ĐẠI HỌC HUẾ
TRƯỜNG ĐẠI HỌC KHOA HỌC
KHOA CÔNG NGHỆ THÔNG TIN

NGUYỄN HỮU TÀI

GIÁO TRÌNH
ĐỒ HỌA MÁY TÍNH
(ĐÀO TẠO CỬ NHÂN CÔNG NGHỆ THÔNG TIN)

NHÀ XUẤT BẢN ĐẠI HỌC HUẾ
Huế, 2017

Mã số sách: GT/ -2017

LỜI NÓI ĐẦU

Giáo trình này nhằm mang đến cho người học là các sinh viên ngành Công nghệ Thông tin những kiến thức cơ bản và chuyên sâu trong lĩnh vực đồ họa máy tính, rèn luyện và phát triển kỹ năng thực hành thực nghiệm, kỹ năng lập trình cho lĩnh vực đồ họa máy tính. Nội dung của giáo trình với thời lượng giảng dạy 3 tín chỉ gồm có 6 chương, 2 phục lục và 7 bài thực nghiệm được trình bày hướng dẫn chi tiết nhằm từng bước phát triển kỹ năng lập trình đồ họa, hiểu sâu và đánh giá chính xác các lý thuyết và giải thuật đồ họa. Bố cục của giáo trình gồm:

LỜI NÓI ĐẦU

Chương 1: Các yếu tố cơ sở của đồ hoạ

Trình bày các khái niệm cơ bản về thiết bị đồ hoạ và điểm ảnh (Pixel). Giới thiệu và trình bày chi tiết các giải thuật dựng các đường cơ bản như: đoạn thẳng, đường tròn, ellipse. Hướng dẫn chi tiết các bước để tạo ứng dụng khung phục vụ cho việc thực hành thực nghiệm thông qua "***Bài thực nghiệm số 1***", để từ đó dẫn dắt làm quen và trang bị từng bước các kiến thức lập trình đồ họa trên windows với VC++ và MFC.

Chương 2: Các hệ màu và cơ chế tổ chức bộ nhớ màn hình

Trình bày đôi nét về cấu trúc màn hình màu. Tính chất giao thoa ánh sáng và nguyên lý tạo điểm màu trên màn hình hay máy in. Giới thiệu sơ bộ về các hệ màu RGB, CMY, HSV. Tìm hiểu về cơ chế tổ chức bộ nhớ màn hình, cách tính địa chỉ để truy xuất thông tin điểm ảnh, qua đó tìm hiểu một mode độ họa căn bản là mode 13H.

Chương 3: Các phép xén hình và tô màu

Giới thiệu phạm vi và ứng dụng của bài toán xén hình. Trình bày chi tiết các giải thuật xén hình căn bản như: Xén đoạn thẳng vào hình chữ nhật, xén đa giác vào hình chữ nhật. Giới thiệu bài toán tô màu và ứng dụng. Trình bày chi tiết 2 giải thuật tô màu gồm: Giải thuật vết dầu loang (Flood fill), và giải thuật tô đa giác theo dòng quét (Scan-line). Tìm hiểu sâu hơn về vấn đề xử

lý đồ họa của hệ thống thông qua "***Bài thực nghiệm số 2***" để xử lý bài toán tô màu theo giải thuật vết dầu loang.

Chương 4: Các phép biến đổi hình học

Trình bày lý thuyết biến đổi hình học affine với căn bản là các phép tính toán ma trận. Hệ tọa độ thuần nhất và lợi ích của nó trên mô hình xử lý máy tính. Ví dụ hướng dẫn về các phân tích bài toán biến hình phức tạp về thành tổng hợp của những phép biến hình cơ bản, dựa trên việc tính tích các ma trận. Phân tích bài toán quan sát vật thể trong không gian 3 chiều và sự mô phỏng thế giới thực.

Chương 5: Mô hình WireFrame

Trình bày chi tiết về mô hình Wireframe. Cách thức tổ chức lưu trữ thông tin. Hướng dẫn xây dựng một ứng dụng mô phỏng việc quan sát vật thể 3 chiều trong không gian theo mô hình Wireframe, áp dụng kết hợp kiến thức của chương 4 và chương 5, thông qua "***Bài thực nghiệm số 3***"

Chương 6: Mô hình các mặt đa giác và vấn đề khử mặt khuất

Giới thiệu mô hình các mặt đa giác, ưu và nhược điểm cùng cách thức tổ chức lưu trữ thông tin. Giới thiệu bài toán khử mặt khuất và trình bày chi tiết các giải thuật sắp xếp theo độ sâu, giải thuật chọn lọc mặt sau, giải thuật vùng đệm độ sâu. "***Bài thực nghiệm số 4***" giúp phát triển ứng dụng 3DViewer mô phỏng việc quan sát vật thể trong không gian 3 chiều trên máy tính, trong đó vấn đề khử mặt khuất được xử lý bởi giải thuật chọn lọc mặt sau. Và "***Bài thực nghiệm số 5***" phát triển một bản nâng cấp của ứng dụng 3DViewer trong đó vấn đề khử mặt khuất được xử lý bởi giải thuật vùng đệm độ sâu.

Cuối cùng là hệ thống 2 phụ lục nhằm giúp sinh viên có thể tìm hiểu và nghiên cứu sâu hơn một số vấn đề mà trong khuôn khổ thời gian hạn hẹp của học phần không cho phép tìm hiểu hết được. Các phục lục đó bao gồm:

Phụ lục I: Các phương pháp dựng đường cong và mặt cong

Trình bày chi tiết các phương pháp tạo đường cong và mặt cong

hiệu quả trên mô hình máy tính.

Phụ lục II: Các mô hình chiếu sáng

Trình bày chi tiết việc tính toán mô phỏng các loại hình chiếu sáng lên vật thể 3 chiều nhằm tăng tính trung thực của hình ảnh mô phỏng, tạo cho hình ảnh mô phỏng được trung thực và "đẹp hơn". Đưa ra các giải pháp công nghệ xử lý kết hợp nhằm giải quyết vấn đề tốc độ cho bài toán mô phỏng xử lý hình ảnh 3 chiều trên máy tính. "***Bài thực nghiệm số 6***" và "***Bài thực nghiệm số 7***" giúp phát triển và hoàn thiện hơn nữa ứng dụng 3DView, để ứng dụng có thể mô phỏng trung thực hơn các đối tượng 3 chiều, và qua đó giúp sinh viên có được kiến thức sâu hơn và kỹ năng vững vàng về vấn đề xử lý mô phỏng đối tượng 3 chiều trên máy tính.

Với mong muốn tạo điều kiện tốt nhất để sinh viên có thể dễ dàng lĩnh hội kiến thức lý thuyết, phát triển năng lực thực nghiệm và kỹ năng giải quyết vấn đề nói chung hay kỹ năng lập trình nói riêng. Tác giả đã cố gắng để trình bày các vấn đề thuộc lĩnh vực đồ họa máy tính một cách chi tiết mạch lạc và chuẩn xác nhất có thể, các kỹ thuật xử lý luôn sát với công nghệ trong thực tiễn. Hy vọng rằng nó sẽ mang lại nhiều bổ ích cho sinh viên cũng như bạn đọc. Tác giả cũng mong nhận được nhiều đóng góp ý kiến của quý đồng nghiệp cùng bạn đọc để cuốn sách được hoàn thiện hơn trong lần tái bản sau.

Tác giả

TS. Nguyễn Hữu Tài

MỤC LỤC

DANH SÁCH HÌNH VẼ

DANH SÁCH BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ CHỮ VIẾT TẮT

Ký hiệu và chữ viết tắt	Giải nghĩa
Affine 2D	**Không gian Affine 2 chiều**
Affine geometry	**Hình học Affine**
Back-Face Detection	**Giải thuật chọn lọc mặt sau**
Bezier/Bézier	**Tên của giải thuật phát sinh đường cong hay mặt cong trong lĩnh vực đồ họa**
Bresenham	Tên của một số giải thuật dựng hình cơ bản (đoạn thẳng, đường tròn, đường ellipse) trong giáo trình này.
B-spline	**Dạng tổng quát hóa của đường cong hay mặt cong Bezier**
CMY	Không gian màu, được xây dựng trên 3 màu cơ sở Cyan (màu lục lam), Magenta (màu đỏ tươi), Yellow (màu vàng)
Cohen-Sutherland	Tên của một giải thuật xén đoạn thẳng vào hình chữ nhật
DC	Ngữ cảnh (hay bối cảnh) thiết bị đồ họa, là một cấu trúc định nghĩa một tập các đối tượng đồ họa và các thuộc tính liên quan của chúng, có 4 loại DC khác nhau trong MFC là: Display, Printer, Memory (or compatible), và Information. *Device context*
Depth-Sorting	**Giải thuật người thợ sơn hay sắp xếp theo chiều sâu**
DIB	**ảnh bitmap không phụ thuộc thiết bị** ***Device-Independent Bitmap***
Flood Fill	Chỉ quá trình tô màu theo giải thuật vết dầu loang
Homogeneous Coordinates	**Tọa độ thuần nhất**
HSV	Không gian màu, được xây dựng trên 3 thành phần cơ

	sở là H (Hue, sắc màu), S (Saturation, độ bão hòa) và V (Value, thể hiện độ sáng)
Liang-Barsky	Tên của một giải thuật xén đoạn thẳng vào hình chữ nhật
MFC	Thư viện chứa các lớp C++ dùng để bao bọc các hàm API của hệ điều hành Windows *Microsoft Foundation Class Library*
MidPoint	Tên của một giải thuật dựng đường tròn
Pixel	Điểm ảnh
Polygon mesh model	**Mô hình các mặt đa giác lưu trữ thông tin của đối tượng trong không gian 3 chiều**
Phong	Tên của một mô hình xử lý ánh sáng, giúp cho đối tượng 3 chiều có hình dáng cong có được hình ảnh bóng sáng sát với thực tế hơn các mô hình tạo bóng sáng khác trong lĩnh vực đồ họa
RGB	Không gian màu, được xây dựng trên 3 màu cơ sở Red, Green và Blue
ScanLine	Tên một thuật toán tô đa giác theo phương pháp quét dòng (scan line) trong giáo trình này
Sutherland-Hodgman	Tên của một giải thuật xén đa giác vào hình chữ nhật
VESA	**Một tổ chức tiêu chuẩn kỹ thuật cho chuẩn hiển thị trên máy tính** ***Video Electronics Standards Association***
WireFrame	**Mô hình khung dây lưu trữ thông tin về hình dáng (bộ khung) của đối tượng trong không gian 3 chiều**
Z-Buffer	**Giải thuật vùng đệm độ sâu**

Chương 1. CÁC YẾU TỐ CƠ SỞ CỦA ĐỒ HỌA

Trong chương này sẽ trình bày các khái niệm về điểm ảnh, tọa độ điểm ảnh và ma trận điểm ảnh trên thiết bị đồ họa. Trình bày các giải thuật giúp dựng hình một cách hiệu quả đối với các đối tượng cơ bản như đoạn thẳng, hình tròn, hình ellipse.

1. CÁC KHÁI NIỆM CƠ BẢN

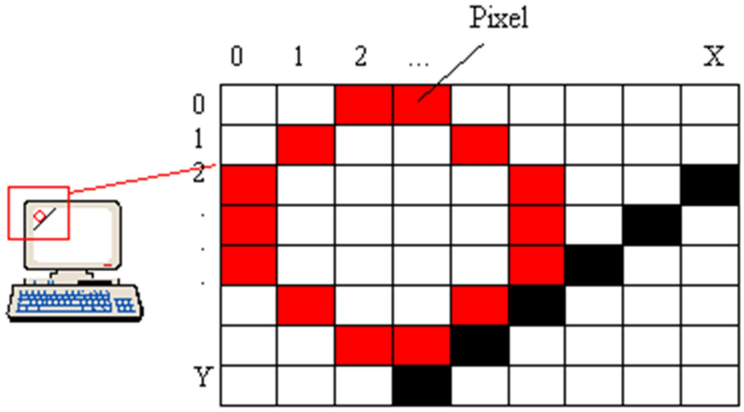
1.1. Thiết bị đồ hoạ và điểm ảnh

Thiết bị đồ họa được hiểu là những phương tiện giúp chúng ta thể hiện được các hình ảnh thông qua sự điều khiển của máy tính. Từ đó chúng ta có thể liệt kê một số thiết bị quen thuộc như màn hình máy tính, máy in, máy vẽ,…Hình 1.1 cho thấy khả năng hiển thị sinh động hình vẽ, chữ, hình ảnh thu được từ camera trên một màn hình máy tính của hãng Dell. Để có thể điều khiển được quá trình hiển thị thông tin (hình vẽ, chữ viết, hình ảnh,…) trên thiết bị đồ họa, chúng ta cần hiểu được tính chất cấu tạo của chúng. Trong chương này chúng ta cần tìm hiểu một số khái niệm cơ bản liên quan đến quá trình dựng hình.

Hình 1.1. Giao diện đồ họa windows 8 thể hiện trên màn hình của hãng Dell

Mỗi thiết bị đồ họa có một mặt phẳng (hai chiều) được phân chia

thành các dòng (rows) và các cột (columns). Giao của các dòng và các cột tạo nên các điểm ảnh, thuật ngữ tiếng anh là Pixel. Kích thước của điểm ảnh phụ thuộc vào diện tích của bề mặt hiển thị và số điểm ảnh tối đa mà thiết bị điều khiển và hiện thị được trên bề mặt đó. Độ phân giải của thiết bị màn hình thường được biểu diễn bởi khả năng phân chia với số cột và số dòng cực đại. Ví dụ màn hình LCD Full HD sẽ cho khả năng phân chia được 1920 cột và 1080 dòng, từ đó tạo nên hơn 2 triệu điểm ảnh. Các cột và các dòng được đánh chỉ số bắt đầu từ 0 tại vị trí góc trên bên trái như minh họa trong Hình 1.2. Từ đó mỗi điểm ảnh được định danh thông qua một cặp chỉ số (x,y) trong đó x và y lần lượt là chỉ số cột và chỉ số dòng tạo nên điểm ảnh đó, cặp chỉ số này còn được gọi là tọa độ điểm ảnh trên thiết bị đồ họa. Từ đó cũng dễ thấy rằng tọa độ điểm ảnh trên thiết bị đồ họa luôn luôn phải là một cặp số nguyên dương hoặc bằng không. Các cặp giá trị tọa độ thực (không nguyên) hoặc âm không được chấp nhận vì nó không giúp hệ thống xác định được điểm ảnh (pixel) cần điều khiển.



