

Infographie - M1

Chapitre 3 - Fenêtrage

V. Padovani

Equipe Preuves, Programmes et Systèmes, Université Paris 7

Dans sa formulation la plus générale, le *fenêtrage* (ou “clipping”) est le problème consistant, étant donné un certain objet et une certaine région de l’espace, à calculer la portion de l’objet intérieure à cette région. Ce problème intervient en particulier dans le calcul de la “partie utile” d’une scène 3D, c’est-à-dire l’élimination des éléments nécessairement non vus par l’observateur.

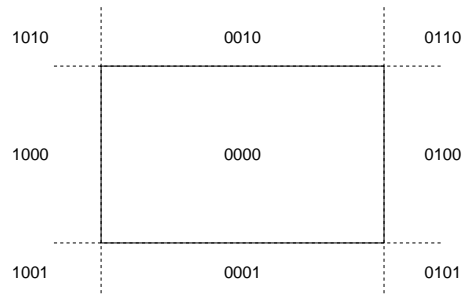
On s’intéressera ici à trois versions de ce problème, en 2D : déterminer la partie d’un segment intérieure à un rectangle (algorithme de Cohen-Sutherland) ; déterminer la partie d’un segment intérieure à un polygone convexe (algorithme de Cyrus-Beck) ; calculer les portions d’un polygone intérieures à un autre polygone (algorithme de Weiler-Atherton).

1 Algorithme de Cohen-Sutherland

Cet algorithme permet de calculer le fenêtrage d’un segment par un rectangle.

1.1 Code de la position d’un point

L’algorithme de Cohen-Sutherland utilise une fonction auxiliaire calculant, pour un rectangle R et un point P donné, un entier C compris entre les nombres binaires 0000 et 1111, qui indique le placement de ce point relativement aux droites passant par chaque bord du rectangle R . On peut par exemple fixer la convention suivante :



- le bit 1 (0001) de C est à 1 si P est strictement sous R
- le bit 2 (0010) de C est à 1 si P est strictement au dessus R
- le bit 3 (0100) de C est à 1 si P est strictement à droite de R
- le bit 4 (1000) de C est à 1 si P est strictement à gauche de R

Les bits 1, 2, 3, et 4 d'un code seront désignés respectivement comme ses bits B, H, D et G. Les propriétés suivantes sont facilement vérifiables :

1. Si le code d'un point est nul, il est intérieur à R .
2. Si le "et" bit à bit des codes de P_1 et P_2 est non nul, alors le segment $[P_1P_2]$ est extérieur à R .

1.1.1 L'algorithme

Etant donnés un rectangle R et un segment $S = [P_1 P_2]$,

- Soient C_1 et C_2 les codes des positions de P_1 et P_2 relativement à R .
- Tant que C_1 et C_2 n'indiquent pas que S est intérieur ou extérieur à R , faire :
 1. Soit C égal à C_1 si C_1 est non nul, C_2 sinon.
 2. On note ℓ_S la droite passant par S , et ℓ_H , ℓ_B , ℓ_G , ℓ_D les droites passant par chacun des bords du rectangle.
 Soit P le point défini comme suit :
 - si le bit G de C vaut 1, P vaut $\ell_S \cap \ell_G$.
 - sinon, et si le bit D de C vaut 1, P vaut $\ell_S \cap \ell_D$.
 - sinon, et si le bit H de C vaut 1, P vaut $\ell_S \cap \ell_H$.
 - sinon, et si le bit B de C vaut 1, P vaut $\ell_S \cap \ell_B$.
 3. Si C vaut C_1 alors
 - redéfinir P_1 comme P
 - redéfinir C_1 comme le code de la position de P
 sinon,
 - redéfinir P_2 comme P
 - redéfinir C_2 comme le code de la position de P

La propriétés suivantes sont vérifiables : à chaque itération, P appartient bien au segment $[P_1P_2]$ courant, et le code de P contient strictement moins de bits allumés que C ; au bout de ce traitement, le segment $[P_1P_2]$ est soit intérieur, soit extérieur à R .

Noter qu'à l'étape (2), le calcul de P est facilité par le fait que la droite dont on calcule l'intersection avec ℓ_S est un bord du rectangle, et est donc soit verticale, soit horizontale.

