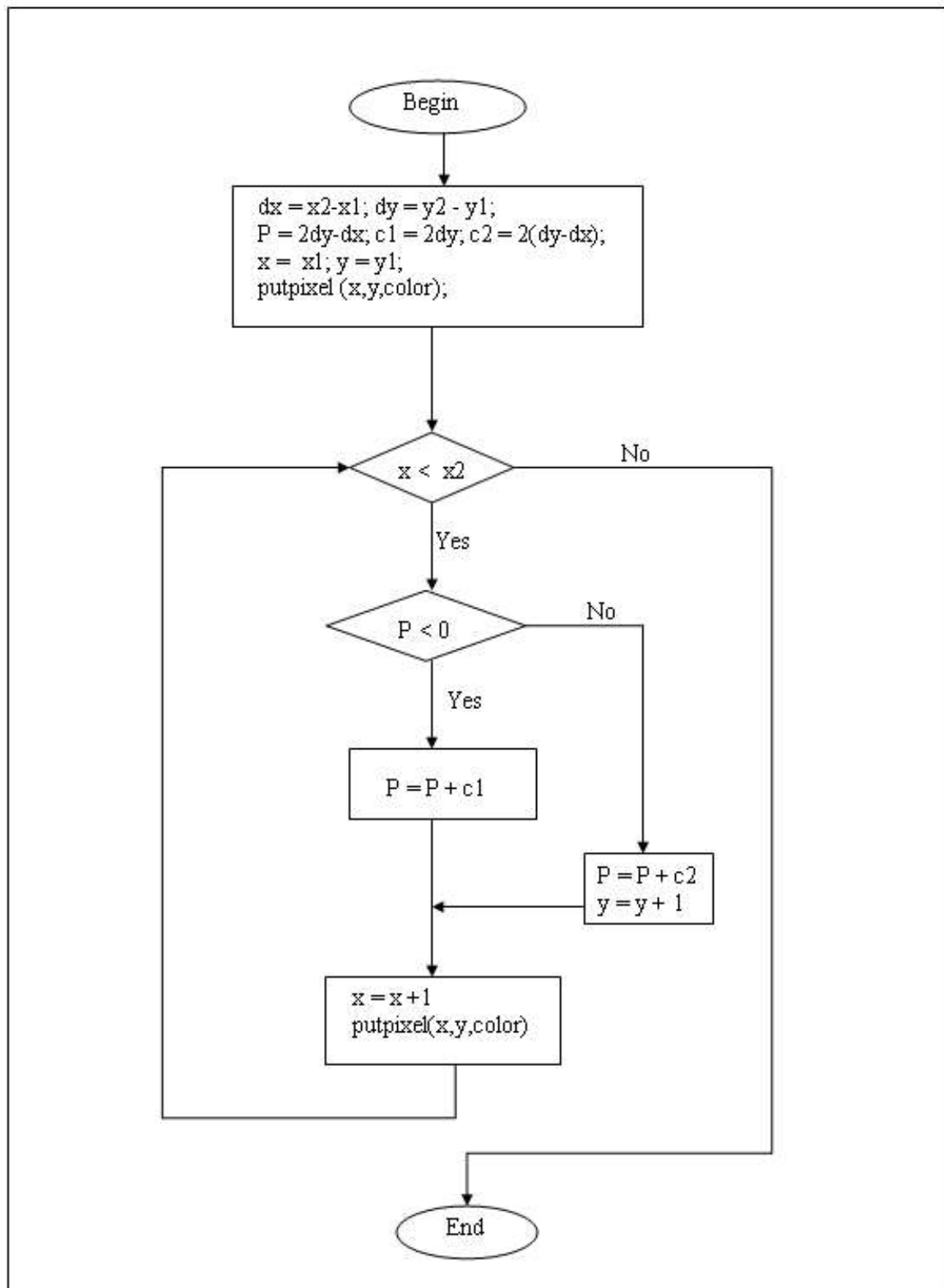
Do (x_0, y_0) là điểm nguyên thuộc về đoạn thẳng nên ta có :

$$y_0 = m .x_0 + b = \frac{Dy}{Dx} .x_0 + b$$

Thế vào phương trình trên ta được :

$$P_0 = 2\Delta y - \Delta x$$

Lưu đồ thuật toán Bresenham



Cài đặt minh họa thuật toán Bresenham

```

Procedure Bres_Line (x1,y1,x2,y2 : integer);
  Var dx, dy, x, y, P, const1, const2 : integer;
  Begin
    dx := x2 - x1; dy := y2 - y1;
    P := 2*dy - dx;
    Const1 := 2*dy ; const2 := 2*(dy - dx) ;
    x:= x1; y:=y1;
    Putpixel ( x, y, Color);
    while (x < x2 ) do
      begin
        x := x +1 ;
        if (P < 0) then P := P + const1
        else
          begin
            y := y+1 ;
            P := P + const2
          end ;
        putpixel (x, y, color) ;
      end ;
    End ;
  
```

Nhận xét :

Thuật toán Bresenham chỉ thao tác trên số nguyên và chỉ tính toán trên phép cộng và phép nhân 2 (phép dịch bit). Điều này là một cải tiến làm tăng tốc độ đáng kể so với thuật toán DDA.

