Calcul d’une scène visible

Algorithmes de lignes et de faces cachées

- Il n'existe pas une « meilleure » solution.
- Plusieurs facteurs interviennent:
  - Type de projection choisi (perspective, perspective simplifiée, …)
  - Complexité de la scène
  - Complexité de l'image
  - Représentation des objets de la scène
  - Degré de réalisme exigé.

Classification des algorithmes

A

Les calculs se font dans l'espace de l'usager

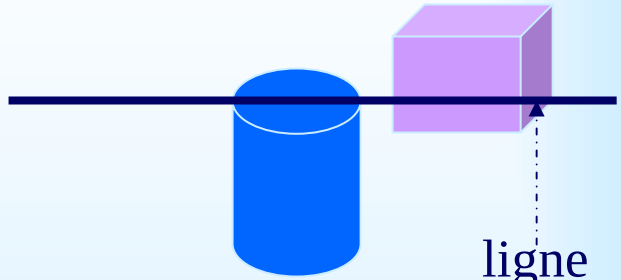
- La précision des calculs dépend de la représentation des nombres dans la machine.
- Ce sont des méthodes applicables à tous les dispositifs graphiques.
- Dépendent de la complexité de la scène et/ou des objets de la scène (visible ou non).
- $O(n^2)$ où n = # objets dans la scène.

Classification des algorithmes

B

Les calculs se font dans le plan de vue

- Comparaison de chaque face avec la droite de balayage.
- Besoin d'un coloriage sélectif
- Dépend du matériel
- Dépend de la complexité de la scène visible
- O(n N) où n = # objets dans la scène
  N = # pixels à l'écran



Si n < N alors les algorithmes de A devraient être retenus.

Ce n'est pas le cas car il est plus facile de tirer profit de la cohérence des objets dans le plan de vue.

C

Type mixte

Points en commun de ces algorithmes

- Ils nécessitent tous une étape de tri.

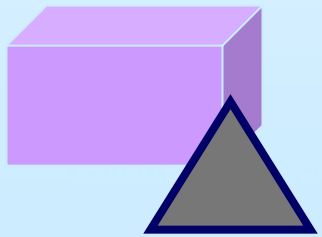
  Hypothèses

  - Plus un objet est « éloigné » de l'observateur, plus il est vraisemblable qu'il soit (partiellement ou non) caché.
  - Les objets d'une scène ont tendance à se regrouper.

- Ils misent tous sur le concept de cohérence (i.e. les caractéristiques d'une scène ont tendance à rester inchangées)

  (a) Cohérence des arêtes

  Une arête devient visible ou invisible lorsqu'elle intercepte une autre arête.



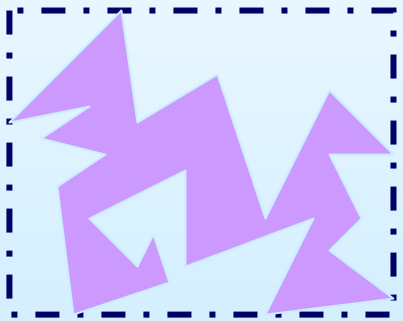
Concepts de cohérence

(b) Cohérence d'une face d'objet

Si l'une des faces d'un objet est partiellement visible, il existe de « bonnes » chances qu'elle soit complètement visible.

(c) Cohérence d'un objet

Tester la visibilité d'un objet plus simple qui renferme l'objet complexe.



rectangle

cercle

(d) Cohérence dans le cas d'un balayage ligne par ligne

Similitude d'une ligne de balayage à une autre.

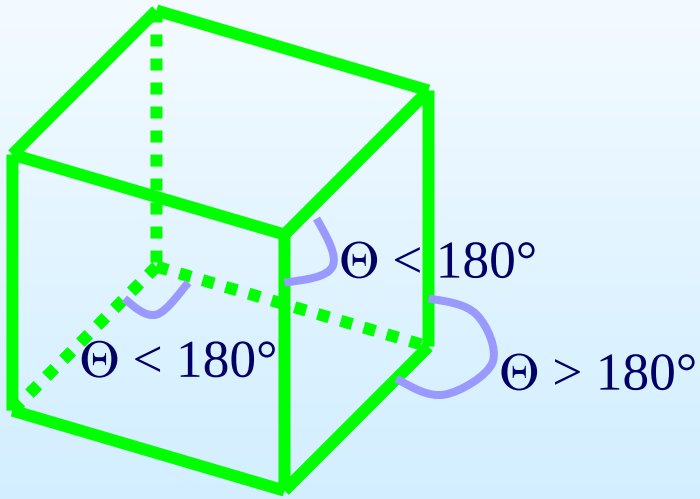
Concepts de cohérence

(e) Cohérence d ’une séquence animée

Distinguer entre les objets du décor et les objets dotés de mouvement.

