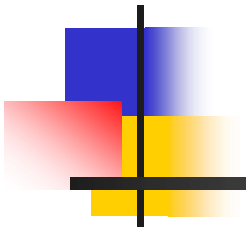
TRƯỜNG ĐẠI HỌC THỦY LỢI
Khoa Công nghệ Thông tin
Bộ môn KHMT

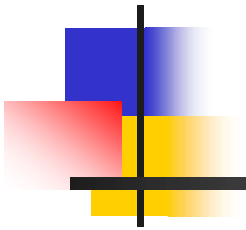


ĐỒ HỌA MÁY TÍNH
(Computer Graphics)

Ngô Trường Giang
E-mail: giangnt@tlu.edu.vn

Nội dung

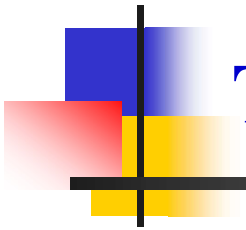
- Tổng quan đồ họa máy tính
- Màu và phối màu
- Thuật toán cơ sở vẽ đồ họa
- **Các kỹ thuật trong đồ họa 2D**
- Phép biến đổi đồ họa 2D
- Phép biến đổi đồ họa 3D
- Quan sát đồ họa 3D
- Mô hình hóa bề mặt



CÁC KỸ THUẬT TRONG ĐỒ HỌA 2D

- Thuật toán cắt xén
- Thuật toán tô màu
- Phông chữ

Tham khảo slides của PGS.TS Đặng Văn Đức – Viện CNTT, Viện HLKH&CN VN



Thuật toán cắt xén

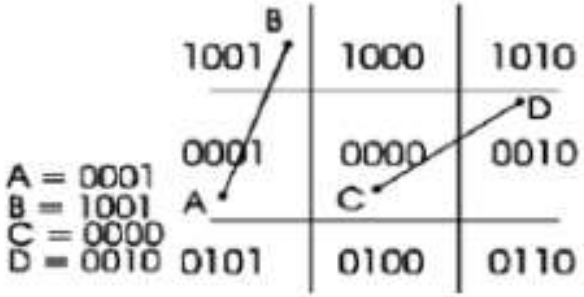
- Cắt xén đoạn thẳng
- Cắt xén vùng bởi đa giác bất kỳ

Thuật toán cắt xén đoạn thẳng

- Các giải pháp:
  - Kiểm tra từng pixel của đoạn thẳng có ở trong chữ nhật?
  - Tính toán các điểm cắt của từng đoạn thẳng với cạnh chữ nhật cắt xén
  - Thuật toán Sutherland-Cohen: loại bỏ các đoạn không cần cắt xén bằng xét tọa độ đầu mút các đoạn thẳng -> đơn giản và hiệu quả.

Thuật toán Sutherland-Cohen

- Mã hóa đầu mút các đoạn thẳng
- Xác định nhanh đoạn thẳng có cần cắt xén hay không nhờ các phép toán logíc AND và OR



bit 1 - left
bit 2 - right
bit 3 - below
bit 4 - above

Endpoint codes		*OR*	*AND*	*Meaning*
0000	0000	0000	0000	No clipping (1)
0001	0001	0001	**0001**	No clipping (2)
1001	0001	1001	**0001**	No clipping (3)
1001	0100	1101	0000	**Partly visible** (4)

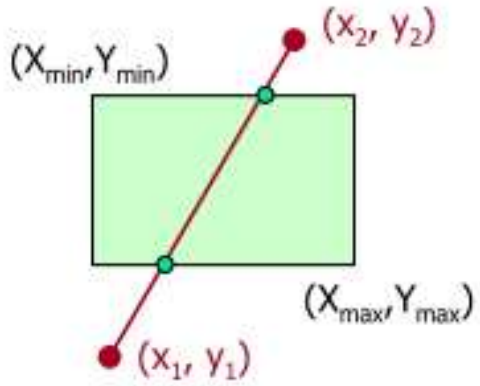
Thuật toán Sutherland-Cohen

- Kết quả phép OR hai mã đầu mút đoạn thẳng cho kết quả 0: cả hai điểm nằm trong chữ nhật
- Kết quả phép AND hai mã đầu mút đoạn thẳng cho kết quả khác 0: cả hai điểm nằm ngoài chữ nhật
- Cắt xén:
  - Giao của đoạn thẳng với các cạnh chữ nhật song song trục tung
    - x có giá trị X_{min}, X_{max} và hệ số góc a= $(y_2-y_1)/(x_2-x_1)$

$$y=y_1 + a(x - x_1)$$

  - Giao đoạn thẳng với các cạnh song song trục hoành
    - y có giá trị Y_{min}, Y_{max} và hệ số góc

$$x=x_1 + (y - y_1)/a$$

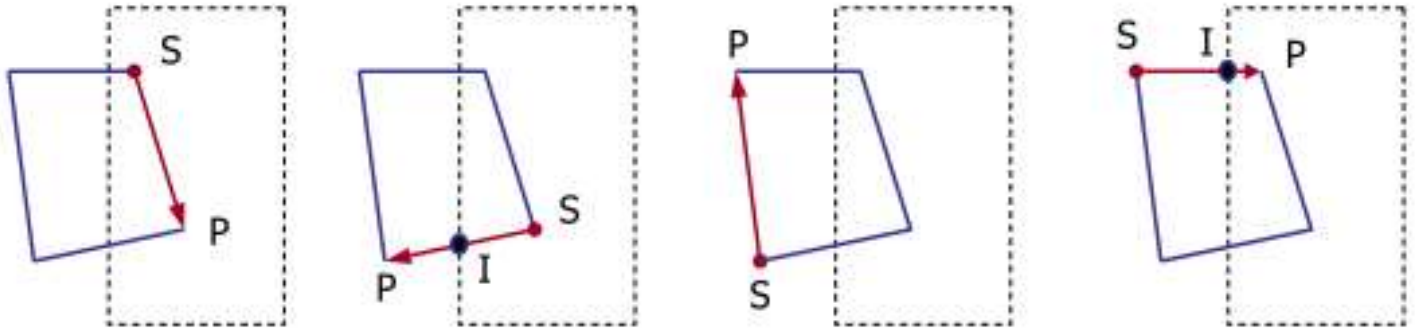


Thuật toán cắt xén vùng

- Tác giả: Sutherland-Cohen
- Input:
  - Chữ nhật cắt xén
  - Vùng là đa giác được xác định bởi trật tự các cặp tọa độ
- Output:
  - Các đa giác nằm trong cửa sổ cắt xén
- Ý tưởng thuật toán
  - So sánh lần lượt các đỉnh đa giác với biên cửa sổ
    - đỉnh nằm ngoài, loại bỏ ngay
    - đỉnh nằm trong, lưu trữ lại làm kết quả
    - tính giao điểm của các cạnh đa giác vùng với cạnh chữ nhật

