Clipping 2D = Fenêtrage 2D= Découpage

Pourquoi le clipping

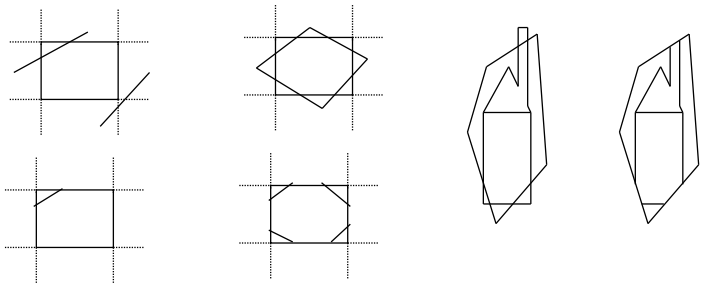
Que faire si l'utilisateur demande de tracer une droite qui sort de la fenêtre ?

Ceci arrive souvent : par exemple pour faire un Zoom.

Définition

On appelle clipping, tout traitement permettant de réduire le dessin d'un objet graphique à une région précise de l'écran.

Cette région est classiquement un rectangle mais peut être de toute autre forme.



Problème

Etant donné un polygone à afficher dans une fenêtre (rectangulaire ou autre), comment ne parcourir que les parties du polygone qui sont à l'intérieur de la fenêtre ?

Les méthodes *de découpage varient suivant le type de formes graphiques à traiter :*

- Points
- Segments de droite
- Polygones
- Courbes
- …

Les algorithmes de découpage dépendent de la forme de la clôture :

- Rectangle
- Polygone convexe
- Polygone concave
- Polygone avec trous
- …

Remarque

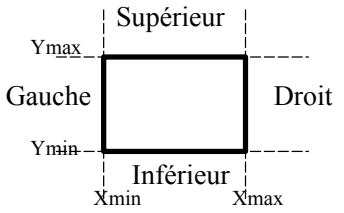
Le découpage se fait sur tous les objets composant la scène alors que l'affichage ne traite que les parties résultant du découpage.

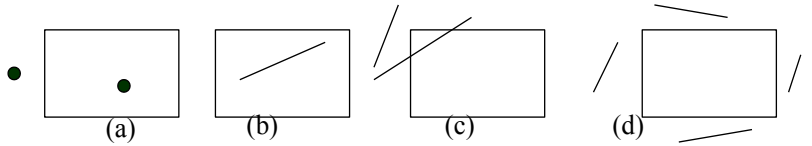
Découpage de segments de droite

La primitive géométrique la plus utilisée en graphique est le segment de droite, pour lequel de nombreux algorithmes de découpage existent.

Définition

- Une fenêtre en 2D est définie par ses côtés :
  - le côte gauche (L)
  - le côte droit (R)
  - le côté supérieur (T)
  - le côté inférieur (B)





- Figure (a) : Un point M(x, y) du plan est intérieur à la fenêtre si $x_{min} <= x <= x_{max}$ et $y_{min} <= y <= y_{max}$

  Le signe égal (=) signifie que les points du bord de la fenêtre sont inclus dans la fenêtre.

- Figure (b) : Les segments de droite sont situés à l'intérieur de la fenêtre et sont donc visibles, si leurs deux extrémités sont toutes deux intérieures à cette fenêtre.

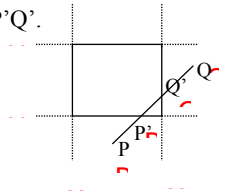
- Figure (c) : Si les deux extrémités d'un segment sont extérieures à la fenêtre, cela ne signifie pas que ce segment est complètement extérieur à cette fenêtre. Le segment peut être partiellement visible (cas du segment gh).

- Figure (d) : Si les deux extrémités d'un segment sont situées toutes deux à droite, toutes deux à gauche, toutes deux au-dessus ou toutes 2 au-dessous de la fenêtre, ce segment est considéré complètement extérieur à la fenêtre et par conséquent invisible.

