BÀI GIẢNG ĐỒ HỌA MÁY TÍNH

CÁC THUẬT GIẢI VẼ ĐƯỜNG THẲNG VÀ CONG

NGÔ QUỐC VIỆT

2010

Nội dung

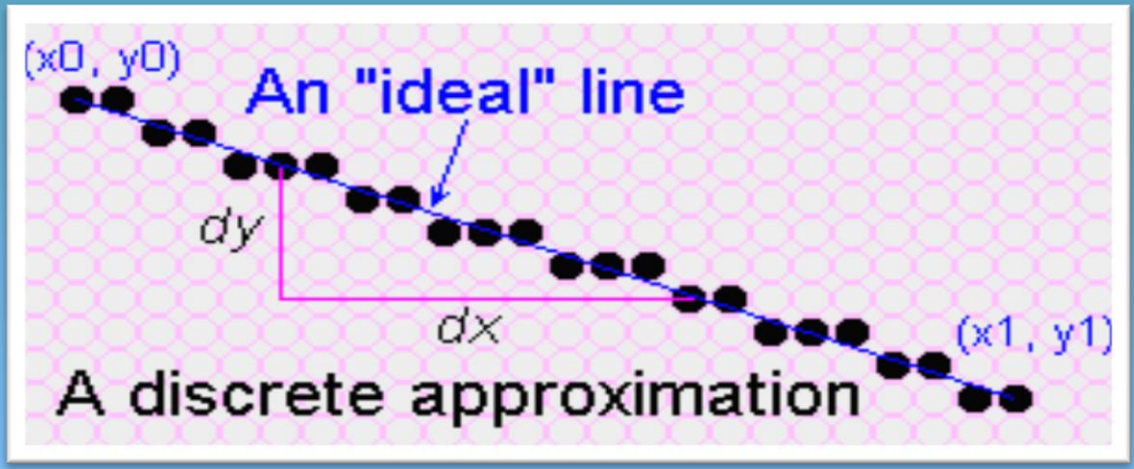
- Thuật giải vẽ đường thẳng
- Thuật giải vẽ đường tròn và conic
- Giải đáp thắc mắc
- Bài tập

Giới thiệu

- Nhu cầu chuyển từ vector sang raster → *rasterization*. Vì tính chất tự nhiên của thiết bị hiển thị raster.
- Các thuật giải là cơ bản cho cả đồ họa 2D và 3D.
- Chuyển từ liên tục (thực tế) sang rời rạc (lấy mẫu).
- Most incremental line-drawing algorithms were first developed for pen-plotters.
- Hầu hết đều dựa trên ý tưởng của Jack Bresenham (kỹ sư IBM)

Thuật giải vẽ đường thẳng

- Vấn đề: Vẽ đoạn thẳng trên thiết bị raster.
- Giải quyết: tiếp cận tốt nhất là xấp xỉ đường lý tưởng.



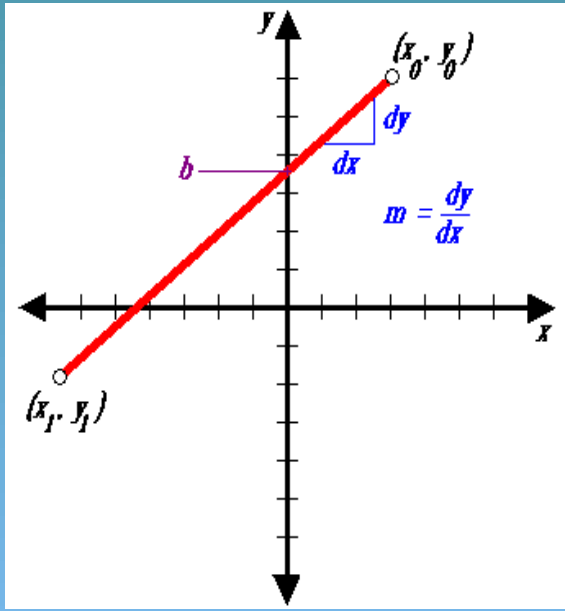
- Yêu cầu: *nhìn liên tục; độ sáng và độ dày đồng nhất; Xấp xỉ gần đường lý tưởng nhất; vẽ nhanh.*