Thuật toán cắt xén vùng

- Duyệt lần lượt (td. theo chiều kim đồng hồ) các cạnh đa giác
- Nếu đỉnh duyệt xuất phát từ trong cửa sổ theo cạnh đa giác đi ra ngoài cửa sổ: lưu trữ giao của cạnh đa giác với biên cửa sổ
- Nếu đường đi từ ngoài vào trong cửa sổ: lưu trữ đỉnh đa giác và giao điểm
- Thí dụ xét hai đỉnh đa giác S và P:

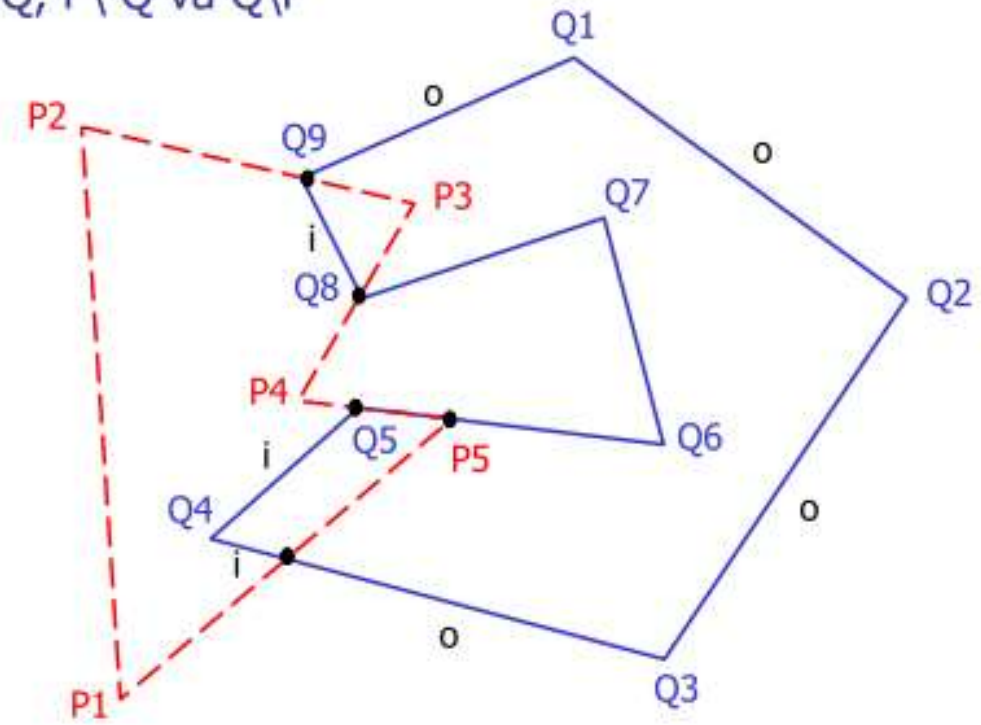


Cắt xén vùng bằng đa giác

- Thuật toán do Clamer Schutte (Đại học Delft, Hà lan) đề xuất
  - cho trước hai đa giác P và Q không có lỗ hổng, không tự cắt và đỉnh của chúng được sắp xếp theo chiều kim đồng hồ
  - Hãy tìm đa giác thuộc tập P∪Q, P\ Q và Q\P
- Các bước của thuật toán
  - phân lớp các đỉnh của hai đa giác vào 2 danh sách: gán vào mỗi đỉnh giá trị i(nside), o(utside) hay b(oundary) phụ thuộc vào vị trí của nó so với đa giác kia

Pv= <(P1, o), (P2, o), (P3, i),
(P4, o), (P5, b)>

Qv=<(Q1, o), (Q2, o), (Q3, o),
(Q4, i), (Q5, b), (Q6, o),
(Q7, o), (Q8, b), (Q9, b)>

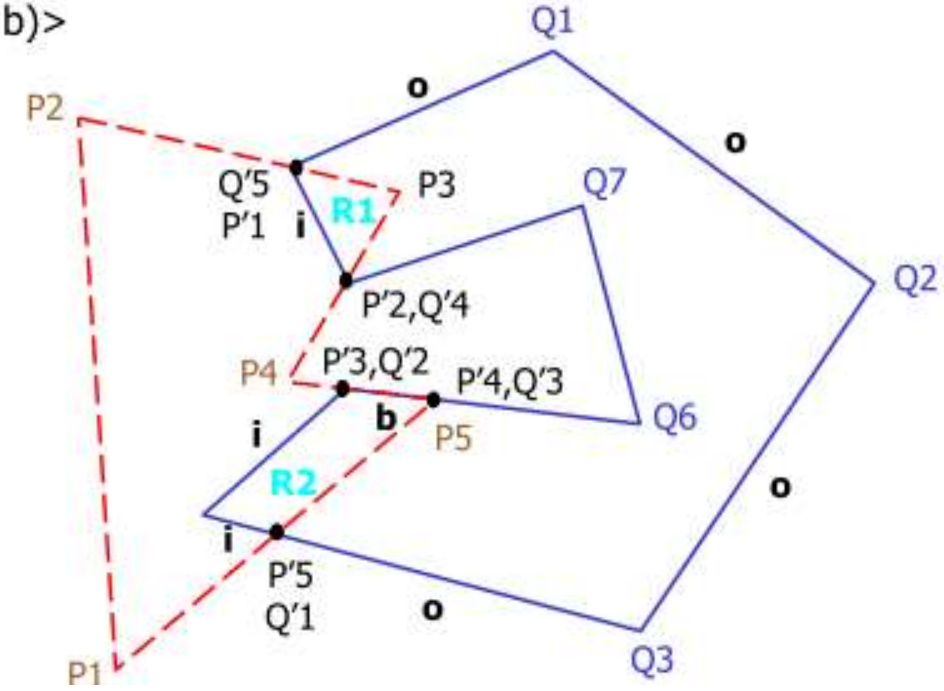


Cắt xén vùng bằng đa giác

- Tìm giao các cạnh của hai đa giác P, Q.
- Mỗi giao điểm được xen vào Pv hay Qv và đánh dấu b(oundary) để có danh sách mới

