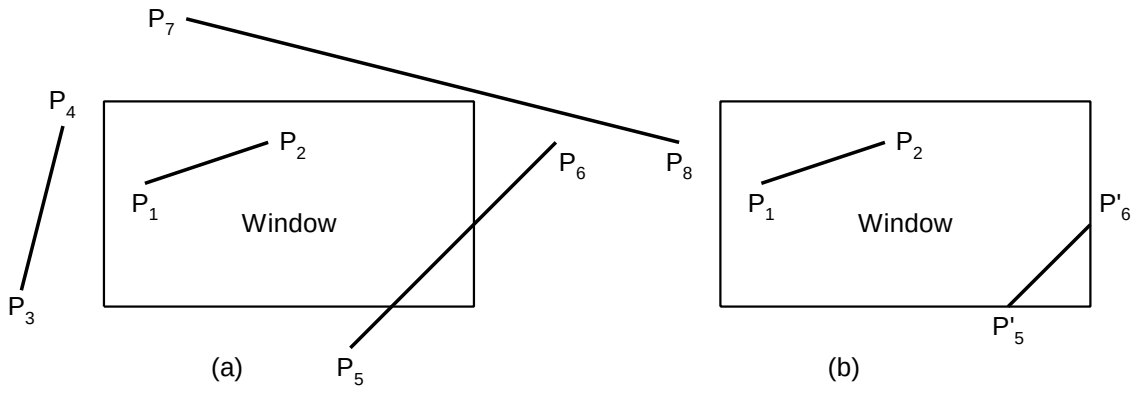
Các thuật toán xén điểm, đoạn thẳng

Dẫn nhập

- Thao tác loại bỏ các phần hình ảnh nằm ngoài một vùng cho trước được gọi là xén hình.
- Vùng được dùng để xén hình gọi là cửa sổ xén (clip window).
- Cho cửa sổ hình chữ nhật có tọa độ của các điểm dưới bên trái và điểm trên bên phải lần lượt là $(x_{\min}, y_{\min})$ và $(x_{\max}, y_{\max})$.
- Một điểm $P(x,y)$ được coi là nằm bên trong cửa sổ nếu thỏa hệ bất phương trình : $\begin{cases} x_{\min} \le x \le x_{\max} \\ y_{\min} \le y \le y_{\max} \end{cases}$.
- Bây giờ, ta sẽ xét bài toán xén đoạn thẳng được cho bởi hai điểm $P_1(x_1, y_1)$ và $P_2(x_2, y_2)$vào cửa sổ hình chữ nhật trên.



Vấn đề tối ưu hóa tốc độ

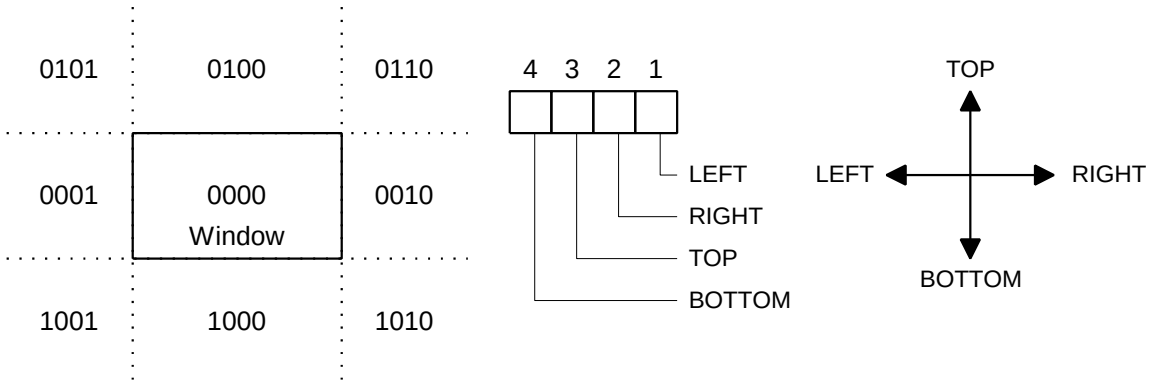
- Ý tưởng chung :
  - Đối với các đoạn thẳng đặc biệt như nằm hoàn toàn trong hoặc hoàn toàn bên ngoài cửa sổ (ví dụ như đoạn P_1P_2 và P_3P_4 trong hình trên) : không cần phải tìm giao điểm.
  - Đối với các đoạn thẳng có khả năng cắt cửa sổ : cần phải đưa ra cách tìm giao điểm nhanh.
- Nhận xét
  - Các đoạn thẳng mà có cả hai điểm nằm hoàn toàn trong cửa sổ thì cả đoạn thẳng nằm trong cửa sổ, đây cũng chính là kết quả sau khi xén (ví dụ như đoạn thẳng P_1P_2), mặt khác đối với các đoạn thẳng mà có hai điểm nằm về cùng một phía của cửa sổ thì luôn nằm ngoài cửa sổ và sẽ bị mất sau khi xén (ví dụ như đoạn thẳng P_3P_4).
  - Với các đoạn thẳng có khả năng cắt cửa sổ (ví dụ như đoạn thẳng P_5P_6 và P_7P_8) để việc tìm giao điểm nhanh cần rút gọn việc tìm giao điểm với những ***biên cửa sổ không cần thiết*** để xác định phần giao nếu có của đoạn thẳng và cửa sổ.
- Người ta thường sử dụng phương trình tham số của đoạn thẳng trong việc tìm giao điểm giữa đoạn thẳng với cửa sổ.