Ý tưởng chính của thuật toán này là ở chỗ xét dấu P_i để quyết định điểm kế tiếp, và sử dụng công thức truy hồi $P_{i+1} - P_i$ để tính P_i bằng các phép toán đơn giản trên số nguyên.

Tuy nhiên, việc xây dựng trường hợp tổng quát cho thuật toán Bresenham có phức tạp hơn thuật toán DDA.

Thuật toán vẽ đường tròn

Trong hệ tọa độ Descartes, phương trình đường tròn bán kính R có dạng:

Với tâm $O(0,0)$: $x^2 + y^2 = R^2$

Với tâm $C(x_c, y_c)$: $(x - x_c)^2 + (y - y_c)^2 = R^2$

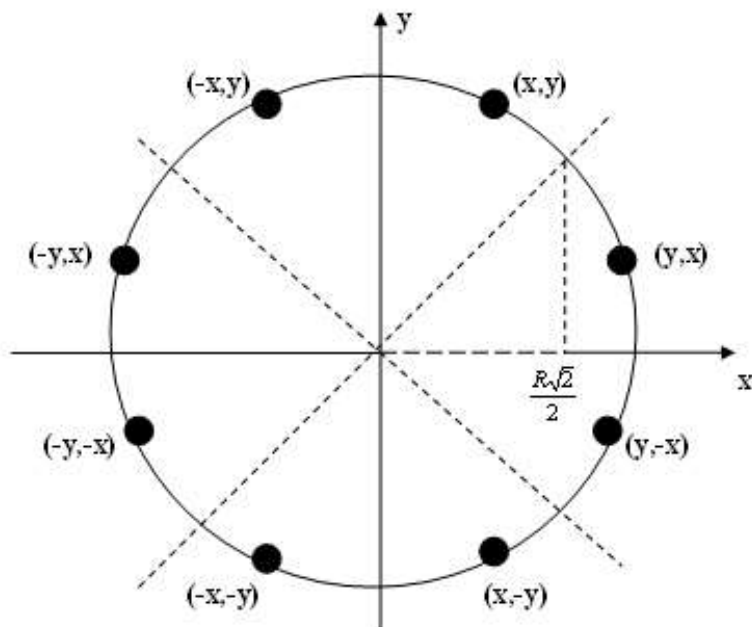
Trong hệ tọa độ cực :

$$x = x_c + R \cdot \cos\theta$$

$$y = y_c + R \cdot \sin\theta$$

với $\theta \in [0, 2\pi]$.

Do tính đối xứng của đường tròn C (xem hình 1.7) nên ta chỉ cần vẽ 1/8 cung tròn, sau đó lấy đối xứng qua 2 trục tọa độ và 2 đường phân giác thì ta vẽ được cả đường tròn.



Hình 1.7 : Đường tròn với các điểm đối xứng.

Thuật toán đơn giản

Cho $x = 0, 1, 2, \dots, \text{int}(\frac{R\sqrt{2}}{2})$ với $R > 1$.

- Tại mỗi giá trị x , tính $\text{int}(y = \sqrt{R^2 - x^2})$.
 - Vẽ điểm (x, y) cùng 7 điểm đối xứng của nó.
- Cài đặt minh họa thuật toán đơn giản.

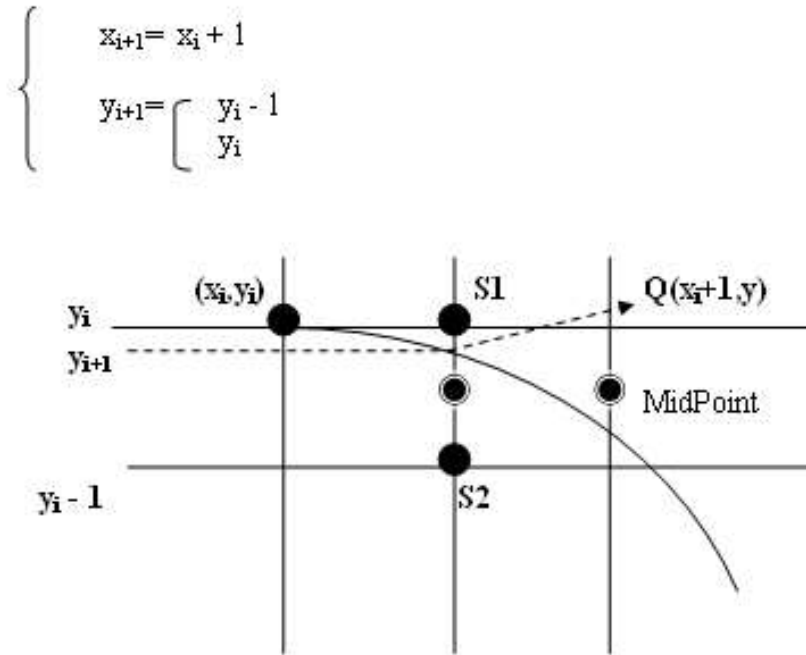
```

Procedure Circle (xc, yc, R : integer) ;
  Var x, y : integer ;
  Procedure DOIXUNG ;
  Begin
    putpixel (xc + x , yc +y, color) ;
    putpixel (xc - x , yc + y, color) ;
    putpixel (xc + x , yc - y, color) ;
    putpixel (xc - x , yc- y, color) ;
    putpixel (xc + y , yc + x, color) ;
    putpixel (xc - y , yc + x, color) ;
    putpixel (xc + y , yc - x, color) ;
    putpixel (xc - y , yc - x, color) ;
  End ;
  Begin
    For x := 0 to round(R*Sqrt(2)/2) do
      Begin
        y := round(Sqrt(R*R - x*x)) ;
        DOIXUNG;
      End ;
    End ;
  End ;