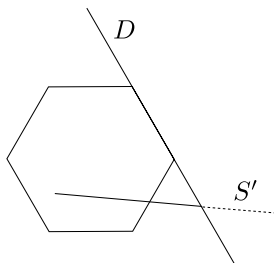
1.2 Algorithme de Cyrus-Beck

L'algorithme de Cyrus-Beck permet de fenêtrer un segment par un polygone *convexe*. On peut supposer sans perte de généralité que tous les sommets du polygone sont distincts (en fusionnant les sommets égaux), qu'il contient au moins trois sommets, et qu'il ne contient pas trois sommets successifs alignés (en éliminant dans ce cas le sommet central). On supposera également que le segment à fenêtrer n'est pas réduit à un point - auquel cas, un autre algorithme (un test d'intériorité) doit être utilisé.

Le principe de l'algorithme est le suivant. Soit S le segment à fenêtrer. Les deux propriétés ci-dessous sont facilement vérifiables :

(1) Etant donné une arête du polygone, soit D la droite contenant cette arête. Le polygone étant convexe, tous les sommets du polygone sont du même côté de D (au sens faible, les deux sommets de l'arête étant sur D).

Soit S' la portion de S formée de tous les points de S qui ne sont pas du même côté de D que les sommets du polygone. Tous les points de S' sont nécessairement extérieurs au polygone, et l'on peut redéfinir S comme $S - S'$.



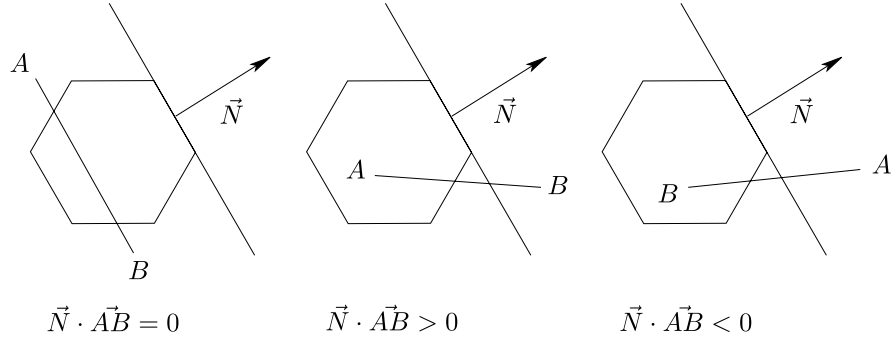
(2) En effectuant la transformation précédente pour chaque arête du polygone, le segment S final est égal au fenêtrage du segment initial par le polygone.

A l'étape (1), il faut, pour calculer S' , déterminer bien sûr si D intersecte S et, le cas échéant, calculer ce point d'intersection. Le simple calcul de ce point n'est cependant pas suffisant pour déterminer quelle portion de S doit être éliminée : il faut aussi savoir de quel côté de D se situent les points du polygone.

L'algorithme de Cyrus-Beck résout cette difficulté en précalculant, pour chaque arête du polygone, une normale à cette arête (un vecteur perpendiculaire à l'arête) allant vers l'extérieur du polygone (en partant d'un point quelconque de l'arête). Il est facile de voir que si $S = [AB]$, si D est la droite contenant une arête donnée, et si \vec{N} est une normale à cette arête allant vers l'extérieur, alors :

- si $\vec{N} \cdot \vec{AB} = 0$, l'arête est parallèle au segment,
- si $[AB]$ et D s'intersectent et si $\vec{N} \cdot \vec{AB} > 0$, B est extérieur au polygone,

- si $[AB]$ et D s'intersectent et si $\vec{N} \cdot \vec{AB} < 0$, A est extérieur au polygone,



On suppose dans la suite le polygone formé de la suite de sommets P_0, \dots, P_{n-1} (avec par convention $P_n = P_0, P_{n+1} = P_1$, etc.). Soit $[AB]$ le segment à fenêtrer.