$$x = x_1 + t(x_2 - x_1) = x_1 + tDx, \quad Dx = x_2 - x_1$$
$$y = y_1 + t(y_2 - y_1) = y_1 + tDy, \quad Dy = y_2 - y_1, \qquad 0 \le t \le 1$$

- Nếu giao điểm ứng với giá trị t nằm ngoài đoạn [0,1] thì giao điểm đó sẽ không thuộc về cửa sổ.

Thuật toán Cohen - Sutherland

- Kéo dài các biên của cửa sổ, ta chia mặt phẳng thành chín vùng gồm cửa sổ và tám vùng xung quanh nó.



- Khái niệm mã vùng (area code)
  - ♦ Một con số 4 bit nhị phân gọi là mã vùng sẽ được gán cho mỗi vùng để mô tả vị trí tương đối của vùng đó so với cửa sổ.
  - ♦ Bằng cách đánh số từ 1 đến 4 theo thứ tự từ phải qua trái, các bit của mã vùng được dùng theo quy ước sau để chỉ một trong bốn vị trí tương đối của vùng so với cửa sổ bao gồm : trái, phải, trên, dưới. Ví dụ :
    Bit 1 : trái (LEFT)
    Bit 2 : phải (RIGHT)
    Bit 3 : trên (TOP)
    Bit 4 : dưới (BOTTOM)
  - ♦ Giá trị 1 tương ứng với vị trí bit nào trong mã vùng sẽ chỉ ra rằng điểm đó ở vị trí ương ứng, ngược lại bit đó sẽ được đặt bằng 0.
  - ♦ Các giá trị bit trong mã vùng được tính bằng cách xác định tọa độ của điểm (x, y) thuộc vùng đó với các biên của cửa sổ. Bit 1 được đặt là 1 nếu $x < x_{\min}$, các bit khác được tính tương tự.

Thuật toán

- Gán mã vùng tương ứng cho các điểm đầu cuối P_1, P_2 của đoạn thẳng cần xén lần lượt là c_1, c_2. Ta có nhận xét :