```

Thuật toán xét điểm giữa (MidPoint)

Do tính đối xứng của đường tròn nên ta chỉ cần vẽ 1/8 cung tròn, sau đó lấy đối xứng là vẽ được cả đường tròn. Thuật toán MidPoint đưa ra cách chọn y_{i+1} là y_i hay y_{i-1} bằng cách so sánh điểm thực $Q(x_{i+1}, y)$ với điểm giữa MidPoint là trung điểm của S1 và S2. Chọn điểm bắt đầu để vẽ là $(0, R)$. Giả sử (x_i, y_i) là điểm nguyên đã tìm được ở bước thứ i (xem hình 1.8), thì điểm (x_{i+1}, y_{i+1}) ở bước $i+1$ là sự lựa chọn giữa S1 và S2.



Hình 1.8 : Đường tròn với điểm $Q(x_i+1, y)$ và điểm MidPoint.

Đặt $F(x,y) = x^2 + y^2 - R^2$, ta có :

. $F(x,y) < 0$, nếu điểm (x,y) nằm trong đường tròn.

. $F(x,y) = 0$, nếu điểm (x,y) nằm trên đường tròn.

. $F(x,y) > 0$, nếu điểm (x,y) nằm ngoài đường tròn.

Xét $P_i = F(\text{MidPoint}) = F(x_i + 1, y_i - 1/2)$. Ta có :

- Nếu $P_i < 0$: điểm MidPoint nằm trong đường tròn. Khi đó, điểm thực Q gần với điểm S1 hơn nên ta chọn $y_{i+1} = y_i$.

- Nếu $P_i \geq 0$: điểm MidPoint nằm ngoài đường tròn. Khi đó, điểm thực Q gần với điểm S2 hơn nên ta chọn $y_{i+1} = y_i - 1$.

Mặt khác :

$$\begin{aligned} P_{i+1} - P_i &= F(x_{i+1} + 1, y_{i+1} - 1/2) - F(x_i + 1, y_i - 1/2) \\ &= [(x_{i+1} + 1)^2 + (y_{i+1} - 1/2)^2 - R^2] - [(x_i + 1)^2 + (y_i - 1/2)^2 - R^2] \\ &= 2x_i + 3 + ((y_{i+1})^2 + (y_i)^2) - (y_{i+1} - y_i) \end{aligned}$$

Vậy :

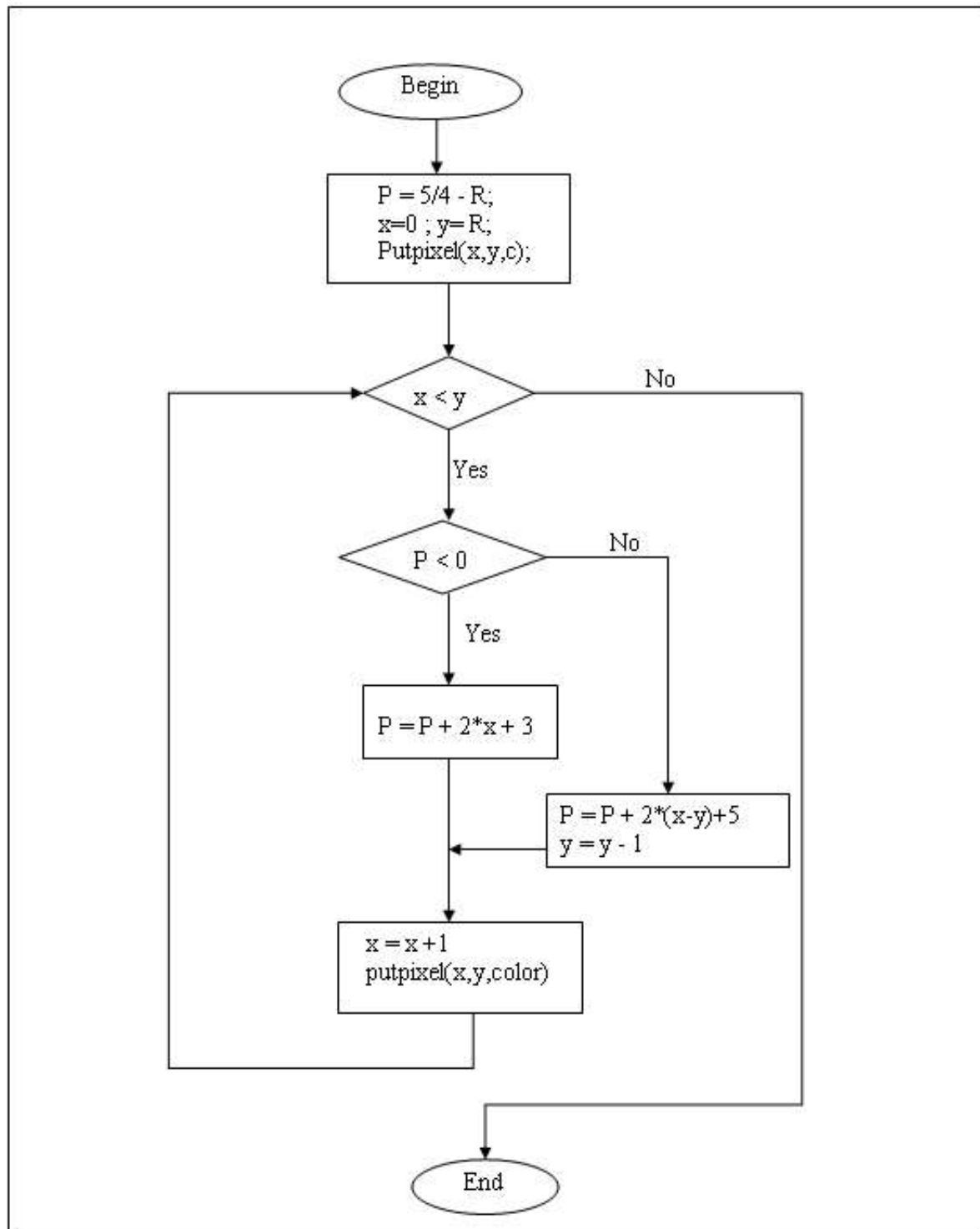
- Nếu $P_i < 0$: chọn $y_{i+1} = y_i$. Khi đó $P_{i+1} = P_i + 2x_i + 3$