Pef=<(P1, o), (P2, o), (P'1, b), (P3, i), (P'2, b), (P4, o), (P'3, b), (P5, b), (P'4, b), (P'5, b)
Qef= (Q1, o), (Q2, o), (Q3, o), (Q'1, b), (Q4, i), (Q5, b),(Q'2, b), (Q'3, b), (Q6, o), (Q7, o),
(Q8, b), (Q'4, b), (Q'5, b), (Q9, b)>

- Lựa chọn các cạnh mới cho đa giác kết quả
  - Các khúc của cạnh là một phần cạnh đa giác gốc, nằm hoàn toàn trong hay hoàn toàn ngoài đa giác kia
  - thí dụ: chọn các khúc nằm trong khi đầu cuối nằm trong, nếu cả hai đầu mút nằm trên cạnh đa giác thì kiểm tra điểm giữa của nó xem có nằm trong đa giác?

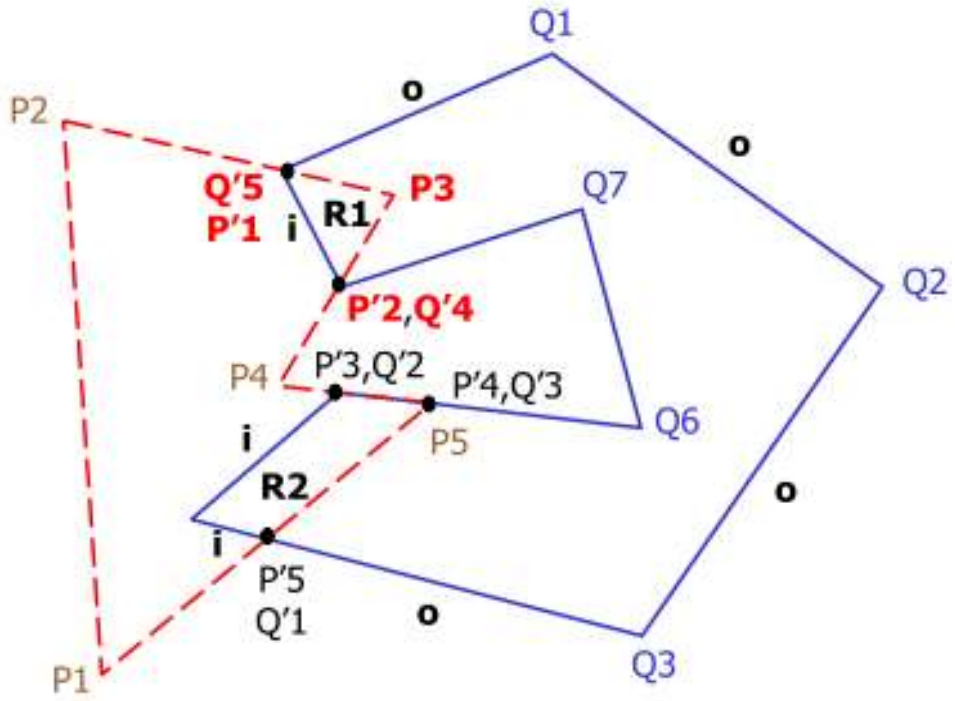


Cắt xén vùng bằng đa giác

- Danh sách Pef và Qef:

Pef=<(P1, o), (P2, o), (P'1, b), (P3, i), (P'2, b), (P4, o), (P'3, b), (P5, b), (P'4, b), (P'5, b)
Qef= (Q1, o), (Q2, o), (Q3, o), (Q'1, b), (Q4, i), (Q5, b),(Q'2, b), (Q'3, b), (Q6, o), (Q7, o),
(Q8, b), (Q'4, b), (Q'5, b), (Q9, b)>

- Tách các đa giác kết quả
  - Tách từng đa giác kết quả. Bắt đầu từ khúc bất kỳ có dấu i hay b sau đó tìm trong 2 danh sách các khúc tiếp theo có đầu mút khớp với điểm cuối của cạnh trước. Lặp cho đến khi trở lại điểm ban đầu
  - Thí dụ: hãy bắt đầu từ khúc (P'1, P3). Sau đó tìm khúc (i, b) có P3 là điểm mút, ta thấy (P3, P'2). Tiếp theo từ P'2, ta tìm trong Qef thấy khúc Q'4Q'5. Hình thành đa giác R1



Thuật toán liên quan đến cắt xén

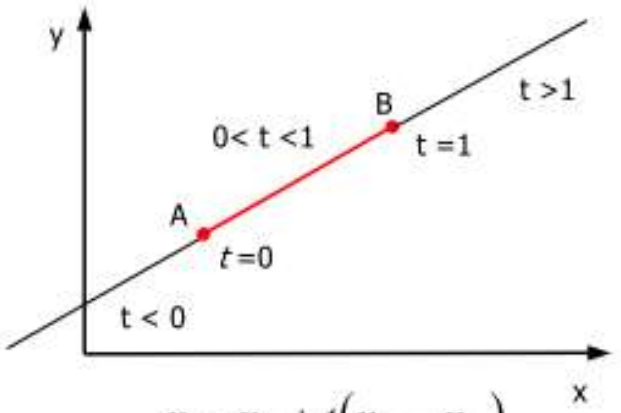
- Xác định giao điểm hai đoạn thẳng
  - Giao của hai đường thẳng đi qua hai điểm
    - thí dụ đơn giản
      - đường thẳng đi qua tọa độ (4,2) và tọa độ (2,0) có giao với đoạn thẳng đi qua (0,4) và (4,0)?
    - giải pháp
      - xác định phương trình đường thẳng qua 2 điểm y=ax+b, trong đó a=(y2-y1)/(x2-x1)
      - từ thí dụ trên có: y=-2+x và y=4-x giao điểm tại (3, 1)
    - Tổng quát: nếu ta có $y=a_1+b_1x$ và $y=a_2+b_2x$ thì giao điểm sẽ ở tại: $x_i=-(a_1-a_2)/(b_1-b_2)$, $y_i=a_1+b_1x_i$
    - Các trường hợp đặc biệt: song song trục x hay y, song song với nhau