Hình 1.2. Minh họa việc hiển thị hình ảnh đồ họa trên thiết bị

1.2. Điểm và đoạn thẳng trong mặt phẳng

Về mặt toán học thì một đoạn thẳng bao gồm một tập vô hạn các điểm trong mặt phẳng với cặp toạ độ thực và không có kích thước (hay kích thước vô cùng bé). Khái niệm này có nhiều khác biệt với khái niệm Pixel trên thiết bị đồ họa mà người học cần nắm vững trước khi bắt đầu

tìm hiểu bài toán dựng hình trong lĩnh vực đồ họa máy tính. Từ Hình 1.2 chúng ta có thể hiểu rằng quá trình dựng hình trên thiết bị đồ họa chính là quá trình xác định một tập các điểm ảnh (pixel) sao cho chúng có thể thể hiện được hình ảnh mà chúng ta mong muốn ở mức tốt nhất (tối ưu nhất) có thể. Ví dụ đoạn thẳng màu đen được thể hiện bằng một tập 6 pixel liên tiếp nhau như minh họa trong Hình 1.2, mỗi pixel có một kích thước cụ thể phụ thuộc vào kích thước và độ phân giải của thiết bị.

2. CÁC GIẢI THUẬT VẼ ĐOẠN THẲNG

Phương trình tổng quát của một đường thẳng được viết dưới dạng:

$$y = ax + b$$

trong đó:

- a là hệ số góc hay còn gọi là độ dốc, nó phản ánh mối tương quan giữa 2 biến số x và y.
- b là khoảng chắn trên trục hoành.

Phương trình đường thẳng đi qua 2 điểm $A(x_a, y_a)$ và $B(x_b, y_b)$ được viết dưới dạng:

$$\frac{y-y_a}{y_b-y_a} = \frac{x-x_a}{x_b-x_a} \qquad (1.1)$$

(*với* $x_a <> x_b$ *và* $y_a <> y_b$. *Khi* $x_a = x_b$ *thì phương trình là* $x = x_a$, *còn khi* $y_a = y_b$ *thì phương trình là* $y = y_a$)

Đặt $\Delta x = x_b - x_a$ và $\Delta y = y_b - y_a$ thì (1.1) trở thành

$$y = \frac{\Delta y}{\Delta x}x - \frac{\Delta y}{\Delta x}x_a + y_a$$

$$\Leftrightarrow y = ax + b \text{ với } \begin{cases} a = \dfrac{\Delta y}{\Delta x} \\ b = -ax_a + y_a \end{cases} \qquad (1.2)$$

2.1. Vẽ đoạn thẳng dựa vào phương trình

Khi biết phương trình của một đường chúng ta hoàn toàn có thể vẽ

được đường biểu diễn nhờ vào các tính toán trên phương trình. Ở đây đường mà chúng ta cần biểu diễn là một đoạn thẳng AB với $A(x_a,y_a)$ và $B(x_b,y_b)$. Phương trình biểu diễn được cho bởi (1.2) với

$$x \in [x_a, x_b], y \in [y_a, y_b]$$

Quy trình có thể tóm tắt như sau:

- Nếu $|\Delta y| \le |\Delta x|$, nghĩa là biến số x biến thiên nhanh hơn biến số y, lúc này để đảm bảo tính liên tục của các điểm vẽ chúng ta cho biến số x thay đổi tuần tự và tính biến số y qua phương trình. Cụ thể như sau:

  Cho x nhận các giá trị nguyên lần lượt từ x_a đến x_b, với mỗi giá trị x chúng ta thực hiện:

  - Tính $y = ax + b$ thông qua phương trình
  - Vẽ điểm (x, round(y))

  Ở đây điểm trên đoạn thẳng có toạ độ là (x, y) song chúng ta không thể vẽ điểm đó bởi giá trị y là một giá trị thực trong khi các hệ thống biểu diễn đồ hoạ chỉ có hữu hạn điểm và mỗi điểm có toạ độ nguyên. Từ đó chúng ta buộc phải minh hoạ cho điểm (x, y) thuộc đoạn thẳng thực bởi một điểm trên hệ thống thiết bị đồ họa gần với nó nhất, đó chính là điểm có toạ độ cột là x và dòng là giá trị làm tròn về số nguyên của y.

- Ngược lại, nghĩa là biến số y biến thiên nhanh hơn biến số x, lúc này để đảm bảo tính liên tục của các điểm vẽ chúng ta cho biến số y thay đổi tuần tự và tính biến số x qua phương trình. Cụ thể như sau:

Cho y nhận các giá trị nguyên lần lượt từ y_a đến y_b, với mỗi giá trị y chúng ta thực hiện:

- Tính $x = \frac{y-b}{a}$ (hay $x = \frac{\Delta x}{\Delta y} y - \frac{\Delta x}{\Delta y} y_a + x_a$)
- Vẽ điểm (round(x), y)

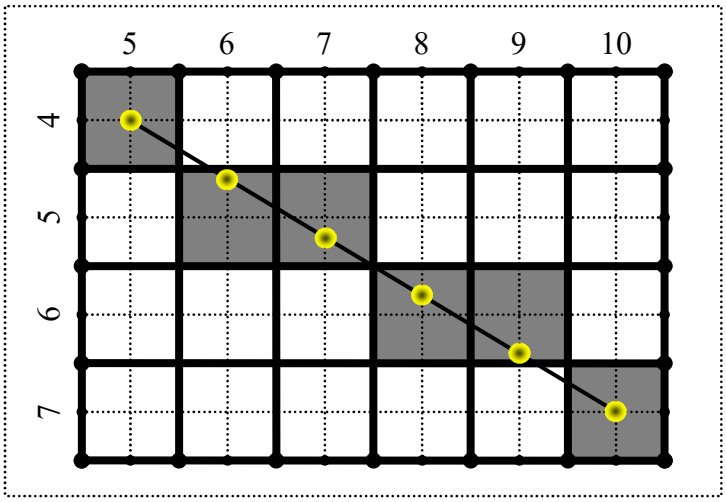
Ví dụ: Cho A(5, 4) đến B(10, 7) để vẽ đoạn thẳng AB chúng ta thực hiện các bước sau:

Tính: $\Delta x = x_b - x_a = 10 - 5 = 5; \Delta y = y_b - y_a = 7 - 4 = 3$

$$\begin{cases} a = \dfrac{\Delta y}{\Delta x} = \dfrac{3}{5} \\ b = -ax_a + y_a = 1 \end{cases}$$

Vì $|\Delta y| \le |\Delta x|$, nên chúng ta cho x nhận các giá trị nguyên lần lượt từ x_a đến x_b, với mỗi giá trị x chúng ta cần thực hiện:

- Tính $y = ax + b$ thông qua phương trình
- Vẽ điểm (x, round(y))



Hình 1.3. Ảnh minh họa một đoạn thẳng từ A(5,4) đến B(10,7)

Cụ thể như sau:

Khi x = x_a = 5:	y = ax+b = 4;	Vẽ điểm (5,4)
Khi x = 6:	y = 23/5 = 4.6;	Vẽ điểm (6,5)
Khi x = 7:	y = 26/5 = 5.2;	Vẽ điểm (7,5)
Khi x = 8:	y = 29/5 = 5.8;	Vẽ điểm (8,6)
Khi x = 9:	y = 32/5 = 6.4;	Vẽ điểm (9,6)
Khi x = 10:	y = 7;	Vẽ điểm (10,7)

Kết quả chúng ta có hình vẽ đoạn thẳng AB có thể minh họa như trong Hình 1.3.

2.2. Vẽ đoạn thẳng dựa vào giải thuật Bresenham

Mục 2.1 đã đưa ra quy trình để vẽ một đoạn thẳng AB bất kỳ trên thiết bị đồ họa. Song phương pháp tính toán còn chưa thật sự hiệu quả. Cụ thể tại mỗi bước lặp để tìm ra được tọa độ của một điểm vẽ chúng ta cần phải tính 1 phép nhân và 1 phép cộng trên trường số thực, cùng với một phép tính làm tròn (round) số thực về số nguyên. Cũng với cách tiếp cận trên, song giải thuật Bresenham hướng tới một sự phân tích bài toán sâu sắc hơn để đi đến một quy trình ít tính toán hơn.

Giả thiết đầu tiên mà giải thuật Bresenham đặt ra là hệ số góc của đoạn thẳng $a \in [0,1]$, các trường hợp còn lại của hệ số góc như $a \in [1,+\infty)$; $a \in [-1,0]$; $a \in (-\infty,-1]$ có thể được quy về trường hợp đoạn thẳng có hệ số góc $a \in [0,1]$ thông qua các phép lấy đối xứng, quy trình xử lý cụ thể đối với các đoạn thẳng có hệ số góc $a \notin [0,1]$ sẽ được bàn thảo và hướng dẫn tại mục 2.2.3.

Từ giả thiết đặt ra là hệ số góc của đoạn thẳng $a \in [0,1]$, chúng ta có thể suy ra rằng trên toàn bộ đoạn thẳng tham số x luôn luôn biến thiên nhanh hơn tham số y. Từ đó đưa đến quy trình: cho x nhận các giá trị nguyên lần lượt từ x_a đến x_b, với mỗi giá trị x chúng ta cần phải tìm ra một giá trị y nguyên để (x,y) chính là toạ độ của điểm cần minh hoạ trên thiết bị. Và điểm mấu chốt ở đây là việc tìm ra giá trị y phải thông qua ít phép tính toán hơn quy trình đã trình bày ở mục 2.1.

Giả thiết với hai điểm đầu mút A(x_a, y_a) và B(x_b, y_b) có toạ độ nguyên và $x_a<x_b$ (nếu cần thì hoán đổi hai đầu mút A và B để thỏa mãn giải thiết $x_a<x_b$). Rõ ràng điểm ảnh đầu tiên cần biểu diễn trên thiết bị chính là điểm A có toạ độ (x_a, y_a). Nếu gọi điểm ảnh được lựa chọn đầu tiên trong quy trình là (x_0, y_0) thì:

$$(x_0, y_0) = (x_a, y_a)$$

Theo lập luận quy nạp:

- Giả thiết rằng đến bước thứ i chúng ta đã chọn được điểm ảnh thứ i, hay nói cách khác là điểm ảnh được chọn ở bước thứ i có tên gọi (x_i, y_i) đã được xác định giá trị.
- Vậy đến bước tiếp theo (bước thứ i+1) chúng ta sẽ chọn điểm ảnh

nào? Nói cách khác là điểm ảnh được chọn ở bước thứ (i+1) với tên gọi (x_{i+1}, y_{i+1}) sẽ được xác định các giá trị ra sao?

Chú ý: *x_i, y_i là tên gọi của toạ độ điểm ảnh chọn thứ i, ví dụ như (x_0, y_0) là tên gọi của điểm ảnh được lựa chọn đầu tiên (i = 0) và nó có giá trị là (x_a, y_a)*

Để trả lời câu hỏi này chúng ta cần dựa vào một số lập luận sau đây:

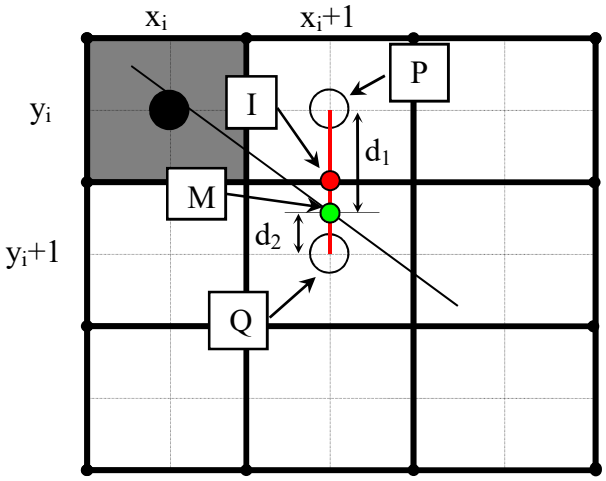
Như đã trình bày thì điểm ảnh chọn thứ i+1 sẽ phải có hoành độ x bằng hoành độ của điểm ảnh được lựa chọn trước đó cộng thêm 1:

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

Gọi M là điểm thuộc AB sao cho $x_M = x_{i+1} = x_i + 1$

Thì: $y_M = ax_M + b = a(x_i + 1) + b = (ax_i + b) + a$

Vậy điểm tiếp theo thuộc đoạn thẳng mà chúng ta cần tìm điểm ảnh minh hoạ trên thiết bị là M($x_i + 1$, $(ax_i + b) + a$). Câu hỏi đặt ra là chúng ta sẽ chọn điểm nào trong 2 điểm ảnh P($x_i + 1$, y_i) và Q($x_i + 1$, $y_i + 1$) để minh hoạ cho M trên thiết bị đồ hoạ (*xem Hình 1.4*).



Hình 1.4. Minh họa việc chọn lựa điểm P hay Q dựa vào các tham số

Để trả lời câu hỏi này chúng ta cần xem xét một biểu thức trung gian:

Đặt $d_1 = (y_M - y_P)$ và $d_2 = (y_Q - y_M)$

Xét biểu thức:

$$d_1 - d_2 = (y_M - y_P) - (y_Q - y_M) = 2y_M - (y_P + y_Q) = 2\left[y_M - \frac{y_P + y_Q}{2} \right]$$

Nếu gọi I là trung điểm của QP thì: $d_1 - d_2 = 2[y_M - y_I]$

Rõ ràng là:

Nếu $d_1 - d_2 < 0$ dẫn đến $y_M < y_I$, suy ra P gần điểm M hơn Q, vậy chúng ta sẽ chọn điểm ảnh P làm điểm minh họa cho điểm M thuộc đường thẳng trên thiết bị đồ hoạ

- Nếu $d_1 - d_2 > 0$ dẫn đến $y_M > y_I$, suy ra Q gần điểm M hơn P, vậy chúng ta sẽ chọn điểm ảnh Q làm điểm minh họa cho M trên thiết bị đồ hoạ
- Nếu $d_1 - d_2 = 0$ dẫn đến $y_M = y_I$, suy ra khả năng lựa chọn P và Q là như nhau, song chúng ta phải quyết định chọn một điểm ảnh. Trong tình huống này giải thuật quy định chọn điểm Q.