Les algorithmes de découpage sont basés sur l'examen des extrémités du segment pour permettre la discrimination rapide des cas triviaux

Algorithme simple de visibilité

Objectif : Découper le segment PQ. Cela signifie déterminer les extrémités du segment P'Q'.



Pour chaque segment

Si le segment **est totalement visible on affiche le segment**

Si le segment **est Invisible on ne fait rien**

Si Le segment **est partiellement visible on d**étermine les intersections puis on affiche la partie visible

Finpour

Remarque

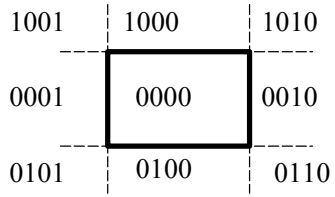
- ✓ L'ordre dans lequel sont effectués les tests de visibilité et d'invisibilité est sans importance.
- ✓ Certains segments vont nécessiter 4 tests pour être acceptés comme totalement visibles ou invisibles.
- ✓ D'autres segments ne demanderont qu'un seul test.
- ✓ Par contre, le calcul de l'intersection du segment avec les bords de la fenêtre est coûteux en calcul et doit être réalisé en dernier.

Algorithmes classiques

Algorithme de Dan Cohen et Ivan Sutherland

Cet algorithme est basé sur une technique permettant de détecter facilement certains cas de figure de position d'un segment vis à vis d'un rectangle.

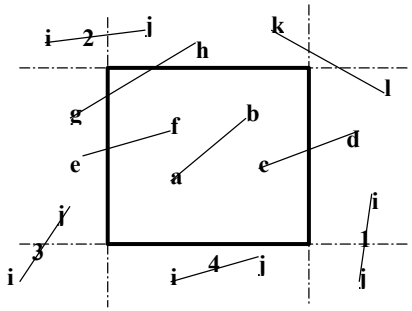
On associe à chaque extrémité du segment un code sur quatre bits indiquant la position (x, y) du sommet par rapport aux 4 droites définissant le rectangle.



- Premier Bit à 1 : si l'extrémité est à gauche de la fenêtre
- Deuxième Bit à 1 : si l'extrémité est à droite de la fenêtre
- Troisième Bit à 1 : si l'extrémité est au-dessous de la fenêtre
- Quatrième Bit à 1 : si l'extrémité est au-dessus de la fenêtre
- Dans le cas contraire, le bit est mis à zéro.

- ➢ Si les deux codes d'extrémités du segment sont à zéro, alors les deux extrémités de ce segment se trouvent à l'intérieur de la fenêtre, et le segment est **visible.**
- ➢ Si l'intersection logique bit à bit des deux côtés d'extrémités est non nulle, alors le segment est totalement invisible
- ➢ Le segment peut être totalement ou partiellement visible, ou même totalement invisible lorsque l'intersection logique est nulle.

Exemple



Segment	Codes d'extrémités	Intersection logique	Commentaire
ab	0000 0000	0000	Visible
ij	0010 0110	0010	Invisible
ij	1001 1000	1000	Invisible
ij	0101 0001	0001	« «
ij	0100 0100	0100	« «
Cd	0000 0010	0000	Partiellement visible
Ef	0001 0000	0000	
Gh	0001 1000	0000	
kl	1000 0010	0000	Invisible

Rappel

L'équation de la droite passant par les points P1(x1, y1) et P2(x2, y2) est :

Y = m*(x - x1) + y1 ou y=m*(x-x2) + y2

Avec m = dy/dx = (y2-y1) / (x2 – x1)

Les intersections du segment avec les bords de la fenêtre sont :