- Nếu $P_i \geq 0$: chọn $y_{i+1} = y_i - 1$. Khi đó $P_{i+1} = P_i + 2x_i - 2y_i + 5$.

- P_i ứng với điểm ban đầu $(x_0, y_0) = (0, R)$ là:

$$P_0 = F(x_0 + 1, y_0 - 1/2) = F(1, R - 1/2) = \frac{5}{4} - R$$

Lưu đồ thuật toán MidPoint vẽ đường tròn



Minh họa thuật toán MidPoint:

```

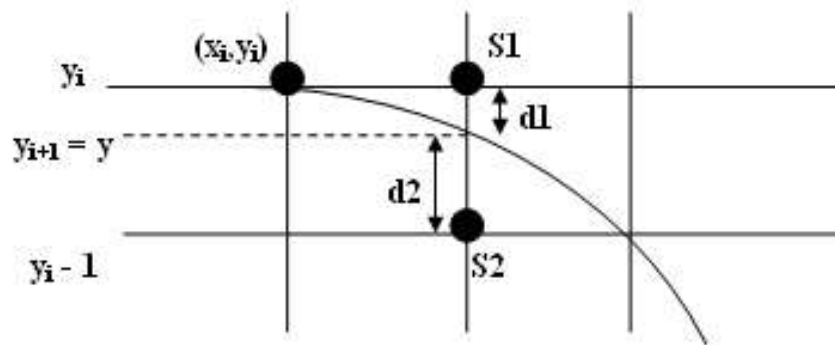
Procedure DTR(xc, yc, r, mau : integer);
Var    dx, dy, step : integer;
      X_inc, y_inc , x, y : real ;
Begin
  dx:=x2-x1;
  dy:=y2-y1;
  if abs(dx)>abs(dy) then  steps:=abs(dx)
  else  steps:=abs(dy);
  x_inc:=dx/steps;
  y_inc:=dy/steps;
  x:=x1;  y:=y1;
  putpixel(round(x),round(y), color);
  for k:=1 to steps do
  begin
    x:=x+x_inc;
    y:=y+y_inc;
    putpixel(round(x),round(y),  color);

  end;
end;

```

Vẽ đường tròn bằng thuật toán Bresenham

Tương tự thuật toán vẽ đường thẳng Bresenham, các vị trí ứng với các tọa độ nguyên nằm trên đường tròn có thể tính được bằng cách xác định một trong hai pixel gần nhất với đường tròn thực hơn trong mỗi bước (xem hình 1.9).



Hình 1.9 : Đường tròn với khoảng cách $d1$ và $d2$.

Ta có :

$$d1 = (y_i)^2 - y^2 = (y_i)^2 - (R^2 - (x_i + 1)^2)$$

$$d2 = y^2 - (y_i - 1)^2 = (R^2 - (x_i + 1)^2) - (y_i - 1)^2$$

$$P_i = d1 - d2$$

Tính $P_{i+1} - P_i$

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

- Nếu $P_i < 0$: chọn $y_{i+1} = y_i$. Khi đó $P_{i+1} = P_i + 4x_i + 6$

- Nếu $P_i \geq 0$: chọn $y_{i+1} = y_i - 1$. Khi đó $P_{i+1} = P_i + 4(x_i - y_i) + 10$.

- P_0 ứng với điểm ban đầu $(x_0, y_0) = (0, R)$ là: $P_0 = 3 - 2R$.

Minh họa thuật toán vẽ đường tròn bằng Bresenham

```

Procedure DTR_BRES(xc,yc,r,mau : integer);
var  x,y,p:integer;
begin
  x:=0 ; y:=r;
  p:= 3 - 2*r ;
  while ( x < y ) do
    begin
      doi_xung;
      if (p < 0) then p:= p + 4*x + 6
      else begin
        p:= p + 4*(x-y) + 10 ;
        y:=y-1;
      end;
      x:=x+1;
    end;{while}
  end;

```

Thuật toán vẽ Ellipse

Tương tự thuật toán vẽ đường tròn, sử dụng thuật toán Bresenham để vẽ, ta chỉ cần vẽ 1/4 ellipse, sau đó lấy đối xứng qua các trục tọa độ sẽ vẽ được toàn bộ ellipse.