Vậy để tìm được điểm minh hoạ tiếp theo chúng ta cần xét dấu của biểu thức $d_1 - d_2$. Song chúng ta thấy biểu thức $d_1 - d_2$ còn khá phức tạp và phải thực hiện tính toán trên trường số thực do trong đó có xuất hiện phép chia:

$$\begin{aligned} y_M &= (ax_i + b) + a = (ax_i + (-ax_a + y_a)) + a \\ &= \frac{\Delta y}{\Delta x} x_i - \frac{\Delta y}{\Delta x} x_a + y_a + \frac{\Delta y}{\Delta x} \end{aligned} \tag{1.3}$$

Để tránh tính biểu thức $d_1 - d_2$ trên trường số thực người ta hướng tới một biểu thức tương đương về dấu đó là

$$P_i = \Delta x(d_1 - d_2)$$

Việc đưa Δx vào nhằm loại bỏ mẫu số trong biểu thức $d_1 - d_2$, từ đó thu được biểu thức P_i tính trên trường số nguyên. Thật vậy:

$$\begin{aligned} P_i &= \Delta x(d_1 - d_2) = \Delta x(2y_M - (y_P + y_Q)) = \Delta x(2y_M - (y_i + y_i + 1)) \\ &= \Delta x(2y_M - 2y_i - 1) = 2\Delta x y_M - 2\Delta x y_i - \Delta x \end{aligned}$$

Thay y_M bởi giá trị ở (1.3) chúng ta được:

$$\begin{aligned} P_i &= 2\Delta y x_i - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - 2\Delta x y_i - \Delta x \\ &= 2\Delta y x_i - 2\Delta x y_i - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - \Delta x \end{aligned} \quad (1.4)$$

Chúng ta thấy biểu thức P_i được xác lập từ toạ độ của điểm chọn thứ i là (x_i, y_i). Vậy P_{i+1} sẽ được xác lập từ điểm chọn thứ i+1 là (x_{i+1}, y_{i+1}) như sau:

$$P_{i+1} = 2\Delta y x_{i+1} - 2\Delta x y_{i+1} - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - \Delta x \quad (1.5)$$

Vì dấu của P_i và dấu của $(d_1 - d_2)$ là tương đương nên có thể tóm tắt quy tắc chọn điểm ảnh tiếp theo như sau:

- Nếu $P_i < 0$: Thì chọn điểm ảnh P làm điểm minh họa cho M trên thiết bị đồ hoạ. Hay nói cách khác là điểm ảnh chọn thứ i+1 là (x_{i+1}, y_{i+1}) sẽ có giá trị bằng P. Nghĩa là $(x_{i+1}, y_{i+1}) = (x_i+1, y_i)$, từ đó thay vào công thức (1.5) chúng ta có:

$$P_{i+1} = 2\Delta y(x_i + 1) - 2\Delta x y_i - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - \Delta x = P_i + 2\Delta y$$

- Nếu $P_i \geq 0$: Thì chọn điểm Q là điểm minh họa cho M trên thiết bị đồ hoạ. Hay nói cách khác là điểm chọn thứ i+1 là (x_{i+1}, y_{i+1}) sẽ có giá trị bằng Q. Nghĩa là: $(x_{i+1}, y_{i+1}) = (x_i+1, y_i+1)$, từ đó thay vào công thức (1.5) chúng ta có:

$$\begin{aligned} P_{i+1} &= 2\Delta y(x_i + 1) - 2\Delta x(y_i + 1) - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - \Delta x \\ &= P_i + 2\Delta y - 2\Delta x \end{aligned}$$

Khi i = 0, ta có $(x_0, y_0) = (x_a, y_a)$ thay vào (1.4) chúng ta có:

$$P_0 = 2\Delta y x_0 - 2\Delta y x_a + 2\Delta x y_a + 2\Delta y - 2\Delta x y_0 - \Delta x = 2\Delta y - \Delta x$$

Vậy từ đây chúng ta thấy được quy trình chọn ra các điểm trên thiết bị để minh hoạ cho đoạn thẳng AB theo giải thuật Bresenham như sau:

- Điểm chọn đầu tiên (i = 0) là $(x_0, y_0) = (x_a, y_a)$ và giá trị $P_0 = 2\Delta y - \Delta x$

- Dựa vào giá trị của P_0 là âm hay dương mà chúng ta lại chọn được điểm tiếp theo (x_1,y_1) và tính được giá trị P_1
- Dựa vào giá trị của P_1 là âm hay dương mà chúng ta lại chọn được điểm tiếp theo (x_2,y_2) và tính được giá trị P_2
- Cứ như vậy chúng ta tìm ra được tập các điểm trên thiết bị đồ hoạ để minh hoạ cho đoạn thẳng AB.

2.2.1. Tóm tắt giải thuật Bresenham:

Đầu vào: Tọa độ (x_a, y_a) và (x_b, y_b) của đoạn thẳng AB thỏa mãn giả thiết hệ số góc thuộc đoạn [0,1] và $x_a < x_b$.

Ra: Vẽ các điểm ảnh nhằm thể hiện hình ảnh đoạn thẳng AB trên mặt phẳng thiết bị đồ họa.

- **Bước 1:** (Bước khởi động, tính toán các giá trị ban đầu)

  $\Delta x = x_b - x_a$; $\Delta y = y_b - y_a$;

  Const1 = $2\Delta y$; Const2 = $2\Delta y - 2\Delta x$

  $P_0 = 2\Delta y - \Delta x$; $(x_0, y_0) = (x_a, y_a)$

  Vẽ điểm (x_0, y_0)

- **Bước 2:** (Bước lặp, thực hiện tính các giá trị điểm ảnh)

  Với mỗi giá trị i (i = 0, 1, 2,…) chúng ta xét dấu P_i

  - Nếu $P_i < 0$: thì chọn điểm tiếp theo là

    $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i)$

    $P_{i+1} = P_i + Const1$

  - Ngược lại (tức $P_i \geq 0$): thì chọn điểm tiếp theo là

    $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i + 1)$

    $P_{i+1} = P_i + Const2$

  Vẽ điểm (x_{i+1}, y_{i+1}) vừa tìm được

- **Bước 3:** (Xác định điều kiện lặp)

  Lặp lại bước 2 với những giá trị i tiếp theo, cho đến khi điểm tìm được trùng với B, nghĩa là $x_{i+1} = x_b$ thì giải thuật

kết thúc.

2.2.2. Ví dụ:

Cho đoạn thẳng AB với A(5,6) và B(10,10). Sử dụng giải thuật Bresenham chúng ta có thể tìm được các điểm ảnh cần vẽ để biểu diễn đoạn AB trên màn hình như sau:

- Bước 1:

  $\Delta x = 10-5 = 5;\quad \Delta y = 10-6 = 4;$

  $Const1 = 2\Delta y = 8;\ Const2 = 2\Delta y-2\Delta x = 8-10 = -2;$

  $P_0 = 2\Delta y-\Delta x = 8-5 = 3;\quad (x_0, y_0) = (x_a , y_a) = (5,6)$

  Vẽ điểm (x_0, y_0)

- **Bước 2:** Bước lặp:

  i = 0:

  Ta có $P_0 = 3 \geq 0$ nên:

  $(x_1, y_1) = (x_0 + 1, y_0 + 1) = (6, 7)$

  $P_1 = P_0 + Const2 = 3 + (-2) = 1$

  Vẽ điểm $(x_1, y_1) = (6, 7)$

  i = 1:

  Ta có $P_1 = 1 \geq 0$ nên:

  $(x_2, y_2) = (x_1 + 1, y_1 + 1) = (7, 8)$

  $P_2 = P_1 + Const2 = 1 + (-2) = -1$

  Vẽ điểm $(x_2, y_2) = (7, 8)$

  i = 2:

  Ta có $P_2 = -1 < 0$ nên:

  $(x_3, y_3) = (x_2 + 1, y_2) = (8, 8)$

  $P_3 = P_2 + Const1 = -1 + 8 = 7$

  Vẽ điểm $(x_3, y_3) = (8, 8)$

i = 3:

Ta có $P_3 = 7 \geq 0$ nên:

$(x_4, y_4) = (x_3 + 1, y_3 + 1) = (9, 9)$

$P_4 = P_3 + Const2 = 7 + (-2) = 5$

Vẽ điểm $(x_4, y_4) = (9, 9)$

i = 4:

Ta có $P_4 = 5 \geq 0$ nên:

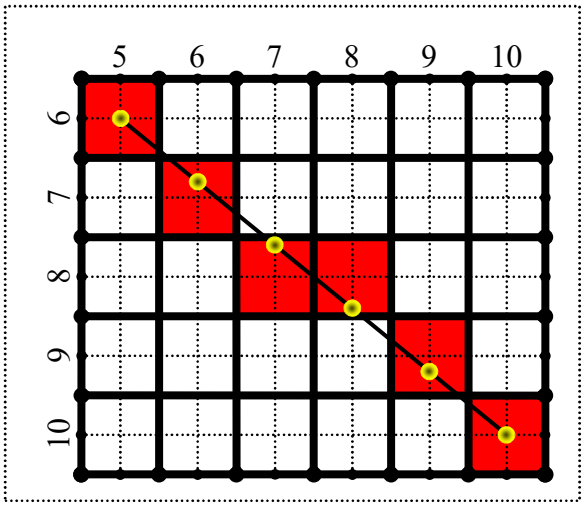
$(x_5, y_5) = (x_4 + 1, y_4 + 1) = (10, 10)$

$P_5 = P_4 + Const2 = 5 + (-2) = 3$

Vẽ điểm $(x_5, y_5) = (10, 10)$

Vì $x_5 = x_b = 10$, nên kết thúc vòng lặp và cũng là kết thúc giải thuật.

Hình vẽ minh họa (*xem Hình 1.5*)



Hình 1.5. Minh họa đoạn thẳng được vẽ trên thiết bị đồ họa. Các Pixel vuông chính là hình ảnh thể hiện của đoạn thẳng AB trên màn hình máy tính

2.2.3. Hướng dẫn cho các trường hợp hệ số góc ngoài đoạn [0, 1]

Giả sử cho A(0,50), B(100,10) để dựng đoạn thẳng AB chúng ta cần

tiến hành một số phân tích:

$\Delta x = x_b - x_a = 100 - 0 = 100$

$\Delta y = y_b - y_a = 10 - 50 = -40$

Suy ra hệ số góc $a = \Delta y/\Delta x = -0.4 \in [-1, 0]$

Lúc này ta cần lấy đối xứng của AB qua trục OX để được CD với C(0,-50) D(100,-10) nên xét trên đoạn thẳng CD chúng ta có

$\Delta x = x_d - x_c = 100 - 0 = 100$

$\Delta y = y_d - y_c = -10 - (-50) = 40$

Suy ra hệ số góc $a = \Delta y/\Delta x = 0.4 \in [1, 0]$ thỏa mãn điều kiện của giải thuật Bresenham. Từ đó chúng ta có thể áp dụng giải thuật Bresenham để tính toán ra các điểm ảnh cần vẽ trên CD nhưng chúng ta sẽ không vẽ nó (vì mục đích chúng ta là dựng hình AB) mà lại lấy đối xứng qua trục OX (tức đối xứng ngược lại với lúc đầu) rồi mới vẽ, thì lúc này các điểm ảnh vẽ ra sẽ là hình ảnh của đoạn thẳng AB. Như thế CD chỉ đóng vai trò trung gian nhằm áp dụng được giải thuật còn kết quả sau cùng chúng ta thu được vẫn là hình ảnh minh họa cho đoạn AB.

Áp dụng tương tự:

- Trường hợp hệ số góc $a \in (1,+\infty)$, lúc này chúng ta cần lấy đối xứng qua đường phân giác của góc phần tư thứ nhất để hệ số góc được quy về [0, 1].
- Trường hợp hệ số góc $a \in (-\infty,-1)$, lúc này chúng ta cần lấy 2 lần đối xứng. Đối xứng qua OX rồi tiếp đến đối xứng qua đường phân giác.

  Xét điểm M(x,y). Đối xứng qua OX chúng ta được tọa độ mới là (x, -y). tiếp đến lấy đối xứng qua tia phân giác góc phần tư thứ nhất chúng ta được (-y, x). Hay nói cụ thể hơn là với một đoạn thẳng đầu vào có tọa độ là $A(x_a, y_a)$ và $B(x_b, y_b)$ thì đoạn thẳng trung gian của nó sẽ là CD với $C(-y_a , x_a)$ và $D(-y_b , x_b)$.

2.3. Môi trường thực nghiệm và các bước thiết lập cơ bản

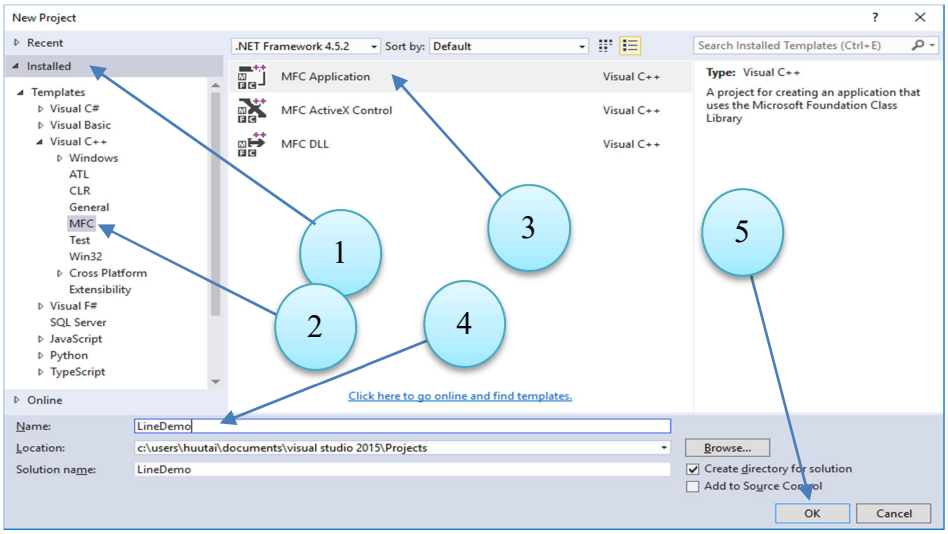
Chúng tôi đề xuất sinh viên thực hành thực nghiệm trên môi trường Microsoft Visual C++ sử dụng thư viện MFC (*Microsoft Foundation*

Class Library) để có thể dễ dàng tương tác sâu với thiết bị đồ họa và mang lại năng lực thực thi mạnh mẽ. Đồng thời nó cũng phù hợp với nền tảng kiến thức lập trình hướng đối tượng và kỹ năng lập trình với ngôn ngữ C++ mà sinh viên đã được trang bị trước khi đến với môn học này. Tuy nhiên sinh viên cũng cần phải trang bị và trau dồi thêm khả năng lập trình xử lý sự kiện trên giao diện đồ họa của Windows và một số kiến thức cơ bản về MFC.

Bài thực nghiệm số 1:

Dưới đây là các bước hướng dẫn để sinh viên có thể tiếp cận với quy trình xây dựng một ứng dụng đồ họa cơ bản, ứng dụng này bước đầu đặt nền tảng cho quá trình thực hành thực nghiệm, là quá trình quan trọng giúp người học có thể kiểm tra tính đúng đắn và tính thực tiễn của các lý thuyết đã được học thông qua kết quả thực nghiệm và các phân tích đánh giá, mang lại hiểu biết sâu sắc và đa chiều trên thực tiễn.