Gauche : X_l , y = m*(X_{min} – x1) + y1 $m \neq \infty$

Droite : Xr , y = m*(Xmax – x1) + y1 $m \neq \infty$

Haut : Y_t , x = (1/m)*(Y_{max} – y1) + x1 $m \neq 0$

Bas : Y_b , x = (1/m)*(Y_{min} – y1) + x1 $m \neq 0$

Remarque

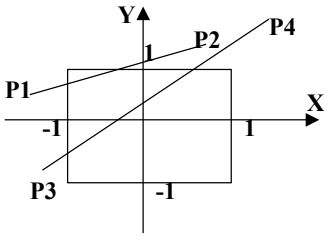
- Si la pente du segment est infinie, il est donc parallèle aux bords droit et gauche

→ L'intersection ne doit être recherchée qu'avec les bords supérieur et inférieur

- De même si la pente du segment est nulle, il est alors parallèle aux bords supérieur et inférieur.

→ L'intersection ne doit être recherchée qu'avec les bords gauche et droit.

Exemple : Fenêtrage 2D explicite



Intersections du segment P1(-3/2, 1/6) , P2(1/2, 3/2) avec les bords de la fenêtre sont :

La pente $m=\frac{y2-y1}{x2-x1}=\frac{\frac{3}{2}-\frac{1}{6}}{\frac{1}{2}-(-\frac{5}{2})}=\frac{2}{3}$

Gauche x = - 1 $y=\frac{2}{3}(-1-(-\frac{3}{2}))+\frac{1}{6}=\frac{1}{2}$

Droite x = 1 $y=\frac{2}{3}(1-(-\frac{5}{2}))+\frac{1}{6}=\frac{11}{2}>Y_{\max}$ est donc rejeté.

Haut y = 1 $x=\frac{3}{2}(1-\frac{1}{6})-\frac{3}{2}=-\frac{1}{4}$

Bas y = -1 $x=\frac{3}{2}(-1--\frac{1}{6})-\frac{3}{2}=-\frac{13}{4}<X_{\min}$ est donc rejeté

De même pour le segment P3(-3/2, -1) , P4(3/2, 2)

Gauche x = - 1 y = -1/2

Droite x = 1 y = 3/2 > Y_{max} est donc rejeté

Haut y = 1 x = 1/2

Bas y = -1 x = -3/2 < X_{min} est donc rejeté

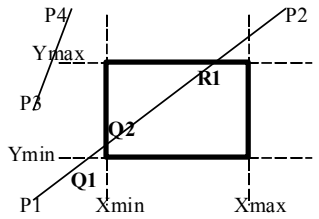
Etapes de l'algorithme

- Calculer les codes des 2 extrémités
- Tester si le segment est totalement visible ou invisible

  Sinon

  - Chercher une extrémité qui se situe à l'extérieur (il y a au moins une)
  - Tester le code pour trouver le côté traversé par le segment et calculer le point d'intersection correspondant.
  - Remplacer le point extrémité extérieur par le point d'intersection
  - Calculer le code de cette nouvelle extrémité afin de préparer l'itération suivante.

Exemples



Supérieur	Inférieur	Droite	Gauche

Cas du segment P1P2

L'algorithme détecte P1 comme point extérieur :

- Son premier bit à 1 correspond à l'horizontal y = ymin
- L'intersection Q1 remplace le point P1. Code Q1 = 0001
- Le et logique Q1 et P2 = 0000
- On recommence

  Q1 est remplace par Q2 de code 0000

  - On recommence avec l'autre extrémité P2 de code 1010.
    - On remplace P2 par l'intersection avec l'horizontal y = ymax. En effet : le premier bit du code P2 est à 1 et correspond à la droite y = ymax.
    - Le segment obtenu Q2R1 est visible
    - Tracer le segment
- Fin de l'algorithme.

Cas du segment P3P4

Le et logique des deux extrémités est différent de zéro. Le segment est donc totalement invisible.