Thuật giải vẽ đường thẳng-dựa trên độ dốc

y = m x + b

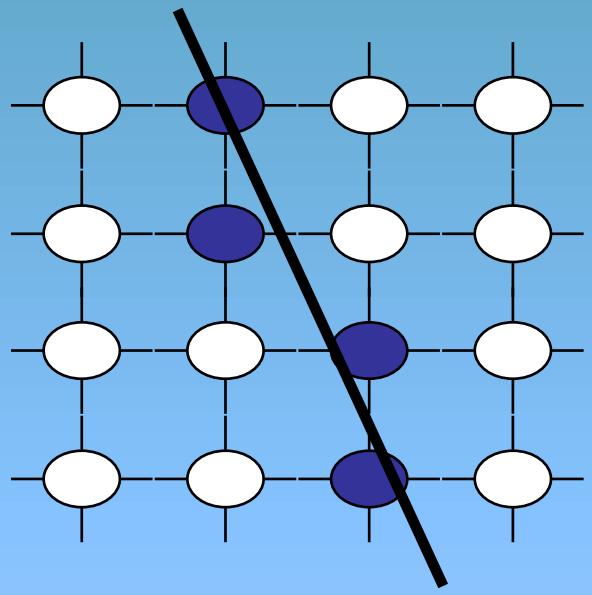
slope the y intercept

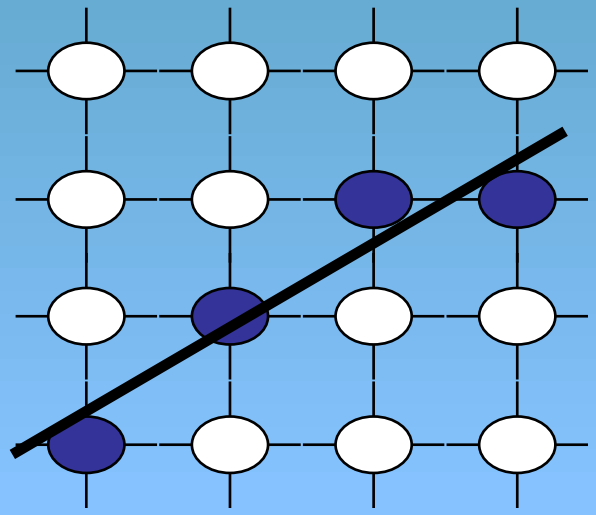
```
public void lineSimple(int x0, int y0, int x1, int y1, Color color)
{
int pix = color.getRGB();
 int dx = x1 - x0;
 int dy = y1 - y0;
raster.setPixel(pix, x0, y0);
if (dx != 0) {
 float m = (float) dy / (float) dx;
float b = y0 - m*x0;
dx = (x1 > x0) ? 1 : -1;
while (x0 != x1) {
x0 += dx;
y0 = Math.round(m*x0 + b);
raster.setPixel(pix, x0, y0);
} } }
```



Thuật giải vẽ đường thẳng-dựa trên độ dốc

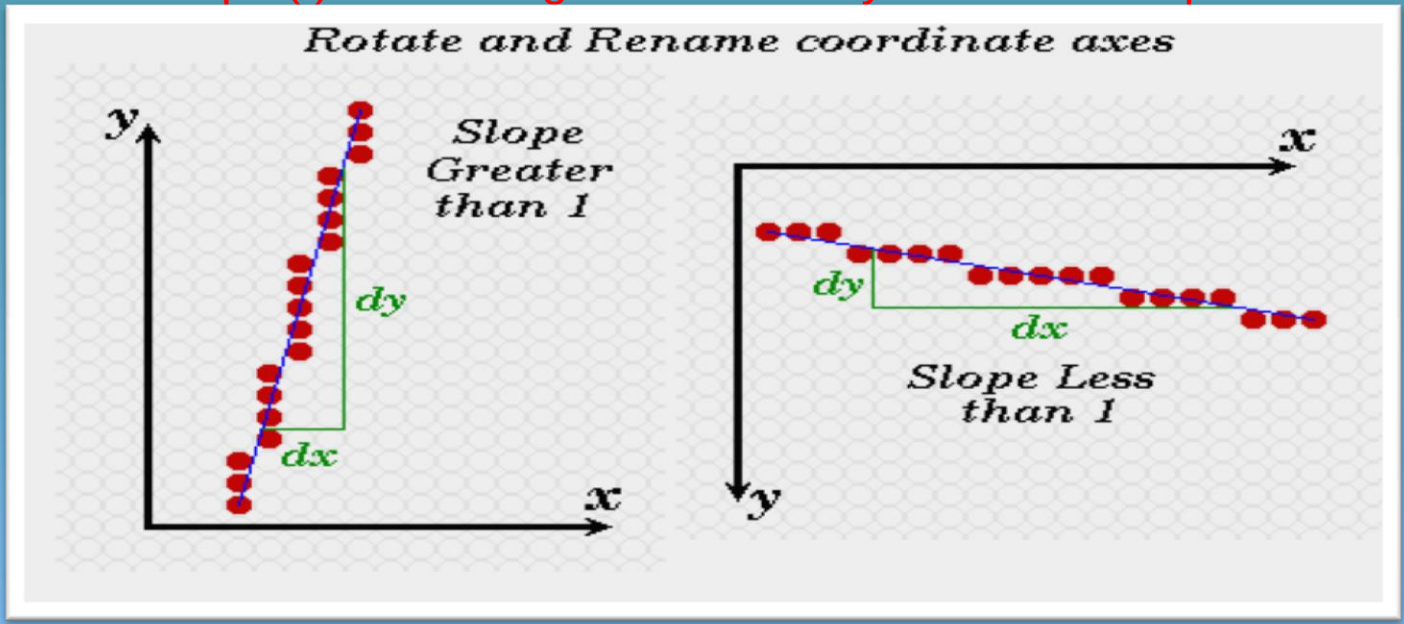
- Mục tiêu: vẽ đường càng mịn càng tốt (một pixel mỗi cột nếu -1 < slope <1, ngược lại một pixel mỗi hàng).