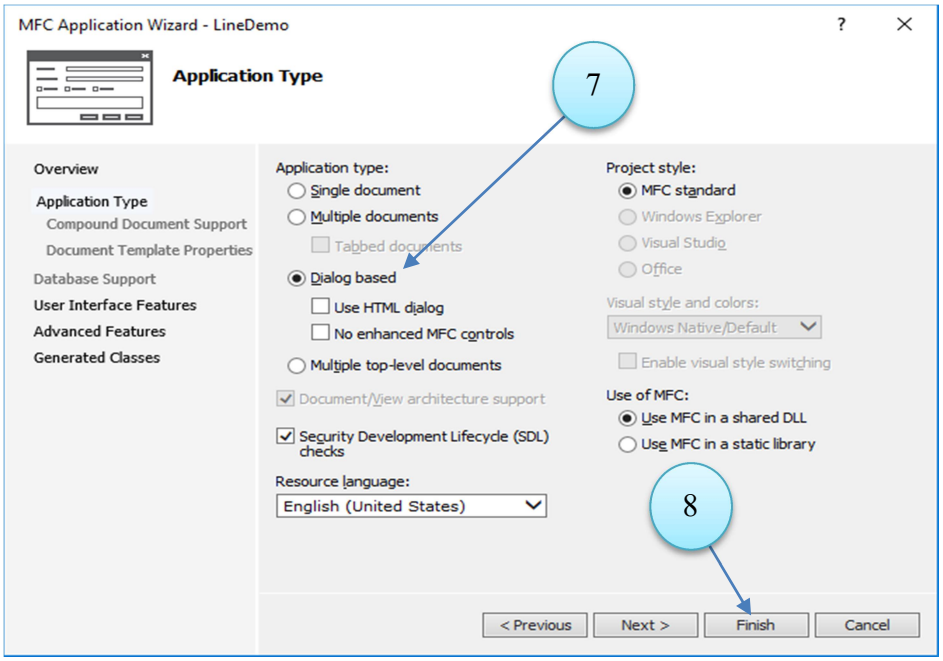
❖ Bước 1: Tạo một dự án (project) mới trong Microsoft Visual Studio sử dụng ngôn ngữ Visual C++ và thư viện MFC:



Hình 1.6. Các bước tạo một project phục vụ cho quá trình thực nghiệm

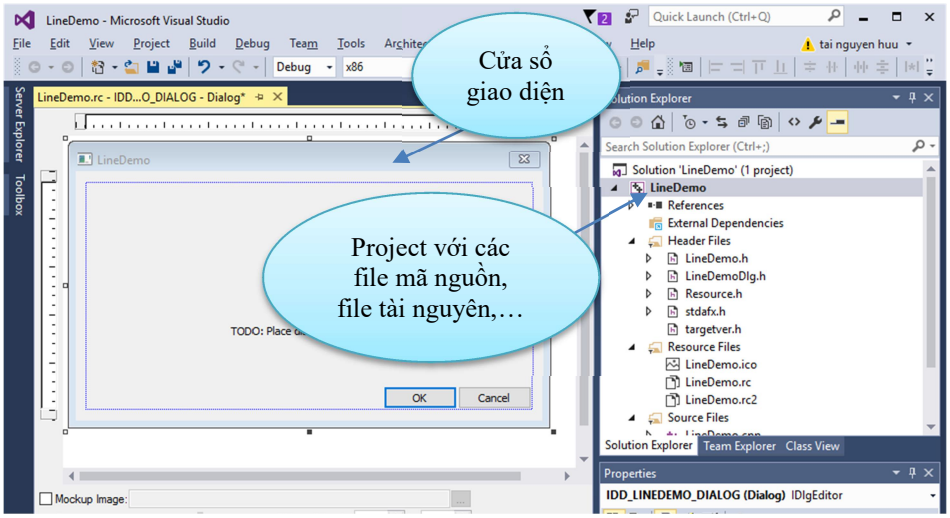
Khi hộp thoại MFC Application Wizard xuất hiện, cần bấm nút Next để chuyển đến mục Application Type. Tiếp đến click chọn mục "**Dialog**

based" như hình dưới đây, rồi cuối cùng bấm nút Finish để kết thúc quá trình tạo khung ứng dụng.



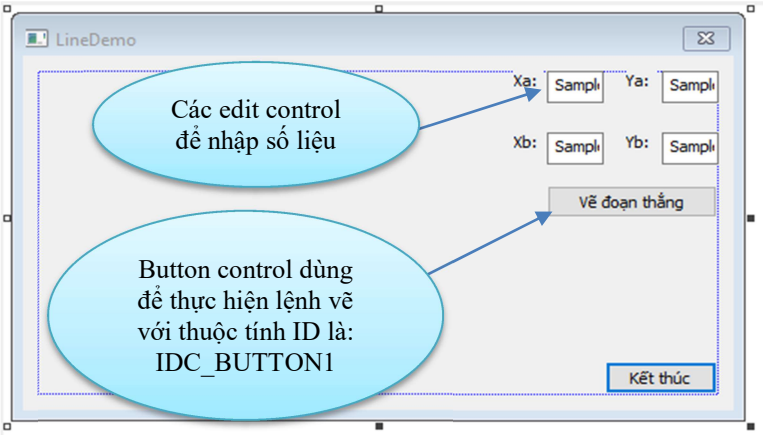
Hình 1.7. Giao diện MFC Application Wizard giúp chọn lựa kiểu ứng dụng

Kết thúc quá trình trên, Microsoft sẽ tạo ra một khung ứng dụng (template – Bản mẫu) với các thành phần và giao diện như hình dưới đây:



Hình 1.8. Hình ảnh của một ứng dụng dạng dialog based làm khuôn mẫu xây dựng các ứng dụng thực nghiệm đồ họa máy tính.

❖ Bước 2: Thiết kế lại giao diện

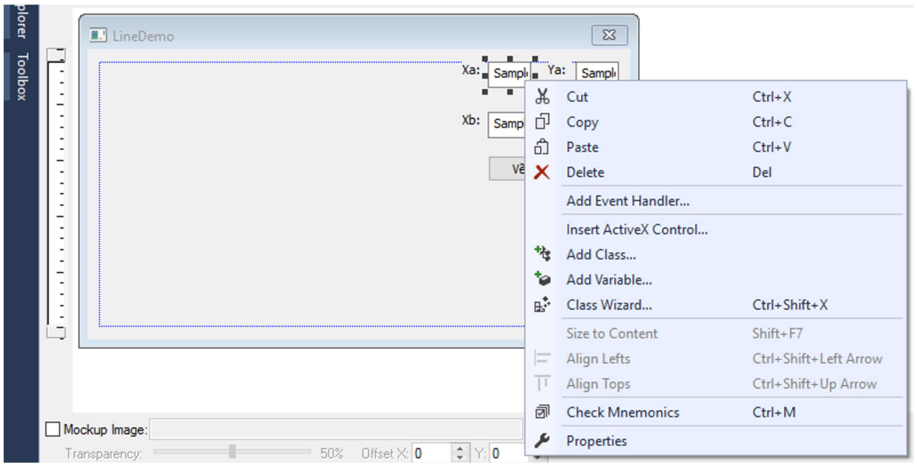


Hình 1.9. Thiết kế giao diện chương trình LineDemo

Tùy thuộc vào yêu cầu của bài toán mà chúng ta có những giải pháp thiết kế giao diện khác nhau. Hình 1.9 là thiết kế đơn giản cho bài toán dựng đoạn thẳng AB với các tọa độ được nhập vào thông qua các hộp nhập liệu (edit control):

❖ Bước 3: Tạo các biến nhận dữ liệu từ các edit control

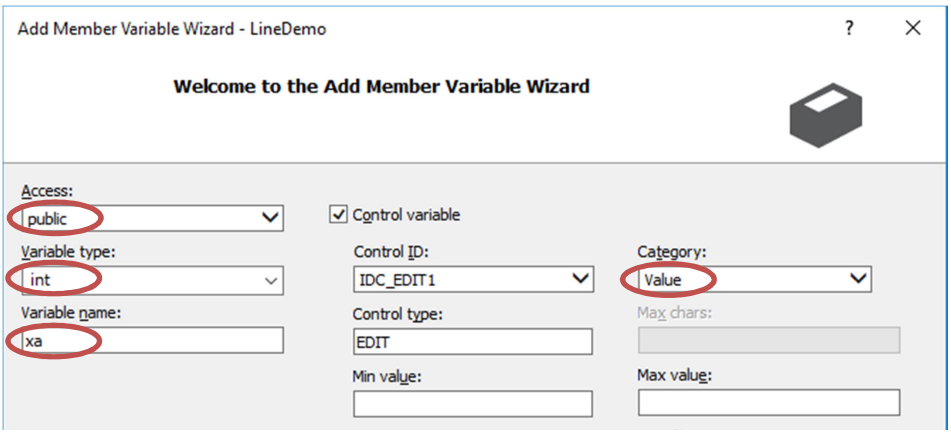
Click chuột phải vào edit control cần tạo biến nhận dữ liệu, tiếp đến chọn mục "Add variable…" trong menu ngữ cảnh.



Hình 1.10. Menu ngữ cảnh trong quá trình tạo biến nhận dữ liệu từ edit control

Sau đó cần xác định tên biến và các thông số cơ bản cho biến như trong *Hình 1.11*.

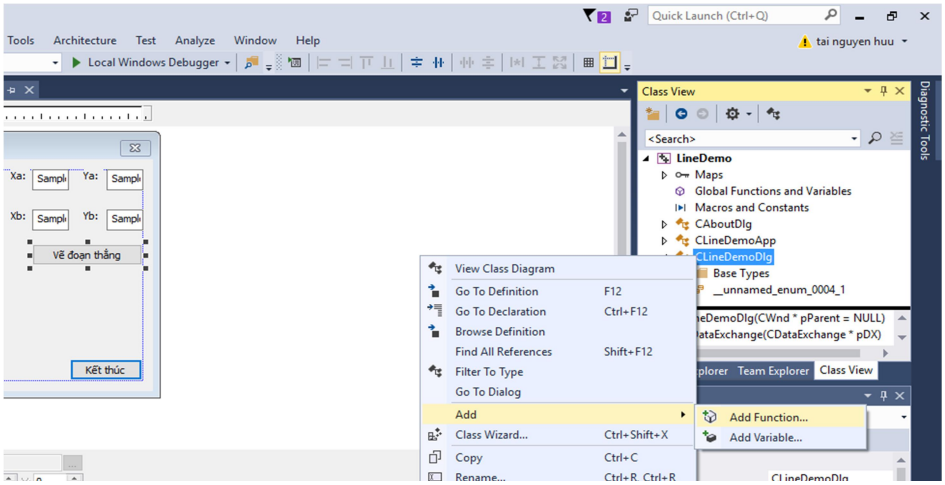
Thực hiện tương tự với các edit control còn lại để tạo các biến nhận dữ liệu tọa độ của đoạn thẳng AB, các biến gồm: xa, ya, xb, yb kiểu số nguyên.



Hình 1.11. Đặt tên và xác định các thông số cho biến

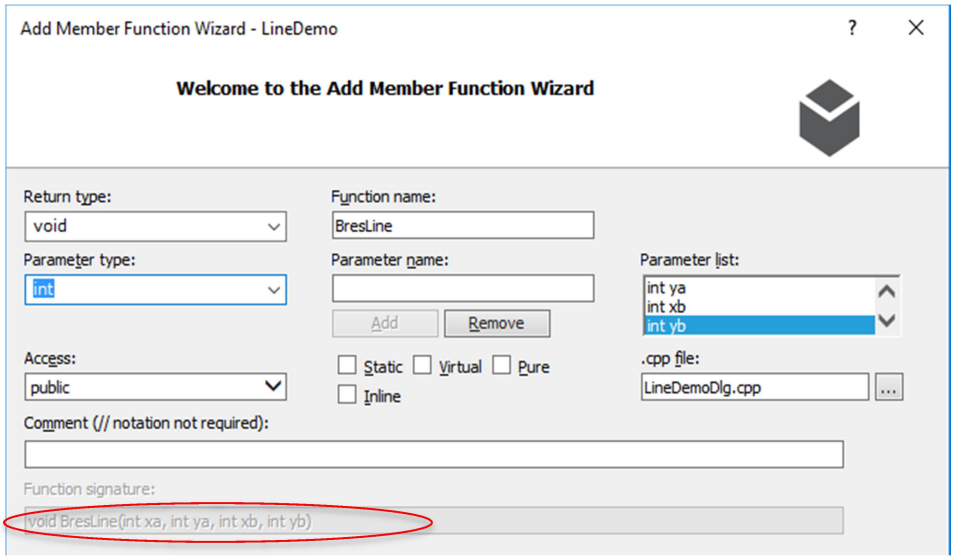
❖ Bước 4: Thêm hàm xử lý BresLine vào lớp CLineDemoDlg với chức

năng tính toán theo thuật giải Bresenham và vẽ hình trên cửa sổ giao diện.



Hình 1.12. Các bước để thêm một hàm xử lý vào lớp CLineDemoDlg

Đầu tiên chúng ta cần click chọn vào tab "Class View", tiếp đến click chuột phải vào mục CLineDemoDlg, rồi chọn mục Add → Add Function…



Hình 1.13. Xác định tên hàm và các tham số

Tiếp đến cần xác định tên hàm, kiểu trả về của hàm và danh sách các

tham số. Sau khi hoàn thành cần bấm nút **Finish** để chuyển sang công đoạn viết mã lệnh cho hàm.