Structure du code

```
Typedef struct {
        Unsigned somme ;
        Unsigned gauche ;
        Unsigned droite ;
        Unsigned inférieur ;
        Unsigned supérieur ;
}CodeSeg;
```

Calcul du code d'un point M(x, y)

CalculCode(x, y, Xmin, Ymin, Xmax, Ymax) → CodeSeg

Paramètres formels
x, y : coordonnées d'un point : Entier : Param en Entré
Xmin, Ymin, Xmax, Ymax : Coord Fenêtre : Entier : Param Entré

Début

Code(Somme) ← 0
Si (y > Ymax) alors
Code(Sup) ← 1
Incrémenter(code(Somme))
Sinon
Si (y < Ymin)
Code(Inf) ← 1
Incrémenter(code(Somme))
Finsi
Finsi
Si (x > Xmax) alors
Code(droite) ← 1
Incrémenter(code(Somme))
Sinon
Si (x < Xmin)
Code(gauche) ← 1
Incrémenter(code (Somme))
Finsi
Finsi
Renvoyer Code ;

FIN

Algorithme de Cohen-Sutherland de fenêtrage d'un segment de droite

Cohen-Sutherland (Xa, Ya, Xb, Yb, Xmin, Ymin, Xmax, Ymax, Attribut)

Paramètre formels :

```
 Xa, Ya, Xb, Yb : coordonnées du segment à tracer. Entiers(E)
Xmin, Ymin, Xmax, Ymax :coordonnées de la fenêtre. Entiers(E)
Attribut : attribut du segment. Entier (E)
```

Début

/* Calcul les codes des extrémités du segment */

```
codeA ← CalculCode(Xa, Ya, Xmin, Ymin, Xmax, Ymax)
codeB ← CalculCode(Xb, Yb, Xmin, Ymin, Xmax, Ymax)
m ← (yb-ya)/(xb-xa)
accept ← faux
fin ← faux
répéter
```

/* Tester si le Segment est totalement visible */

```
si somme(codeA)=0 et somme(codeB)=0 alors
    accepte ← vrai
    fin ← vrai
sinon
```

/* Tester si le Segment est totalement invisible */

```
    si Et_logique(codeA, codeB)#0 alors
        fin ← vrai
    sinon
```

/* Segment peut être partiellement Visible */

```
        codeExt ← codeA
        si somme(codeA)=0 alors
            codeExt ← codeB
        finsi
```

// Calcul l'Intersection avec les bords de la fenêtre

```
// Déterminer l'intersection avec le côte supérieur de la fenêtre
si supérieur(codeExt) alors
    x ← Xa + ( Ymax – Ya)/m
    y ← Ymax
sinon
    // Déterminer l'intersection avec le côte inférieur de F
    si inférieur(codeExt) alors
        x ← Xa + ( Ymin– Ya)/m
        y ← Ymin
    sinon
        // Déterminer l'intersection avec le côte droit de F
        si droite(codeExt) alors
            y ← Ya + ( Xmax– Xa)*m
            x ← Xmax
        sinon
        // Déterminer l'intersection avec le côte droit de F
            si gauche(codeExt) alors
                y ← Ya + ( Xmin– Xa)*m
                x ← Xmin
            finsi
        finsi
    finsi
finsi
```



```
// Calcul le code la nouvelle extrémité
si codeExt= codeA alors
    Xa ← x
    Ya ← y
    codeA ← CalculCode(Xa, Ya, Xmin, Ymin,
            Xmax, Ymax)
Sinon
    Xb ← x
    Yb ← y
    CodeB ← CalculCode(Xb, Yb, Xmin, Ymin,
            Xmax, Ymax)
Finsi
Jusqu'à fin
Si accept alors
    AfficheSegment(Xa, Ya, Xb, Yb, attribut)
Finsi
FIN
```

Autres algorithmes :

- **Algorithme de Cyrus Beck : Découpage selon une clôture polygonage convexe**
- **) Découpage selon une clôture polygonage non convexe**
- **)…**