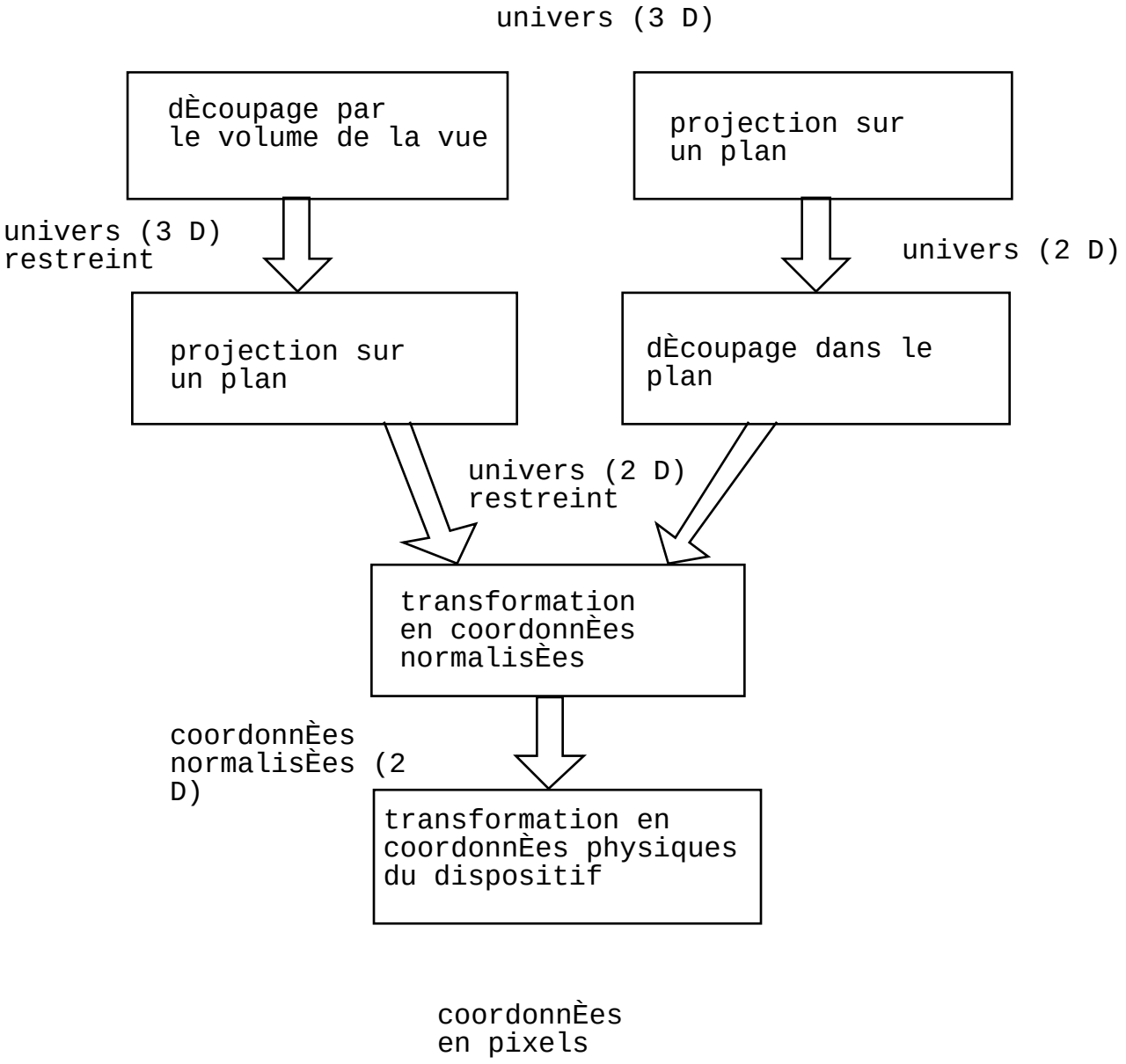
PROBLÈMES DE DÉCOUPAGE

- **Le pipeline de la vue**
- **Fenêtres et cadres**
- **Le découpage dans le plan**
- **Le découpage dans l'espace**
- **Les problèmes voisins**
- **La transformation fenêtre-cadre**

