TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
MÔN: **ĐỒ HỌA MÁY TÍNH**
LỚP: 16CNTN

LAB 01
THUẬT TOÁN VẼ ĐƯỜNG

LÊ THÀNH CÔNG
MSSV: 1612842

TP.HCM, ngày 13 tháng 10 năm 2018

Mục lục

I. Nội dung

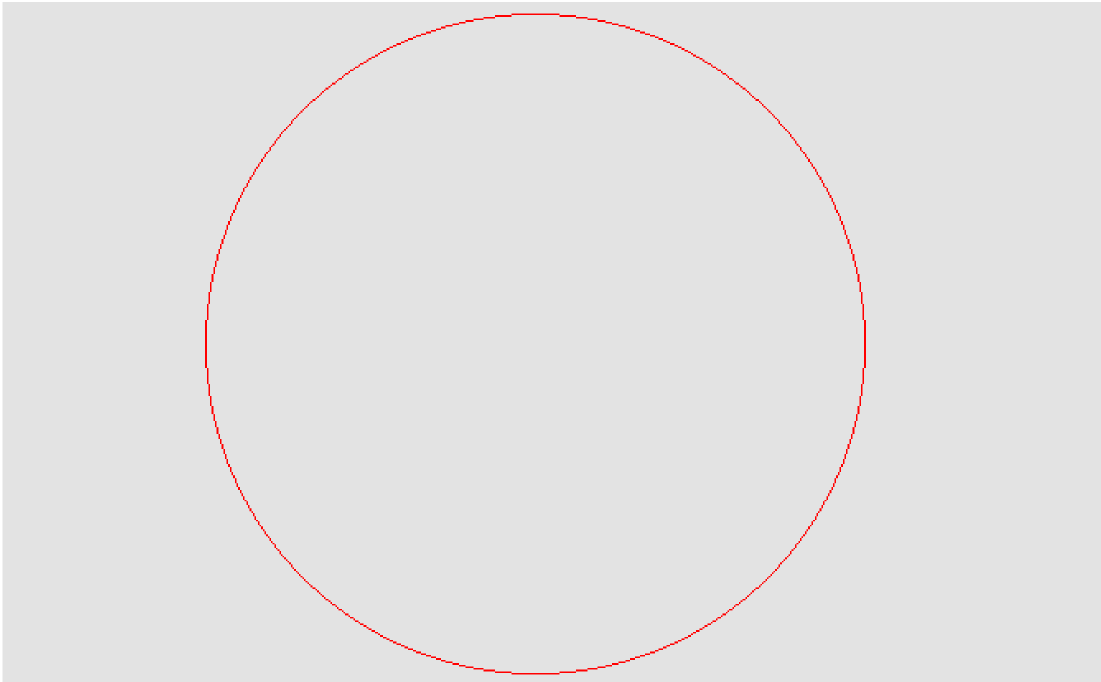
1. Giải thích các thuật toán

a. *Thuật toán DDA*

Vẽ đường thẳng

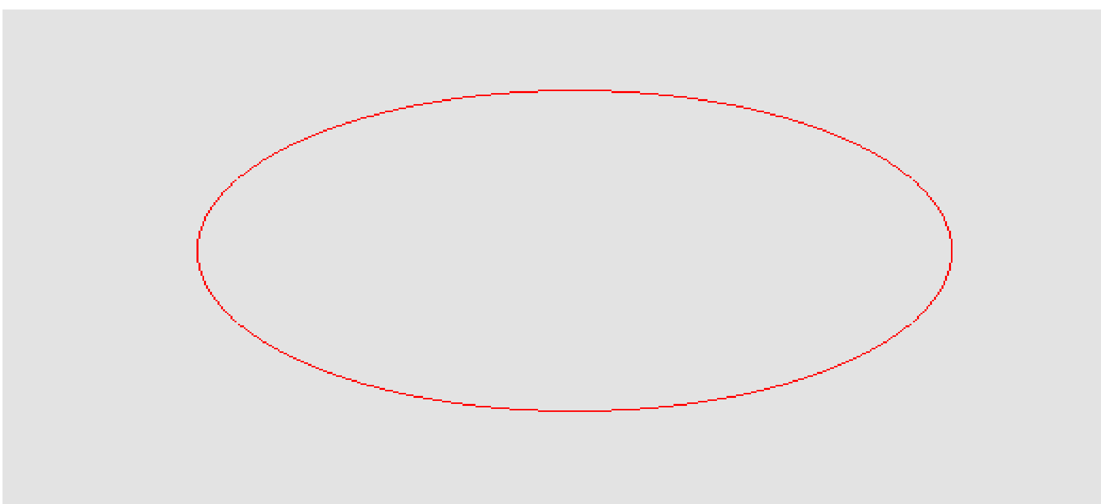
- DDA (Digital Differential Analyzer) về cơ bản là thuật toán vẽ đường thẳng theo cách làm tròn tọa độ các điểm mà đường thẳng đó đi qua.
- Xét 2 điểm (x_0,y_0) là điểm đầu và (x_1,y_1) là điểm cuối của đường thẳng cần vẽ
- Thuật toán thực hiện tính dx (độ lệch giữa x_0 và x_1), dy (độ lệch giữa y_0 và y_1) sau đó so sánh độ lớn của dx và dy. Mục đích của việc này là xác định khoảng cách nào lớn hơn để dùng đó vẽ được nhiều điểm hơn làm đường thẳng đẹp hơn.
- Steps là số vòng lặp sẽ thực hiện để vẽ, càng lớn càng tốt nên steps=max(dx,dy)
- Lấy dx/steps và dy/steps xác định độ tăng mỗi vòng lặp. Ta luôn đảm bảo nếu dx > dy thì mỗi vòng lặp x luôn tăng 1 đơn vị, y tăng một lượng (0..1).
- Sau đó chỉ cần làm tròn về số nguyên để setPixel.