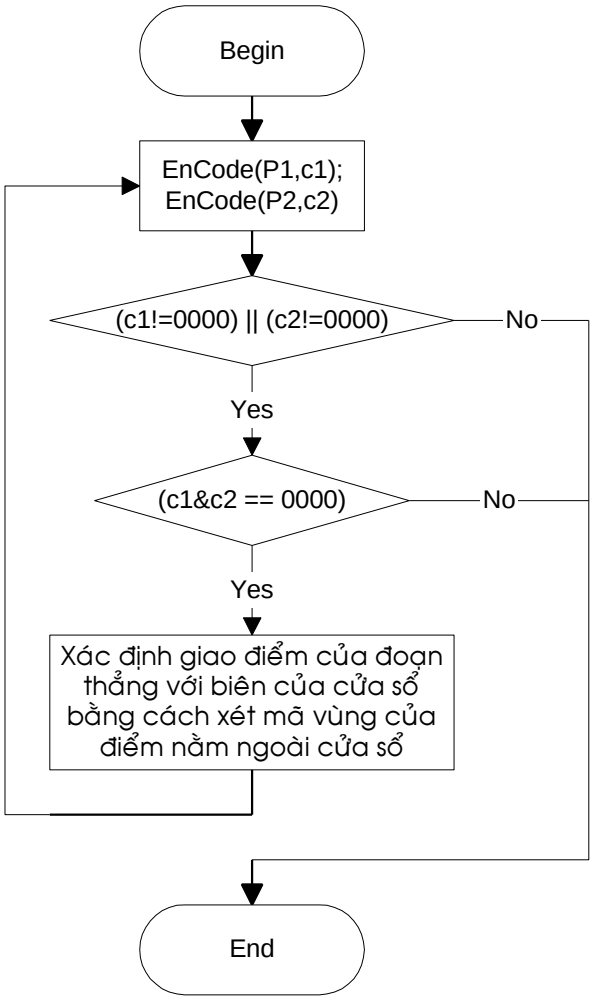
  - Các đoạn thẳng nằm hoàn toàn bên trong cửa sổ sẽ có $c_1 = c_2 = 0000$, ứng với các đoạn này, kết quả sau khi xén là chính nó.

  - Nếu tồn tại $k \in 1,..4$, sao cho với bit thứ k của c_1, c_2 đều có giá trị 1, lúc này đoạn thẳng sẽ nằm về cùng phía ứng với bit k so với cửa sổ, do đó nằm hoàn toàn ngoài cửa sổ. Đoạn này sẽ bị loại bỏ sau khi xén. Để xác định tính chất này, đơn giản chỉ cần thực hiện phép toán logic AND trên c_1, c_2. Nếu kết quả khác 0000, đoạn thẳng sẽ nằm hoàn toàn ngoài cửa sổ.

  - Nếu c_1, c_2 không thuộc về hai trường hợp trên, đoạn thẳng có thể hoặc không cắt ngang cửa sổ, chắc chắn sẽ tồn tại một điểm nằm ngoài cửa sổ, không mất tính tổng quát giả sử điểm đó là P_1. Bằng cách xét mã vùng của P_1 là c_1 ta có thể xác định được các biên mà đoạn thẳng có thể cắt để từ đó chọn một biên và tiến hành tìm giao điểm P_1' của đoạn thẳng với biên đó. Lúc này, đoạn thẳng ban đầu được xén thành P_1P_1'. Sau đó chúng ta lại lặp lại thao tác đã xét cho đoạn thẳng mới P_1P_1' cho tới khi xác định được phần nằm trong hoặc loại bỏ toàn bộ đoạn thẳng.

  - Các điểm giao với các biên cửa sổ của đoạn thẳng có thể được tính từ phương trình tham số. Ví dụ : tung độ y của điểm giao đoạn thẳng với biên đứng của cửa sổ có thể tính từ công thức $y = y_1 + m(x - x_1)$, trong đó x có thể là $x_{\min}$ hay $x_{\max}$.