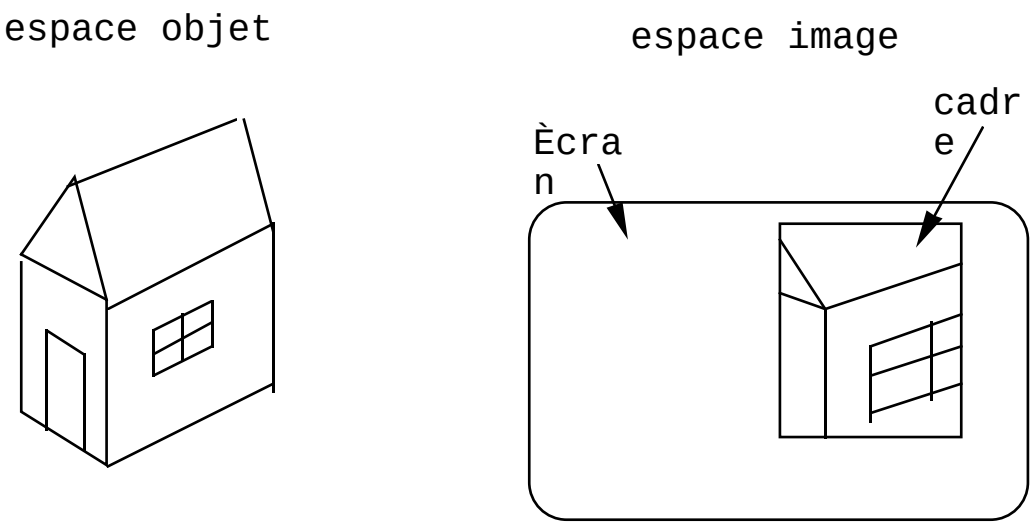
LE PIPELINE DE LA VUE



FENÊTRES-CADRES

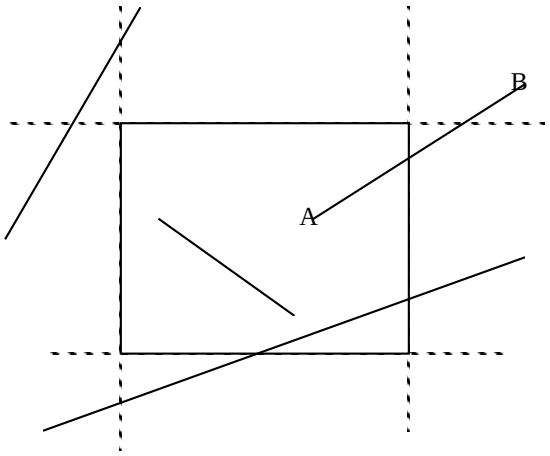
On distingue

- ***la fenêtre*** **(*window*) qui est un rectangle dans l'espace objet (l'univers) de l'application (une maison, un diagramme, une puce électronique, etc ...)**
- **la *clotûre* ou *cadre* (*viewport*) dans l'espace image (unité le pixel)**

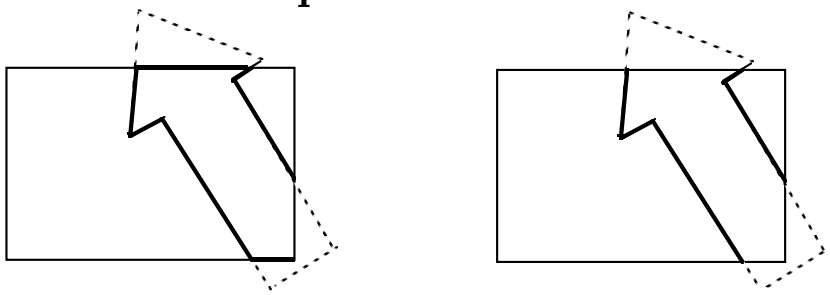


DÉCOUPAGE

Problème: étant donné un segment de droite et une fenêtre quelconque définie par un polygone simple, déterminer quelle partie du segment appartient à l'intérieur de la fenêtre.



Remarque: le découpage d'un polygone convexe par une fenêtre régulière ne se réduit pas à celui des segments dont il est composé



ALGORITHME DE COHEN-SUTHERLAND

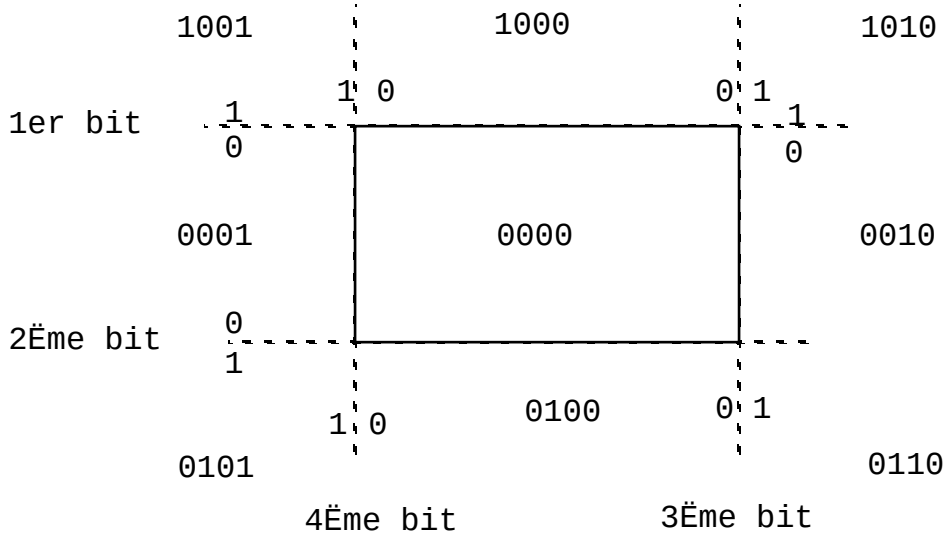
La fenêtre est un rectangle régulier. On considère les 9 régions du plan définies par les 4 droites supports des côtés. On code par un bit la position d'un point par rapport à chacune des droites (1 à l'extérieur, 0 à l'intérieur):

bit 1: droite "haut"

bit 2: droite "bas"

bit 3: droite "droit"

bit 4: droite "gauche"