Vẽ đường tròn



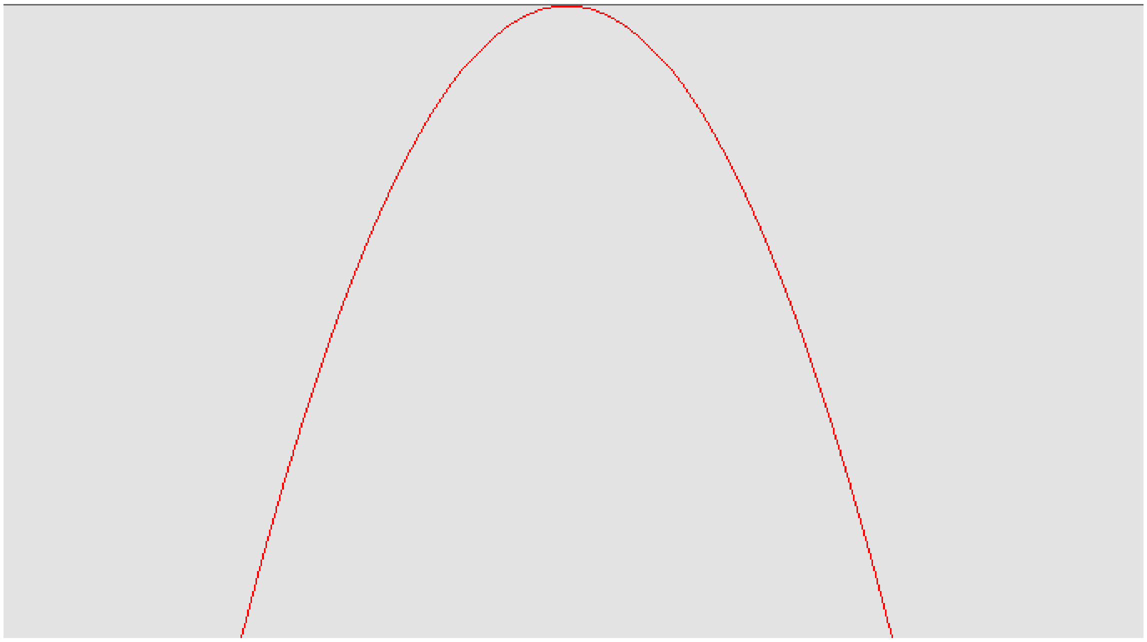
- Công thức đường tròn tại tâm $(0,0)$ bán kính R là $x^2 + y^2 = R^2$
- Ta dùng vòng lặp vẽ 1/8 cung tròn sau đó lấy các điểm đối xứng qua Ox, Oy, y=x, y=-x
- Mỗi lần trong vòng lặp ta tính y theo x: $y = \sqrt{R^2 - x^2}$
- Sau đó làm tròn y và set8pixel (vẽ 8 điểm pixel đối xứng qua Ox, Oy, y=x và y = -x).

Vẽ ellipse



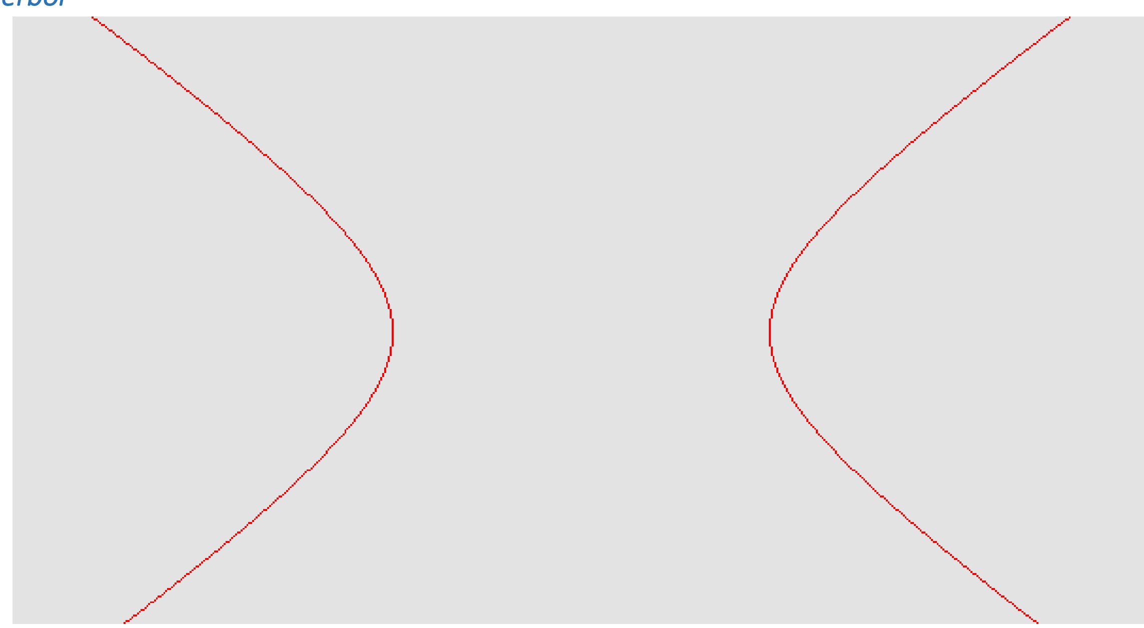
- Ta có công thức ellipse bán trục lớn a, bán trục nhỏ b, tâm tại $(0,0)$ là $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
- Ta tìm x và y sao cho y' = - 1, ta được $x = \frac{a^2}{\sqrt{a^2+b^2}}$ và $y = \frac{b^2}{\sqrt{a^2+b^2}}$
- Dùng vòng lặp vẽ ¼ ellipse, với $0 \le x \le \frac{a^2}{\sqrt{a^2+b^2}}$. Ta tính y theo x, sau đó làm tròn y và set4pixel
- Với $0 \le y \le \frac{b^2}{\sqrt{a^2+b^2}}$ ta tính x theo y, làm tròn x và set4pixel (vẽ 4 pixel đối xứng qua Ox, Oy)