(f) Cohérence géométrique

Exemple : Projection d’un cube dans le plan de vision



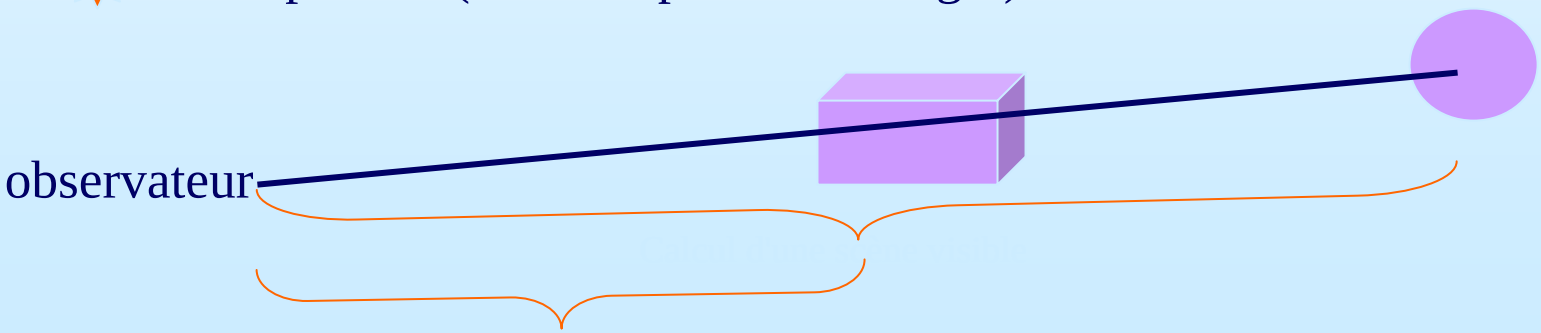
Soit Θ = angle maximum entre les arêtes adjacentes à un sommet donné,

Θ > 180° les 2 arêtes extrêmes sont visibles.

Θ ≤ 180° les arêtes ont toutes même visibilité.