Thuật giải vẽ đường thẳng-dựa trên độ dốc

Problem: lineSimple( ) does not give satisfactory results for slopes > 1



Thuật giải không tốt khi độ dốc > 1. Giải pháp: **làm đối xứng**. Nghĩa là hoán vị vai trò của trục *x* và *y*. Nhờ vậy, độ dốc luôn nhỏ hơn 1.

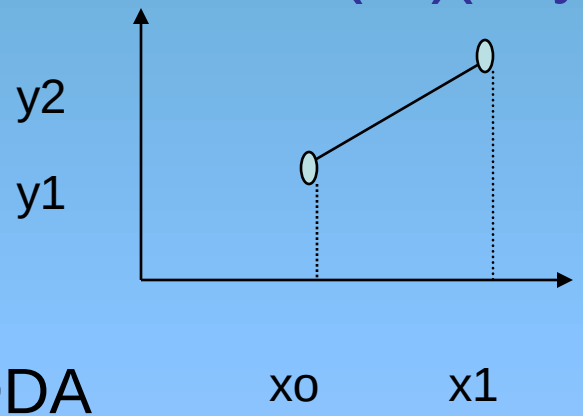
Thuật giải vẽ đường thẳng-dựa trên độ dốc => Cải tiến

- Cải tiến đoạn code nào làm tốn thời gian. Thường là các vòng lặp trong.
- Bỏ các lệnh không cần thiết. Ví dụ:
- Thay **Math.round(m*x0 + b)** bởi **(int)(m*y0 + b + 0.5)**;
- Sử dụng kết quả của bước trước: **(int)(m*y0 + b + 0.5)**

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

hoặc

$y_{i+1} = y_i - m;$



- Phát sinh ra thuật giải DDA

Thuật giải vẽ đường thẳng-Cải tiến thêm

Nguyên tắc:

- Cộng/trừ thì nhanh hơn nhân. Nhân nhanh hơn chia.
- Dùng bảng tra nếu được.
- Tính toán số nguyên nhanh hơn số thực.
- Tránh tính toán thừa bằng cách kiểm tra các trường hợp đặc biệt

Thuật giải DDA

- Xét: **m = (y1 - y0) / (x1 - x0)** . Giả sử: 0< m < 1
- Nhận xét: y mới không lớn hơn y cũa quá một đơn vị.
- $y_{i+1} = y_i + m$
- Như vậy chỉ cần xét giá trị cộng dồn cho *y* khi tổng giá trị cộng dồn vượt quá 1. Khi đó, thay đổi lại giá trị này cho hợp lý. Nghĩa là:
- **fraction += m;**
- **if (fraction >= 1) { y = y + 1; fraction -= 1; }**

Thuật giải Bresenham

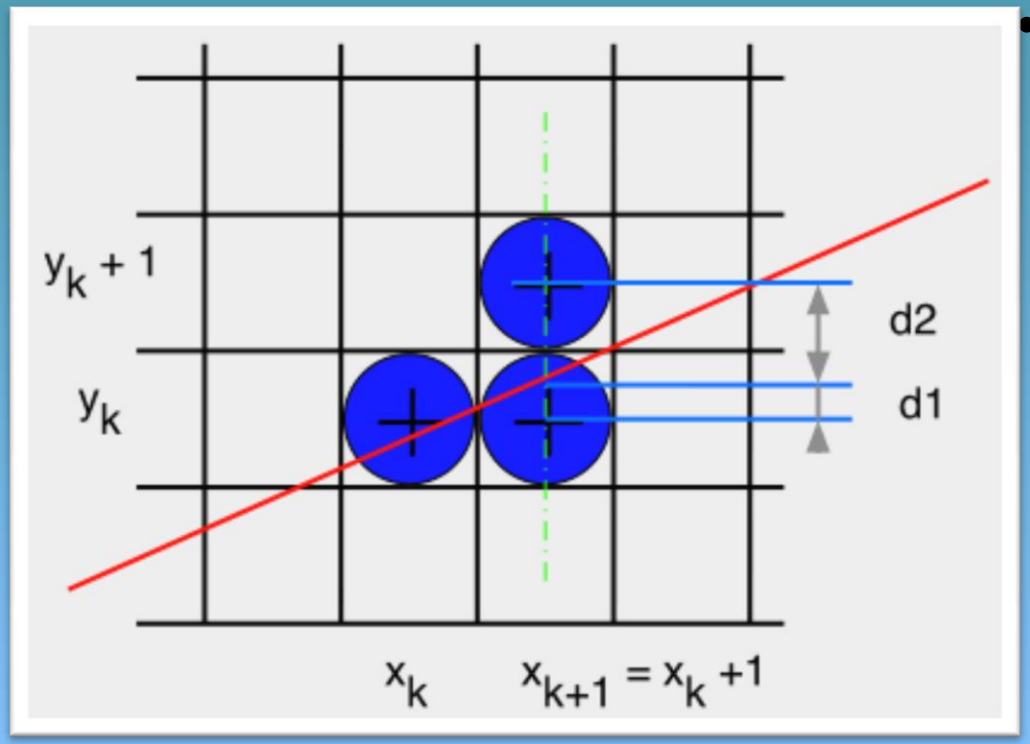
- Có thể dùng số nguyên cho thừa số cộng dồn => thuật giải chỉ dùng số nguyên.
- Sau khi vẽ pixel đầu tiên.
- fraction = 1/2 + dy/dx.
- Nhân với *2*dx:* scaledFraction = dx + 2*dy
- scaledFraction += 2*dy // 2*dx*(dy/dx)
- Biểu thức kiểm tra trở thành:
- if (scaledFraction >= 2*dx) { ... }