Vẽ parabol



- Công thức parabol tại $O(0,0)$ có hệ số a là $y = ax^2$
- Tìm x và y sao cho y' = 1, ta được $x = \frac{1}{2a}$ và $y = \frac{1}{4a}$
- Như vậy, dùng vòng lặp biến chạy là x với $0 \leq x \leq \frac{1}{2a}$, tính y theo x rồi set2pixel, nếu a<0 lấy đối xứng qua Oy
- Tiếp theo dùng vòng lặp biến chạy là y với $\frac{1}{4a} \leq y \leq height$ với height là chiều cao khung vẽ để vẽ hết khung, tính x theo y rồi set2pixel, nếu a<0 lấy đối xứng Oy

Vẽ hyperbol



- Công thức hyperbol trục thực a và trục ảo b là $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$ nếu a>b và $\frac{x^2}{a^2} - \frac{y^2}{b^2} = -1$ nếu a<b

- Tương tự, ta tìm x và y để y' = 1, nếu a>b thì $x = \frac{a^2}{\sqrt{a^2-b^2}}$ và $y = \frac{b^2}{\sqrt{a^2-b^2}}$, nếu a<b thì $x = \frac{a^2}{\sqrt{b^2-a^2}}$ và $x = \frac{b^2}{\sqrt{b^2-a^2}}$
- Dùng vòng lặp y chạy từ 0 cho tới điểm y vừa tìm được, tính x theo y, làm tròn x và set4pixel
- Sau đó lặp x chạy từ x vừa tìm được cho đến x = width, tức cho x chạy hết vùng hệ số góc tăng chậm hết hết màn hình, tính y theo x, làm tròn y và set4pixel.

b. Thuật toán Bresenham

Vẽ đường thẳng

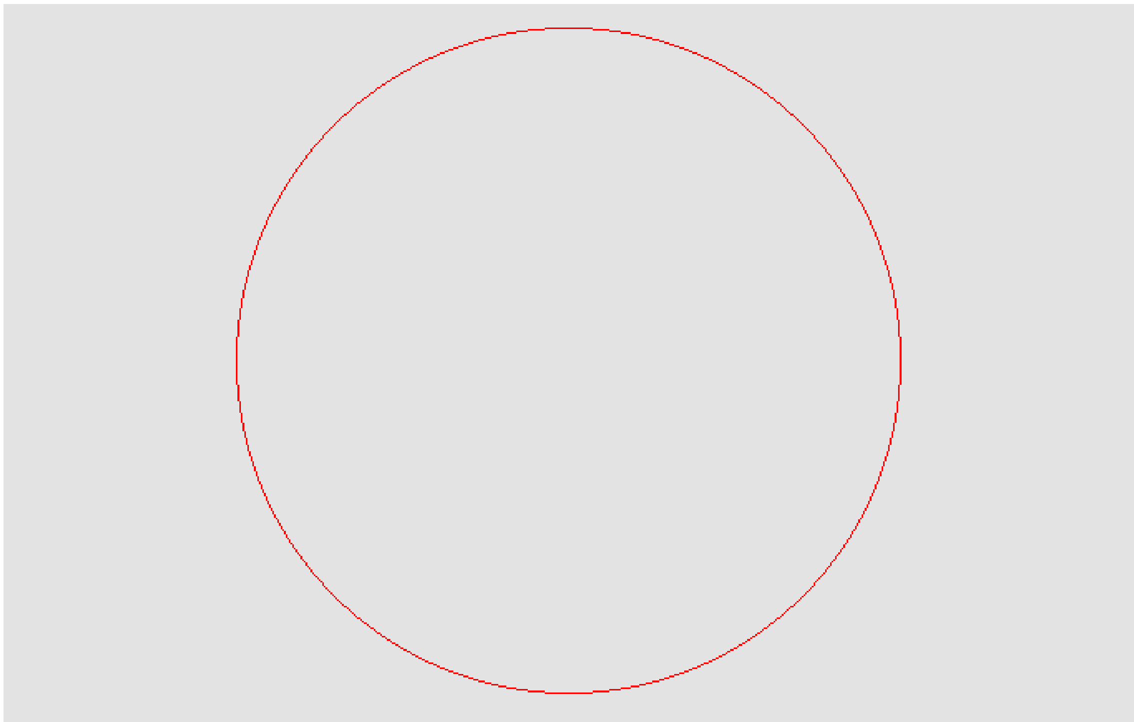
- Thuật toán Bresenham thay thế các phép toán số thực bằng phép toán số nguyên
- Hạn chế các phép toán thực hiện để giảm tải thời gian
- Thuật toán trên được viết để vẽ đường thẳng trong trường hợp tổng quát
- So sánh khoảng cách giữa điểm thực y với 2 pixel gần kề nó nhất
- Gọi P_i là biến để xét hiệu khoảng cách từ điểm y_i và y_{i+1} đến điểm y_{i+1} thực sự.
  Đặt $P_i = dx((y - y_i) - (y_i + 1 - y))$, ta dễ dàng chứng minh được:
  - $P_{i+1} = P_i + 2dy - 2dx(y_{i+1} - y_i)$, với $dy = |y_{end} - y_{start}|$, $dx = |x_{end} - x_{start}|$
  - Nếu $P_i < 0 \Rightarrow y_{i+1} = y_i \Rightarrow P_{i+1} = P_i + 2dy$
  - Nếu $P_i > 0 \Rightarrow y_{i+1} = y_i + 1 \Rightarrow P_{i+1} = P_i + 2(dy - dx)$
  - $P_0 = 2dy - dx$, P_0 là khoảng cách từ điểm y_0 đến điểm y_1 thực
- Mục đích của biến steep để kiểm tra dy > dx tức là dy/dx = m > 1, khi đó hệ số góc > 1 nên ta hoán đổi vai trò của x và y, để có thể vẽ đường thẳng trong trường hợp đó
- Biến xInc và yInc có ý nghĩa như biến chạy của x và y. Tuy nhiên, hướng chạy tăng hay giảm phụ thuộc vào x_1 có lớn hơn x_0 hay không? Nếu x_1 nhỏ hơn x_0 thì biến x khi vẽ pixel cần chạy lùi phải sang trái (tức là trừ 1) để vẽ được đường thẳng, nếu $x_1 > x_0$ thì ngược lại.