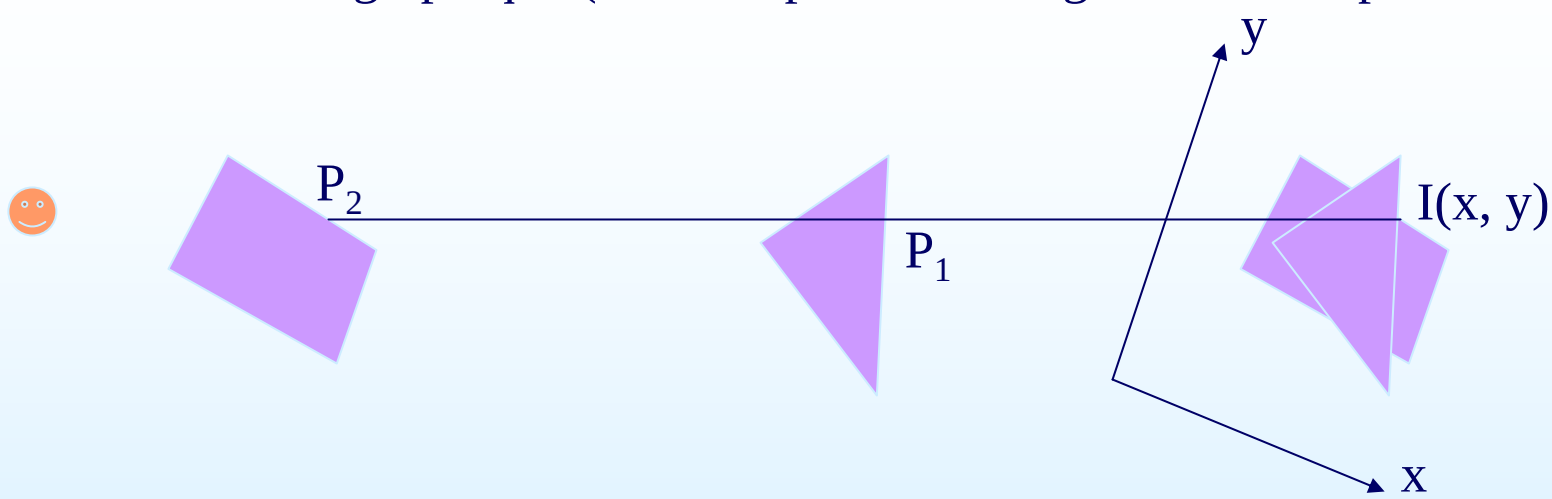
Opérations et tests de base

- A Opération de projection
- B Test de recouvrement
  - Intersection de 2 objets après projection
  - Intersection des objets avec une ligne de balayage
  - Intersection des objets avec un rayon issu de l'observateur
- C Test d'appartenance
- D Test de profondeur : comparaison des valeurs de la coordonnée en z des points
  - Perspective (dans l'espace de l'usager)



Opérations et tests de base

- Parallèle orthographique (dans l'espace de l'usager et dans le plan de vue)

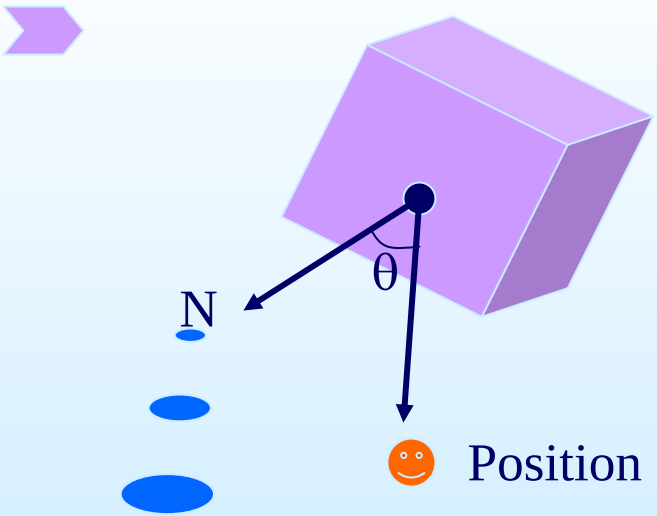


Les coordonnées de I sont remplacées dans l'équation du plan coïncidant avec l'objet O_1 et celle de O_2 pour comparer les profondeurs.

Opérations et tests de base

E Test de visibilité

Il s'agit de déterminer si une face est à l'avant ou à l'arrière de l'objet.



Normale à la facette pointant vers l'extérieur

Si $\theta \leq 90°$ alors la facette est potentiellement visible.

Intéressant dans le cas d'un polyèdre convexe.

Opérations et tests de base

E Test de visibilité (suite)

Il s'agit de déterminer si une face est à l'avant ou à l'arrière de l'objet.

- Exigences : La scène est formée d'objets convexes et chaque facette est un polygone convexe.
- Chaque objet est représenté par un ensemble fini de plans frontières:

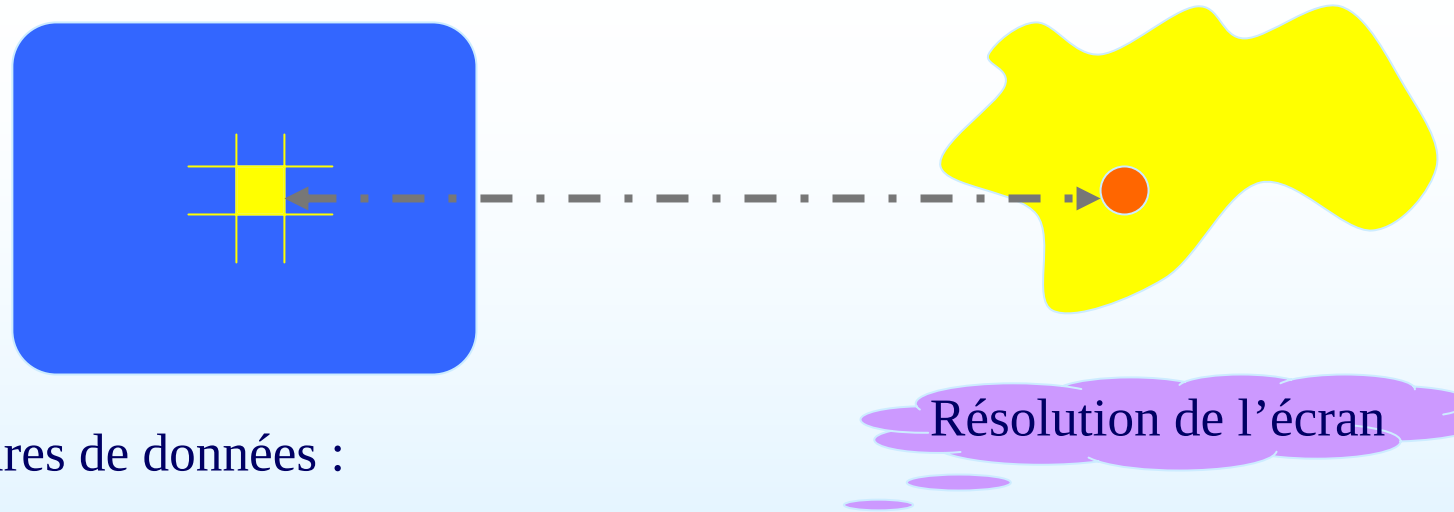
$$a_i x + b_i y + c_i z + d_i = 0 \qquad \forall i = 1, 2, \ldots, m$$

(le i^e plan renferme la i^e facette)

- Convention : Si $a_i x + b_i y + c_i z + d_i > 0$ alors
  $(x, y, z) \in$ demi-espace qui contient l'objet.
- Soit $(x_{OBS}, y_{OBS}, z_{OBS})$ la position de l'observateur,
  si $a_i x_{OBS} + b_i y_{OBS} + c_i z_{OBS} + d_i < 0$ alors
  la i^e facette est potentiellement visible.

NOTE Si l'observateur se trouve sur l'axe des z positifs à $(0, 0, \infty)$,
alors tester le signe de c_i.

Algorithme du z-buffer (tampon Z)



Structures de données :

- INTENSITE : tableau de dimension u x v renfermant la couleur de chaque pixel
- PROFONDEUR : tableau de dimension u x v renfermant la profondeur du point le plus proche de l'observateur.

Algorithme du z-buffer (description)

1. ∀position (x, y),
   PROFONDEUR[x, y] ← valeur maximale
   INTENSITE[x, y] ← couleur de fond
2. ∀ objet de la scène, trouver toutes les positions (x, y) à l'intérieur de l'objet, une fois projetée à l'écran.
3. Pour chacune des positions (x, y) obtenues à l'étape 2,
   a) calculer la profondeur z de l'objet en (x, y).
   b) si z < PROFONDEUR[x, y]
      { PROFONDEUR[x, y] ← z
      { INTENSITE[x, y] ← couleur associée à ce point.

Exigences :

1) La scène doit être constituée d'objets opaques.
2) Cela doit être facile d'énumérer toutes les positions à l'intérieur des objets projetés.

Algorithme du z-buffer (Simplification)

- L'algorithme précédent est simplifié lorsque la scène est constituée de polygones.

  Chaque polygone est représenté par l'équation plane $ax + by + cz + d = 0$ dans le système de coordonnées de l'écran.

- Comment simplifier le calcul de la profondeur z au point (x, y) d'un polygone ?

  Si $ax_1 + by_1 + cz_1 + d = 0$
  alors le point (x_1+1, y_1) est de profondeur $z_1 - a / c$ et
  le point $(x_1, y_1 -1)$ est de profondeur $z_1 + b/c$.

Ex. : $a(x_1+1) + b\,y_1 + c\,z + d = 0 \Rightarrow ax_1 + b\,y_1 + c\,z_1 + d + c\,z - c\,z_1 + a = 0$
$\Rightarrow c\,z = c\,z_1 - a \Rightarrow z = z_1 - a / c.$

- Une scène est constituée d'une liste de polygones dans le système de coordonnées de l'écran. Les coefficients de l'équation de chaque plan correspondant à un polygone sont aussi donnés.

Algorithme du z-buffer (Accélération des calculs)

- Calcul des points à l'intérieur d'un polygone
  - En calculant les extremums du polygone, on obtient un rectangle renfermant le polygone.
  - Pour chaque point appartenant au rectangle, vérifier s'il appartient au polygone.