1.2.1 Calcul préliminaire des normales

Pour chaque $i \in [0 \dots n[$, le calcul d'une normale \vec{N}_i à l'arête $[P_i P_{i+1}]$ peut se faire comme suit. Soit $\vec{U} = \vec{P_i P_{i+1}}$. Les points du polygone sont supposés distincts, dont \vec{U} est non nul. Soit $\vec{V} = (-U_y, U_x)$. On a $\vec{U} \cdot \vec{V} = 0$, dont \vec{V} est un vecteur non nul orthogonal à l'arête. On calcule ensuite le produit scalaire $\vec{V} \cdot \vec{P_i P_{i+2}}$. Il ne peut pas être nul, sinon P_i, P_{i+1}, P_{i+2} seraient alignés. S'il est négatif, l'angle entre \vec{V} et $\vec{P_i P_{i+2}}$ dépasse l'angle droit, donc \vec{V} va vers l'extérieur du polygone, et l'on pose $\vec{N}_i = \vec{V}$. S'il est positif, on pose $\vec{N}_i = -\vec{V}$.

1.2.2 Représentation paramétrique du segment et des intersections

Le segment $[AB]$ est représentable sous forme paramétrique : il s'agit de l'ensemble des points de coordonnées $S(t) = A + t \times (B - A)$ pour t variant entre 0 et 1 inclus. Pour chaque $i \in [0, \dots, n[$, le réel t_i tel que $S(t_i)$ appartienne à la droite $(P_i P_{i+1})$ est l'unique réel tel que l'on ait $\vec{N}_i \cdot (P_i \vec{S}(t_i)) = 0$, c'est-à-dire $\vec{N}_i \cdot (P_i \vec{A} + AS(t_i)) = 0$, soit encore $\vec{N}_i \cdot (P_i \vec{A} + t \times \vec{AB}) = 0$, ce qui donne :

$$t_i = \frac{\vec{N}_i \cdot \vec{AP_i}}{\vec{N}_i \cdot \vec{AB}}$$

Noter que t_i n'est défini que si $\vec{N}_i \cdot \vec{AB} \neq 0$, c'est-à-dire si $[AB]$ n'est pas parallèle à l'arête $n^o i$.

1.2.3 Algorithme de Cyrus Beck

1. On précalcule les normales $\vec{N}_0, \dots, \vec{N}_{n-1}$ comme ci-dessus.
2. On pose $t_E = 0, t_S = 1$.

3. Pour chaque $i \in [0, \dots, n]$,
 - (a) On calcule $\vec{N}_i \cdot \vec{AB}$.
 Si cette quantité est nulle, on passe au i suivant.
 Sinon, on calcule t_i comme ci-dessus, $t_i = (\vec{N}_i \cdot \vec{AP}_i) / (\vec{N}_i \cdot \vec{AB})$.
 - (b) Si $\vec{N}_i \cdot \vec{AB} > 0$ on redéfinit t_S comme $\min(t_S, t_i)$.
 Sinon, on redéfinit t_E comme $\max(t_E, t_i)$.
 - (c) Si $t_E > t_S$, le segment est totalement extérieur :
 on interrompt le traitement, en renvoyant une valeur singulière.
4. On renvoie le couple (t_E, t_S) .

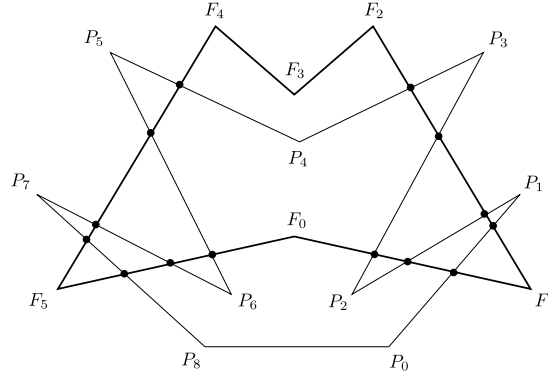
En fin de traitement, et si une partie au moins du segment est intérieure au polygone, cette partie est $\{S(t) \mid t_E \leq t \leq t_S\}$. Si une interruption s'est produite, le segment est totalement extérieur au polygone. A l'étape (3), et à chaque itération, les propriétés suivantes sont vérifiables :

1. Tous les points de $\{S(t) \mid 0 \leq t < t_E\}$ et $\{S(t) \mid t_S < t \leq 1\}$ sont extérieurs au polygone,
2. En (a), si $\vec{N}_i \cdot \vec{AB} = 0$, le segment est parallèle à l'arête courante, et celle-ci n'intervient pas dans le calcul de la partie visible.
3. En (b), si $\vec{N}_i \cdot \vec{AB} > 0$, les points de $\{S(t) \mid t_i < t \leq 1\}$ sont extérieurs au polygone. Sinon, c'est-à-dire si $\vec{N}_i \cdot \vec{AB} < 0$, les points de $\{S(t) \mid 0 \leq t < t_i\}$ sont extérieurs au polygone. Dans tous les cas, les nouvelles valeurs de t_E, t_S vérifient encore la propriété (1).
4. Si $t_E > t_S$, le segment est totalement extérieur au polygone (toujours à cause de (1)).