c. Thuật toán Mid point

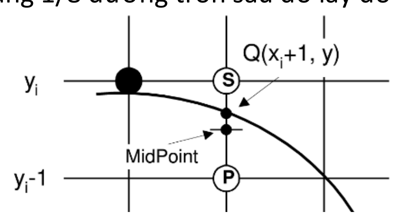
Vẽ đường thẳng

- Thuật toán Mid point cho kết quả hoàn toàn giống với Bresenham nhưng cách xây dựng đơn giản hơn
- Các biến steep, xInc và yInc có tác dụng tương tự ở phần vẽ đường thẳng bằng Bresenham
- Ở đây, ta có dạng tổng quát của phương trình đường thẳng:
    - $Ax + By + C = 0$
    - $A = y_1 - y_0$
    - $B = -(x_1 - x_0)$
    - $C = x_1y_0 - x_0y_1$
- Ta xét điểm Midpoint là trung điểm của 2 pixel kề nhau y_i và y_{i+1}. Vị trí tương đối của Midpoint với đường thẳng:
    - $F(x,y) = Ax + By + C$
    - Nếu $F(x,y) < 0$ thì (x,y) nằm phía trên đường thẳng
    - Nếu $F(x,y) = 0$ thì (x,y) thuộc về đường thẳng
    - Nếu $F(x,y) > 0$ thì (x,y) nằm phía dưới đường thẳng
- Giả sử ta đã cho biến x chạy lần lượt các pixel hoành độ x nguyên từ x_0 đến x_1, việc còn lại là xét và chọn các điểm y thích hợp. Tại mỗi bước nên chọn y_i hay y_{i+1} để vẽ. Điều này phụ thuộc vào việc xét dấu $F(x,y)$
- Ta thay Midpoint(x_i+1, $y_{i+½}$) vào $F(x,y)$ để xét dấu. Để đơn giản được ½ ta nhân thêm cho 2. Ta được:
    - $p = 2(Ax_i + By_i + C) + 2A + B$
    - $p >= 0$ tức là điểm Midpoint nằm phía dưới đường thẳng thực, khi đó điểm y thực lại gần với y_{i+1}. Vậy nên ta chỉ cần cho y+= yInc
    - $p<0$ thì ngược lại nên y gần với y_i do đó không cần tăng y.