    Si oui, { calcul de la profondeur
    comparaison

    Coût : Appartenance d'un point à l'intérieur d'un polygone.

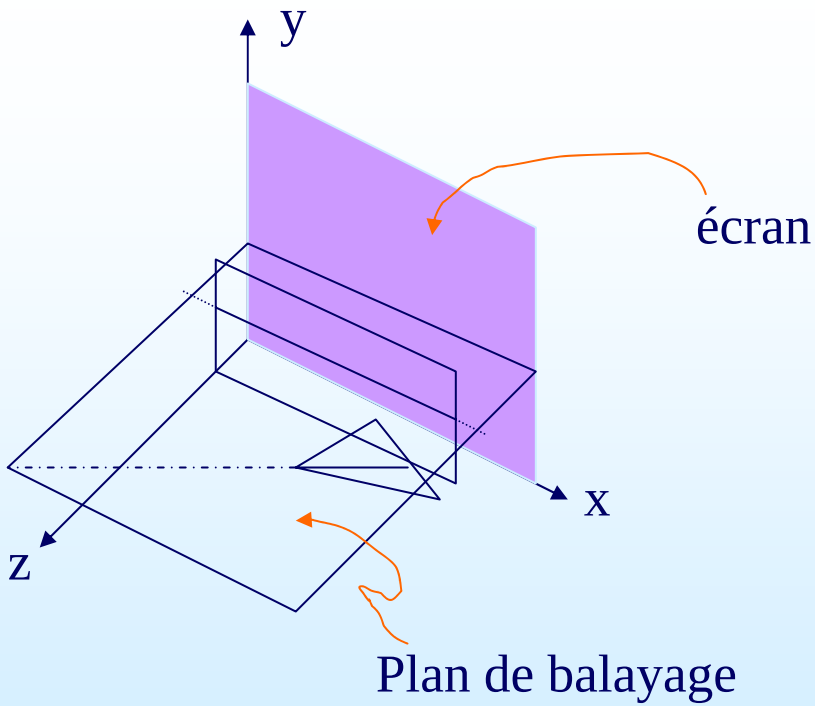
    ↳ Pour améliorer l'algorithme, on fait passer le test de profondeur avant le test d'appartenance.

    ∴ Malgré cela, l'implantation devient impraticable.
  - Balayage horizontal du polygone

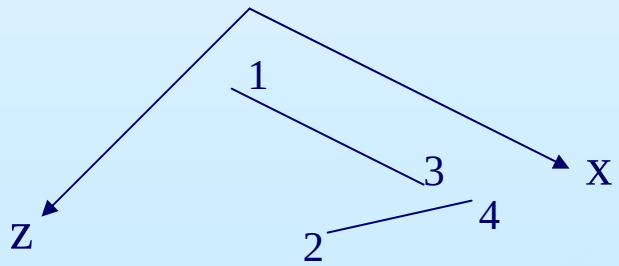
    ↳ Facilite le calcul de la profondeur en chaque point.

Algorithmes de balayage

- Définir un plan de balayage à partir de l'observateur et d'une droite de balayage.



- L'élimination des parties cachées est effectuée dans le plan de découpage :



Algorithmes de balayage

Tentative :

- Construire la liste ordonnée des extrémités des arêtes selon l'abscisse
- Soient (x_i, z_i), i = 1, 2, …, N les coordonnées des points de la liste ordonnée,

$\Rightarrow$ (x_{2i-1}, x_{2i}), i= 1, 2, … sont les segments visibles.

FAUX FAUX FAUX FAUX FAUX FAUX FAUX FAUX

Application du principe de balayage à l'algorithme z-buffer

INTENSITE — Couleur de chaque pixel de la ligne de balayage courante

PROFONDEUR — Profondeur de chaque point sur la ligne de balayage courante le plus près de l'observateur.

2 vecteurs de longueur N (# pixels / ligne)

Algorithme

∀ligne de balayage d'ordonnée y,

1. ∀position horizontale x de la ligne de balayage y
   - PROFONDEUR[x] ← valeur maximale;
   - INTENSITE[x] ← couleur de fond
2. ∀ objet de la scène, projeté dans le plan de vue
   - ∀ point d'intersection (x, y) entre la ligne de balayage et l'objet,
     - calculer la profondeur z du point (x, y).
     - si z < PROFONDEUR[x]
       - PROFONDEUR[x] ← z; INTENSITE[x] ← couleur;
3. Afficher cette ligne.