Thuật toán liên quan đến cắt xén

- Xác định giao điểm hai đoạn thẳng
  - Nếu sử dụng phương pháp tìm giao đường thẳng: đòi hỏi kiểm tra tọa độ của giao đường thẳng có nằm trong các đoạn thẳng?
  - Phương pháp khác: biểu diễn đoạn thẳng bằng tham số
    - đoạn thẳng 1 từ (x_A, y_A) đến (x_B, y_B)
    - đoạn thẳng 2 từ (x_C, y_C) đến (x_D, y_D)
    - tính toán giao của 2 đoạn thẳng tại tọa độ có t, s:

$$t = \frac{(x_C - x_A)(y_C - y_D) - (x_C - x_A)(y_C - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

$$s = \frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

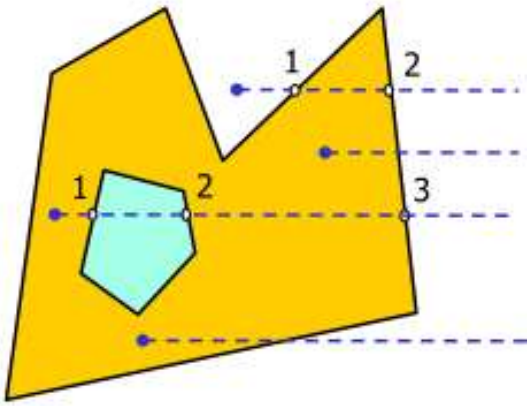


$$x = x_A + t(x_B - x_A)$$
$$y = y_A + t(y_B - y_A)$$
$$x = x_C + s(x_D - x_C)$$
$$y = y_C + s(y_D - y_C)$$

$0 \le t \le 1$ và $0 \le s \le 1$

Thuật toán liên quan đến cắt xén

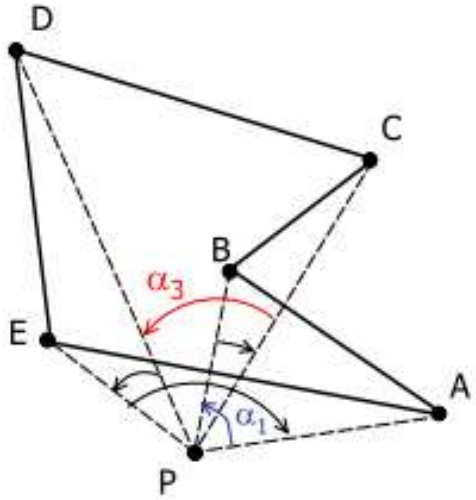
- Xác định điểm trong đa giác
  - Thuật toán này ứng dụng để xác định điểm độc lập hay đầu mút đoạn thẳng nằm trong đa giác.
  - Định lý nửa đường thẳng của Jordan
    - từ điểm cho trước, hãy vẽ tia ra cạnh tận cùng của đa giác
    - tính tổng giao điểm của tia với các cạnh đa giác
      - nếu tổng số điểm là lẻ thì điểm đó nằm trong đa giác, ngược lại tổng số điểm chẵn thì điểm nằm ngoài đa giác
    - để dễ tính toán hãy dựng tia song song với trục tọa độ
    - thuật toán này đúng cả với trường hợp đa giác có lỗ hổng
    - cần chú ý khi tia đi qua đỉnh đa giác hay trùng với cạnh đa giác.



Tăng tốc độ tính toán bằng cách sử dụng chữ nhật bao đa giác

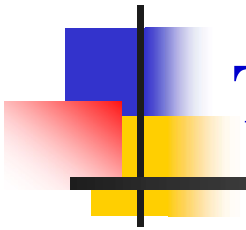
Thuật toán liên quan đến cắt xén

- Xác định điểm trong đa giác
  - Phương pháp kiểm tra góc
    - Thí dụ, xét xem điểm P cho trước có ở trong đa giác ABCDE?
    - từ điểm P nối với các đỉnh đa giác để tạo thành các góc theo thứ tự ngược chiều kim đồng hồ
    - các góc này có giá trị dương hoặc âm tùy theo hướng đo
    - tính tổng các góc
      - nếu tổng các góc bằng 0 thì P nằm ngoài đa giác
      - nếu tổng các góc bằng 360^0 thì P nằm trong đa giác



Bài tập

- Mã hóa thành chuỗi 4 bít hai đầu đoạn thẳng AB theo giải thuật Cohen Sutheland, với A (1, -2) và B(1, 4), tọa độ cửa sổ cắt xén có góc trái dưới (-1,0) và góc phải trên (2, 3). Xác định tọa độ mới của đoạn thẳng sau khi cắt xén.
- Xây dựng thuật toán cắt xén đa giác bằng chữ nhật
- Xây dựng thuật toán tìm giao hai đa giác lồi
- Xây dựng thuật toán tìm giao hai đa giác bất kỳ (có thể là đa giác phức)
- Cài đặt các thuật toán trên



Thuật toán tô mầu

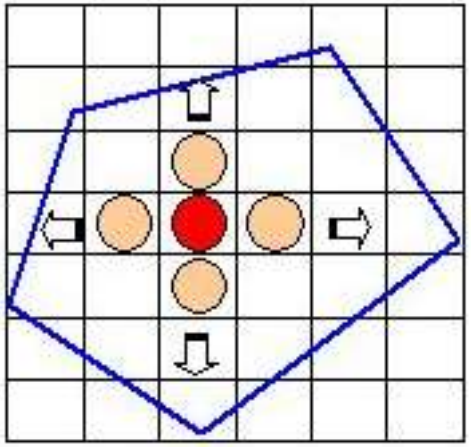
- Thuật toán tô màu tràn
- Thuật toán tô màu theo đường quét

Các thuật toán tô màu

- Đặt vấn đề
  - Cho trước đa giác trong không gian 2D, hãy tô đa giác theo màu xác định trên màn hình đồ họa
- Giả thiết
  - Đa giác đơn, không tự cắt
- Giải pháp
  - Tô màu tràn (Flood fill)
  - Tô màu theo đường quét(Scan conversion)