Vẽ đường tròn

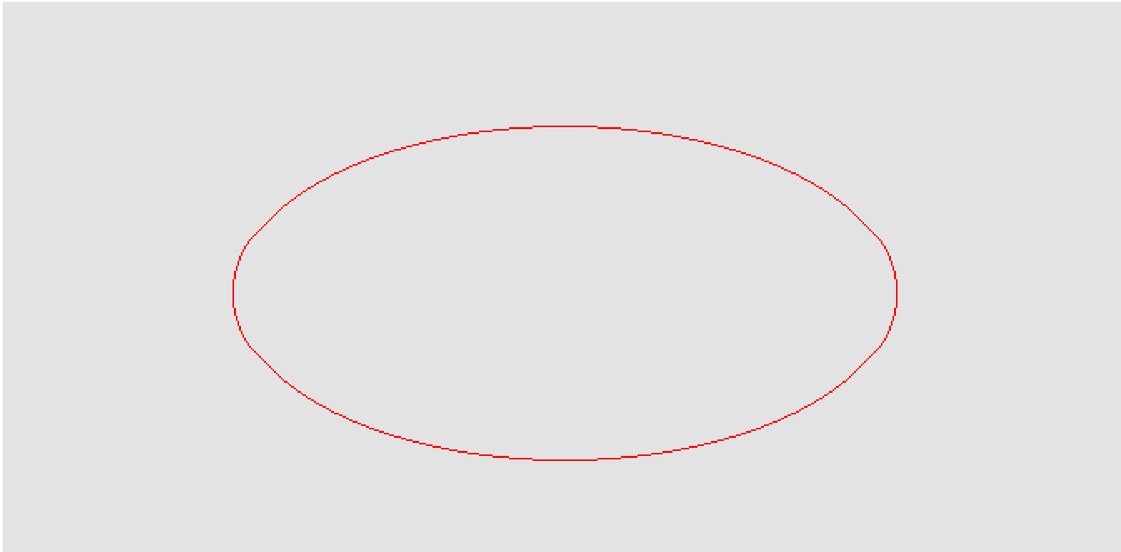


- Ở hàm set8Pixel thực hiện vẽ 8 điểm tương ứng với 8 cung trên đường tròn, đối xứng qua Ox, Oy, y=x, y=-x. Tuy nhiên, có tâm (x_0, y_0) nên cần thực hiện các phép tịnh tiến, đối xứng bằng cách sửa lại tọa độ truyền vào khi setPixel tương ứng với (x_0, y_0).
- Ý tưởng để vẽ đường tròn: vẽ cung 1/8 đường tròn sau đó lấy đối xứng qua Ox, Oy, y=x và y=-x.
- Ta vẽ cung tròn trong đoạn:
  - $0 \leq x \leq \frac{R\sqrt{2}}{2}$
  - $\frac{R\sqrt{2}}{2} \leq y \leq R$



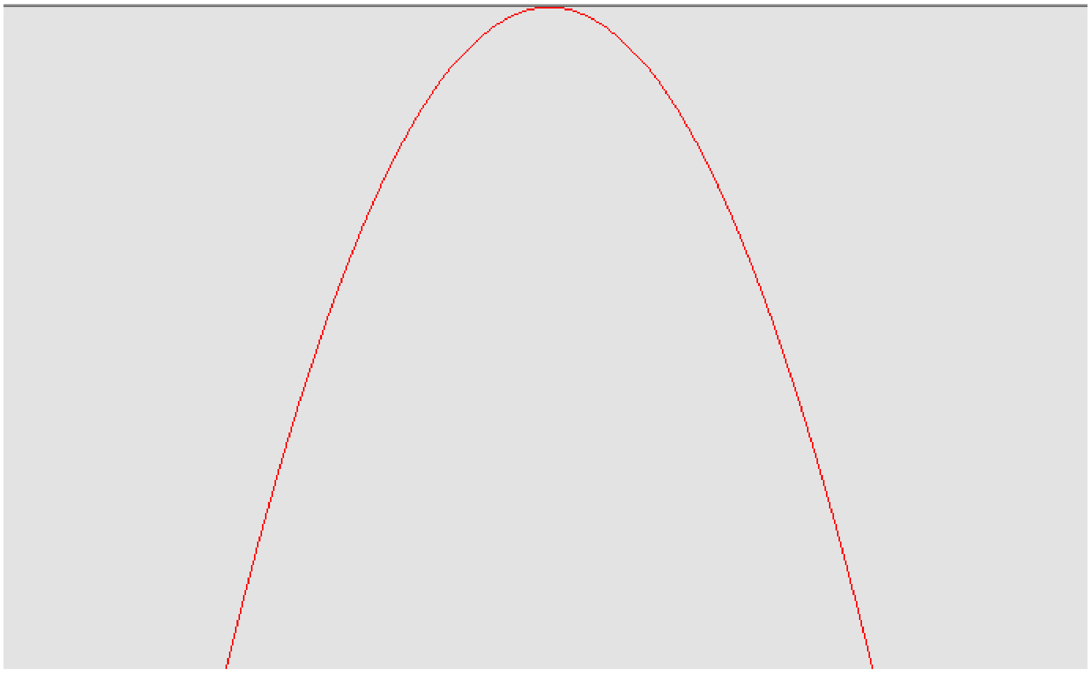
- Nếu x_i và y_i nguyên đã tìm được ở bước i thì bước i+1 ta cần chọn lựa giữa y_i hoặc $y_i - 1$
- Xét P_i= F(Midpoint) = F(x_i+1, y_i – ½). Ta chứng minh được:
  - Nếu $P_i < 0$ thì $P_{i+1} = p_i + 2x_i + 3$, Midpoint trong đường tròn, ta chọn $y_{i+1}=y_i$
  - Nếu $P_i > 0$ thì $P_{i+1} = p_i + 2x_i – 2y_i +5$, Midpoint nằm ngoài đường tròn, ta chọn $y_{i+1}=y_i – 1$
  - P_0 ứng với điểm đầu (0,R) nên P_0 = F(0+1, R – ½) thay vào phương trình đường tròn ta được $P_0 = 5/4 – R \approx 1 - R$
- Như vậy, chúng ta chỉ cần cài đặt theo công thức ở trên.