❖ Bước 5: Viết mã lệnh thực thi cho hàm **BresLine**:

```
void CLineDemoDlg::BresLine(int xa, int ya, int xb, int yb)
{ /* Giả thiết đầu vào thỏa mãn hai điều kiện của giải thuật
Bresenham là hệ số góc a thuộc [0, 1] và xa < xb. Từ đó chúng ta
chỉ cần thực hiện theo đúng các bước đã được nêu ra trong giải
thuật.
  Để giải quyết với đầu vào tổng quát, sinh viên cần tham khảo thêm
phần hướng dẫn xử lý cho các tình huống hệ số góc a ngoài đoạn [0,
1]. Phần này được xem như một yêu cầu nâng cấp mã lệnh mà sinh viên
cần thực hiện */

  /* Lấy con trỏ quản lý đối tượng Device Context của cửa sổ giao
diện chương trình để có thể tiến hành vẽ các điểm ảnh trên cửa sổ
*/

   CDC *p_DC = this->GetDC();
   COLORREF Color = RGB(255, 0, 0); //Màu đỏ

   //Thực hiện Bước 0:

   int Dx = xb - xa, Dy = yb - ya;
   int P = 2 * Dy - Dx, Const1 = 2 * Dy, Const2 = 2 * Dy - 2 * Dx;
   int x = xa, y = ya;

   p_DC->SetPixel(x, y, Color);

   // Thực hiện Bước 2 (bước lặp):

   while (x < xb)  // Điều kiện dừng (Bước 3)
   {
      x++;
      if (P < 0)
      {
         P += Const1;
      }
      else
      {
         y++;
```

```
            P += Const2;
        }
        p_DC->SetPixel(x, y, Color);
    }
}
```

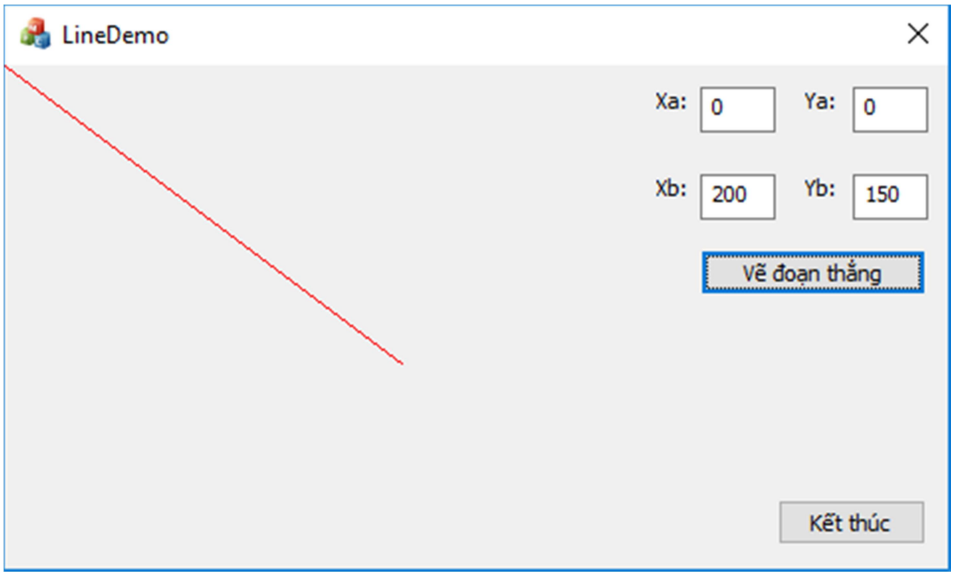
❖ Bước 6: Bấm DoubleClick vào nút "Vẽ đoạn thẳng" (tức nút có ID là IDC_BUTTON1) để chương trình MS Visual Studio tự động thêm vào một hàm đáp ứng sự kiện khi người sử dụng click (hay bấm chọn) vào nút "Vẽ đoạn thẳng" có dạng:

```
void CLineDemoDlg::OnBnClickedButton1()
{

}
```

Tiếp đến chúng ta cần viết mã lệnh cho hàm này như sau:

```
void CLineDemoDlg::OnBnClickedButton1()
{
   /* Gọi lệnh thực thi việc cập nhật dữ liệu vào các biến xa, ya, xb, yb */
   UpdateData(true);

   // Gọi hàm vẽ đoạn thẳng theo giải thuật Bresenham đã cài đặt
   BresLine(xa, ya, xb, yb);
}
```

❖ Bước 7: Biên dịch và thực thi chương trình, nhập tọa độ của AB vào các edit control rồi click nút "Vẽ đoạn thẳng" sẽ thấy được hình ảnh biểu diễn cho đoạn thẳng AB được hiện thị trên cửa sổ như hình dưới đây:



Hình 1.14. Kết quả thực thi chương trình với hình ảnh biểu diễn cho một đoạn thẳng AB được tính toán theo giải thuật Bresenham.

2.4. So sánh đánh giá hai giải thuật dựng đường thẳng

Sau đây chúng ta sẽ tiến hành so sánh và đánh giá hai giải thuật dựng đoạn thẳng trên một số tiêu chí quan trọng:

1. **Yêu cầu về phần cứng thực thi tính toán:**
   Rõ ràng rằng hai giải thuật có hai yêu cầu khác nhau về phần cứng thực thi tính toán. Cụ thể:

- Giải thuật vẽ đoạn thẳng dựa vào phương trình yêu cầu tính toán với kiểu dữ liệu số thực, vì vậy một hệ thống đồ họa nếu sử dụng giải thuật này sẽ cần có vi xử lý hỗ trợ tính toán số thực, yêu cầu này không phải khi nào cũng được đáp ứng. Thực tế cho thấy, rất nhiều hệ thống thiết bị trong công nghiệp cho đến giải trí và gia dụng có màn hình hiển thị đồ họa cấp thấp (độ phân giải thấp với nhiệm vụ hiển thị các thông tin đơn giản), được xây dựng trên các vi xử lý số nguyên 8-bit, 16-bit, hay 32-bit nhằm đáp ứng yêu cầu về giá thành. Vậy nên các hệ thống này không thể đáp ứng được yêu cầu của giải thuật, hoặc chỉ đáp ứng miễn cưỡng theo một cách phức tạp hơn đó là xây dựng các modul phần mềm hỗ trợ xử

lý số thực dựa trên các lệnh tính toán số nguyên. Ngay cả vi xử lý máy tính 16-bít của hãng Intel có mã hiệu là 8086 được thiết kế trong giai đoạn 1976-1978, *và trang bị trên các máy IBM PC*, có kiến trúc bộ xử lý số học và logic (Arithmetic Logic Unit, hay ALU) chỉ thực hiện các tính toán trên trường số nguyên, vì vậy các tính toán số thực trên máy tính giai đoạn đó phải được thực hiện thông qua giải pháp phần mềm mất nhiều thời gian thực hiện.

- Giải thuật vẽ đoạn thẳng dựa vào phương trình chỉ yêu cầu các tính toán căn bản trên trường số nguyên gồm: so sánh, cộng, trừ, và phép dịch bít để thực hiện phép nhân 2. Những yêu cầu tính toán này là hết sức căn bản và có thể thực hiện được trên hầu hết các hệ thống vi xử lý.

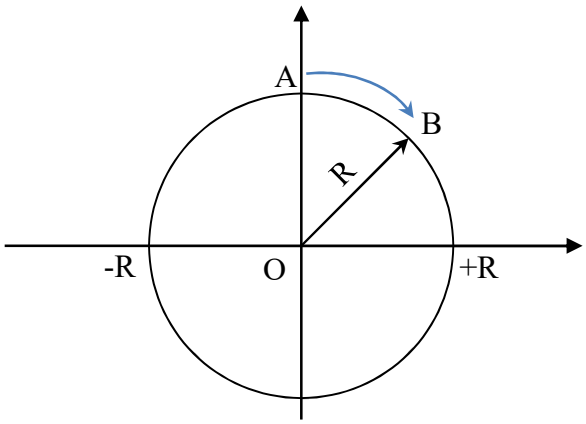
2. Số lệnh máy cần thực hiện để có thể tính toán được tọa độ điểm ảnh

Giải thuật vẽ đoạn thẳng dựa vào phương trình	**Giải thuật Bresenham vẽ đoạn thẳng**
Tính toán trong phần khởi động	
$\Delta x = x_b - x_a$ $\Delta y = y_b - y_a$ $a = \Delta y / \Delta y$ $b = -a * x_a + y_a$ $x = x_a$ $y = y_a$	$\Delta x = x_b - x_a$ $\Delta y = y_b - y_a$ $Const1 = 2\Delta y$ $Const2 = 2\Delta y - 2\Delta x$ $P = 2\Delta y - \Delta x$ $x = x_a$ $y = y_a$
Nhận xét: Số lệnh máy cần thực hiện ở giai đoạn khởi động của cả hai giải thuật không khác nhau đáng kể. Mặt khác, do bước khởi động chỉ thực hiện một lần, nên góp phần không đáng kể vào chi phí thời gian tính toán của toàn bộ quy trình thực thi của giải thuật.	
Tính toán trong vòng lặp để thu được tọa độ một điểm ảnh minh họa	

x = x +1 y = int (a.x + b + 0.5)	x = x +1 Nếu P < 0: P = P + Const1 Ngược lại: y = y + 1 P = P + Const2
Tổng cộng: - 3 lệnh cộng số thực - 1 lệnh nhân số thực - 1 lệnh lấy phần nguyên của số thực	Tổng cộng trong tình huống xấu nhất: - 3 lệnh cộng số nguyên - 1 lệnh chuyển hướng thực thi (hay lệnh nhảy). Tổng cộng trong tình huống tốt nhất: - 2 lệnh cộng số nguyên - 1 lệnh chuyển hướng thực thi (hay lệnh nhảy).
Nhận xét: Ở đây chúng ta chỉ so sánh mang tính tương đối, vì 2 giải thuật có yêu cầu nền tảng tính toán cùng các lệnh máy khác nhau nên sẽ rất khác nhau khi thực hiện trên các nền tảng kiến trúc vi xử lý khác nhau. Nhưng nhìn chung giải thuật dựa vào phương trình ngoài yêu cầu phải thực hiện trên dữ liệu số thực, nó còn nhiều hơn một lệnh nhân số thực, vì thế khi đoạn thẳng càng dài, kéo theo số lần lặp càng lớn thì hiệu quả cải thiện về thời gian của giải thuật Bresenham càng cao.	

Từ những so sánh và đánh giá trên cho thấy, chúng ta nên lựa chọn giải thuật Bresenham để cài đặt trên các hệ thống đồ họa để mang lại hiệu quả thực thi tốt nhất, còn giải thuật dựng đường tròn dựa vào phương trình chỉ mang tính tham khảo và so sánh.

3. CÁC GIẢI THUẬT VẼ ĐƯỜNG TRÒN



Hình 1.15. Đồ thị toán học của hình tròn tâm O bán kính R

Phương trình đường tròn tâm O (gốc toạ độ) bán kính R (nguyên) là:

$$X^2 + Y^2 = R^2$$

Trong mục này chúng ta chỉ cần tìm phương pháp vẽ đường tròn tâm tại gốc toạ độ. Nếu chúng ta vẽ được đường tròn tâm tại gốc toạ độ thì bằng cách thêm vào phép tịnh tiến chúng ta được đường tròn tâm (x,y) bất kỳ.

Dễ thấy, để vẽ được đường tròn tâm gốc toạ độ chúng ta chỉ cần tìm phương pháp vẽ cung một phần tám AB như trên Hình 1.15, kết hợp với các phép lấy đối xứng chúng ta sẽ có các phần còn lại của đường tròn.

Với cung AB thì rõ ràng độ dốc của nó thuộc đoạn [-1, 0]. Điều này chúng ta có thể dễ dàng thấy qua góc của tiếp tuyến với cung AB hay qua đạo hàm phương trình biểu diễn cung AB.

Vì cung AB có độ dốc trong khoảng [-1, 0], nên chúng ta suy ra rằng trên toàn bộ cung AB khi biến số x tăng thì biến số y giảm, và tốc độ thay đổi của y chậm hơn của x. Từ đây chúng ta có thể đề ra một quy trình dựng cung AB nhằm đảm bảo tính liên tục (hay nối tiếp, liền kề) của các pixel được chọn, cụ thể:

- Cho biến số x nhận lần lượt các giá trị nguyên từ x_a đến x_b. Với mỗi giá trị x chúng ta thực hiện:

- Tìm giá trị y nguyên tương ứng để pixel có toạ độ (x,y) sẽ là điểm gần nhất với điểm có tọa độ thực (x, y_{circle}) vốn thuộc đường tròn về mặt toán học.
- Vẽ pixel (x,y) tìm được và các đối xứng của nó để có được đường tròn

Trong mục này chúng ta sẽ đi tìm hiểu 2 giải thuật cho phép dựng đường tròn (*thực chất là dựng cung AB và các đối xứng của nó*) một cách hiệu quả về mặt tốc độ tính toán.

3.1. Giải thuật vẽ đường tròn MidPoint

Giải thuật MidPoint hay còn gọi là giải thuật xét điểm giữa.

- Điểm ảnh (hay Pixel) đầu tiên được chọn để dựng cung AB sẽ chính là điểm A(0,R), nghĩa là điểm chọn đầu tiên của quy trình dựng hình với tên gọi (x_0,y_0) sẽ có tọa độ (0,R). Nói cách khác chúng ta có:

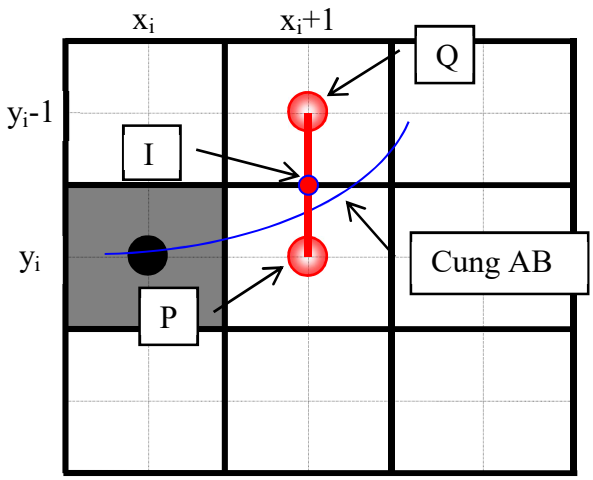
$$(x_0,y_0) = (0,R)$$

- Giả sử ở bước thứ i chúng ta đã xác định được giá trị điểm ảnh cần vẽ của bước này là (x_i,y_i), hay nói cách khác là điểm vẽ thứ i là (x_i,y_i) đã được xác định giá trị. Câu hỏi đặt ra là đến bước thứ i+1 chúng ta sẽ chọn điểm vẽ thứ (i+1) như thế nào? Hay nói theo một cách khác là điểm chọn thứ (i+i) với tên gọi là (x_{i+1},y_{i+1}) có giá trị bằng bao nhiêu?

Vì điểm ảnh chọn lựa tiếp theo của quy trình dựng hình phải được chọn lựa trên cơ sở tuân thủ quy tắc đã đề ra ở cuối mục 3, nên hoành độ x của nó phải tăng một giá trị so với giá trị của điểm chọn trước đó, hay nói cách khác là:

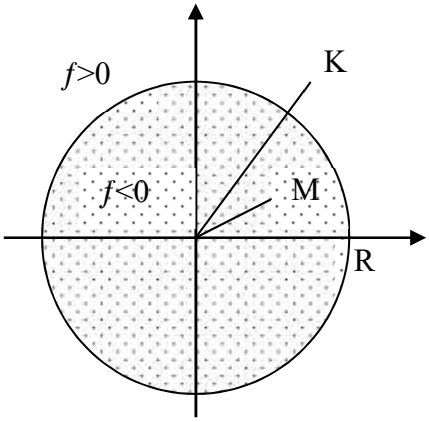
$$x_{i+1} = x_i + 1$$

Đồng thời vì trên cung AB (*xem Hình 1.16*) khi x tăng thì y giảm và tốc độ thay đổi của y chậm hơn của x, vì thế với giá trị x tăng 1 đơn vị (hay một cột) thì giá trị y sẽ thay đổi một lượng Δy với ràng buộc $-1 \leq \Delta y \leq 0$. Mà điểm ảnh chọn bước trước là (x_i,y_i) nên điểm ảnh chọn ở bước tiếp theo chỉ có thể là một trong hai điểm ảnh P(x_i+1, y_i) và Q(x_i+1, y_i - 1).