Soit cA et cB sont les codes associés aux extrémités A et B du segment. Il y a deux cas faciles

- **cA = 0000 et cB= 0000: le segment est intérieur**
- **(cA and cB) $\neq$ 0000: alors le segment est extérieur**

Dans les autres cas, l'un au moins des points, est extérieur à la fenêtre. On annule alors l'un de ses bits égaux à 1 en calculant l'intersection du segment avec la droite correspondante. On répète ainsi la procédure (au plus 4 fois).

LA PROCÉDURE

Codage de la position d'un point par rapport aux 4 axes bottom, top, left, right. On suppose définis

```
type edge = (left, right, bottom, top);
outcode = set of edge;


procedure Code (x, y : integer; var c : outcode);
début
    c := [];
    si x < clipl alors
        c := [left]
    sinon si x > clipr alors
        c := [right];
    si y < clipb alors
        c := c + [bottom]
    sinon si y > clipt alors
        c := c + [top]
fin;
```

Découpe d'un segment d'extrémités (x1,y1) et (x2,y2) par une fenêtre definie par son côté inférieur gauche (clipl,clipb) et son côté supérieur droit (clipr,clipt).

```
procedure Sutherland(x1, y1, x2, y2, clipl, clipr, clipt, clipb: real);
var
   c, c1, c2 : outcode;
   x, y: real;
début
   code(x1, y1, c1);code(x2, y2, c2);
   tantque (c1 <> []) ou (c2 <> []) faire début
      si (c1 * c2) <> [] alors aller retour;
      c := c1;
      si c = [] alors c := c2;
      si left dans c alors début
         y := y1 + (y2 - y1) * (clipl - x1) /(x2 - x1);
         x := clipl
         fin
         sinon si right dans c alors début
            y := y1 + (y2 - y1) * (clipr - x1)/(x2 - x1);
            x := clipr
            fin
            sinon si bottom dans c alors début
               x := x1 + (x2 - x1) * (clipb - y1)/ (y2 - y1);
               y := clipb
               fin
               sinon si top dans c alors début
                  x := x1 + (x2 - x1) * (clipt - y1)/ (y2 - y1);
                  y := clipt
               fin;
```

```
        si c = c1 alors début
            x1 := x; y1 := y; code(x1, y1, c1)
        fin sinon début
            x2 := x; y2 := y; code(x2, y2, c2)
            fin
    fin;
    DrawLine(x1, y1, x2, y2);
retour:fin;
```

DÉCOUPAGE EN 3 D

Deux façons de procéder à la projection d'un objet en 3 D sur le plan de l'écran (traité ultérieurement).

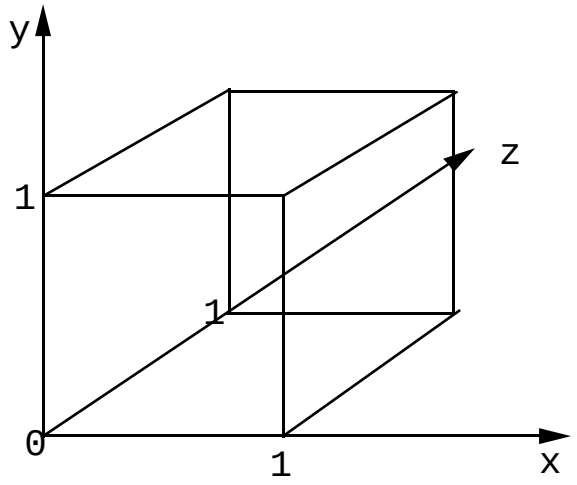
- **projection parallèle**
- **perspective**

Dans les deux cas on restreint la scène au *volume de la vue canonique*

- **éliminer les objets éloignés dont la taille dans la perspective, est inférieure au pixel**
- **extraire l'information significative pour l'application**