Application de principes de cohérence

0. Déterminer pour chaque polygone le sommet d'ordonnée maximale (noté y_{max})

- Trier la liste de polygones en ordre décroissant de y_{max}.
- Ranger dans une liste chaînée,

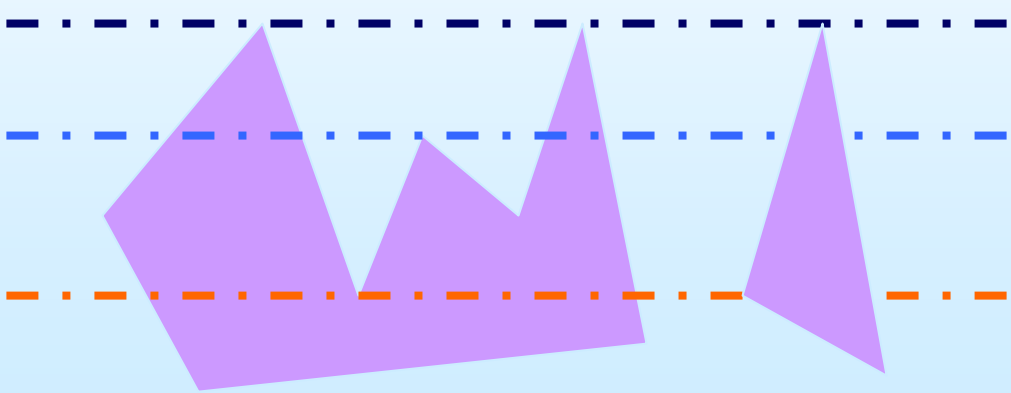
  $\Delta y \equiv$ le nombre de lignes de balayage qui interceptent le polygone,
  la liste des arêtes du polygone,
  les coefficients a, b, c, d de l'équation du plan,
  la couleur du polygone

  pour chaque polygone de la scène.
- La liste des polygones actifs et la liste des arêtes actives sont vides.

Application de principes de cohérence

1. ∀ligne de balayage d'ordonnée y,

   - Initialisation des vecteurs PROFONDEUR et INTENSITE.
   - Ajouter à la liste des polygones actifs ceux dont $y_{max} \equiv y$.
   - Parmi ces polygones actifs, ajouter à la liste des arêtes actives toutes les nouvelles paires d'arêtes.



Application de principes de cohérence

La liste d'arêtes actives contient pour chaque paire d'arêtes :

- x_G : abscisse du point d'intersection entre la droite de balayage courante et l'élément de gauche de la paire,
- Δx_G : incrémentation de x_G d'une droite de balayage à la suivante,
- Δy_G : # lignes de balayage qui interceptent l'élément de gauche,
- x_D , Δx_D , Δy_D : l'équivalent pour l'élément de droite,
- z_G : profondeur du point appartenant à l'arête de gauche sur la ligne de balayage courante,
- Δz_x : incrémentation en z le long de la droite de balayage (= a / c),
- Δz_y : incrémentation en z d'une droite de balayage à une autre (= b / c),

Application de principes de cohérence

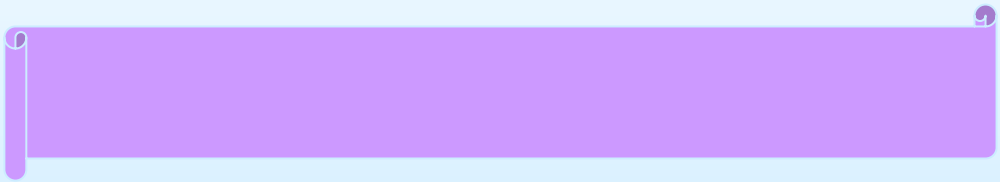
2. ∀paire d'arêtes dans la liste des arêtes actives,
   - Poser $z \leftarrow \mathbf{z_G}$
   - $\forall x = \mathbf{x_G}, \mathbf{x_G+1}, \ldots, \mathbf{x_D}$
     - calculer z(x, y), **i.e.** $z \leftarrow z - \Delta z_x$
     - comparer avec PROFONDEUR[x],
     - mettre à jour INTENSITE[x] s'il y a lieu.
3. Afficher cette ligne.
4. Mise à jour de la liste des arêtes actives.
   - $\Delta y_G \leftarrow \Delta y_G - 1$; $\Delta y_D \leftarrow \Delta y_D - 1$;
   - **Si** $\Delta y_G < 0$ enlever l'arête de la liste
     Si $\Delta y_D < 0$ enlever l'arête de la liste
     ⇒ Conserver l'emplacement de l'arête enlevée dans la liste et un accès au polygone dont cette arête fait partie.
   - $\mathbf{x_G} \leftarrow \mathbf{x_G} + \Delta \mathbf{x_G}$; $\mathbf{x_D} \leftarrow \mathbf{x_D} + \Delta \mathbf{x_D}$;
   - $\mathbf{z_G} \leftarrow \mathbf{z_G} + \Delta \mathbf{z_y}$;