Lưu đồ thuật toán Cohen - Sutherland



// Đoạn CT tính mã vùng

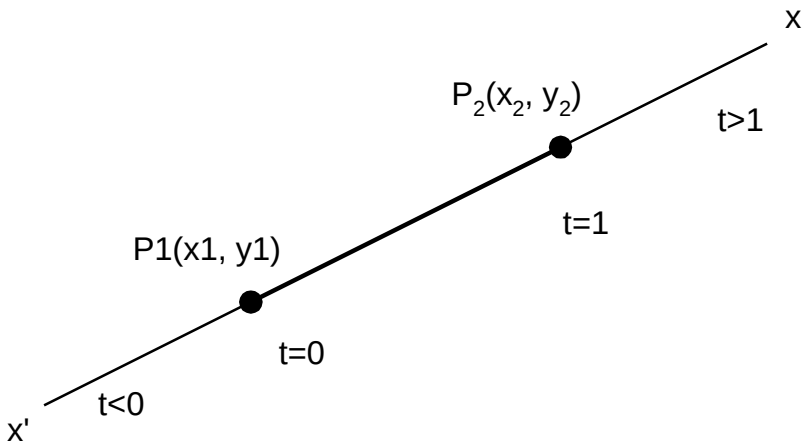
```
void EnCode(POINT p, CODE &c, RECT rWin)
{
        c = 0;
    if(p.x < rWin.Left)
        c |= LEFT;
    if(p.x > rWin.Right)
        c |= RIGHT;
    if(p.y > rWin.Top)
        c |= TOP;
    if(p.y < rWin.Bottom)
        c |= BOTTOM;
}
```

Thuật toán Liang - Barsky

- Thuật toán Liang-Barsky được phát triển dựa vào việc phân tích dạng tham số của phương trình đoạn thẳng.

$$x = x_1 + t(x_2 - x_1) = x_1 + tDx, \quad Dx = x_2 - x_1$$
$$y = y_1 + t(y_2 - y_1) = y_1 + tDy, \quad Dy = y_2 - y_1, \qquad 0 \le t \le 1$$

- Ứng với mỗi giá trị t, ta sẽ có một điểm P tương ứng thuộc đường thẳng.
  - Các điểm ứng với $t \ge 1$ sẽ thuộc về tia P_2x.
  - Các điểm ứng với $t \le 0$ sẽ thuộc về tia P_2x'.
  - Các điểm ứng với $0 \le t \le 1$ sẽ thuộc về đoạn thẳng P_1P_2.



- Tập hợp các điểm thuộc về phần giao của đoạn thẳng và cửa sổ ứng với các giá trị t thỏa hệ bất phương trình : $\begin{cases} x_{\min} \le x_1 + tDx \le x_{\max} \\ y_{\min} \le y_1 + tDy \le y_{\max} \\ 0 \le t \le 1 \end{cases}$

- Đặt
$$\begin{aligned} p_1 &= -Dx, & q_1 &= x_1 - x_{\min} \\ p_2 &= Dx, & q_2 &= x_{\max} - x_1 \\ p_3 &= -Dy, & q_3 &= y_1 - y_{\min} \\ p_4 &= Dy, & q_4 &= y_{\max} - y_1 \end{aligned}$$

- Lúc này ta viết hệ phương trình trên dưới dạng :

$$\begin{cases} p_k t \le q_k, & k = 1,2,3,4 \\ 0 \le t \le 1 \end{cases}$$

- Như vậy việc tìm đoạn giao thực chất là tìm nghiệm của hệ bất phương trình này. Có hai khả năng xảy ra đó là :
  - Hệ bất phương trình vô nghiệm, nghĩa là đường thẳng không có phần giao với cửa sổ nên sẽ bị loại bỏ.
  - Hệ bất phương trình có nghiệm, lúc này tập nghiệm sẽ là các giá trị t thỏa $t \in [t_1, t_2] \subseteq [0,1]$.

- Ta xét các trường hợp :
  - Nếu $\exists\ k \in \{1,2,3,4\} : (p_k = 0) \wedge (q_k < 0)$ thì rõ ràng bất phương trình ứng với k trên là vô nghiệm, do đó hệ vô nghiệm.
  - Nếu $\forall\ k \in \{1,2,3,4\} : (p_k \neq 0) \vee (q_k \ge 0)$ thì với các bất phương trình mà ứng với $p_k = 0$ là các bất phương trình hiển nhiên, lúc này hệ bất phương trình cần giải tương đương với hệ bất phương trình có $p_k \neq 0$.
  - Với các bất phương trình $p_k t \le q_k$ mà $p_k < 0$, ta có $t \ge q_k / p_k$.
  - Với các bất phương trình $p_k t \le q_k$ mà $p_k > 0$, ta có $t \le q_k / p_k$.