Vẽ ellipse



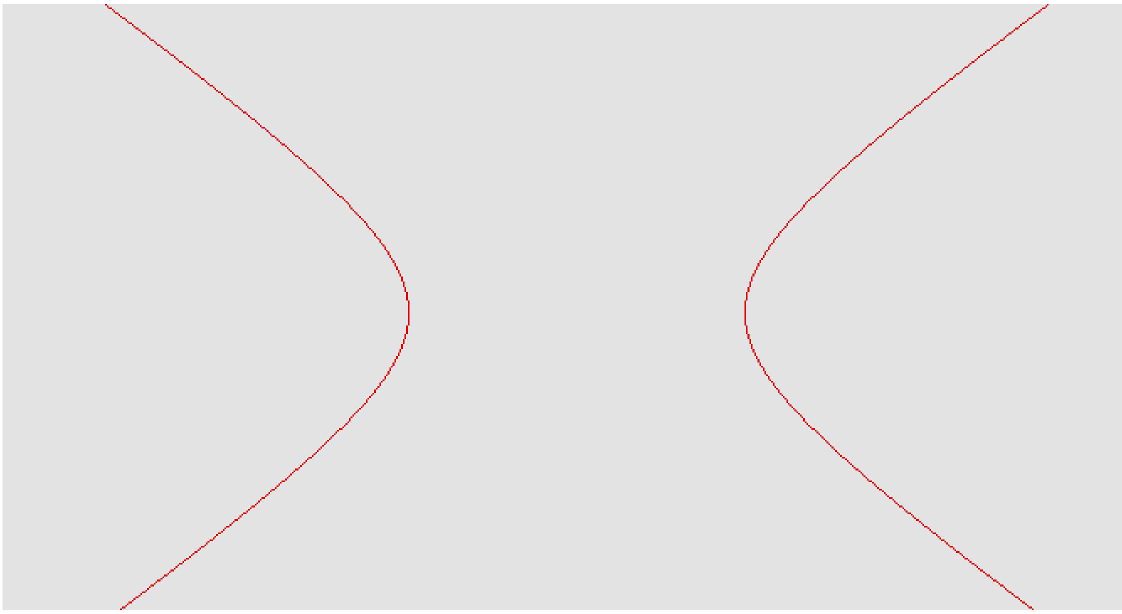
- Ta xét phương trình Elip có tâm ở gốc tọa độ: F(x,y) = $b^2x^2 + a^2y^2 - a^2b^2$
- Tương tự đường tròn, nhưng lần này ta vẽ $\frac{1}{4}$ Elip sau đó lấy đối xứng qua trục Ox và Oy.
- Để vẽ nhánh $\frac{1}{4}$ Elip này ta cần chia nhánh làm 2 phần tại nơi có vector gradient bằng 1 (nơi hệ số góc của tiếp tuyến bằng -1)
- Giải sử $fx < fy \leftrightarrow 2b^2x < 2a^2y$ => Với điều kiện này thì nếu chọn x để vẽ nhánh Elip sẽ được nhiều điểm hơn y nên ta chọn x làm biến chạy (x++). Mỗi lần lặp lưu $fx = fx + 2b^2$ nếu chạy x lần thì $fx = 2b^2x$
- Ta đi xây dựng công thức vẽ Elip trong trường hợp fx < fy:
    - Tại mỗi bước trên nhánh này ta cần lựa chọn y_i hay y_{i-1}
    - Lấy midpoint giữa 2 điểm y này thay vào phương trình Elip để xác định điểm nào được chọn. Cách chọn điểm tương tự phần đường tròn
    - Xét $p_i = f(x_i + 1, y_i - \frac{1}{2})$, ta chứng minh được:
        - $p_i < 0 \rightarrow p_{i+1} = p_i + b^2(2x_i + 3)$
        - $p_i \geq 0 \rightarrow p_{i+1} = p_i + b^2(2x_i + 3) + a^2(2 - 2y)$
        - $p_1 = f(0, b) = b^2 - a^2b + \frac{a^2}{4}$
- Khởi tạo fy ban đầu là $fy = 2a^2y$ vì đi từ điểm (0,b). Sau mỗi lần chọn được điểm $y_i = y_i - 1$ ta cho $fy = fy - 2a^2$
- Đối với nhánh còn lại, tức $fx > fy$. Ta thực hiện tương tự nhưng với x và y đổi vai trò. Lúc này y là biến chạy (y--) cho mỗi vòng lặp. Mục tiêu là chọn điểm $x_i \; hay \; x_{i+1}$
- Cấu trúc ở nhánh 2 này tương tự nhánh 1. Công thức tính p_i ở nhánh 2:
    - $p_i > 0 \rightarrow p_{i+1} = p_i + a^2(3 - 2y_i)$ chọn điểm x_i
    - $p_i < 0 \rightarrow p_{i+1} = p_i + b^2(2x_i + 2) + a^2(3 - 2y_i)$ chọn điểm x_i
    - $p_1 = f\left(x_k + \frac{1}{2}, y_k - 1\right)$ với (x_k, y_k) là vị trí cuối cùng ở phần nhánh 1
- Công thức được xây dựng bằng cách thay Midpoint vào p_i sau đó tìm mối liên hệ p_i và p_{i+1}. Nếu p_i>0 thì chọn được điểm x_{i+1}=x_i, nếu p_i<0 thì x_{i+1}=x_i+1 thay ngược vào biểu thức.