Xét ellipse có tâm O, các bán kính là a và b, phương trình là : $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

Chọn tọa độ pixel đầu tiên cần hiển thị là $(x_i, y_i) = (0, b)$. Cần xác định pixel tiếp theo là (x_{i+1}, y_{i+1}) . Ta có :

$$\begin{cases} x_{i+1} = x_i + 1 \\ y_{i+1} = \left\lfloor y_i - 1 \right\rfloor \end{cases}$$

$$d1 = (y_i)^2 - y^2$$

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

$$P_i = d1 - d2$$

Tính $P_{i+1} - P_i$

$$\Rightarrow P_{i+1} = P_i + 2((y_{i+1})^2 - (y_i)^2) - 2(y_{i+1} - y_i) + \frac{2b^2}{a^2}(2x_i + 3)$$

- Nếu $P_i < 0$: chọn $y_{i+1} = y_i$. Khi đó $P_{i+1} = P_i + \frac{2b^2}{a^2} (2x_i + 3)$

- Nếu $P_i \geq 0$: chọn $y_{i+1} = y_i - 1$. Khi đó $P_{i+1} = P_i + \frac{2b^2}{a^2} (2x_i + 3) + 4(1 - y_i)$

- P_i ứng với điểm ban đầu $(x_0, y_0) = (0, b)$ là: $P_0 = \frac{2b^2}{a^2} - 2b + 1$

Minh họa thuật toán vẽ Ellipse

```

Procedure Ellipse(xc,yc,a,b : integer);
  var x,y : integer;
  z1, z2, P : real;
  procedure dx;
  begin
    putpixel (xc + x , yc +y, color) ;
    putpixel (xc - x , yc + y, color) ;
    putpixel (xc + x , yc - y, color) ;
    putpixel (xc - x , yc- y, color) ;
  end;
  begin
    x:=0 y:=b;
    z1:= (b*b)/(a*a);
    z2:= 1/ z1;
    P:= 2*z1 - 2*b +1;
    while (z1* (x/y) ≤ 1) do
      begin
        dx;
        if P < 0 then P:= P + 2*z1*(2*x+3)
        else begin
          P:= P + 2*z1*(2*x+3) + 4*(1-y);
          y:= y -1;
        end;
        x:= x+1;
      end;
      x:=a ; y:= 0;
      P:= 2*z2 - 2*a +1;
      while (z2* (y/x) < 1) do
        begin
          dx;
          if P < 0 then P:= P + 2*z2*(2*y+3)
          else begin
            P:= P + 2*z2*(2*y+3) + 4*(1-x);
            x:= x -1;
          end;
          y:= y +1;
        end;
      end;
    end;
  end;

```

Vẽ đường conics và một số đường cong khác

Phương trình tổng quát của các đường conics có dạng :

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

Giá trị của các hằng số A, B, C, D, E, F sẽ quyết định dạng của đường conics, cụ thể là nếu:

$B^2 - 4AC < 0$: dạng đường tròn (nếu $A=C$ và $B=0$) hay ellipse.

$B^2 - 4AC = 0$: dạng parabol.

$B^2 - 4AC > 0$: dạng hyperbol.

Áp dụng ý tưởng của thuật toán Midpoint để vẽ các đường conics và một số đường cong khác theo các bước theo các bước tuần tự sau:

- Bước 1: Dựa vào dáng điệu và phương trình đường cong, để xem thử có thể rút gọn phần đường cong cần vẽ hay không.

- Bước 2: Tính đạo hàm, từ đó phân thành các vùng vẽ.

. Nếu $0 \leq f'(x) \leq 1$: $x_{i+1} = x_i + 1$; $y_{i+1} = y_i$ (hoặc $y_i + 1$)

. Nếu $-1 \leq f'(x) \leq 0$: $x_{i+1} = x_i + 1$; $y_{i+1} = y_i$ (hoặc $y_i - 1$)

. Nếu $f'(x) > 1$: $y_{i+1} = y_i + 1$; $x_{i+1} = x_i$ (hoặc $x_i + 1$)

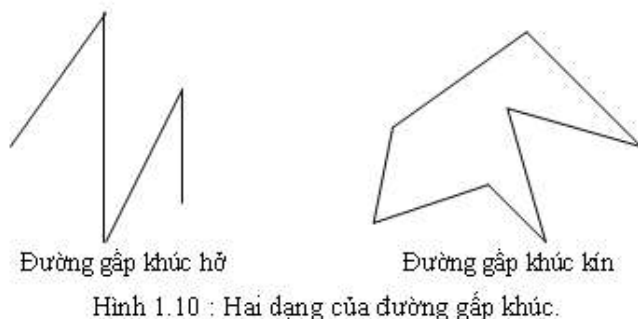
. Nếu $f'(x) < -1$: $y_{i+1} = y_i + 1$; $x_{i+1} = x_i$ (hoặc $x_i + 1$)

- Bước 3 : Tính P_i cho từng trường hợp để quyết định $f'(x)$ dựa trên dấu của P_i . P_i thường là hàm được xây dựng từ phương trình đường cong. Cho $P_i=0$ nếu (x_i, y_i) thuộc về đường cong. Việc chọn P_i cần chú ý sao cho các thao tác tính P_i sau này hạn chế 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_0 và hoàn chỉnh thuật toán.