Hình 1.16. Minh họa việc chọn lựa điểm P hay Q dựa vào các tham số khi dựng cung AB trên màn hình

Để quyết định được điểm chọn là P hay Q chúng ta hướng đến một biểu thức mà dấu của nó cho phép chúng ta ra quyết định chọn điểm nào.



Hình 1.17. Minh họa hàm f_{circle}

Trước hết chúng ta xét hàm (xem Hình 1.17):

$$f_{circle}(x, y) = x^2 + y^2 - R^2$$

Với một điểm M(x,y) thì rõ ràng chúng ta có:

- $f_{circle}(M) = f(x,y) = x^2 + y^2 - R^2 < 0$ khi và chỉ khi điểm M nằm trong

đường tròn *(tâm O bán kính R)*

- $f_{circle}(M) = f(x,y) = x^2 + y^2 – R^2 > 0$ khi và chỉ khi điểm M nằm ngoài đường tròn *(tâm O bán kính R)*
- $f_{circle}(M) = f(x,y) = x^2 + y^2 – R^2 = 0$ khi và chỉ khi điểm M nằm trên đường tròn

Từ kết quả trên, nếu chúng ta gọi I là trung điểm của PQ thì $I(x_i + 1,y_i - 0.5)$ và:

$$\begin{aligned} \text{Đặt } P_i &= f_{circle}(I) = f_{circle}(x_i +1,y_i - 0.5) \\ &= (x_i +1)^2 + (y_i - 0.5)^2 – R^2 \end{aligned} \qquad (1.6)$$

- Khi $P_i = f_{circle}(I) < 0$ thì điểm I nằm trong đường tròn *(tâm O bán kính R)*, vì thế điểm P sẽ gần với đường tròn hơn điểm Q. Suy ra điểm P sẽ được lựa chọn làm điểm ảnh biểu diễn cho cung AB ở bước thứ i+1.
- Khi $P_i = f_{circle}(I) > 0$ thì điểm I nằm ngoài đường tròn, vì thế điểm Q sẽ gần với đường tròn hơn điểm P. Suy ra điểm Q sẽ được lựa chọn làm điểm ảnh biểu diễn cho cung AB ở bước thứ i+1.
- Khi $P_i = f_{circle}(I) = 0$ thì điểm I nằm trên đường tròn, suy ra khả năng lựa chọn P và Q là như nhau, song chúng ta phải quyết định chọn một điểm. Trong tình huống này giải thuật quy định chọn điểm Q.

Từ đây chúng ta thấy, dấu của biểu thức P_i có thể được sử dụng như một hàm ra quyết định cho việc lựa chọn điểm ảnh điểm tiếp trong quá trình dựng hình.

Để giải thuật được đơn giản người ta tối ưu hoá việc tính P_i theo công thức truy hồi, cụ thể:

$$P_{i+1} = f(x_{i+1} +1,y_{i+1} - 0.5) = (x_{i+1} + 1)^2 + (y_{i+1} - 0.5)^2 – R^2 \qquad (1.7)$$

Dấu của P_i sẽ quyết định giá trị P_{i+1} như sau:

- Nếu $P_i < 0$: thì điểm chọn tiếp theo là $P(x_i +1,y_i)$, điều đó có nghĩa là $(x_{i+1},y_{i+1}) = (x_i +1,y_i)$. Thay vào (1.7) chúng ta thu được:

$$P_{i+1} = (x_i+1+1)^2 + (y_i -0.5)^2 – R^2 = P_i + 2(x_i+1)+1 = P_i + 2x_i +3$$

- Nếu $P_i \geq 0$: thì điểm chọn tiếp theo là $Q(x_i+1,y_i-1)$, điều đó có nghĩa là $(x_{i+1},y_{i+1}) = (x_i+1,y_i - 1)$. Thay vào (1.7) chúng ta được:

$$P_{i+1} = (x_i+1+1)^2 + (y_i -1 - 0.5)^2 - R^2 = P_i + 2(x_i+1)+1-2(y_i-0.5)+1$$

$$= P_i + 2(x_i - y_i) +5$$

Đầu tiên chúng ta chọn điểm A(0,R), nghĩa là $(x_0,y_0) = (0,R)$, thay vào (1.6) chúng ta có:

$$P_0 = 1 - R + \frac{1}{4} = \frac{5}{4} - R$$

Từ đây quy trình dựng hình có thể được thực hiện theo logic sau:

- Tính P_0, vẽ điểm $(x_0,y_0) = (0,R)$
- Dựa vào dấu của P_0 chúng ta lại chọn được điểm vẽ tiếp theo (x_1,y_1) và giá trị P_1
- Dựa vào dấu của P_1 chúng ta lại chọn được điểm vẽ tiếp theo (x_2,y_2) và giá trị P_2
- Quá trình trên được lặp đi lặp lại cho đến khi chúng ta vẽ được điểm nguyên gần nhất với B.

❖ Một điểm đáng chú ý ở đây các giá trị P tiếp theo có được bằng cách cộng với giá trị P trước đó với một lượng nguyên $2x_i +3$ hoặc $2(x_i - y_i) +5$ tuỳ theo dấu của P. Song nếu giá trị P khởi đầu là $P_0 = 1 - R + \frac{1}{4}$ là một giá trị thực sẽ làm cho việc tính các giá trị P tiếp theo cũng phải xử lý trên trường số thực. Một điều dễ thấy là nếu chúng ta thay đổi giá trị P_0 khởi đầu là 1-R thì dấu của P_0 và các P_i có được sau đó không hề thay đổi (mặt dù giá trị có bị giảm một lượng 0.25), dẫn đến kết quả chọn lựa các điểm ảnh của giải thuật không hề bị thay đổi, nhưng các tính toán giá trị P chỉ cần thực hiện trên trường số nguyên, điều này giúp giảm độ phức tạp tính toán nếu xét trên khía cạnh thực hiện bằng phần mềm (software) hay giảm yêu cầu về kiến trúc phần cứng nếu xét trên khía cạnh thực hiện bằng phần cứng (Hardware)

3.1.1. Tóm tắt giải thuật vẽ đường tròn MidPoint

- **Bước 1:** (Bước khởi động, tính toán các giá trị ban đầu)

  $P_0 = 1 - R$; $(x_0,y_0) = (0,R)$

  Vẽ điểm (x_0,y_0)

- **Bước 2:** (Bước lặp, thực hiện tính các giá trị điểm ảnh)

  Với mỗi giá trị i (i=0,1,2,…) chúng ta xét dấu P_i

  - Nếu $P_i<0$: thì chọn điểm tiếp theo là

    $(x_{i+1},y_{i+1}) = (x_i+1,y_i)$

    $P_{i+1}=P_i+ 2x_i +3$

  - Ngược lại (tức $P_i \geq 0$): thì chọn điểm tiếp theo là

    $(x_{i+1},y_{i+1}) = (x_i+1,y_i -1)$

    $P_{i+1} = P_i + 2(x_i - y_i) +5$

  Vẽ điểm (x_{i+1},y_{i+1}) vừa tìm được

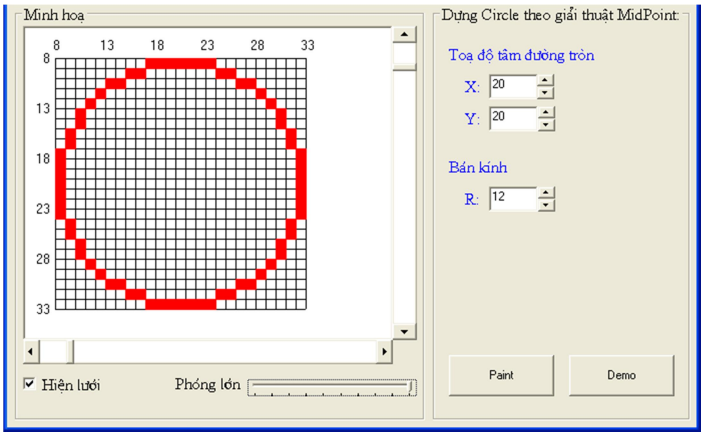
- **Bước 3:** (Xác định điều kiện lặp)

  Lặp lại bước 2 với những giá trị i tiếp theo, cho đến khi chúng ta vẽ được điểm nguyên gần nhất với B, nghĩa là $x_{i+1} = round(x_b) = Round(\frac{R}{\sqrt{2}})$ thì giải thuật kết thúc.

3.1.2. Cài đặt

❖ Sinh viên cần xây dựng hàm vẽ đường tròn theo giải thuật đã trình bày trên. Trên cơ sở đó mở rộng chương trình LineDemo với chức năng mới là vẽ đường tròn.

❖ Nâng cấp chương trình với khả năng vẽ phóng lớn, với thiết kế giao diện như hình dưới đây:



Hình 1.18. Minh họa việc hiển thị các điểm ảnh của đường tròn với các kích cỡ khác nhau

3.2. Giải thuật vẽ đường tròn Bresenham

Lập luận tương tự giải thuật trên song không dùng hàm f_{circle} mà dùng biểu thức $(d_1 - d_2)$. Thuật giải được trình bày chi tiết như sau:

- Điểm ảnh đầu tiên được chọn để vẽ sẽ là điểm A(0, R), nghĩa là: $(x_0, y_0)=(0, R)$
- Giả sử đến bước thứ i chúng ta đã chọn được điểm ảnh (x_i, y_i) để vẽ. Câu hỏi đặt ra là đến bước thứ i+1 chúng ta sẽ chọn điểm ảnh có tên gọi (x_{i+1}, y_{i+1}) với giá trị bằng bao nhiêu?

Vì điểm ảnh tiếp theo sẽ chọn theo quy tắc đã nói trên sẽ có hoành độ x tăng một giá trị so với giá trị của điểm chọn trước, hay nói cách khác là:

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

Đồng thời vì trên cung AB khi x tăng thì y giảm và tốc độ thay đổi của y chậm hơn của x, nên rõ ràng chúng ta thấy rằng với giá trị x tăng 1 thì giá trị y sẽ thay đổi một lượng Δy với điều kiện ràng buộc là $-1 \leq \Delta y \leq 0$. Mà điểm ảnh chọn bước trước là (x_i,y_i) nên điểm ảnh chọn tiếp theo chỉ có thể là một trong hai điểm $P(x_i + 1, y_i)$ và $Q(x_i + 1, y_i - 1)$.

Để quyết định được điểm ảnh cần chọn là P hay Q chúng ta hướng đến một biểu thức mà dấu của nó cho phép chúng ta ra quyết định chọn

điểm nào.

Đặt: $d_1 = y_P^2 - y^2$ và $d_2 = y^2 - y_Q^2$

giá trị y ở đây chính là tung độ của cung AB ứng với hoành độ $x = x_i + 1$

$$\text{Đặt: } P_i = d_1 - d_2 = y_P^2 + y_Q^2 - 2y^2 = y_i^2 + (y_i-1)^2 - 2(R^2 - x_{i+1}^2)$$
$$= y_i^2 + (y_i-1)^2 - 2(R^2 - (x_i+1)^2) = y_i^2 + (y_i-1)^2 - 2R^2 + 2(x_i+1)^2 \quad (1.8)$$

Dấu của biểu thức P_i cho phép xác định điểm ảnh được chọn tiếp theo là P hay Q.

- Khi $P_i < 0$: thì điểm ảnh P sẽ gần với đường tròn hơn điểm ảnh Q, do đó chúng ta sẽ chọn điểm ảnh P làm điểm biểu diễn (vẽ).
- Khi $P_i > 0$: thì điểm ảnh Q sẽ gần với đường tròn hơn P, do đó chúng ta sẽ chọn điểm ảnh Q làm điểm biểu diễn.
- Khi $P_i = 0$: khoảng cách từ P và Q đến đường tròn đều bằng nhau, nên chúng ta có thể chọn P hay Q đều được. Trong tình huống này giải thuật quy ước chọn điểm ảnh Q làm điểm biểu diễn.

Vậy từ đây chúng ta thấy có thể dựa vào dấu của biểu thức P_i để ra quyết định chọn điểm tiếp theo.

Để giải thuật được đơn giản người ta tối ưu hoá việc tính P_i theo công thức truy hồi:

$$P_{i+1} = y_{i+1}^2 + (y_{i+1}-1)^2 - 2R^2 + 2(x_{i+1}+1)^2 \quad (1.9)$$

Dấu của P_i sẽ quyết định giá trị P_{i+1} cụ thể như sau:

- Nếu $P_i < 0$: thì điểm ảnh chọn tiếp theo là $P(x_i+1, y_i)$, nghĩa là $(x_{i+1},y_{i+1}) = (x_i + 1,y_i)$.

  Thay vào (1.9) chúng ta được:

  $P_{i+1} = y_i^2 + (y_i-1)^2 - 2R^2 + 2((x_i+1)+1)^2 = P_i + 2(2(x_i+1)+1)$

  $= P_i + 4x_i + 6$

- Nếu $P_i \geq 0$: thì điểm ảnh chọn tiếp theo là $Q(x_i+1, y_i-1)$, nghĩa là:

$$(x_{i+1},y_{i+1}) = (x_i + 1,y_i - 1).$$

Thay vào (1.9) chúng ta được:

$$P_{i+1} = (y_i -1)^2 + (y_i-2)^2 - 2R^2 + 2((x_i +1)+1)^2$$

$$= P_i +(-4y_i+4) + 2(2(x_i+1)+1) = P_i + 4(x_i - y_i) +10$$

Đầu tiên chúng ta chọn điểm ảnh A(0,R), nghĩa là $(x_0,y_0) = (0,R)$, thay vào (1.8) chúng ta có:

$$P_0 = y_0^2 + (y_0-1)^2 - 2R^2 + 2(x_0 +1)^2 = R^2 + (R-1)^2 -2R^2 + 2$$

$$= R^2 + R^2 -2R +1 -2R^2 + 2 = 3 - 2R$$

Vậy quy trình vẽ được thực hiện như sau:

- Tính P_0, vẽ điểm ảnh $(x_0,y_0) = (0,R)$
- Dựa vào dấu của P_0 chúng ta lại chọn được điểm vẽ tiếp theo (x_1,y_1) và giá trị P_1
- Dựa vào dấu của P_1 chúng ta lại chọn được điểm vẽ tiếp theo (x_2,y_2) và giá trị P_2
- Quá trình trên được lặp đi lặp lại cho đến khi chúng ta vẽ được điểm ảnh nguyên gần nhất với B.