LE CAS DE LA PROJECTION

Cube unité (le côté avant inférieur gauche à l'origine et arêtes sont parallèles aux axes)



bit 1: égal à 1 **ssi le point est au-dessus** $y > 1$**.**

bit 2: égal à 1 **ssi le point est en-dessous** $y < 0$**.**

bit 3: égal à 1 **ssi le point est à droite** $x > 1$**.**

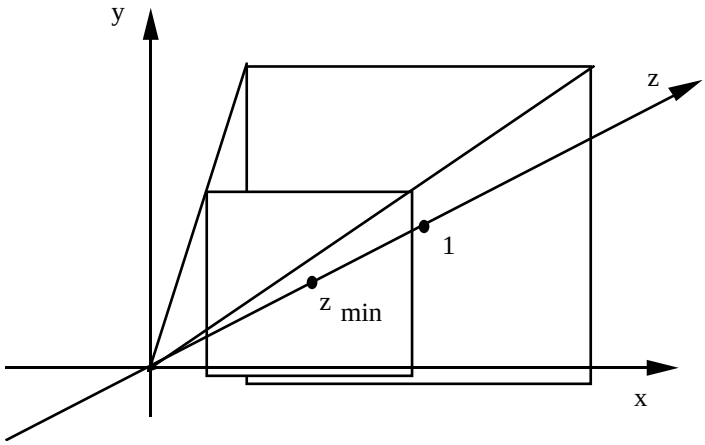
bit 4: égal à 1 **ssi le point est à gauche** $x < 0$**.**

bit 5: égal à 1 **ssi le point est devant** $z < 0$**.**

bit 6: égal à 1 **ssi le point est derrière** $z > 1$**.**

LE CAS DE LA PERSPECTIVE

Pyramide tronquée de sommet à l'origine, de base le carré de côté 2, parallèle au plan x, y, centré sur l'axe des z à la coordonnée $z = 1$ et tronquée en $z = zmin$.



Volume canonique dans le cas d'une perspective

bit 1: égal à 1 ssi le point est au-dessus $y > z$.

bit 2: égal à 1 ssi le point est en dessous $y < -z$.

bit 3: égal à 1 ssi le point est à droite $x > z$.

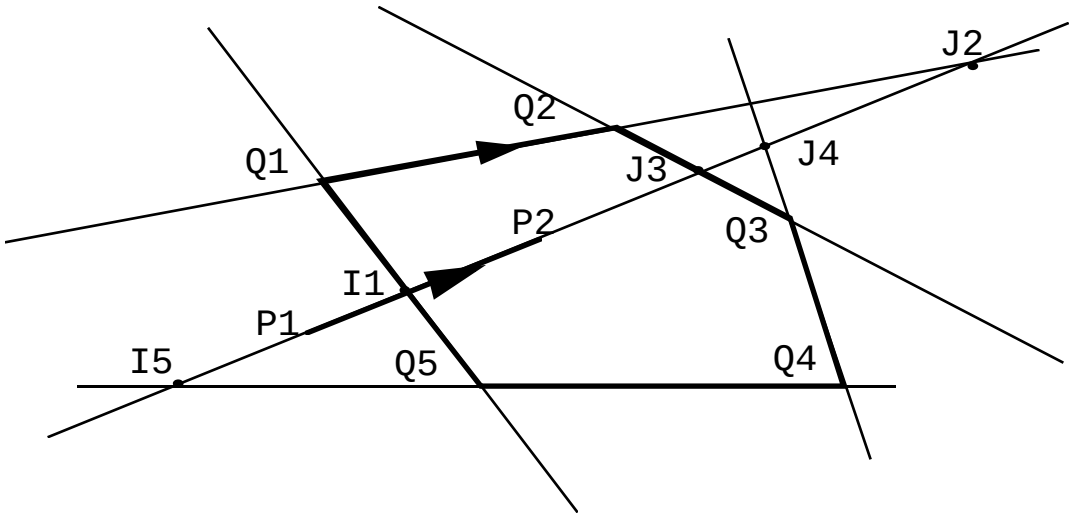
bit 4: égal à 1 ssi le point est à gauche $x < -z$.

bit 5: égal à 1 ssi le point est devant $z < zmin$.

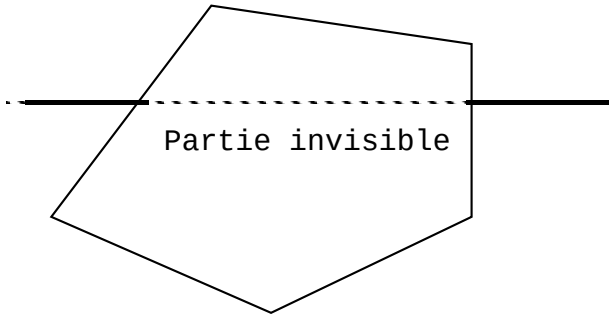
bit 6: égal à 1 ssi le point est derrière $z > 1$.