Vẽ đa giác



Định nghĩa đa giác (Polygone): Đa giác là một đường gấp khúc kín có đỉnh đầu và đỉnh cuối trùng nhau (xem hình 1.10)

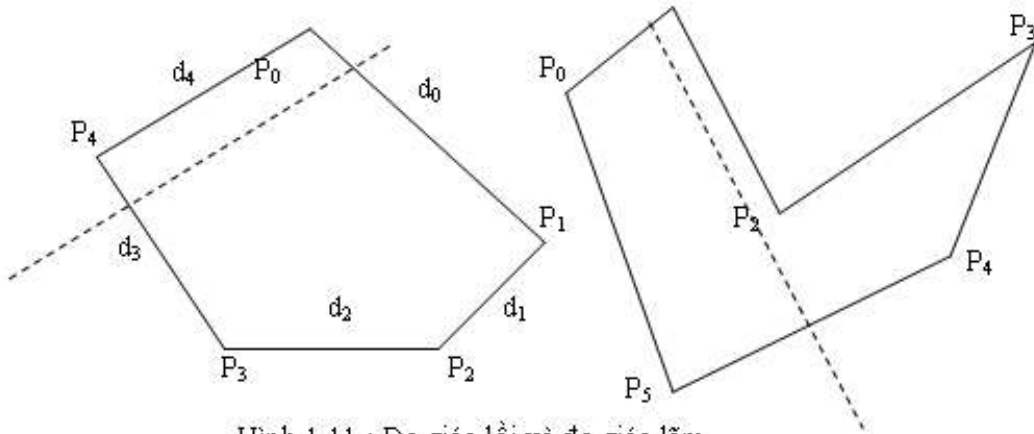
Xây dựng cấu trúc dữ liệu để vẽ đa giác

Type

```
d_dinh = record
  x,y: longint;
end;
dinh = array[0..10] of d_dinh;
var
  d: dinh;
```

Với cách xây dựng cấu trúc dữ liệu như thế này thì chúng ta chỉ cần nhập vào tọa độ các đỉnh và sau đó gọi thủ tục vẽ đường thẳng lần lượt qua 2 đỉnh như $(0, 1)$, $(1, 2)$, ..., $(n-1, n)$, trong đó đỉnh n trùng với đỉnh 0 thì ta sẽ vẽ được toàn bộ đa giác.

Đa giác được gọi là **lồi**: nếu bất kỳ đường thẳng nào đi qua một cạnh của đa giác thì toàn bộ đa giác nằm về một phía của đường thẳng đó. Ngược lại, nếu tồn tại ít nhất một cạnh của đa giác chia đa giác làm 2 phần thì gọi là **đa giác lõm** (xem hình 1.11).



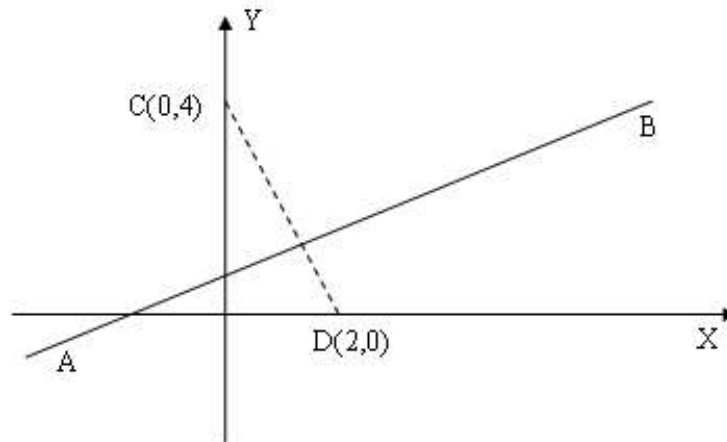
Hình 1.11 : Đa giác lồi và đa giác lõm

Thuật toán kiểm tra một đa giác là lồi hay lõm

Thuật toán 1: Lần lượt thiết lập phương trình đường thẳng đi qua các cạnh của đa giác. Ứng với từng phương trình đường thẳng, xét xem các đỉnh còn lại có nằm về một phía đối với đường thẳng đó hay không? Nếu đúng thì kết luận đa giác lồi, ngược lại là đa giác lõm.

Nhận xét : Phương trình đường thẳng $y = ax + b$ chia mặt phẳng ra làm 2 phần. Các điểm nằm $C(x_c, y_c)$ trên đường thẳng sẽ có $y_c = ax_c + b$ và các điểm $D(x_d, y_d)$ nằm phía dưới đường thẳng sẽ có $y_d < ax_d + b$.

Ví dụ : Cho đường thẳng AB có phương trình $y = \frac{1}{2}x + 1$ và hai điểm C, D có tọa độ là $C(0, 4)$, $D(2, 0)$ (xem hình 1.12).



Hình 1.12 : Đường thẳng AB và 2 điểm C, D.

Ta có : $Y_c = 4 > ax_c + b = \frac{1}{2} \cdot 0 + 1$

và $Y_d = 0 < ax_d + b = \frac{1}{2} \cdot 2 + 1$

Vậy hai điểm C, D nằm về hai phía đối với đường thẳng AB.