Thuật toán tô màu tràn

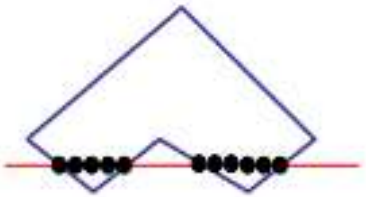
- Input:
  - Cho trước đa giác P có n đỉnh v0 đến vn-1 (vn trùng với v0)
  - Màu tô đa giác: C
  - Tọa độ điểm xuất phát tô: p = (x, y) ∈ P (là điểm bên trong đa giác P)
- Thuật toán

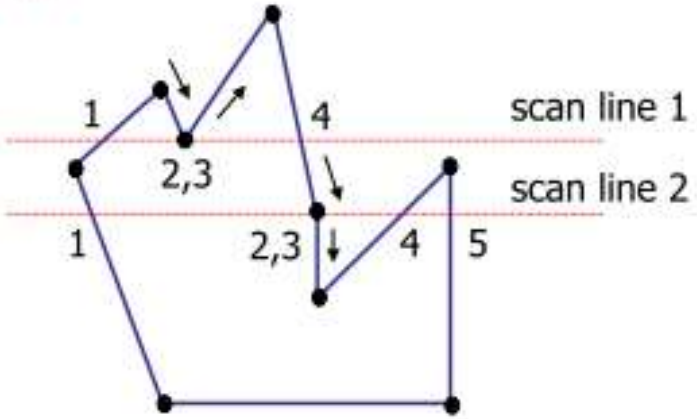


```
FloodFill (Polygon P, int x, int y, Color C)
if not (OnBoundary (x, y, P) and Colored (x, y, C))
Begin
    PlotPixel (x, y, C);
    FloodFill (P, x+1, y, C);
    FloodFill (P, x, y+1, C);
    FloodFill (P, x-1, y, C);
    FloodFill (P, x, y-1, C);
End;
```

Thuật toán tô màu theo đường quét

- Ý tưởng: Sử dụng giao điểm giữa các biên đa giác và đường quét để nhận ra pixel có trong đa giác?





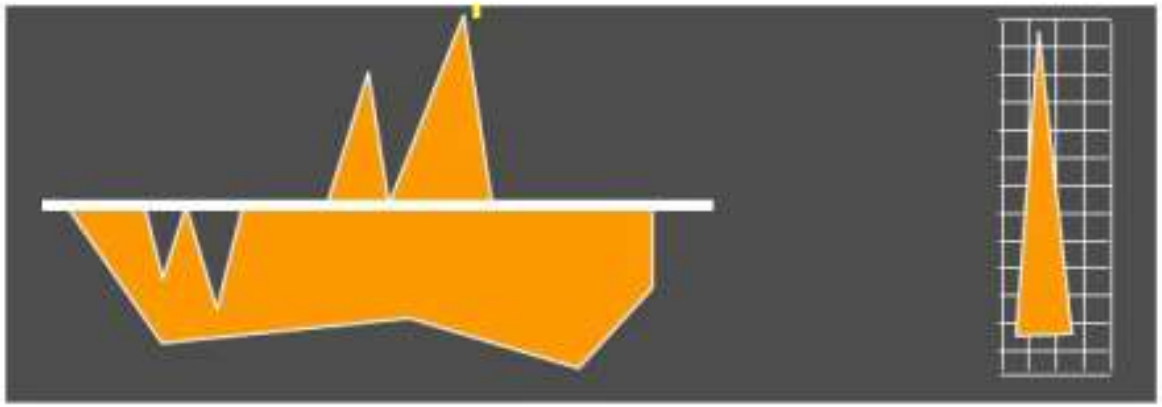
- Thuật toán:
  - Cho trước đa giác P với n đỉnh v_0 đến v_{n-1} (v_n trùng với v_0)
  - Cho trước C là màu tô đa giác
  - Giao của mỗi đường quét với các cạnh đa giác thì sẽ là điểm vào hay điểm ra đa giác
  - Tìm ra các đoạn thẳng nằm trong đa giác để vẽ theo màu C

Thuật toán tô màu theo đường quét

```
ScanConvert( Polygon P, Color C)
{
    For (y=0 ;y<=ScreenYMax;y++)
    {
        I <= Các giao điểm của cạnh đa giác P với đường Y = y;
        Sắp xếp I: X tăng dần và
        Vẽ đoạn thẳng cách quãng theo màu C;
    }
}
```

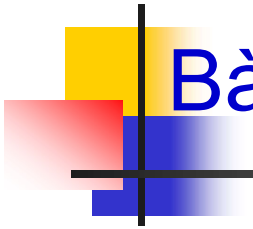
Thuật toán tô màu theo đường quét

- Tối ưu chuyển đổi đường quét
  - Sử dụng đồng thời thuật toán Bresenham hay thuật toán trung điểm vẽ đoạn thẳng
- Các trường hợp đặc biệt



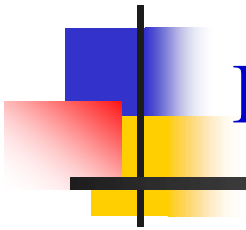
So sánh các thuật toán tô màu

Flood Fill	Scan Conversion
Đơn giản	Phức tạp hơn
Thuật toán rời rạc hóa trong không gian màn hình	Thuật toán rời rạc hóa trong đối tượng hoặc/và không gian màn hình
Yêu cầu gọi hệ thống GetPixel/Val	Độc lập với thiết bị
Đòi hỏi điểm seed	Không đòi hỏi điểm seed
Yêu cầu stack rất lớn	Yêu cầu stack nhỏ



Bài tập

- Cài đặt tô màu tràn đa giác.
- Cài đặt tô màu đa giác theo đường quét
- Viết chương trình tô màu elíp , hình tròn

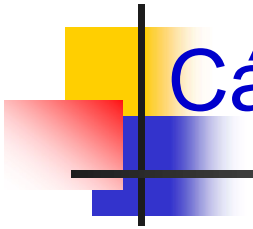


Phông chữ

- Một vài khái niệm về phông chữ
- Phông bitmap (raster)
- Phông véctơ
- Phông True Type