Vẽ parabol



- Xét parabol dạng $y = ax^2$ đỉnh tại O(0,0)
- Ý tưởng: đối với parabol ta vẽ $\frac{1}{2}$ ở nhánh $x \in [0, +\infty)$ sau đó lấy đối xứng qua trục Oy
- Tương tự như phần elip, ta tìm điểm chia nhánh làm 2 phần: phần tăng chậm (y'<1) và phần tăng nhanh (y'>1). Với $y = ax^2$ thì $y' = 2ax$
- Ta được 2 trường hợp:
    - y' < 1 tức 2ax < 1 :
        - $x_{i+1} = x_i + 1$
        - y_{i+1} nên chọn y_i hay $y_i + 1$
    - y' > 1 tức 2ax > 1 :
        - x_{i+1} nên chọn x_i hay $x_i + 1$
        - $y_{i+1} = y_i + 1$
- Với 2ax < 1, đặt $p_i = 2f(M_i) = 2(ax^2 - y)$ với M_i điểm Midpoint $(x_i + 1, y_i + \frac{1}{2})$. Ta chứng minh được:
        - Nếu $p_i < 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i$ và $p_{i+1} = p_i + a(2x_i + 3)$
        - Nếu $p_i > 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i + 1$ và $p_{i+1} = p_i + a(2x_i + 3) - 1$
        - $p_1 = 2f(0,0) = 2a - 1$
- Với 2ax > 1, đặt $p_i = 4f(M_i) = 4(ax^2 - y)$ với M_i điểm Midpoint $(x_i + \frac{1}{2}, y_i + 1)$. Ta chứng minh được:
        - Nếu $p_i < 0$ thì $x_{i+1} = x_i + 1, y_{i+1} = y_i + 1$ và $p_{i+1} = p_i + 2a(x_i + 1) - 1$
        - Nếu $p_i > 0$ thì $x_{i+1} = x_i, y_{i+1} = y_i + 1$ và $p_{i+1} = p_i - 1$
        - $p_1 = 4f\left(\frac{1}{2a}, \frac{1}{4a}\right) = a - 2$
- Ở đoạn while (y<height) để có thể vẽ đến hết độ cao của khung vẽ

Vẽ hyperbol



- Các tiếp cận vẽ Hyperbol tương tự như vẽ elip ở phần trên. Ta xét Hyperbol $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$
- Ta chọn nhánh :
  - $y = \frac{b}{a}\sqrt{x^2 - a^2}$
  - $x \in [a, +\infty]$
  - $F(x, y) = b^2x^2 - a^2y^2 - a^2b^2$
- Nếu vẽ được nhánh này ta chỉ cần lấy đối xứng qua Ox và Oy để hoàn thành Hyperbol
- Ban đầu gán x=a và y=0 vì tại nhánh đang xét có điểm đầu là $(a, 0)$
- Tìm y để y' = 1 ta được hệ số góc $\frac{b^2}{\sqrt{a^2-b^2}}$
- Trường hợp hệ số góc tăng nhanh, ta cho y_i chạy (y++) cho đến khi $y_i = \frac{b^2}{\sqrt{a^2-b^2}}$ thì dừng, tại mỗi bước xét lựa chọn giữa $x_i$hay $x_i + 1$. Gọi Midpoint $M(x_i + \frac{1}{2}, y_i + 1)$ và đặt $p_i = F(M_i)$ với F là phương trình Hyperbol. Tương tự Elip ta tìm được công thức
  - $p_{i+1} = p_i + (x_i + 1)2b^2 - (2y_i + 3)a^2$ nếu $p_i < 0$, ta chọn $x_i + 1$
  - $p_{i+1} = p_i - a^2(2y_i + 3)$ nếu $p_i > 0$, ta chọn x_i
- Trường hợp hệ số góc tăng chậm, ta đảo lại, tức x chạy (x++) cho tới width tức là tối đa độ rộng màn hình vẽ. Tại mỗi bước lại xét y_i hay $y_i + 1$. Gọi Midpoint $M(x_i + 1, y_i + \frac{1}{2})$
  - $p_{i+1} = p_i + (x_i + 1)2b^2$ nếu $p_i < 0$, ta chọn y_i
  - $p_{i+1} = (x_i + 1)2b^2 - 2a^2(y_i + 1)$ nếu $p_i > 0$, ta chọn $y_i + 1$
  - $p_1 = F\left(\frac{a^2}{\sqrt{a^2-b^2}}, \frac{b^2}{\sqrt{a^2-b^2}}\right) = b^2 - \frac{a^2}{4}$
- Như vậy, ta chỉ cần cài đặt thuật toán theo công thức ở trên.