Thuật toán 2 :

Nhận xét :

Trong mặt phẳng Oxy, cho 2 véc tơ \vec{a} và \vec{b} , Tích vô hướng của 2 véc tơ là :

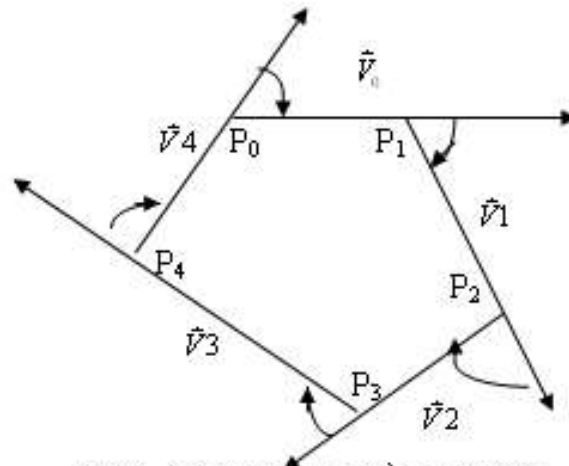
$$T(\vec{a}, \vec{b}) = \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix} = a_x \cdot b_y - a_y \cdot b_x$$

Khi đó :

\vec{a} quẹo trái sang \vec{b} nếu $T \geq 0$

\vec{a} quẹo phải sang \vec{b} nếu $T < 0$

Một đa giác là lồi khi đi dọc theo biên của nó thì chỉ đi theo một hướng mà thôi. Nghĩa là chỉ quẹo phải hay quẹo trái. Ngược lại là đa giác lõm (xem hình 1.13).



Hình 1.13 : Đa giác lồi có 5 đỉnh.

Xét đa giác gồm các đỉnh P_0, P_1, \dots, P_n , ($P_0 = P_n$), $n \geq 3$ (xem hình 1.13).

Tính $V_i = P_{i+1} - P_i$, $\forall i = 0, 1, \dots, n-1$.

Tính $T_i = T(V_i, V_{i+1})$

Nếu với mọi T_i đều cùng dấu thì kết luận đa giác lồi.

Ngược lại, là đa giác lõm.

Tổng kết chương

- Chương này đã trình bày khái niệm về một hệ độ họa, sự hiển thị của điểm trên màn hình với tọa độ phải là số nguyên.
- Phân biệt thế nào là hệ tọa độ thế giới thực, hệ tọa độ thiết bị và hệ tọa độ chuẩn.
- Cần lưu ý về hệ số góc của đường thẳng. Bởi vì, với hệ số góc khác nhau thì giải thuật có thay đổi. Nhất là trong giải thuật Bresenham.
- Chú ý hơn trong cách xây dựng cấu trúc dữ liệu để lưu tọa độ của các đỉnh đa giác.
- So sánh các trường hợp sử dụng công thức của các đường cong (có tham số và không có tham số).

Bài tập chương

Viết chương trình vẽ bầu trời có 10.000 điểm sao, mỗi điểm sao xuất hiện với một màu ngẫu nhiên. Những điểm sao này hiện lên rồi từ từ tắt cũng rất ngẫu nhiên.

Viết chương trình thực hiện 2 thao tác sau :

- Khởi tạo chế độ đồ họa, đặt màu nền, đặt màu chữ, định dạng chữ (`settextstyle(f,d,s)`), xuất một chuỗi ký tự ra màn hình. Đổi font, hướng, kích thước.
- Xuất một chuỗi ra màn hình, chuỗi này có tô bóng.

(Lưu ý rằng nội dung chuỗi ký tự, màu tô, màu bóng là được nhập từ bàn phím).

Viết chương trình vẽ đoạn thẳng AB với màu color theo giải thuật DDA. Biết rằng tọa độ A,B, color được nhập từ bàn phím.

Trang trí màu nền, ghi chú các tọa độ A, B ở hai đầu đoạn thẳng.

Tương tự như bài tập 3 nhưng sử dụng giải thuật Bresenham. Lưu ý các trường hợp đặc biệt của hệ số góc.

Tổng hợp bài tập 4, viết chương trình vẽ đường thẳng bằng giải thuật Bresenham cho tất cả các trường hợp của hệ số góc. Lưu ý xét trường hợp đặc biệt khi đường thẳng song song với trục tung hay với trục hoành.

Viết chương trình nhập tọa độ 3 điểm A, B, C từ bàn phím. Tìm tọa độ điểm D thuộc AB sao cho CD vuông góc AB. Vẽ đoạn thẳng AB và CD.

Viết chương trình xét vị trí tương đối của 2 đoạn thẳng AB và CD. Biết rằng trong màn hình đồ họa đoạn thẳng AB và CD được gọi là cắt nhau khi hai điểm A, B ở về hai phía của CD và ngược lại.

Viết chương trình vẽ đường tròn theo giải thuật đơn giản (đối xứng).

Viết chương trình vẽ đường tròn theo giải thuật Bresenham.

Viết chương trình vẽ đường tròn theo giải thuật MidPoint.

Viết chương trình vẽ một đường tròn tâm O bán kính R. Vẽ các đường tròn đồng tâm với O, có bán kính chạy từ 1 đến R.

Sau đó xóa các đường tròn đồng tâm này và vẽ các đường tròn đồng tâm khác đi từ R đến 1.

Viết chương trình vẽ một đường tròn tâm O bán kính R. Hãy vẽ một đoạn thẳng từ tâm O độ dài R. Hãy quay đoạn thẳng này quanh đường tròn.