Các khái niệm

- Phông chữ được Guttenberg thiết kế. Được sử dụng từ nhiều thế kỷ. Ngày nay rất phong phú.
- Phông là tập đầy đủ các ký tự có kiểu dáng (style)
  - Weight: light, normal, bold
  - Shape: round, oval, straight
  - Posture: Oblique, Italic
  - Serif, sans-serif
- Font family
  - Có face name (thí dụ Times Bold, Times Italic) cho biết weight và posture (không cho biết size)



Các loại phông

- Công nghiệp máy tính sử dụng 3 loại phông
  - Phông bitmap (raster)
  - Phông véctơ
  - Phông TrueType

Raster Font

- Là loại phông đầu tiên của màn hình máy tính
- Ngày nay vẫn đang sử dụng
- Ban đầu font bitmap được nhúng trong các vỉ điều khiển màn hình, máy in
- Ánh xạ bố trí pixels của mỗi ký tự lên màn hình
  - Mỗi bit trong bitmap sẽ bật sáng điểm ảnh trên CRT

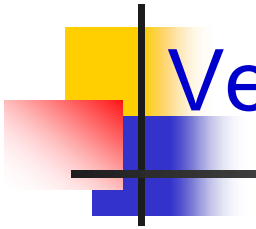
Offset	Hex value	Binary value
0	00h	00000000
1	3Ch	00111100
2	18h	00011000
3	18h	00011000
4	18h	00011000
5	18h	00011000
6	3Ch	00111100
7	00h	00000000

Raster Fonts

- Có độ rộng và độ cao cố định
- Lưu trữu: tách biệt các ảnh phông. Vị trí byte thứ nhất của khối bitmap trong bộ font:

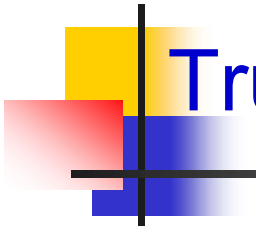
  Offset = (ASCII code) * (Bytes per character)

- Ưu điểm
  - Hiển thị nhanh
  - Dễ tạo lập và dễ sửa đổi
- Nhược điểm
  - Vấn đề co dãn
  - Dung lượng lưu trữ lớn



Vector Fonts

- Không sử dụng Bitmap để mô tả ký tự
- Sử dụng ngôn ngữ mô tả nào đó
  - Ngôn ngữ mô tả bao gồm các lệnh như Line, Curve, Polygon...
  - Tọa độ: tương đối trong chữ nhật chứa ký tự
  - Chương trình con xử lý các lệnh để hiển thị
- Ưu điểm
  - Dễ co giãn, có tính propotional, trơn tru, dễ tạo lập hiệu ứng đặc biệt: xoay, gập, cong...
- Nhược điểm:
  - Khi hiển thị font nhỏ: chậm hơn bitmap font.
  - Vấn đề hiển thị font nét chữ dày.



True Type Fonts

- True Type là công nghệ font của Apple Computer Inc. (1987); Tác giả: Kathryn Weisberg, Sampo Kaasila...
- MS bắt đầu sử dụng True Type vào đầu 1992 trên Windows 3.1
- Công nghệ True Type bao gồm:
  - Các tệp chứa True Type Fonts (TTF)
  - Bộ raster hóa True Type của hệ điều hành (MacOS, Windows...) trước khi hiển thị, in trên giấy.
- OpenType: 5-1996 MS và Adobe System kết hợp công nghệ True Type với PostScript.

Tệp TTF

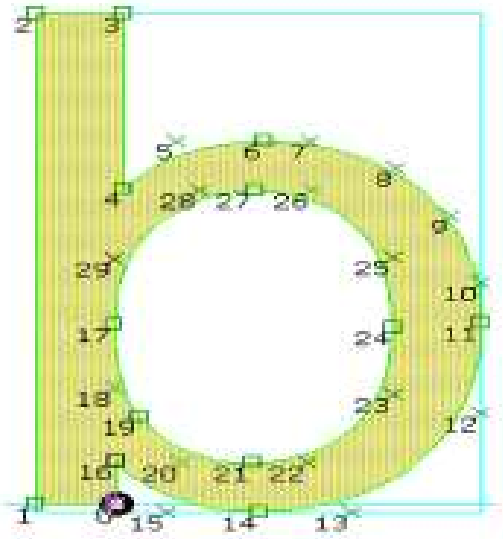
- Tệp TTF chứa
  - Mô tả hình dạng ký tự
  - Các thong tin khác: tên font, bản quyền, hãng sản xuất...
- Mô tả ký tự trong TTF:
  - Mô tả đường viền font -> TTF là font outline
  - Mô tả toán học của ký từ từ dãy các điểm
  - Viền ký tự được tạo bởi các line và B-splines toàn phương
    - Mỗi B-Spline tương ứng với dãy các Bezier bậc 2 được xác định bởi ba điểm điều khiển.
  - Thí dụ có ba điểm điều khiển (A_x, A_y), (B_x B_y), (C_x, C_y)
  - Các điểm P trên đường cong được tính với t trong khoảng [0,1]

$$p_x = (1-t)^2.A_x + 2t(1-t).B_x + t^2.C_x$$

$$p_y = (1-t)^2.A_y + 2t(1-t).B_y + t^2.C_y$$

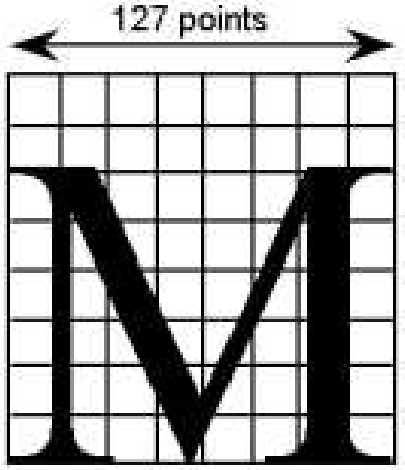
Tệp TTF

- Thí dụ mô tả outline của chữ b Arial:
  - các điểm 4, 5, 6 xác định Bézier
  - các điểm từ 6 đến 11 xác định B-Spline, chứa 4 Béziers
- Contour là đường khép kín bao gồm đường thẳng hay đường cong.
  - Ký tự 7 có 1, ký tự i có 2 và chữ B có 3 contours
- Glyph (nét chữ): là đường viền tạo bởi 1 hay nhiều contour.
- Đánh số điểm điều khiển
  - Theo thứ tự: tô màu phía phải
  - Đánh số kế tiếp giữa các contour