Application de principes de cohérence

5. **Mise à jour de la liste des polygones actifs,**

∀polygone de la liste active,

- $\Delta y \leftarrow \Delta y - 1$
- Si $\Delta y < 0$ enlever le polygone de la liste.



Algorithme de Watkins

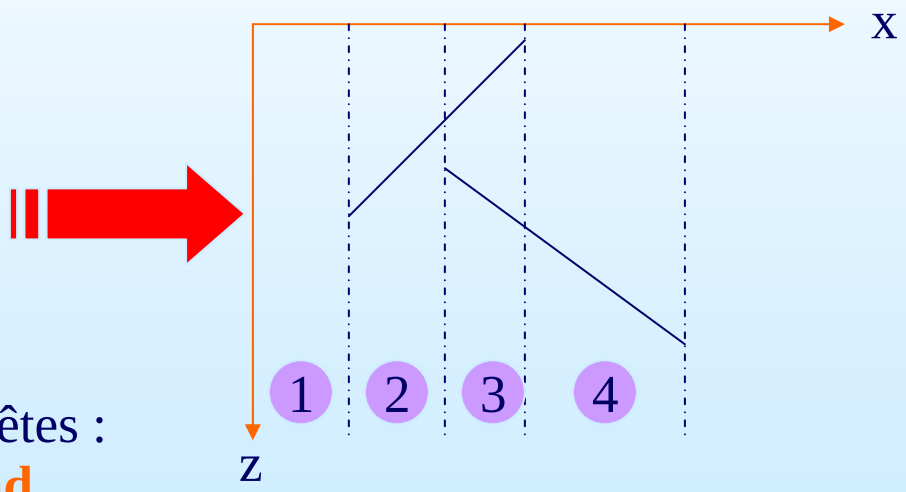
But : L'algorithme du Z-Buffer exige un grand nombre de calculs de profondeur. Comment réduire ce nombre ?

Il s'agit de calculer à chaque itération i l'intersection des polygones de la scène avec le plan de balayage $y = y_i$ ce qui donne une liste d'arêtes dans ce plan.

Des zones sont construites dans le plan x°z à l'aide de la liste ordonnée selon x des extrémités des arêtes.

On détermine alors le segment visible dans chaque zone le cas échéant.

Considérons par exemple l'intersection de 2 polygones avec un plan de balayage :



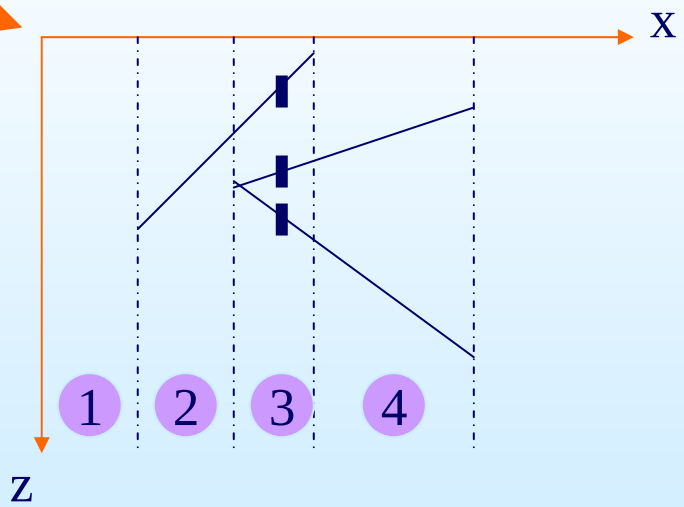
Déf[n] de zones à l'aide des extrémités des arêtes :