Thuật giải Bresenham

- Nhằm so sánh với giá trị zero (tự nhiên hơ) Nên đặt: **OffsetScaledFraction = dx + 2*dy - 2*dx = 2*dy – dx**.
-

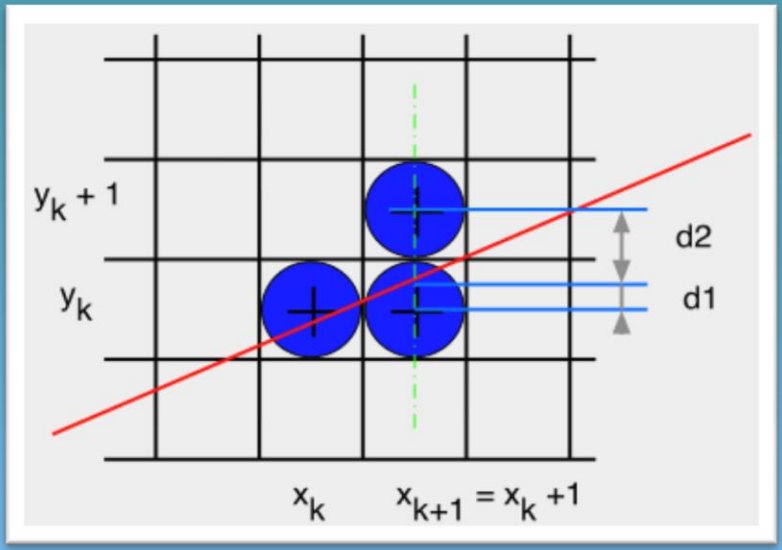
```
OffsetScaledFraction += 2*dy
if (OffsetScaledFraction >= 0) {
        y = y + 1;
        fraction - = 2*dx;
}
```

Thuật giải Bresenham



- **Decision : we'll study the sign of a integer parameter whose value is proportional to the difference between the separations of the two pixel positions from the actual line path.**

Thuật giải Bresenham



- **step 0**
- **from k to k+1 : choice (xk + 1, yk) or (xk + 1, yk + 1)**

y = m (xk + 1) + b

d1 = y - yk = m (xk + 1) + b - yk

d2 = (yk + 1) - y = yk + 1 - m (xk + 1) -b

what we want to know : which of d1 and d2 is smaller, what we'll study : the sign of d1 - d2

d1 - d2 = 2 m (xk + 1) - 2 yk + 2b -1

Decision parameter: pk=△x(d1- d2)

Thuật giải Bresenham

- 1.Input the two line endpoints and store the left endpoint in (x_0,y_0)
- 2. Load (x_0,y_0)into the frame buffer , that is plot the first point .
- 3.Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y-2\Delta x$, and obtain the value for the decision parameter as:

  $p_0 = 2\Delta y- \Delta x$
- 4. At each x_k along the line, starting at k=0, perform the following test:

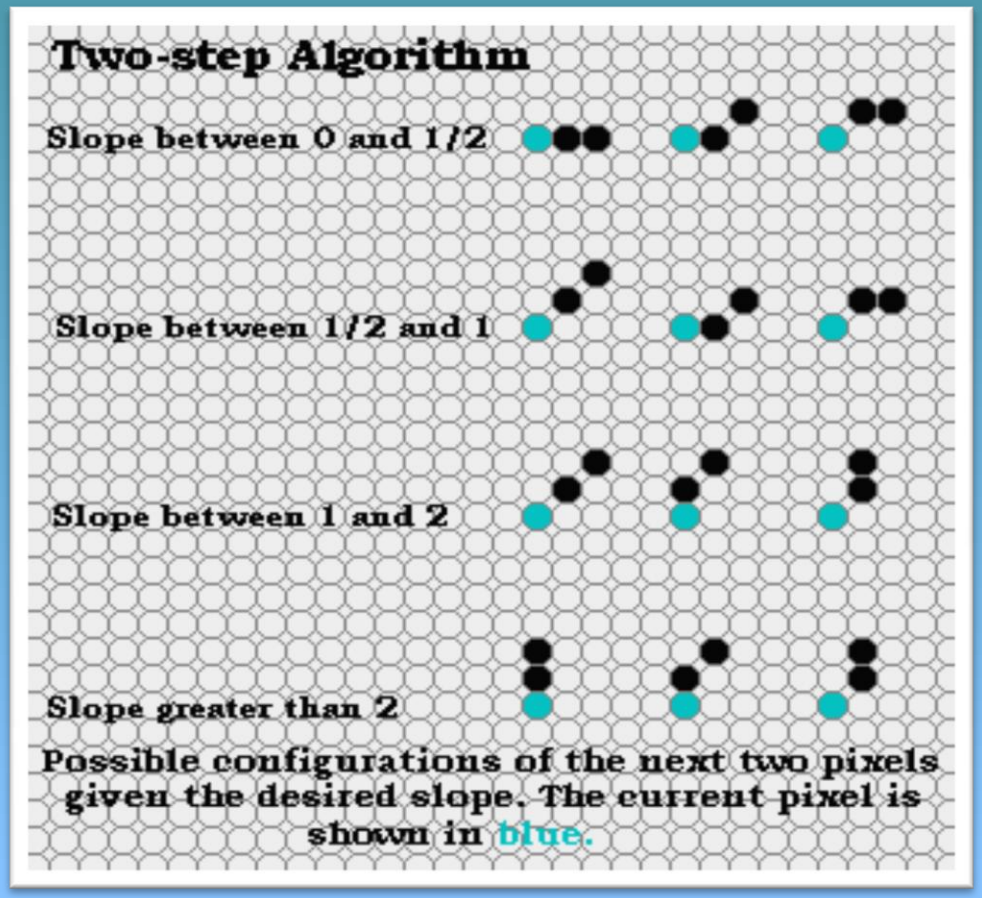
  If $p_k<0$, the next point to plot is (x_k+1,y_k) and