PROBLÈMES VOISINS

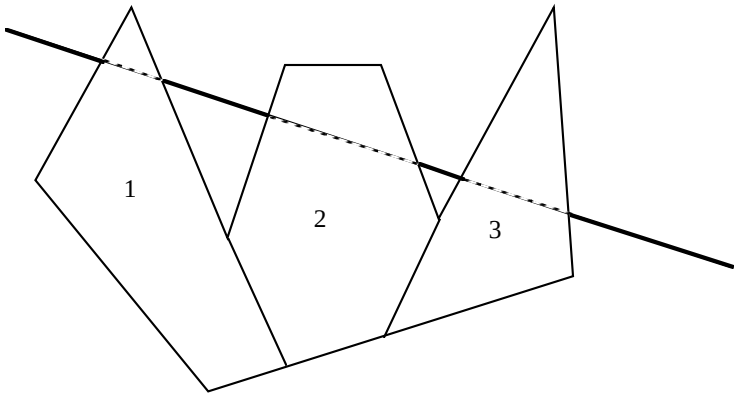
- **la fenêtre est un polygone *convexe*: algorithme de Cyrus-Beckman (Rappel: un point quelconque P de la droite support du segment P_1P_2 est défini par l'égalité: $P = P_1 + t\ P_1P_2$)**



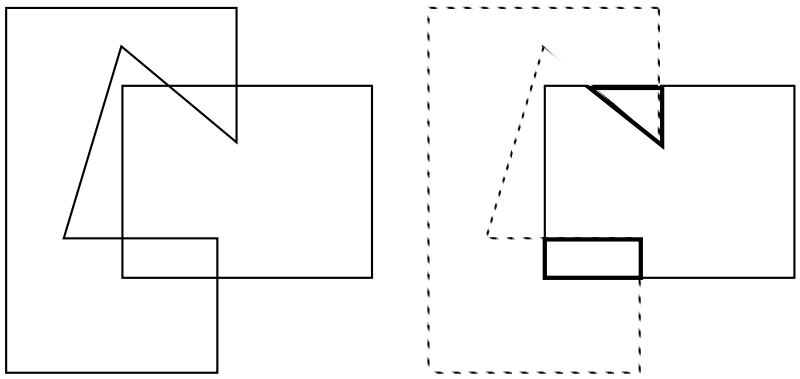
- **découpage *extérieur* d'un segment par un polygone convexe**

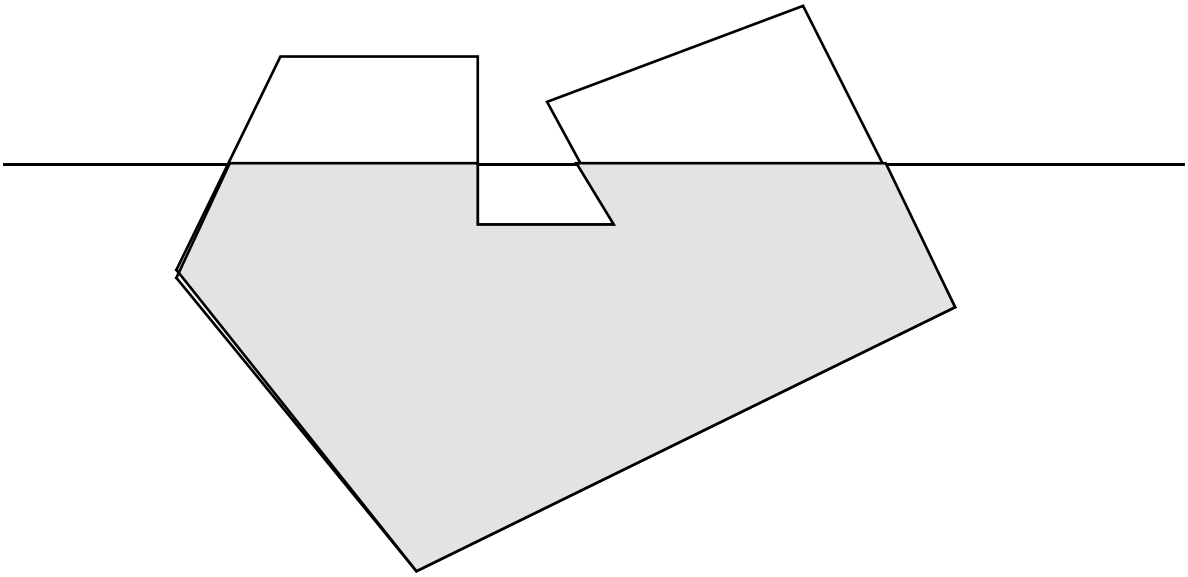


- **découpage d'un *segment* par un polygone *concave simple***

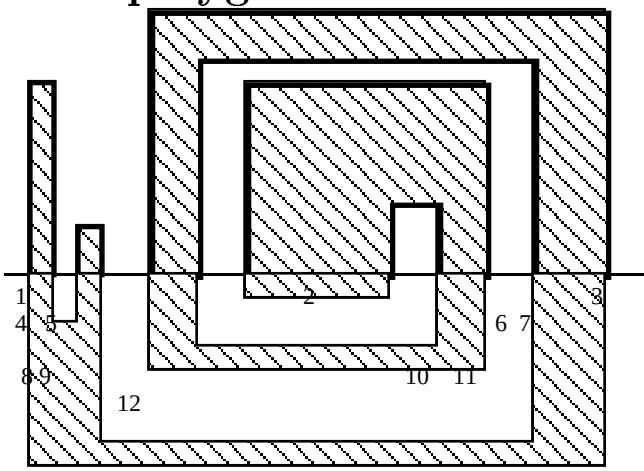


- **découpage d'un *polygone* par une fenêtre (algorithme de Sutherland-Hodgman)**





L'ordre des intersections dans le polygone et la droite



qui coupe doit être le même

Algorithme de Weiler-Atherton

Découpage d'un polygone quelconque par un autre polygone quelconque. Le résultat est en général un ensemble de polygones.