d. Thuật toán Xiaolin Wu

Vẽ đường thẳng

- Đường thẳng vẽ bằng Xiaolin Wu cho kết quả đẹp hơn so với 3 thuật toán trên, độ mịn cao.
- Thuật toán Xiaolin Wu về cơ bản có chức năng khử răng cưa và kiểm soát được trường hợp cuối của đoạn thẳng không nằm trên một điểm có tọa độ nguyên. Nhìn chung, thuật toán này khử răng cưa có tốc độ khá nhanh nhưng không nhanh bằng Bresenham
- 2 điểm ở 2 đầu đoạn thẳng được kiểm soát riêng và vẽ các cặp điểm gần nhau hai bên đoạn thẳng và tô màu dựa trên độ ưu tiên khoảng cách đến đoạn thẳng.
- Phân tích thuật toán:
    - Nếu dy > dx thì steep =1, tức là hệ số góc m của đường thẳng : m> 1. Lúc này ta đổi vai trò của x và y bằng cách thực hiện swap(x0,y0) và swap(x1,y1).
    - Nếu x0 > x1 thì swap(x0,x1) và swap(y0,y1) bởi vì khi vẽ ta cho x chạy (x++) từ giá trị thấp lên cao nên cần đổi vai trò. Ta chủ yếu xét trường hợp điểm đầu nhỏ hơn điểm cuối nên nếu vi phạm thì đổi vai trò.
    - Biến gradient đại diện dy/dx tức hệ số góc của đường thẳng.
    - Lấy x0 đem làm tròn gán cho xEnd, ta tính được yEnd tương ứng theo công thức đường thẳng $y = y_0 + m(x - x_0)$, m là gradient. Đây thực chất là xử lí điểm pixel đầu đoạn thẳng. xGap lấy phần bù của phần thập phân (x0+0.5).
    - Lấy phần nguyên của x0 và y0 ta được xPixel1 và yPixel1 cần vẽ đầu đoạn thẳng.
    - Mỗi lần vẽ 2 pixel (x,y) và (x,y+1) kèm theo con số alpha coi như độ sáng của màu gốc. Ví dụ màu đỏ, nhưng với một thông số alpha khác nhau sẽ cho độ đậm nhạt khác nhau:

      ```
      Color color = Color.FromArgb(alpha, Color.Red);
      ```
    - Khi vẽ nếu steep = 1 thì hoán đổi vai trò của x và y do hệ số góc > 1.
    - Tiếp theo vẽ pixel cuối với (x1, y1) tương tự (x0,y0).

- Bước cuối vẽ các pixel còn lại giữa 2 đầu mút, thấy biến intery ở đây được sử dụng như biến tung độ y ở mỗi vòng lặp với độ tăng là hệ số góc giống như thuật toán DDA (intery += gradient). Tại mỗi bước truyền vào độ đậm nhạt tương ứng với phần thập phân của intery, số càng lớn độ đậm càng cao, nhỏ thì ngược lại.

2. So sánh các thuật toán

a. Thời gian

- Lần lượt thực hiện vẽ 1000, 5000 và 10000 đường thẳng ta được kết quả thời gian của các thuật toán như sau:

	1000	5000	10000
DDA	52568 (ms)	408707 (ms)	939252 (ms)
Bresenham	40242 (ms)	354370 (ms)	780760 (ms)
Mid point	35965 (ms)	272357 (ms)	751236 (ms)
Xiaolin Wu	158598 (ms)	510911 (ms)	1451311 (ms)

- Thuật toán Xiaolin Wu cho kết quả thời gian chạy lâu nhất bởi vì tính năng khử răng cưa và có tính toán số thực.
- Thuật toán Bresenham và Midpoint gần như tương đương nhưng Midpoint có nhỉnh hơn. Cả 2 thuật toán này nhanh hơn DDA rõ rệt bởi vì DDA có tính toán với số thực còn Bresenham và Midpoint hạn chế tối đa tính toán và tính toán trên số nguyên.

b. Độ chính xác

- Khi thuật toán vẽ đường thẳng thực hiện thay vì mỗi lần setPixel(x,y), ta lấy tọa độ pixel đó tính độ chênh lệch với điểm (x_r, y_r) là tọa độ điểm thực được tính khi thay vào phương trình đường thẳng. Nếu dx > dy ta cho x nguyên và tính y theo x theo phương trình đường thẳng. Sau đó tính độ chênh lệch hay tổng sai số: $v += |x_r - x| + |y_r - y|$
- Lần lượt vẽ 1, 5, 10 đường thẳng ta được tổng sai số:

	1	5	10
DDA	9924	170158	1699502
Bresenham	9927	170158	1699509
Mid point	10397	171263	1018883
Xiaolin Wu	19929	709412	4664473

- Xiaolin Wu mỗi lần vẽ 2 pixel và mục tiêu chính là khử răng cưa nên tính độ lệch ở đây không có ý nghĩa.
- DDA và Bresenham cho kết quả khá tương đồng, khả năng sai số hay độ chênh lệch với điểm thực gần như bằng nhau.
- Mid point cho kết quả sai số cao hơn DDA và Bresenham ở số lượng nhỏ nhưng vẽ càng nhiều sai số lại thấp hơn.

II. Nguồn tham khảo

- DDA:
  - https://www.tutorialspoint.com/computer_graphics/line_generation_algorithm.htm
  - https://www.geeksforgeeks.org/dda-line-generation-algorithm-computer-graphics/
- Bresenham:
  - https://en.wikipedia.org/wiki/Bresenham%27s_line_algorithm
  - https://tuhoclaptrinh.cachhoc.net/2017/03/03/bai-4-thuat-toan-ve-duong-thang-bresenham/
- Midpoint:
  - http://monhoc.vn/tai-lieu/do-hoa-may-tinh-cac-thuat-toan-ve-duong-1421/ (Circle)
  - http://baythi.vnweblogs.com/gallery/20982/07k4114_BaiTapSo1.pdf (Elip, Parabol, Hyperbol)
  - https://tuhoclaptrinh.cachhoc.net/2017/03/12/bai-7-thuat-toan-ve-duong-ellipse-midpoint/
- Xiaolin Wu:
  - https://rosettacode.org/wiki/Xiaolin_Wu%27s_line_algorithm#C.23