2 Algorithme de Weiler-Atherton

Cet algorithme résout le problème du fenêtrage d'un polygone par un autre. Soit $F = (F_0, \dots, F_{n-1})$ le polygone de fenêtrage (avec par convention $F_n = F_0$, $F_{n+1} = F_1$, etc.), et soit $P = (P_0, \dots, P_{m-1})$ le polygone à fenêtrer par F (avec encore, par convention $P_m = P_0$, $P_{m+1} = P_1$, etc.).

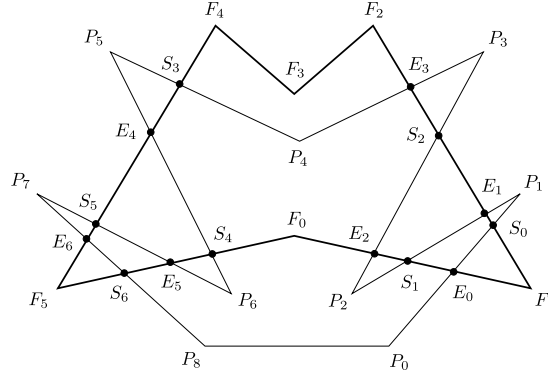
On suppose que les sommets de chaque polygone sont énumérés en parcourant son bord dans le sens *direct* (le sens inverse des aiguilles d'une montre) : si par exemple F_i est le sommet de F le plus en bas et le plus à droite de F , le produit en croix $\vec{F_{i-1}F_i} \times \vec{F_iF_{i+1}}$ est positif. Ou encore, plus informellement, en se déplaçant le long du bord de F en suivant la suite F_0, \dots, F_{n-1} , l'intérieur de F est toujours vers la gauche.



2.1 Etape 1 : calcul des points d'intersection et étiquetage

On commence par ajouter aux polygones chaque point d'intersection entre les arêtes des deux polygones - en négligeant les couples d'arêtes superposées, c'est-à-dire dont l'intersection est un segment, ainsi que les points d'intersection égaux à l'un des sommets de F ou de P .

En se servant de l'hypothèse faite sur l'énumération des sommets de F , on détermine d'autre part pour chaque intersection si elle est *entrante* ou *sortante*. Un point $I = [P_i P_{i+1}] \cap [F_j F_{j+1}]$ est entrant si P_i est à l'extérieur de F (avec dans ce cas, P_{i+1} nécessairement intérieur à F), et sortant sinon. On sait que l'intérieur de F est vers la gauche en se plaçant en F_j et en regardant vers F_{j+1} . Le point P_i est donc extérieur à F si $F_j \vec{F}_{j+1} \times F_j \vec{P}_i < 0$, et intérieur sinon.



Ces points d'intersection sont insérés dans la liste des sommets de F , P . On obtient deux nouveaux polygones étendus F , P . Si aucun point d'intersection n'est ajouté à cette étape, le polygone P est extérieur à F si au moins un point de P est strictement extérieur à F , et intérieur à F sinon. Nous verrons au cours suivant comment implémenter un test d'intériorité générique permettant de résoudre ce cas particulier.

2.2 Etape 2 : calcul des fragments intérieurs

1. Soit \mathcal{E} la liste des intersections entrantes calculées à l'étape 1.
2. Tant que \mathcal{E} n'est pas vide, faire :
 - (a) soit E_0 un élément de \mathcal{E} .
 - (b) en partant de $E = E_0$,
 en supprimant de \mathcal{E} chaque intersection entrante visitée,
 en mémorisant chaque sommet visité,
 faire :
 - aller dans P de E jusqu'à la première intersection sortante,
 (re)définir S comme celle-ci,
 - aller dans F de S jusqu'à la première intersection entrante,
 redéfinir E comme celle-ci,
 tant que E n'est pas égal à E_0 ,
 - (c) stocker la liste des sommets mémorisés en (b) comme un polygone.

Dans l'exemple dessiné ci-dessus, les trois polygones extraits par la boucle interne sont de la forme suivante :