Entrée: un polygone découpant Coupant un polygone sujet Sujet

Sortie:

- **la suite PE1,PE2,...,PEn, des polygones extérieurs à Coupant obtenue par découpage de Sujet**
- **la suite PI1,PI2,...,PIm, des polygones extérieurs à Coupant obtenue par découpage de Sujet**

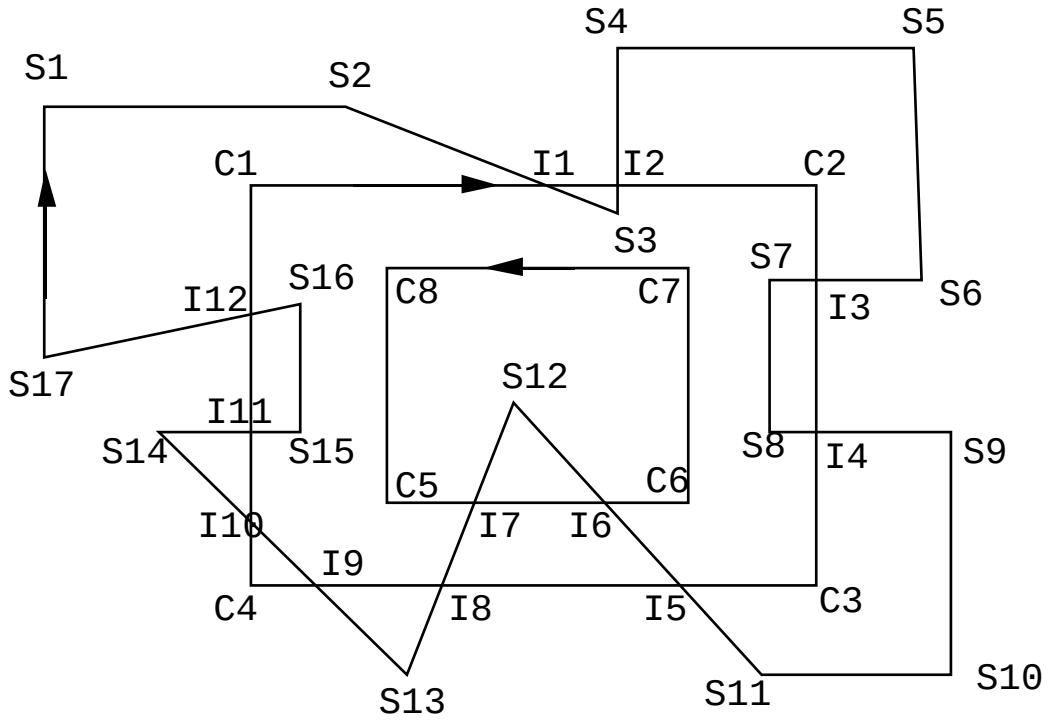
Hypothèses

- **Les polygones Coupant et Sujet sont des polygones généraux (éventuellement concaves et possédant des trous)**
- **Deux cotés consécutifs d'un polygone ne s'intersectent pas**
- **Le bord extérieur d'un polygone est parcouru dans le sens des aiguilles d'une montre et les bords intérieurs dans le sens inverse (le sens "naturel" de parcours).**

EXEMPLE

S = (S1,S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13, S14, S15, S16, S17)

C = (C1, C2, C3, C4)(C5, C6, C7, C8)



1 – Calcul des intersections 2 à 2 des côtés des polygones

```
pour tout côté CiCi+1 de Coupant faire
   pour tout côté SjSj+1 de Sujet faire
   début
      I= Intersection(CiCi+1,SjSj+1);
      Ajouter(I, Coupant);
      Ajouter(I, Sujet);
   fin;
```

S = (S1, S2, I1, S3, I2, S4, S5, S6, I3, S7, S8, I4, S9, S10, S11, I5, I6, S12, I7, I8, S13,I9, I10, S14, I11, S15, S16, I12, S17)

C = (C1, I1 I2, C2, I3, I4, C3, I5, I8, C4, I10, I11, I12)(C5, I7, I6, C6, C7, C8) 2 – Etablir la liste des intersections entrantes et sortantes de Sujet vers Coupant

Choisir un point quelconque du contour extérieur de Sujet, soit S1 par exemple et déterminer s'il est extérieur ou intérieur à Coupant. En suivant les sommets de Sujet déterminer si les points d'intersection sont entrants ou sortants. Si le polygone Sujet n'est pas épuisé prendre un autre sommet non visité du contour extérieur et recommencer. Répéter jusqu'à avoir visité tous les sommets de Sujet. Procéder identiquement avec l'ensemble des contours intérieurs.