  $p_{k+1} = p_k +2\Delta y$

  Otherwise, the next point to plot is (x_k+1,y_k+1) and

  $p_{k+1} = p_k +2\Delta y - 2\Delta x$
- 5. Repeat step 4 Δx times

Thuật giải Bresenham



The two-step algorithm takes the interesting approach of treating line drawing as a automaton, or finite state machine. If one looks at the possible configurations that the next two pixels of a line, it is easy to see that only a finite set of possibilities exist.

The two-step algorithm also exploits the symmetry of line-drawing by simultaneously drawn from both ends towards the midpoint.

Thuật giải Bresenham

- Vẽ đoạn (2,3) → (12,8).
- Xác định *p0*, *dx* và *dy*.
- Xác định *p* ở mỗi bước lặp.
- Xác định tọa độ điểm ở mỗi bước lặp theo thuật giải Bresenham.

dx = 12 – 2 = 10
dy = 8 – 3 = 5
p0 = 2dy – dx = 0

2dy = 10
2dy – 2dx = -10

k	p	P(x)	P(y)
0	0	2	3
1	-10	3	4
2	0	4	4
3	-10	5	5
4	0	6	5
5	-10	7	6
6	0	8	6
7	-10	9	7
8	0	10	7
9	-10	11	8
10	0	12	8

Bài tập

1. Sửa thuật giải ra sao nếu hai điểm đầu cuối không phải số nguyên. (thường dùng trong 3D).
2. Vẽ đường có độ dày lớn hơn 1 (0.5đ - điểm thực hành).
3. **Làm tại lớp**: hãy xác định các giá trị P_i và toạ độ 06 điểm đầu tiên khi vẽ đường thẳng theo thuật giải Bresenham xác định bởi hai điểm đầu mút sau.
   - Điểm đầu: (3, 12).
   - Điểm cuối: (25, 19).

Thuật giải vẽ đường tròn

- Xét:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

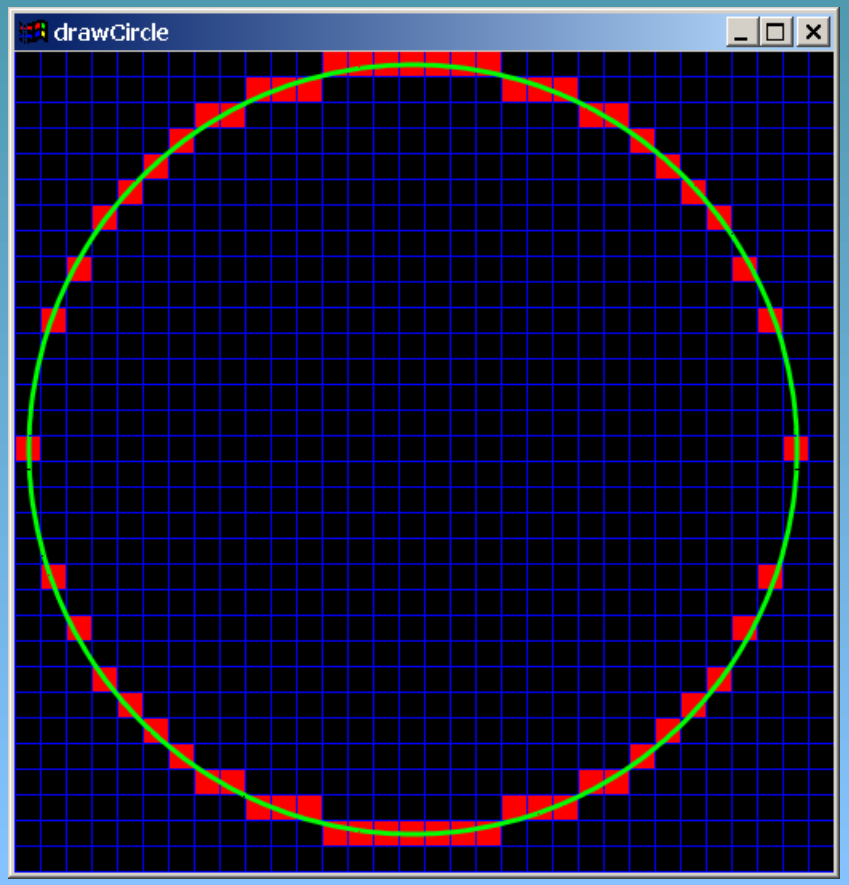
$$y = y_0 \pm \sqrt{r^2 - (x - x_0)^2}$$

```
void circleSimple(int xCenter, int yCenter, int
radius, Color c) {
   int x, y, r2;

   r2 = radius * radius;
   for (x = -radius; x <= radius; x++) {
      y = (int)(sqrt(r2 - x*x) + 0.5);
      setPixel(xCenter + x, yCenter + y, c);
      setPixel(xCenter + x, yCenter - y, c);
   }
}
```