- Vậy nghiệm của hệ bất phương trình là $[t_1, t_2]$ với :

$$\begin{cases} t_1 = \max(\left\{\frac{q_k}{p_k}, p_k < 0\right\} \cup \{0\}) \\ t_2 = \min(\left\{\frac{q_k}{p_k}, p_k > 0\right\} \cup \{1\}) \\ t_1 \le t_2 \end{cases}$$

- Nếu hệ trên có nghiệm thì đoạn giao Q_1Q_2 sẽ là $Q_1(x_1 + t_1 Dx, y_1 + t_1 Dy), Q_2(x_1 + t_2 Dx, y_1 + t_2 Dy)$.

- Nếu xét thuật toán này ở khía cạnh hình học ta có :

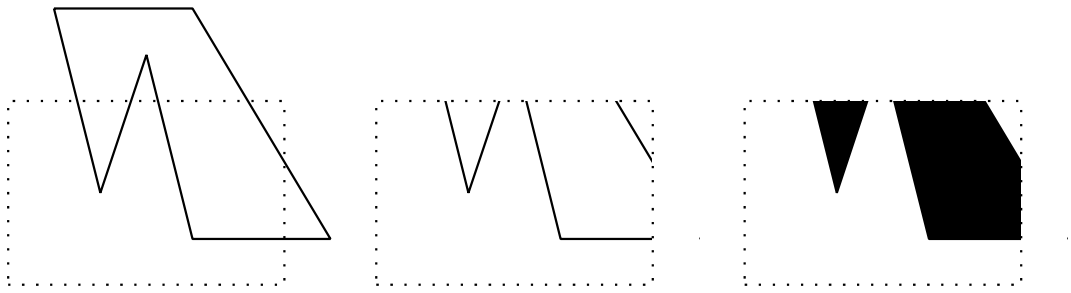
  - Trường hợp $\exists\ k \in \{1,2,3,4\} : (p_k = 0) \wedge (q_k < 0)$ tương ứng với trường hợp đoạn thẳng cần xét song song với một trong các biên của cửa sổ ($p_k = 0$) và nằm ngoài cửa sổ ($q_k < 0$) nên sẽ bị loại bỏ sau khi xén.

  - Với $p_k \neq 0$, giá trị $t = r_k = q_k / p_k$ sẽ tương ứng với giao điểm của đoạn thẳng với biên k kéo dài của cửa sổ. Trường hợp $p_k < 0$, kéo dài các biên cửa sổ và đoạn thẳng về vô cực, ta có đường thẳng đang xét sẽ có hướng đi từ bên ngoài vào bên trong cửa sổ. Nếu $p_k > 0$, đường thẳng sẽ có hướng đi từ bên trong cửa sổ đi ra. Do đó hai đầu mút của đoạn giao sẽ ứng với các giá trị t_1, t_2 được tính như sau : Giá trị t_1 chính là giá trị lớn nhất của các $r_k = q_k / p_k$ mà $p_k < 0$ (đường thẳng đi từ ngoài vào trong cửa sổ) và 0; giá trị t_2 chính là giá trị nhỏ nhất của các $r_k = q_k / p_k$ mà $p_k > 0$ (đường thẳng đi từ trong cửa sổ đi ra) và 1.

Thuật toán xén đa giác Sutherland - Hodgemand

Dẫn nhập

- Chúng ta có thể hiệu chỉnh các thuật toán xén đoạn thẳng để xén đa giác bằng cách xem đa giác như là một tập các đoạn thẳng liên tiếp nối với nhau. Tuy nhiên, kết quả sau khi xén nhiều khi lại là tập các đoạn thẳng rời nhau.
- Điều chúng ta mong muốn ở đây đó là kết quả sau khi xén phải là một các đa giác để sau này có thể chuyển thành các vùng tô.