3.2.1. Tóm tắt giải thuật vẽ đường tròn Bresenham :

- **Bước 1:** (Bước khởi động, tính toán các giá trị ban đầu)

  $P_0 = 3 - 2R$; $(x_0,y_0) = (0,R)$

  Vẽ điểm (x_0,y_0)

- **Bước 2:** (Bước lặp, thực hiện tính các giá trị điểm ảnh)

  Với mỗi giá trị i (i=0,1,2,…) chúng ta xét dấu P_i

  - Nếu $P_i<0$: thì chọn điểm tiếp theo là

    $(x_{i+1},y_{i+1}) = (x_i+1,y_i)$

    $P_{i+1} = P_i+ 4x_i +6$

➢ Ngược lại (tức $P_i \geq 0$): thì chọn điểm tiếp theo là

$(x_{i+1},y_{i+1}) = (x_i+1,y_i -1)$

$P_{i+1} = P_i + 4(x_i - y_i) +10$

Vẽ điểm (x_{i+1},y_{i+1}) vừa tìm được

- **Bước 3:** (Xác định điều kiện lặp)

  Lặp lại bước 2 với những giá trị i tiếp theo, cho đến khi chúng ta vẽ được điểm nguyên gần nhất với B, nghĩa là $x_{i+1} = Round(x_b) = Round(\frac{R}{\sqrt{2}})$ thì giải thuật kết thúc.

3.2.2. Cài đặt

❖ Sinh viên cần cài đặt một hàm vẽ đường tròn theo giải thuật Bresenham và chương trình sử dụng hàm để vẽ các đường tròn ngẫu nhiên.

3.3. So sánh đánh giá hai giải thuật dựng đường tròn

Sau đây chúng ta sẽ tiến hành so sánh và đánh giá hai giải thuật dựng đường tròn trên một số tiêu chí quan trọng:

1. **Yêu cầu về phần cứng thực thi tính toán:**
   Rõ ràng rằng cả hai giải thuật MidPoint và Bresenham đều chỉ yêu cầu tính toán trên trường số nguyên, vì vậy yêu cầu về phần cứng thực thi tính toán của chúng là như nhau chứ không có khác biệt.
2. **Số lệnh máy cần thực hiện để có thể tính toán được tọa độ điểm ảnh**

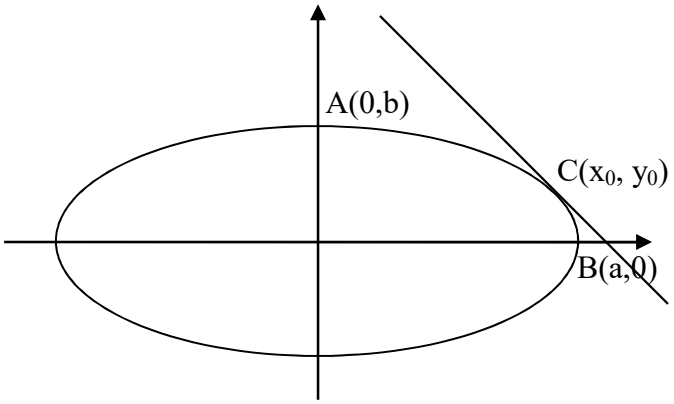
Giải thuật vẽ đường tròn MidPoint	**Giải thuật vẽ đường tròn Bresenham**
Tính toán trong phần khởi động	
P = 1 - R x = 0	P = 3 - 2R x = 0

<table>
<tr><td>y = R</td><td>y = R</td></tr>
<tr><td colspan="2">Nhận xét: Số lệnh máy cần thực hiện ở giai đoạn khởi động của cả hai giải thuật không khác nhau đáng kể. Mặt khác, do bước khởi động chỉ thực hiện một lần, nên góp phần không đáng kể vào chi phí thời gian tính toán của toàn bộ quy trình thực thi của giải thuật.</td></tr>
<tr><td colspan="2">Tính toán trong vòng lặp để thu được tọa độ một điểm ảnh minh họa</td></tr>
<tr><td>x = x +1
Nếu P < 0:
P = P + 2x +3
Ngược lại:
y = y - 1
P = P + 2(x - y) + 5</td><td>x = x +1
Nếu P < 0:
P = P + 4x +6
Ngược lại:
y = y - 1
P = P + 4(x - y) + 10</td></tr>
<tr><td>❖ Tổng cộng trong tình huống xấu nhất:
- 3 lệnh cộng số nguyên
- 2 lệnh trừ số nguyên
- 1 lệnh nhân số nguyên
- 1 lệnh chuyển hướng thực thi (hay lệnh nhảy).
❖ Tổng cộng trong tình huống tốt nhất:
- 3 lệnh cộng số nguyên
- 1 lệnh nhân số nguyên
- 1 lệnh chuyển hướng thực thi (hay lệnh nhảy).</td><td>❖ Tổng cộng trong tình huống xấu nhất:
- 3 lệnh cộng số nguyên
- 2 lệnh trừ số nguyên
- 1 lệnh nhân số nguyên
- 1 lệnh chuyển hướng thực thi (hay lệnh nhảy).
❖ Tổng cộng trong tình huống tốt nhất:
- 3 lệnh cộng số nguyên
- 1 lệnh nhân số nguyên
- 1 lệnh chuyển hướng thực thi (hay lệnh nhảy).</td></tr>
<tr><td colspan="2">Nhận xét: Dễ thấy rằng cả hai giải thuật là giống nhau về các lệnh và số lệnh cần thực thi.</td></tr>
</table>

Qua kết quả so sánh trên có thể giúp chúng ta đi đến kết luận rằng cả hai giải thuật là tương đương nhau trên các tiêu chí đánh giá. Vì vậy,

chúng ta có thể chọn lựa một trong hai giải thuật trên để cài đặt trên các hệ thống đồ họa, và chúng mang lại hiệu quả thực thi ngang nhau.

4. GIẢI THUẬT VẼ ELLIPSE



Hình 1.19. Hình Ellipse với các cung AC và BC

Phương trình chính tắc của Ellipse có dạng:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \qquad (1.10)$$

Để dựng được ellipse rõ ràng là chúng ta chỉ cần tìm cách dựng cung AB, còn các phần còn lại dễ dàng có được bằng cách lấy đối xứng. Song với tư tưởng chung để dựng một đường bất kỳ là cần phải xác định ra các miền mà trên toàn miền đó một biến số biến thiên nhanh hơn một biến số khác.

Rõ ràng trên cung AB thì độ dốc giảm liên tục từ điểm A (độ dốc bằng 0) đến B (độ dốc tiến đến $-\infty$). Xét về tốc độ biến thiên của 2 biến số thì:

- Tốc độ biến thiên của biến số X giảm dần từ A đến B.
- Tốc độ biến thiên của biến số Y tăng dần từ A đến B.

Rõ ràng trên cung AB phải có một điểm mà tại đó tốc độ biến thiên của X và Y là bằng nhau (song x tăng thì y giảm), đó chính là điểm mà tại đó có độ dốc bằng -1.

Gọi C(x_0,y_0) là điểm nằm trên cung AB của ellipse mà tiếp tuyến tại đó có độ dốc bằng -1. Khi đó tiếp tuyến d của ellipse sẽ có dạng:

$$\frac{x.x_0}{a^2} + \frac{y.y_0}{b^2} = 1$$

Mặt khác do độ dốc của d là: $-\frac{x_0 b^2}{y_0 a^2} = -1$ nên chúng ta suy ra

$$y_0 = \frac{b^2}{a^2} x_0 \text{ (ở đây chúng ta có } x_0 \geq 0) \Rightarrow y_0^2 = \frac{b^4}{a^4} x_0^2 \qquad (1.11)$$

Đồng thời do C thuộc ellipse nên chúng ta có:

$$\frac{x_0^2}{a^2} + \frac{y_0^2}{b^2} = 1 \qquad (1.12)$$

Từ (1.11) và (1.12) chúng ta suy ra:

$$x_0 = \frac{a^2}{\sqrt{a^2 + b^2}} \text{ và } y_0 = \frac{b^2}{\sqrt{a^2 + b^2}}$$

Chúng ta sẽ trình bày giả thuật để vẽ cung AC (Đi từ A đến C theo chiều kim đồng hồ). Cung CB được thực hiện một cách tương tự khi chúng ta đổi vai trò của x và y. Các phần còn lại của ellipse có được bằng cách lấy đối xứng.

Trên cung AC độ dốc nằm trong đoạn [0, -1], nghĩa là x tăng thì y giảm và tốc độ biến thiên của x lớn hơn của y. Vậy nên tư tưởng của thuật giải dựng cung AC sẽ là cho tham số x biến thiên tuần tự từ x_a đến x_c với các giá trị nguyên, và với mỗi giá trị x như vậy chúng ta tìm một giá trị y nguyên gần nhất với giá trị y thực của ellipse.

Sau đây chúng ta sẽ tìm hiểu giải thuật Bresenham áp dụng cho dựng ellipse.

4.1. Giải thuật Bresenham cho vẽ hình Ellipse

Rõ ràng điểm ảnh đầu tiên được chọn để vẽ sẽ là điểm A(0, b), nghĩa là:

$$(x_0, y_0) = (0, b)$$

Giả sử đến bước thứ i chúng ta đã chọn được điểm ảnh (x_i, y_i) để vẽ. Câu hỏi đặt ra là đến bước thứ i+1 chúng ta sẽ chọn điểm ảnh có tên gọi (x_{i+1}, y_{i+1}) với giá trị bằng bao nhiêu?

Vì điểm tiếp theo sẽ có hoành độ x tăng một giá trị so với giá trị của điểm chọn trước, hay nói cách khác là:

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

Đồng thời vì trên cung AC khi x tăng thì y giảm và tốc độ thay đổi của y chậm hơn của x, nên rõ ràng chúng ta thấy là với giá trị x tăng 1 thì giá trị y thay đổi một lượng Δy với ràng buộc $-1 \le \Delta y \le 0$. Mà điểm chọn bước trước là (x_i, y_i) nên điểm chọn tiếp theo (x_{i+1}, y_{i+1}) chỉ có thể là một trong hai điểm $P(x_i+1, y_i)$ và $Q(x_i+1, y_i-1)$.

Để quyết định được điểm chọn là P hay Q chúng ta hướng đến một biểu thức mà dấu của nó cho phép chúng ta ra quyết định chọn điểm nào.

Đặt: $d_1 = y_P^2 - y^2$ và $d_2 = y^2 - y_Q^2$

(*giá trị y ở biểu thức trên là tung độ của cung AC ứng với hoành độ đang xét* x_{i+1})

Đặt $P_i = (d_1 - d_2)a^2$

$$= [y_P^2 + y_Q^2 - 2y^2].a^2 = [y_i^2 + (y_i-1)^2 - 2(\frac{b^2(a^2 - x_{i+1}^2)}{a^2})].a^2$$

$$= [y_i^2 + (y_i-1)^2 - \frac{2b^2(a^2 - (x_i+1)^2)}{a^2}].a^2$$

$$= y_i^2.a^2 + (y_i-1)^2.a^2 - 2b^2.[a^2 - (x_i+1)^2]$$

$$= y_i^2.a^2 + (y_i-1)^2.a^2 - 2b^2.a^2 + 2b^2.(x_i+1)^2 \qquad (1.13)$$

(*lượng* a^2 *được đưa vào nhằm mục đích khử mẫu của* (d_1-d_2) *song không làm cho* P_i *và* $(d_1 - d_2)$ *khác dấu*)

Dấu của biểu thức P_i cho phép xác định điểm chọn tiếp theo là P hay Q. Cụ thể:

- Khi $P_i < 0$: thì điểm P sẽ sát với cung AC hơn điểm Q, do đó chúng ta

sẽ chọn điểm P làm điểm biểu diễn (vẽ).

- Khi $P_i > 0$: thì điểm Q sẽ sát với cung AC hơn điểm P, do đó chúng ta sẽ chọn điểm Q làm điểm biểu diễn.
- Khi $P_i = 0$: khoảng cách từ P và Q đến cung AC đều bằng nhau, nên chúng ta có thể chọn P hay Q đều được. Trong tình huống này giải thuật quy ước chọn điểm Q làm điểm biểu diễn

Vậy từ đây chúng ta thấy có thể dựa vào dấu của biểu thức P_i để ra quyết định chọn điểm tiếp theo.

Để giải thuật được đơn giản người ta tối ưu hoá việc tính P_i theo công thức truy hồi:

$$P_{i+1} = y_{i+1}^2.a^2 + (y_{i+1}-1)^2.a^2 - 2b^2.a^2 + 2b^2.(x_{i+1}+1)^2 \qquad (1.14)$$

Dấu của P_i sẽ quyết định giá trị P_{i+1} cụ thể như sau:

- Nếu $P_i < 0$: thì điểm chọn tiếp theo là $P(x_i+1,y_i)$, nghĩa là:

$$(x_{i+1}, y_{i+1}) = (x_i+1, y_i).$$

Thay vào (1.14) chúng ta được:

$$\begin{aligned} P_{i+1} &= y_i^2.a^2 + (y_i-1)^2.a^2 - 2b^2.a^2 + 2b^2.[(x_i+1)+1]^2 \\ &= P_i + 2b^2.[2(x_i+1)+1] \\ &= P_i + 2b^2(2x_i+3) \end{aligned}$$

- Nếu $P_i \geq 0$: thì điểm chọn tiếp theo là $Q(x_i+1, y_i-1)$, nghĩa là:

$$(x_{i+1}, y_{i+1}) = (x_i+1, y_i-1)$$

Thay vào (1.14) chúng ta được:

$$\begin{aligned} P_{i+1} &= (y_i-1)^2.a^2 + (y_i-2)^2.a^2 - 2b^2.a^2 + 2b^2.[(x_i+1)+1]^2 \\ &= P_i + a^2(-4y_i+4) + 2b^2[2(x_i+1)+1] \\ &= P_i + 4a^2(1-y_i) + 2b^2(2x_i+3) \\ &= P_i + 2b^2(2x_i+3) + 4a^2(1-y_i) \end{aligned}$$

Đầu tiên chúng ta chọn điểm A(0, b), nghĩa là $(x_0, y_0) = (0, b)$, thay vào (1.13) chúng ta có:

$$\begin{aligned} P_0 &= y_0^2.a^2 + (y_0-1)^2.a^2 - 2b^2.a^2 + 2b^2.(x_0+1)^2 \\ &= b^2.a^2 + (b-1)^2.a^2 - 2a^2.b^2 + 2b^2 \\ &= b^2.a^2 + a^2.b^2 - 2a^2.b + a^2 - 2a^2.b^2 + 2b^2 = -2a^2.b + a^2 + 2b^2 \\ &= a^2.(1-2b) + 2b^2 \end{aligned}$$

Vậy quy trình vẽ được thực hiện như sau:

- Tính P_0, vẽ điểm $(x_0, y_0) = (0, b)$
- Dựa vào dấu của P_0 chúng ta lại chọn được điểm vẽ tiếp theo (x_1, y_1) và giá trị P_1
- Dựa vào dấu của P_1 chúng ta lại chọn được điểm vẽ tiếp theo (x_2, y_2) và giá trị P_2
- Quá trình trên được lặp đi lặp lại cho đến khi chúng ta vẽ được điểm nguyên gần nhất với C.