Thuật giải vẽ đường tròn

- Vấn đề: nhiều vị trí trên đường tròn có độ dốc của đường tiếp tuyến lớn hơn 1. Vì vậy, không nên lặp theo x.

- Lặp theo y có được không?
- Tận dụng tính đối xứng của đường tròn.



Thuật giải vẽ đường tròn

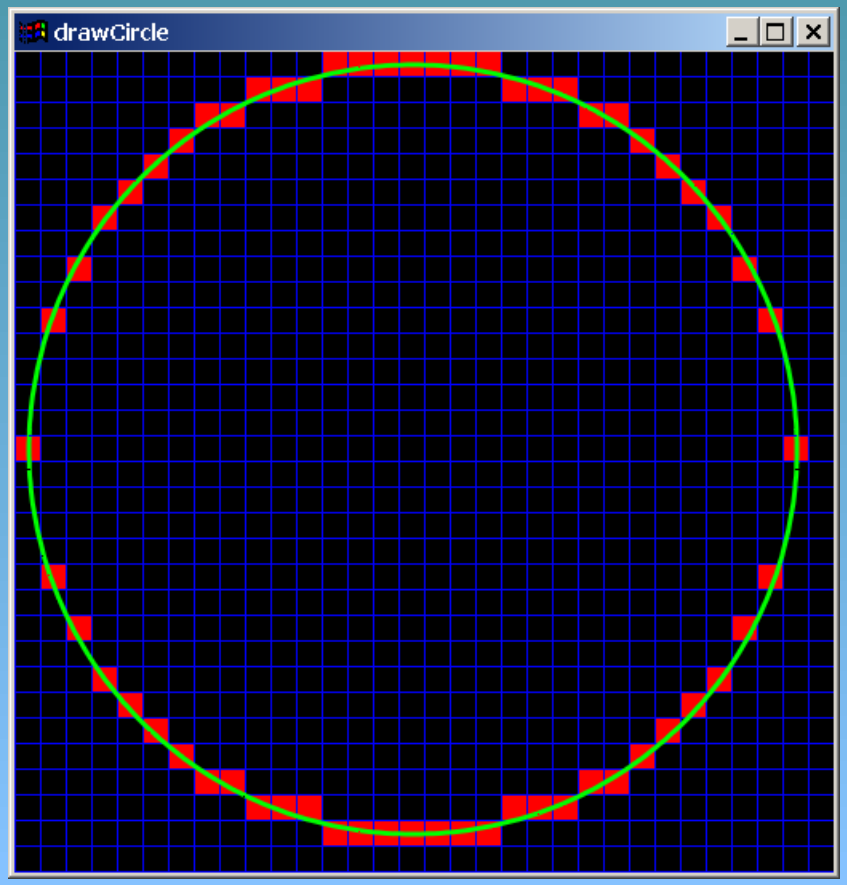
Sử dụng tính đối xứng của 4 góc ¼.

```
void circle4Way(int xCenter, int yCenter, int radius, Color c) {
   int x, y, r2;

   setPixel(xCenter, yCenter + radius, c);
   setPixel(xCenter, yCenter - radius, c);

   r2 = radius * radius;
   for (x = 1; x <= radius; x++) {
      y = (int)(sqrt(r2 - x*x) + 0.5);
      setPixel(xCenter + x, yCenter + y, c);
      setPixel(xCenter + x, yCenter - y, c);
      setPixel(xCenter - x, yCenter + y, c);
      setPixel(xCenter - x, yCenter - y, c);
   }
}
```

Thuật giải vẽ đường tròn

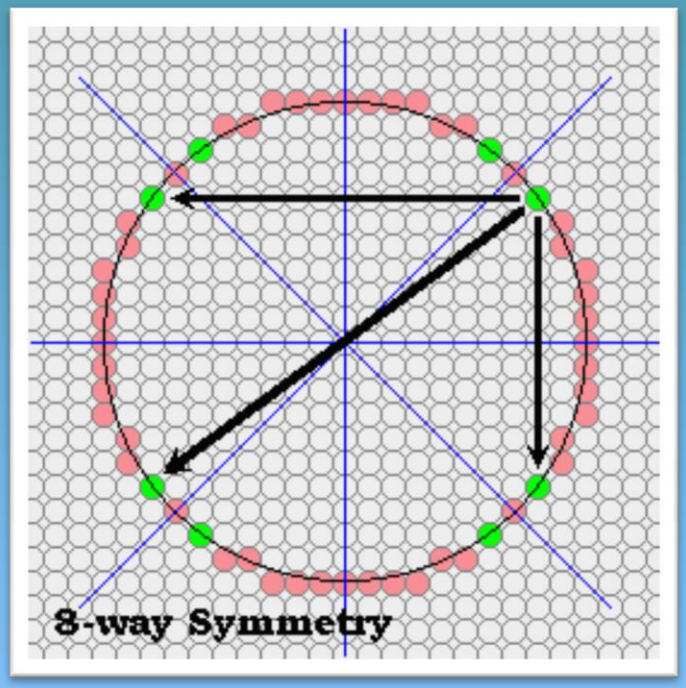
- Nhanh hơn, nhưng vẫn chưa đúng.