Intersections entrantes: {I1, I3, I5, I7, I9, I11}

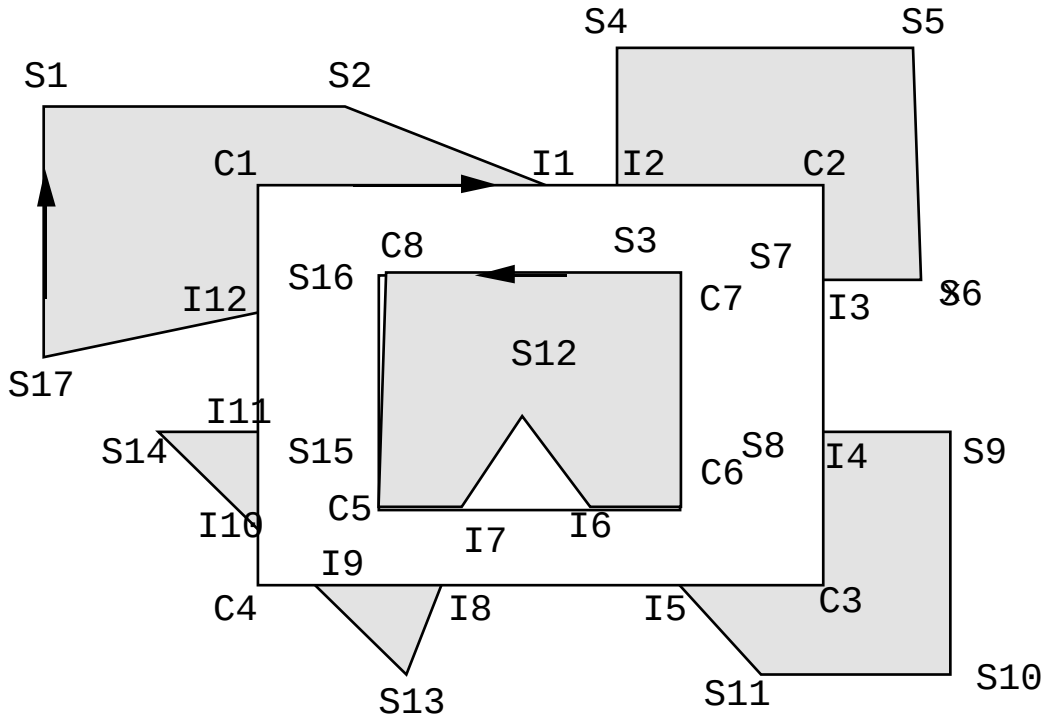
Intersections sortantes: {I2, I4, I6, I8, I10, I12}

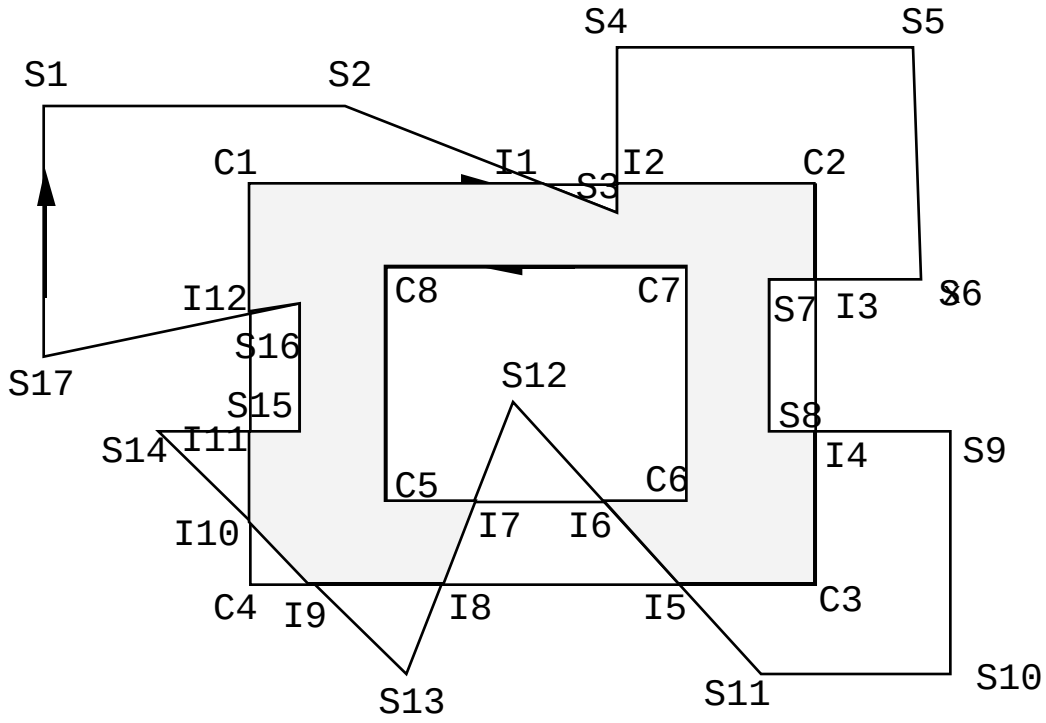
3 – Etablir la liste des polygones extérieurs

```
tantque la liste des intersections sortantes n'est pas vide faire
   début
      choisir une intersection sortante I;
      tantque le parcours ne revient pas en I faire
      début
         suivre Sujet dans le sens naturel jusqu'à la prochaine intersect
         supprimer J de la liste des intersections entrantes;
         suivre Coupant dans le sens inverse jusqu'à la prochaine
                  intersection K;
         supprimer K de la liste des intersections sortantes;
      fin;
      supprimer I de la liste des intersections sortantes;
   fin;
```