Tệp TTF

- Đơn vị đo của các điểm điều khiển
  - Hình vuông 'em':
    - là không gian vuông tưởng tượng có độ phân giải cao để co dãn, hiệu chỉnh các nét font
    - Tổng số đơn vị trong em (upem) do nhà thiết kế xác định (2048)
    - upem đủ lớn để biểu diễn toàn bộ các glyph.
    - upem là cố định cho một thiết kế font (không thay đổi khi co dãn)
  - FUnit:
    - là đơn vị mô tả vị trí các điểm điều khiển trong em.
    - có giá trị trong khoảng [-16384,+16383]
    - là đơn vị tương đối, nó thay đổi khi em thay đổi



Phương pháp co dãn Glyph

- Không gian thiết bị:
  - Độ phân giải hình vuông phải được chuyển đổi thành kích thước của thiết bị (pixel/dot)

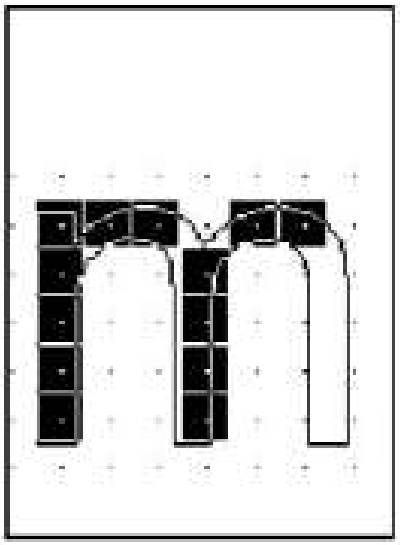
$$p' = p * pointSize * \frac{resolution}{72\ dpi * upem}$$

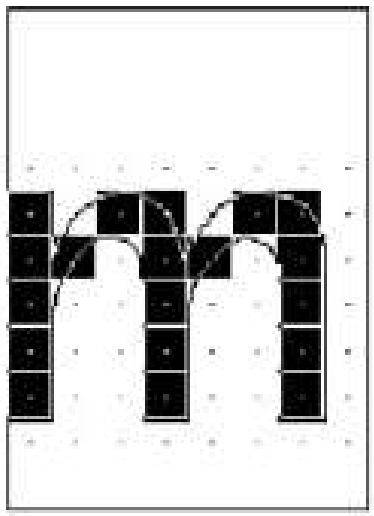
    - p – FUnit; p' – pixel; pointSize – co chữ yêu cầu
    - resolution- pixel/in của thiết bị; upem – độ phân giải thiết kế font
  - Thí dụ:
    - Glyph có 550 FUnits; Độ phân giải màn hình 72 dpi
    - upem thiết kế là 2048
    - Tại điểm 18 của glyph tương đương với 4.83 pixel

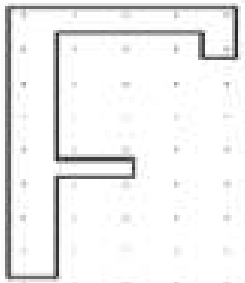
$$550 * 18 * \frac{72}{72 * 2048} = 4.83$$

True Type Fonts

- Hinting
  - Sau khi chuyển outline sang bitmap thường phải hiệu chỉnh nét vẽ để có chất lượng cao
  - Mỗi nét trong phông có chương trình con bằng ngôn ngữ thông dịch (tựa asm) kèm theo để thực hiện hiệu chỉnh.
    - Ngôn ngữ này có khoảng 150 lệnh, kích thước lệnh 1 byte