Viết chương trình vẽ Ellipse.

Viết chương trình vẽ Ellipse có bán kính lớn là a, bán kính nhỏ là b và một đường tròn nội tiếp Ellipse. Tô đường tròn bằng các đường tròn đồng tâm. Sau đó tô ellipse bằng các ellipse đồng tâm có bán kính lớn chạy từ b đến a, bán kính nhỏ là b.

Viết chương trình vẽ một hình chữ nhật, một hình vuông và một hình bình hành. Yêu cầu chú thích tọa độ các đỉnh.

Viết chương trình vẽ một tam giác. Tọa độ các đỉnh được nhập từ bàn phím, mỗi cạnh có một màu khác nhau.

Viết chương trình vẽ một đa giác có n đỉnh.

Viết chương trình xét tính lồi lõm của một đa giác bằng cách thiết lập phương trình đường thẳng đi qua các cạnh của đa giác.

Viết chương trình xét tính lồi lõm của một đa giác bằng cách thiết lập các véc tơ chỉ phương của các cạnh.

Thích 7

Chia sẻ 7

Tweet

0

1 bình luận

Sắp xếp theo

Cũ nhất

Viết bình luận...



Hương Ly


đây là thuật toán cho mọi trường hợp hả

Thích · Phản hồi · 3 năm

Plugin bình luận trên Facebook

TÀI VỀ

TÁI SỬ DỤNG(/user/reuse/m/6d58afd1/1)



unknown (/profile/3)

0 GIÁO TRÌNH (/PROFILE/3?TYPES=2) | 1060 TÀI LIỆU (/PROFILE/3?TYPES=1)

(/profile/3)

ĐÁNH GIÁ:

0 dựa trên 0 đánh giá

NỘI DUNG CÙNG TÁC GIẢ

- Hàm và Script file (/m/ham-va-script-file/e984717a)
- Bài khí (/m/bai-khi/eaae03df)
- Mục tiêu của quá trình tiết trùng (/m/muc-tieu-cua-qua-trinh-tiet-trung/bff925a5)
- Vị trí, đối tượng, phương pháp và chức năng của chủ nghĩa xã hội khoa học (/m/vi-tri-doi-tuong-phuong-phap-va-chuc-nang-cua-chu-nghia-xa-hoi-khoa-hoc/c9a76cb8)
- Trình tự, nội dung lập quy hoạch sử dụng đất chi tiết và kế hoạch sử dụng đất chi tiết kỳ đầu (/m/trinh-tu-noi-dung-lap-quy-hoach-su-dung-dat-chi-tiet-va-ke-hoach-su-dung-dat-chi-tiet-ky-dau/e3ebd4f7)
- Khái niệm về các chất dinh dưỡng và thành phần lương thực thực phẩm (/m/khai-niem-ve-cac-chat-dinh-duong-va-thanh-phan-luong-thuc-thuc-pham/b6536f36)
- Vai trò của đạo đức mới trong việc xây dựng chủ nghĩa xã hội (/m/vai-tro-cua-dao-duc-moi-trong-viec-xay-dung-chu-nghia-xa-hoi/71036e56)
- Nguyên lý đánh bắt lưới kéo (/m/nguyen-ly-danh-bat-luoi-keo/b06be6f8)
- Các hệ vi sinh vật trong đồ hộp (/m/cac-he-vi-sinh-vat-trong-do-hop/693261f4)
- Phương pháp móc mối và kỹ thuật câu (/m/phuong-phap-moc-moi-va-ky-thuat-cau/cd61c2f8)

TRƯỚC | TIẾP

NỘI DUNG TƯƠNG TỰ

- Bài toán về hệ thống đại diện chung (/m/bai-toan-ve-he-thong-dai-dien-chung/b0eb39a7)
- Các thuật toán phân rã (/m/cac-thuat-toan-phan-ra/505fec27)
- SINH MÃ TÁCH ĐƯỢC (Decypherable Coding) (/m/sinh-ma-tach-duoc-decypherable-coding/5ef4b4d3)
- Lược đồ giải mã (/m/luoc-do-giai-ma/fe4959d1)
- MÁY BIẾN ÁP (/m/may-bien-ap/393fce9a)
- Bài toán nội suy (/m/bai-toan-noi-suy/b6083ffd)
- Natri aluminat (/m/natri-aluminat/0e1073fb)
- Ngôn ngữ và hàm tính được (/m/ngon-ngu-va-ham-tinh-duoc/76c46bed)
- Bài toán đám cưới vùng quê (/m/bai-toan-dam-cuoi-vung-que/4fbe7a14)
- Giới thiệu thuật toán vẽ và tô các đường cơ bản (/m/gioi-thieu-thuat-toan-ve-va-to-cac-duong-co-ban/817a7a93)

TRƯỚC | TIẾP



Thư viện Học liệu Mở Việt Nam (VOER) được tài trợ bởi Vietnam Foundation (<http://www.vnfoundation.org>) và vận hành trên nền tảng Hanoi Spring (<http://www.hanoispring.com>). Các tài liệu đều tuân thủ giấy phép Creative Commons Attribution 3.0 trừ khi ghi chú rõ ngoại lệ.

 [Connect with Facebook](https://www.facebook.com/voer.edu.vn) (<https://www.facebook.com/voer.edu.vn>)