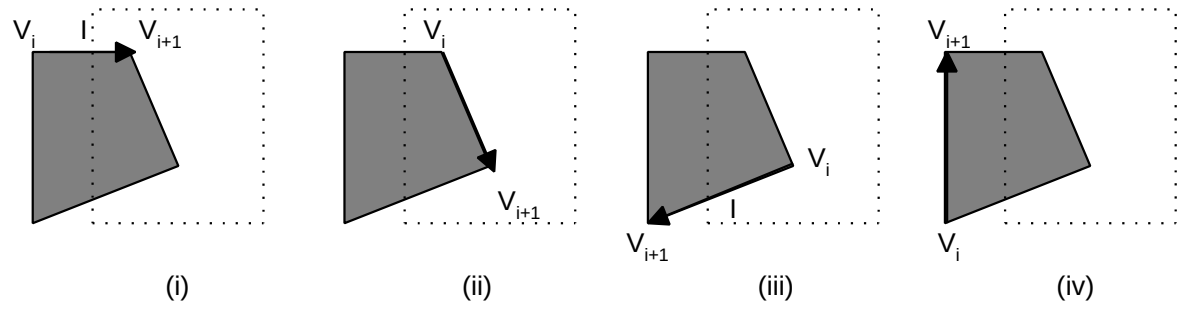
(a) (b) (c)

Thuật toán Sutherland - Hodgeman

- Thuật toán này sẽ tiến hành xén đa giác lần lượt với các biên cửa sổ. Đầu tiên, đa giác sẽ được xén dọc theo biên trái của cửa sổ, kết quả sau bước này sẽ được dùng để xén tiếp biên phải, rồi cứ tương tự như vậy cho các biên trên, dưới. Sau khi xén hết với bốn biên của cửa sổ, ta được kết quả cuối cùng.

- Với mỗi lần xén đa giác dọc theo một biên nào đó của cửa sổ, nếu gọi V_i, V_{i+1} là hai đỉnh kề cạnh V_iV_{i+1}, ta có 4 trường hợp có thể xảy ra khi xét từng cặp đỉnh của đa giác ban đầu với biên của cửa sổ như sau:

  - Nếu V_i nằm ngoài, V_{i+1} nằm trong, ta lưu giao điểm I của V_iV_{i+1} với biên của cửa sổ và V_{i+1}.
  - Nếu cả V_i, V_{i+1} đều nằm trong, ta sẽ lưu cả V_i, V_{i+1}.
  - Nếu V_i nằm trong, V_{i+1} nằm ngoài, ta sẽ lưu V_i và I.
  - Nếu cả V_i, V_{i+1} đều nằm ngoài, ta không lưu gì cả.



Cài đặt hàm xén đa giác theo một cạnh của cửa sổ

```
void ClipEdge(POINT *pIn, int N, POINT *pOut, int &Cnt, int Edge, RECT rWin)
{
    int FlagPrevPt = FALSE;
    Cnt = 0;
    POINT pPrev;

    pPrev = pIn[0];
    if(Inside(pPrev, Edge, rWin)) // Save point
    {
        pOut[Cnt] = pPrev;
        Cnt++;
        FlagPrevPt = TRUE;
    }

    for(int i=1; i<N; i++)
    {
        if(FlagPrevPt) // Diem bat dau nam trong
        {
            if(Inside(pIn[i], Edge, rWin)) // Save point P
            {
            pOut[Cnt] = pIn[i];
            Cnt++;
            }
            else // Save I
            {
            FlagPrevPt = FALSE;
            pOut[Cnt] = Intersect(pPrev, pIn[i], Edge, rWin);
            Cnt++;
            }
        }
        else // Diem bat dau canh nam ngoai
        {
            if(Inside(pIn[i], Edge, rWin)) // Save point I, P
            {
            FlagPrevPt = TRUE;
            pOut[Cnt] = Intersect(pPrev, pIn[i], Edge, rWin);
            Cnt++;
            pOut[Cnt] = pIn[i];
            Cnt++;
            }
        }
        pPrev = pIn[i];
    }

    // Neu Diem cuoi va dau giao voi bien cua cua so Save point I
    if(!(Inside(pIn[N], Edge, rWin) == Inside(pPrev, Edge, rWin)))
    {
        pOut[Cnt] = Intersect(pPrev, pIn[N], Edge, rWin);
        Cnt++;
    }
    pOut[Cnt] = pOut[0];
}// ClipEdge
```