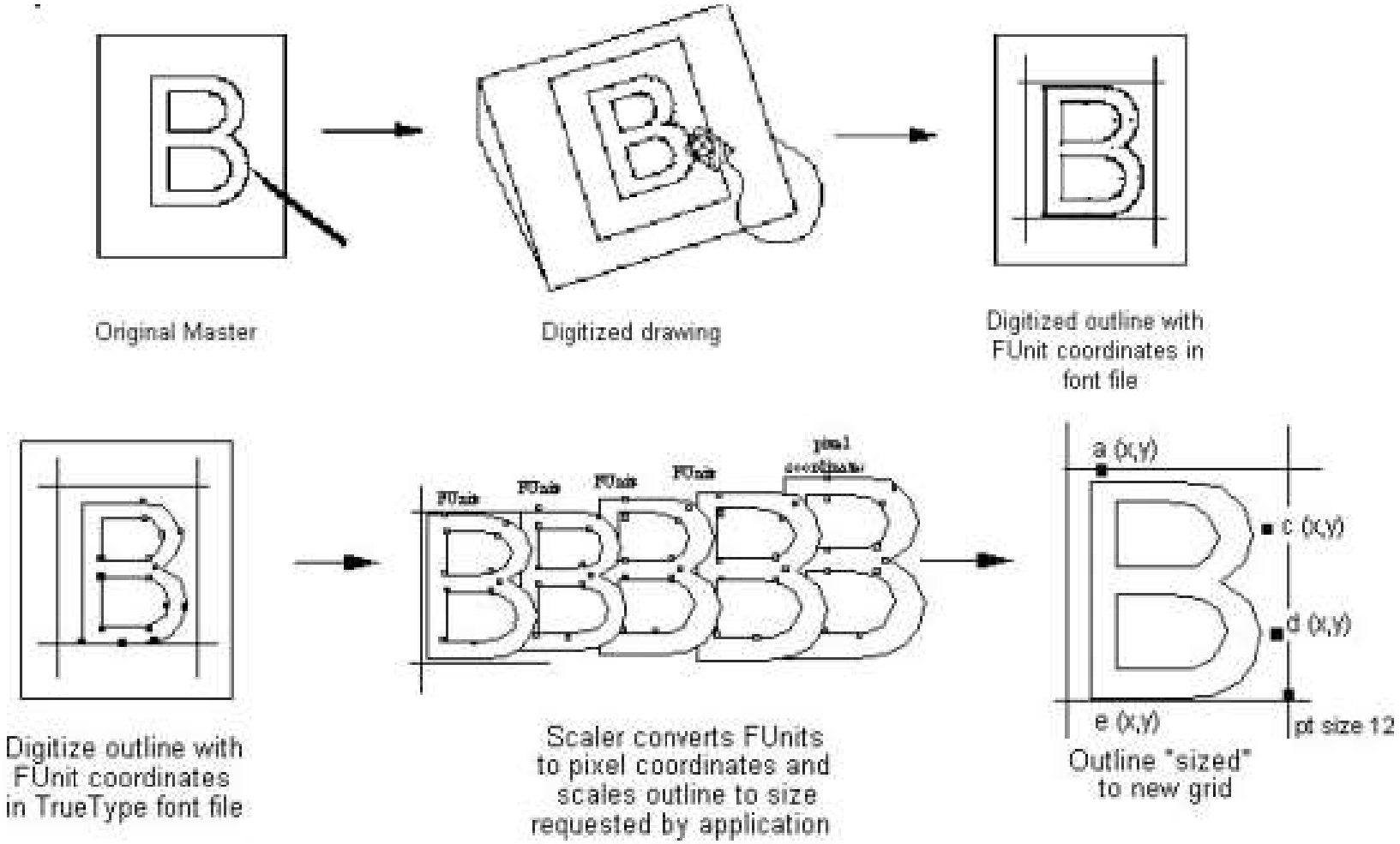




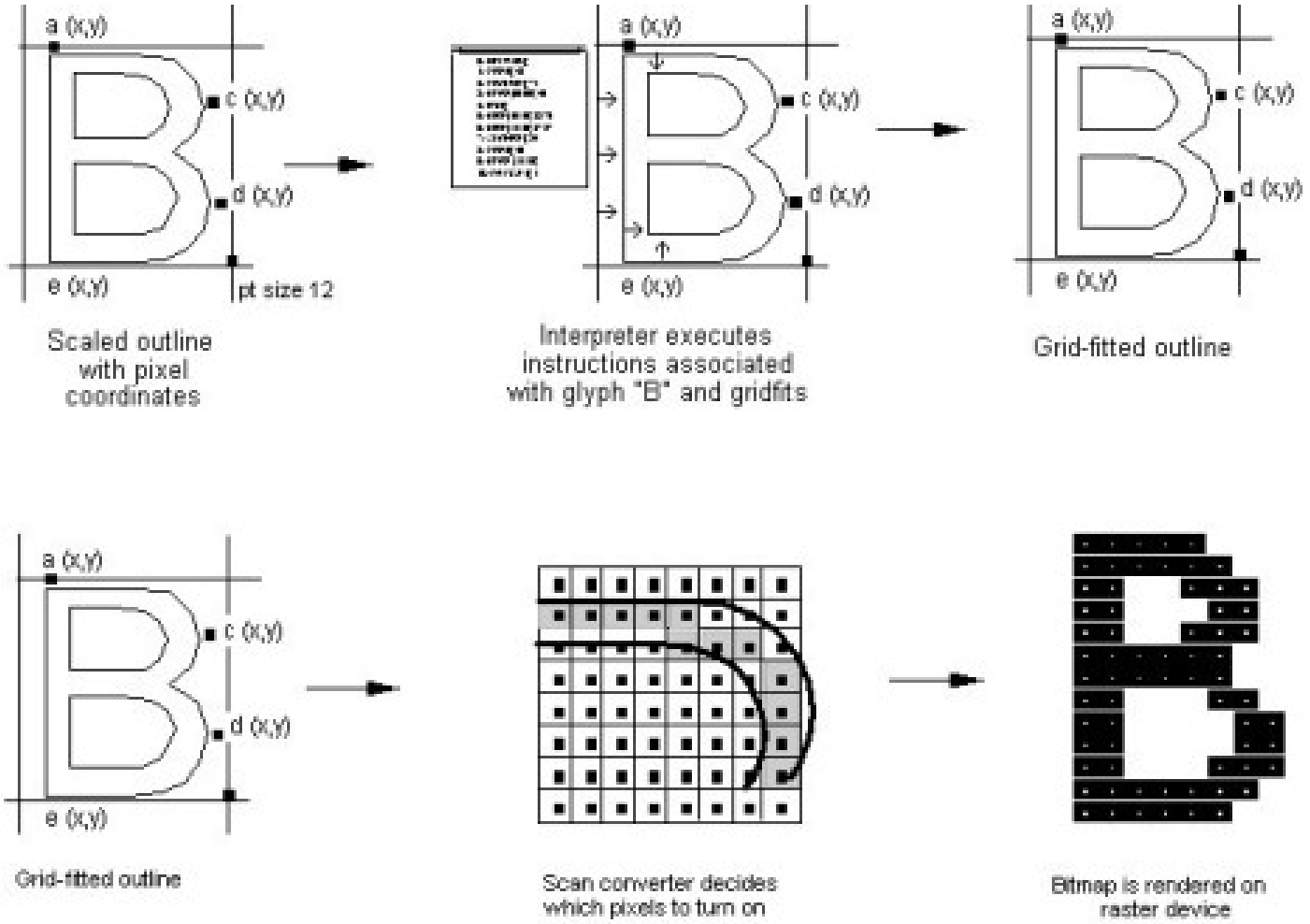
True Type Fonts

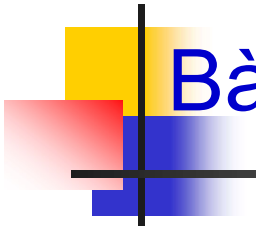
- MS Windows sử dụng TTF?
  - Nạp font từ tệp
  - Trình Scaler:
    - Ánh xạ tọa độ điểm điều khiển của glyph từ FUnit sang tọa độ thiết bị (pixel/dot). (Co dãn đường viền đến kích thước yêu cầu theo mật độ cho trước trên thiết bị ra).
  - Trình Interpreter:
    - Áp dụng 'hint' cho đường viền: biến đổi đường cong để hình thành đường viền đã hiệu chỉnh (grid-fitting).
  - Trình Scan-converter:
    - Tô trong đường viền đã hiệu chỉnh bằng các pixel để tạo bitmap cho chữ đặc.
  - Hiển thị / in các bitmap ký tự.

TTF: Từ thiết kế đến hiển thị



TTF: Từ thiết kế đến hiển thị





Bài tập

- Viết chương trình hiển thị ký tự 12x16 bằng font bitmap, hiển thị tên của mình (sinh viên)