- Sử dụng 8 phần đối xứng?



Thuật giải vẽ đường tròn

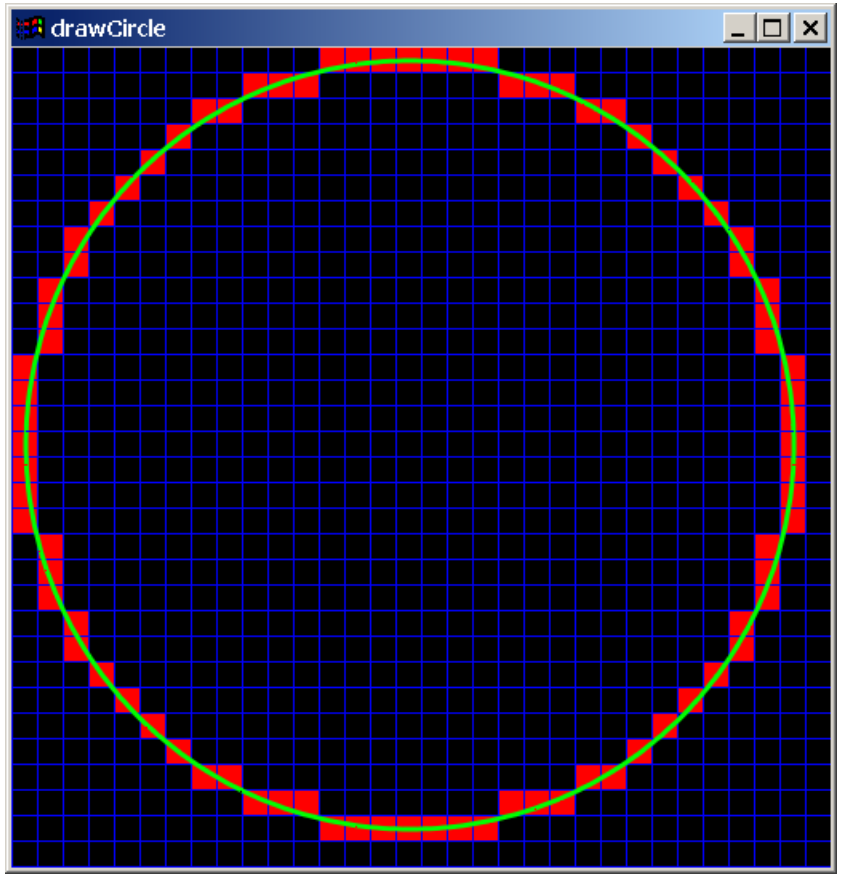
- Đối xứng qua đường thẳng *x*=*y*.
- Lặp theo *x*, hoán vị các tọa độ (đổi x và y). Đồng thời lặp theo *y* ở những phần khác



- Nhanh hơn. Kết quả tốt hơn.

Thuật giải vẽ đường tròn

```
void circle8Way(int xCenter, int yCenter, int radius, Color c) {
   int x, y, r2;

   setPixel(xCenter, yCenter + radius, c);
   setPixel(xCenter, yCenter - radius, c);
   setPixel(xCenter + radius, yCenter, c);
   setPixel(xCenter - radius, yCenter, c);

   r2 = radius * radius;
   x = 1;
   y = (int)(sqrt(r2 - 1) + 0.5);
   while (x < y) {
      setPixel(xCenter + x, yCenter + y, c);
      setPixel(xCenter + x, yCenter - y, c);
      setPixel(xCenter - x, yCenter + y, c);
      setPixel(xCenter - x, yCenter - y, c);
      setPixel(xCenter + y, yCenter + x, c);
      setPixel(xCenter + y, yCenter - x, c);
      setPixel(xCenter - y, yCenter + x, c);
      setPixel(xCenter - y, yCenter - x, c);

      x += 1;
      y = (int)(sqrt(r2 - x*x) + 0.5);
   }
   if (x == y) {
      setPixel(xCenter + x, yCenter + y, c);
      setPixel(xCenter + x, yCenter - y, c);
      setPixel(xCenter - x, yCenter + y, c);
      setPixel(xCenter - x, yCenter - y, c);
   }
}
```



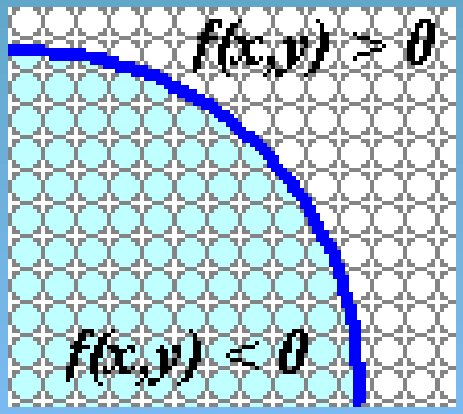
Thuật giải vẽ đường tròn

- Vấn đề 1: vẫn còn tính toán căn bậc 2 trong biểu thức.
- Vấn đề 2: chưa tận dụng kết quả của bước lặp trước.
- Giải pháp: suy nghĩ và tận dụng phương pháp vẽ đường thẳng theo thuật giải Bresenham nhưng áp dụng cho đường cong.