4 – Etablir la liste des polygones intérieurs

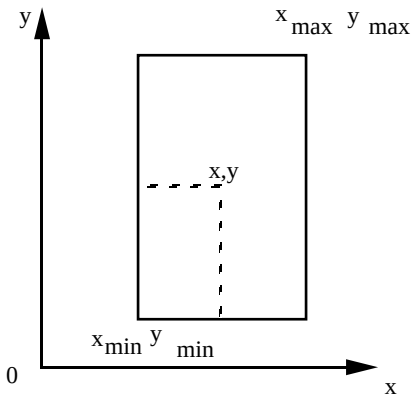
```
tantque la liste des intersections entrantes n'est pas vide faire
   début
      choisir une intersection entrante I;
      tantque le parcours ne revient pas en I faire
      début
         suivre Sujet dans le sens naturel jusqu'à la prochaine intersect
         supprimer J de la liste des intersections sortantes;
         suivre Coupant dans le sens naturel jusqu'à la prochaine
                  intersection K;
         supprimer K de la liste des intersections entrantes;
      fin;
      supprimer I de la liste des intersections entrantes;
   fin;
```


S4
S5
S1
S2
C1
I1
I2
C2
C8
S3
S7
S16
C7
S6
I12
I3
S12
S17
I11
S14
S15
S8
I4
S9
C6
C5
I10
I7
I6
I9
C4
I8
I5
C3
S13
S11
S10

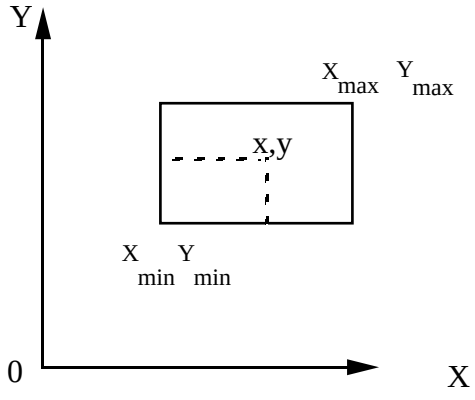

S4
S5
S1
S2
C1
I1
I2
C2
S3
C8
C7
S6
I12
S7
I3
S16
S12
S17
S15
S8
S14
I11
I4
S9
C5
C6
I10
I7
I6
C4
I9
I8
I5
C3
S13
S11
S10

PASSAGE DE LA FENÊTRE À L'ÉCRAN

Passer du point (x,y) de l'espace objet au point (X,Y) de l'espace image



Fenêtre dans
l'espace objet



Cadre dans
l'espace image

L'application qui fait passer de (x, y) à (X, Y) est définie par les égalités suivantes qui expriment simplement la conservation des rapports de longueur:

$$\frac{X - X_{min}}{X_{max} - X_{min}} = \frac{x - x_{min}}{x_{max} - x_{min}}$$
$$\frac{Y - Y_{min}}{Y_{max} - Y_{min}} = \frac{y - y_{min}}{y_{max} - y_{min}}$$

qui peut s'écrire:

$$X = s_x.x + d_x \text{ où } s_x = \frac{X_{max} - X_{min}}{x_{max} - x_{min}} \text{ et } d_x = X_{min} - x_{min}s_x$$

$$Y = s_y.y + d_y \text{ où } s_y = \frac{Y_{max} - Y_{min}}{y_{max} - y_{min}} \text{ et } d_y = Y_{min} - y_{min}s_y$$

Représentation matricielle en coordonnées homogènes (composition d'une homothétie et d'une translation)

$$\begin{pmatrix} s_x & 0 & d_x \\ 0 & s_y & d_y \\ 0 & 0 & 1 \end{pmatrix}$$

En fait on passe de l'espace objet à l'espace image par l'intermédiaire de l'espace normalisé (carré de côté unité). On doit donc appliquer deux fois l'opération précédente.