4.2. Tóm tắt giải thuật Bresenham cho vẽ Ellipse:

- **Bước 1:** (Bước khởi động, tính toán các giá trị ban đầu)

  $P_0 = a^2(1-2b) + 2b^2$; $(x_0, y_0) = (0, b)$

  Vẽ điểm (x_0, y_0)

- **Bước 2:** (Bước lặp, thực hiện tính các giá trị điểm ảnh)

  Với mỗi giá trị i (I = 0, 1, 2,…) chúng ta xét dấu P_i

  - Nếu $P_i<0$: thì chọn điểm tiếp theo là

    $(x_{i+1}, y_{i+1})=(x_i+1, y_i)$

    $P_{i+1} = P_i + 2b^2(2x_i + 3)$

  - Ngược lại (tức $P_i \geq 0$): thì chọn điểm tiếp theo là

    $(x_{i+1}, y_{i+1}) = (x_i + 1, y_i - 1)$

    $P_{i+1} = P_i + 2b^2(2x_i + 3) + 4a^2(1 - y_i)$

  Vẽ điểm (x_{i+1}, y_{i+1}) vừa tìm được

- **Bước 3:** (Xác định điều kiện lặp)

  Lặp lại bước 2 với những giá trị i tiếp theo, cho đến khi chúng ta vẽ được điểm nguyên gần nhất với C, nghĩa là $x_{i+1} = Round(x_C) = Round(\frac{a^2}{\sqrt{a^2+b^2}})$ thì giải thuật kết thúc.

Chú ý: *Tóm tắt giải thuật trên chỉ áp dụng cho đoạn AC. Để dựng đoạn BC chúng ta cần có sự thay đổi vai trò của của x và y cũng như a và b. Cụ thể để dựng được cung BC cần hoán đổi trong toàn bộ giải thuật: x thành y và y ngược lại thành x, a thành b và b ngược lại thành a.*

4.3. Cài đặt giải thuật

Sau đây là phần cài đặt minh họa cho giải thuật vẽ ellipse gồm có: (1) Hàm Bre_Ellipse thực hiện chức năng vẽ đường ellipse (không tô màu phần bên trong). (2) Hàm FillEllipse thực hiện chức năng vẽ hình ellipse có tô màu phần bên trong hình.

1. Hàm Bre_Ellipse

Vào: Con trỏ ngữ cảnh thiết bị pDC, tọa độ tâm của ellipse (X_center, Y_center), bán kính theo trục x là a và bán kính theo trục y là b, màu sắc của đường ellipse là Color.

Ra: Hình ảnh thể hiện của ellipse trên ngữ cảnh thiết bị.

```
void CLineDlg::Bre_Ellipse(CDC * pDC, int X_center, int Y_center,
int a, int b, COLORREF Color)
{
   if ((a <= 0) || (b <= 0))
      return;

   int x, y;
   int P;
   int Const1, Const2;

   Const1 = int(double(a*a) / sqrt(double(a*a + b*b)) + 0.5);
   Const2 = int(double(b*b) / sqrt(double(a*a + b*b)) + 0.5);

   // Vẽ cung AC
```

```
x = 0; y = b; P = a*a*(1 - 2 * b) - 2 * b*b;

pDC->SetPixel(x + X_center, y + Y_center, PenColor);
pDC->SetPixel(x + X_center, -y + Y_center, PenColor);

if (Const1 > 0.5) // Cung AC có độ rộng 1 pixel
{
   while (x < Const1)
   {
      if (P < 0)
         P += 2 * b*b*(2 * x + 3);
      else
      {
         P += 2 * b*b*(2 * x + 3) + 4 * a*a*(1 - y);
         if (y > Const2)
            y--;
      }
      x++;

      pDC->SetPixel(x + X_center, y + Y_center, PenColor);
      pDC->SetPixel(x + X_center, -y + Y_center, PenColor);
      pDC->SetPixel(-x + X_center, y + Y_center, PenColor);
      pDC->SetPixel(-x + X_center, -y + Y_center, PenColor);
   }
}

// Vẽ cung BC
y = 0; x = a; P = b*b*(1 - 2 * a) - 2 * a*a;

pDC->SetPixel(x + X_center, y + Y_center, PenColor);
pDC->SetPixel(-x + X_center, y + Y_center, PenColor);

if (Const2 > 0.5) // Cung BC có độ cao hơn 1 pixel
{
   while (y < Const2)
   {
      if (P < 0)
         P += 2 * a*a*(2 * y + 3);
      else
      {
         P += 2 * a*a*(2 * y + 3) + 4 * b*b*(1 - x);
         if (x > Const1)
```

```
                x--;
            }
            y++;

            pDC->SetPixel(x + X_center, y + Y_center, PenColor);
            pDC->SetPixel(x + X_center, -y + Y_center, PenColor);
            pDC->SetPixel(-x + X_center, y + Y_center, PenColor);
            pDC->SetPixel(-x + X_center, -y + Y_center, PenColor);
        }
    }
}
```

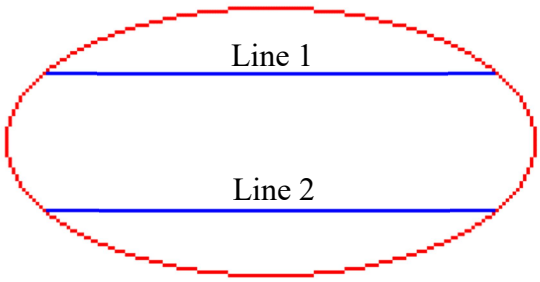
2. Hàm FillEllipse

Phần bên trong của ellipse được tiến hành tô theo kiểu quét dòng. Tại mỗi bước, chúng ta tính được điểm (x_i, y_i) theo giải thuật và xác định được thêm 3 điểm đối xứng của nó lần lượt là $(-x_i, y_i)$, $(x_i, -y_i)$, $(-x_i, -y_i)$. Từ đó xác định được 2 dòng quét là:

Line 1: Từ điểm $(-x_i, y_i)$ đến điểm (x_i, y_i)

Line 2: Từ điểm $(-x_i, -y_i)$ đến điểm $(x_i, -y_i)$

Tiến hành tô màu cho Line 1 và Line 2 như Hình 1.20 dưới đây.



Hình 1.20. Minh họa kỹ thuật tô ellipse theo dòng quét.

Tiến hành xây dựng hàm FillEllipse với thông tin vào ra:

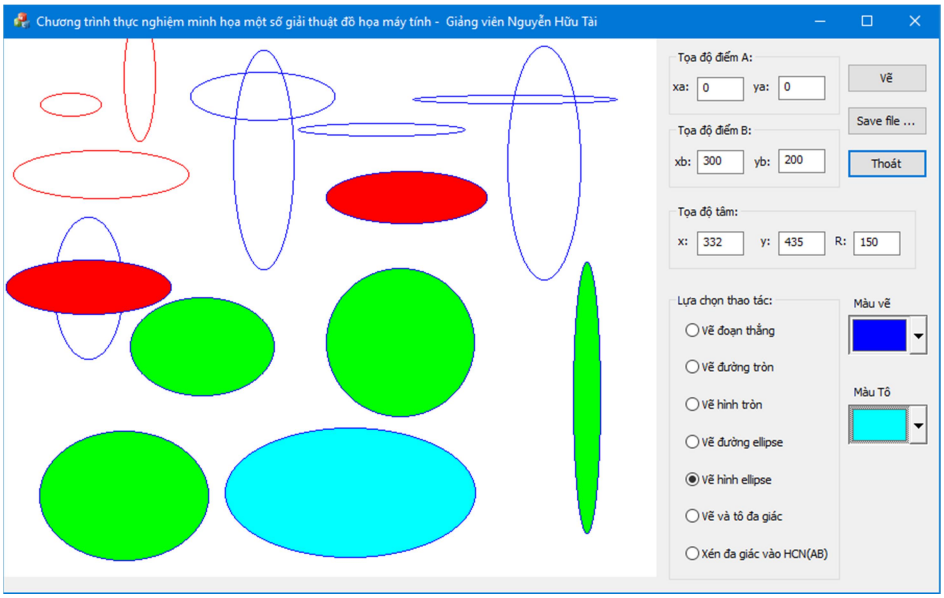
Vào: Con trỏ ngữ cảnh thiết bị pDC, tọa độ tâm của ellipse (X_center, Y_center), bán kính theo trục x là a và bán kính theo trục y là b, màu sắc của đường ellipse là PenColor và màu tô bên trong đường ellipse là BrushColor.

Ra: Hình ảnh thể hiện của ellipse trên ngữ cảnh thiết bị.

```
void FillEllipse(CDC * pDC, int X_center, int Y_center, int a, int
b, COLORREF PenColor, COLORREF BrushColor)
{
   if ((a <= 0) || (b <= 0))
      return;

   CPen MyPen(PS_SOLID, 1, BrushColor);
   HGDIOBJ OldPen = pDC->SelectObject(MyPen);
   int x, y, Old_y;
   int P;
   int Const1, Const2;

   Const1 = int((double(a*a) / sqrt(double(a*a + b*b))) +0.5);
   Const2 = int((double(b*b) / sqrt(double(a*a + b*b))) +0.5);
   // Vẽ cung AC
   x = 0; y = b; P = a*a*(1 - 2 * b) - 2 * b*b;

   pDC->SetPixel(x + X_center, y + Y_center, PenColor);
   pDC->SetPixel(x + X_center, -y + Y_center, PenColor);
   Old_y = y;

   if (Const1 > 0.5) // Cung AC có độ rộng hơn 1 pixel
   {
      while (x < Const1)
      {
         if (P < 0)
            P += 2 * b*b*(2 * x + 3);
         else
         {
            P += 2 * b*b*(2 * x + 3) + 4 * a*a*(1 - y);
            if (y > Const2)
               y--;
         }
         x++;

         pDC->SetPixel(x + X_center, y + Y_center, PenColor);
         pDC->SetPixel(x + X_center, -y + Y_center, PenColor);
         pDC->SetPixel(-x + X_center, y + Y_center, PenColor);
         pDC->SetPixel(-x + X_center, -y + Y_center, PenColor);
```

```
      if (y != Old_y)
      {
         pDC->MoveTo(-x + 1 + X_center, y + Y_center);
         pDC->LineTo(x + X_center, y + Y_center);
         pDC->MoveTo(-x + 1 + X_center, -y + Y_center);
         pDC->LineTo(x + X_center, -y + Y_center);
         Old_y = y;
      }
   }
}

// Vẽ cung BC
y = 0; x = a; P = b*b*(1 - 2 * a) - 2 * a*a;

pDC->SetPixel(x + X_center, y + Y_center, PenColor);
pDC->SetPixel(-x + X_center, y + Y_center, PenColor);
if (y != Old_y)
{
   pDC->MoveTo(-x + 1 + X_center, y + Y_center);
   pDC->LineTo(x + X_center, y + Y_center);
   Old_y = y;
}

if (Const2 > 0.5) // Cung BC có độ cao hơn 1 pixel
{
   while (y < Const2)
   {
      if (P < 0)
         P += 2 * a*a*(2 * y + 3);
      else
      {
         P += 2 * a*a*(2 * y + 3) + 4 * b*b*(1 - x);
         if (x > Const1)
            x--;
      }
      y++;

      pDC->SetPixel(x + X_center, y + Y_center, PenColor);
      pDC->SetPixel(x + X_center, -y + Y_center, PenColor);
      pDC->SetPixel(-x + X_center, y + Y_center, PenColor);
      pDC->SetPixel(-x + X_center, -y + Y_center, PenColor);
```

```
            if (x > 0)
            {
               pDC->MoveTo(-x + 1 + X_center, y + Y_center);
               pDC->LineTo(x + X_center, y + Y_center);
               pDC->MoveTo(-x + 1 + X_center, -y + Y_center);
               pDC->LineTo(x + X_center, -y + Y_center);
            }
         }
      }
      pDC->SelectObject(OldPen);
}
```



Hình 1.21. Hình ảnh thực nghiệm giải thuật Bresenham dựng đường ellipse và hình ellipse.

5. BÀI TẬP CUỐI CHƯƠNG

1. Cho điểm A(5,7) và B(15,15). Sử dụng giải thuật Bresenham đã cho để tìm toạ độ các điểm vẽ (x_i , y_i).
2. Kế thừa dự án ***LineDemo*** trong ***bài thực nghiệm số 1*** đã được

trình bày chi tiết tại ***mục 2.3***, sinh viên cần xây dựng một hàm vẽ đoạn thẳng tổng quát cho phép vẽ đoạn thẳng AB trong mọi trường hợp hệ số góc.

3. Sử dụng giải thuật vẽ đường tròn Midpoint để tính giá trị các điểm vẽ (x_i , y_i) biết rằng R = 20.
4. Cài đặt một hàm vẽ đường tròn theo giải thuật MidPoint.
5. Cài đặt một hàm cho phép tô màu phần diện tích bên trong của đường tròn. Hàm có dạng ***void FillCircle(int x, int y, int R, COLORREF FillColor);***
6. Cài đặt hàm vẽ đường ellipse theo giải thuật Bresenham
7. Cài đặt một hàm tô màu phần bên trong của một Ellipse.
8. Hãy xây dựng một giải thuật để dựng đường cong bậc 2 (Parabol) dạng tổng quát, $y = ax^2 + bx + c$, trên một đoạn xác định $[x_1, x_2]$
9. Viết chương trình vẽ đường Parabol.
10. Xây dựng chương trình cho phép vẽ biểu đồ (Chart) từ số liệu đầu vào như trong chức năng vẽ đồ thị của chương trình Microsoft Excel.
11. Hãy xây dựng một thư viện đồ hoạ riêng với các hàm vẽ các đường cơ bản do bạn tự viết dựa trên những kiến thức đã được lĩnh hội.

TÀI LIỆU THAM KHẢO

Bắt buộc:

1. John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley (2014), ***Computer Graphics: Principles and Practice***, third edition, Addison-Wesley Publishing Company, USA.

Không bắt buộc:

2. Francis S. Hill (1990), ***Computer Graphics***, Jr. Macmillan Publishing Company, New York.
3. John R. Rankin (1989), ***Computer Graphics Software Contruction***, Prentice Hall of Australia.
4. Harrington (1986), ***Computer Graphics: A Programming Approach***, McGraw-Hill Book Company, Singapore.
5. Video Electronics Standards Association (1998), ***VESA BIOS Extensions Core Functions Standard – Version 3.0***.
6. http://www.tftcentral.co.uk/articles/content/pointers_gamut.htm , 20/06/2016.
7. https://msdn.microsoft.com/en-us/library/windows/desktop/dd183567(v=vs.85).aspx, 20/06/2016.
8. https://msdn.microsoft.com/en-us/library/windows/desktop/dd145049(v=vs.85).aspx, 20/06/2016.
9. https://msdn.microsoft.com/en-us/library/windows/desktop/dd183494(v=vs.85).aspx, 20/06/2016.
10. https://msdn.microsoft.com/en-us/library/windows/desktop/dd162467, 20/06/2016.