Thuật giải Midpoint

- Đặt: $f(x,y) = x^2 + y^2 - r^2$

- Một số tính chất:
  - $f(x,y) < 0$: điểm nằm trong đường tròn.
  - $f(x,y) > 0$: điểm nằm ngoài đường tròn
  - $f(x,y) = 0$: điểm nằm trên đường tròn



Thuật giải Midpoint

- Nếu đang ở điểm (x, y) trên đường tròn → cần chọn một trong hai điểm sau $(x+1, y)$ hoặc $(x+1, y-1)$ để vẽ cho điểm kế tiếp.
- Nếu ở trong đường tròn → chọn $(x+1, y)$.
- Nếu ở ngoài đường tròn → chọn $(x+1, y-1)$.

Thuật giải Midpoint

Nếu điểm hiện tại nằm trong vòng tròn: f(x,y) < 0. Đặt là *p*.

Điểm vẽ sẽ là $f(x+1, y)$, cập nhật p:

$$f(x+1, y) = (x + 1)^2 + y^2 - r^2$$

$$f(x+1, y) = (x^2 + 2x + 1) + y^2 - r^2$$

$$f(x+1, y) = f(x, y) + 2x + 1$$

Vậy: $p += 2x + 1$

Thuật giải Midpoint

Nếu điểm hiện tại nằm ngoài vòng tròn: f(x,y) > 0. Đặt là *p*.

Điểm vẽ sẽ là $f(x+1, y-1)$, cập nhật p:

$$f(x+1, y-1) = (x + 1)^2 + (y - 1)^2 - r^2$$

$$f(x+1, y-1) = (x^2 + 2x + 1) + (y^2 - 2y + 2) - r^2$$

$$f(x+1, y-1) = f(x, y) + 2x - 2y + 2$$

Vậy: $p \mathrel{+}= 2x - 2y + 2$

Thuật giải Midpoint

- Điểm vẽ bắt đầu là $(0, r)$, lặp theo x tăng dần theo cung 1/8 thứ hai → cần xác định giá trị *p* khởi đầu.
- Nhận xét: điểm đầu tiên trên đường tròn, dẫn đến điểm kế tiếp sẽ là $(x+1, y)$ (tại sao?)
- Tìm giá trị p_0.

$$p_0 = f(1, r-0.5) = 1^2 + (r - 0.5)^2 - r^2$$

$$p_0 = f(1, r-0.5) = 1 + (r^2 - r + 0.25) - r^2$$

$$p_0 = 1.25 - r$$

Thuật giải Midpoint

```
void circleMidpoint(int xCenter, int yCenter, int radius, Color c) {
   int x = 0;
   int y = radius;
   int p = (5 - radius*4)/4;

   circlePoints(xCenter, yCenter, x, y, c);
   while (x < y) {
      x++;
      if (p < 0) {
         p += 2*x+1;
      } else {
         p += 2*(x-y+1);
         y--;
      }
      circlePoints(xCenter, yCenter, x, y, c);
   }
}
```

Thuật giải Midpoint

```
void circlePoints(int cx, int cy, int x, int y, Color c) {
   if (x == 0) {
      setPixel(cx, cy + y, c);
      setPixel(cx, cy - y, c);
      setPixel(cx + y, cy, c);
      setPixel(cx - y, cy, c);
   } else if (x == y) {
      setPixel(cx + x, cy + y, c);
      setPixel(cx - x, cy + y, c);
      setPixel(cx + x, cy - y, c);
      setPixel(cx - x, cy - y, c);
   } else if (x < y) {
      setPixel(cx + x, cy + y, c);
      setPixel(cx - x, cy + y, c);
      setPixel(cx + x, cy - y, c);
      setPixel(cx - x, cy - y, c);
      setPixel(cx + y, cy + x, c);
      setPixel(cx - y, cy + x, c);
      setPixel(cx + y, cy - x, c);
      setPixel(cx - y, cy - x, c);
   }
}
```

Bài tập

1. **Thực hiện tại lớp**: hãy xác định các giá trị P_i và toạ độ 07 điểm đầu tiên của góc 1/8 (góc bắt đầu từ 90 độ xuống dần đến 45 độ) khi vẽ đường tròn theo thuật giải MidPoint với các tham số.
   - Tâm đường tròn: **(3, 9)**.
   - Bán kính: **9**
   - Điểm bắt đầu vẽ: **(3, 18)**.
2. Làm bài tập trong các trang:

Bài tập thực hành

- Cài đặt các chương trình vẽ đường thẳng, đường tròn (ellipse, conic) theo thuật giải Bresenham hay MidPoint.

Đọc thêm & Hỏi đáp

- Các thuật giải MidPoint cho các đường Conic.