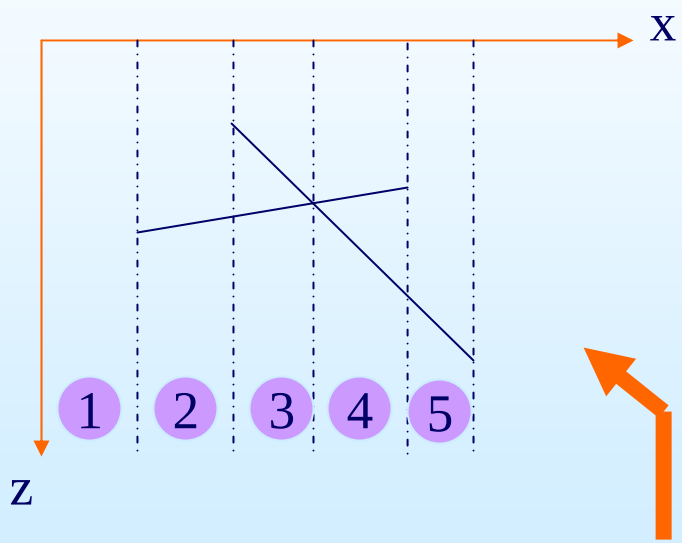
- Zone vide : 1 **Couleur de fond**
- Zone contenant un seul segment : 2 et 4 **Couleur du segment**
- Zone contenant plusieurs segments : 3 **Dépend de la profondeur des segments.**

Algorithme de Watkins

Si les polygones ne se pénètrent pas, il faut calculer la profondeur des segments à l'une des extrémités.

(Si 2 segments se touchent à cette extrémité, on calcule la profondeur des segments au milieu de la zone.)





Si les polygones se pénètrent, on construit une nouvelle zone à chaque extrémité d'arêtes et à chaque point d'intersection.

Approche de subdivision (Lane - Carpenter 80)

Extension des techniques de balayage au cas où les surfaces de la scène sont non polygonales

Au lieu de construire directement un maillage polygonal, on opte plutôt pour une approche qui tient compte de la courbure de la surface :

0. Les surfaces sont triées selon leur ordonnée maximale y.
1. Pour chaque ligne de balayage d'ordonnée y = y',
   Pour chaque surface active (interceptée par la ligne de balayage),
   si la surface est « quasi-plane »
   alors ajouter l'approximation de la surface à la liste des polygones
   sinon subdiviser la surface en « sous-surfaces »,
   si une « sous-surface » intercepte la ligne de balayage,
   alors la « sous-surface » est active
   sinon la « sous-surface » est enlevée.

Note :

- Les surfaces « quasi-planes » sont approximées par des polygones ou bien un maillage est construit à partir de chaque surface.
- Les algorithmes de balayage dans le cas d'une scène polygonale peuvent être utilisés À L'EXCEPTION que les attributs d'un « pixel » sont déterminés à partir de la définition de la surface et non du polygone approximatif.
- La subdivision peut se faire avant le balayage [Clark, 79].

ALGORITHME DE WARNOCK

PRINCIPES DE COHÉRENCE

- La scène a tendance à constituer plusieurs « tas » d ’objets.
- Plus facile de résoudre le problème de faces cachées si la fenêtre est petite.

ALGORITHME

- Subdiviser l’écran en fenêtres, sous-fenêtres, ...
- Pour chaque fenêtre, on distingue 3 cas :

  1) la fenêtre est vide.
  2) l’affichage de la fenêtre se ramène à un cas simple.
  3) Sinon, on subdivise la fenêtre en sous-fenêtres, et on recommence.

Critère d’arrêt

lorsque nous avons parcouru l’écran au complet.

fenêtre vide | cas simple | une fenêtre de la taille d’un « pixel »

ALGORITHME DE WARNOCK

Scène constituée de polygones

CAS SPÉCIAUX :

- Tous les polygones sont à l'extérieur de la fenêtre → Remplir la fenêtre avec la couleur de fond.
- Un seul polygone intercepte la fenêtre ou est inclus dans la fenêtre → Remplir la fenêtre avec la couleur de fond et remplir la partie commune.
- La fenêtre est incluse dans un polygone et il n'y a pas d'autres polygones dans la fenêtre.
- La fenêtre est incluse dans un polygone et celui-ci est le plus près de l'observateur.

→ Remplir la fenêtre avec la couleur appropriée.

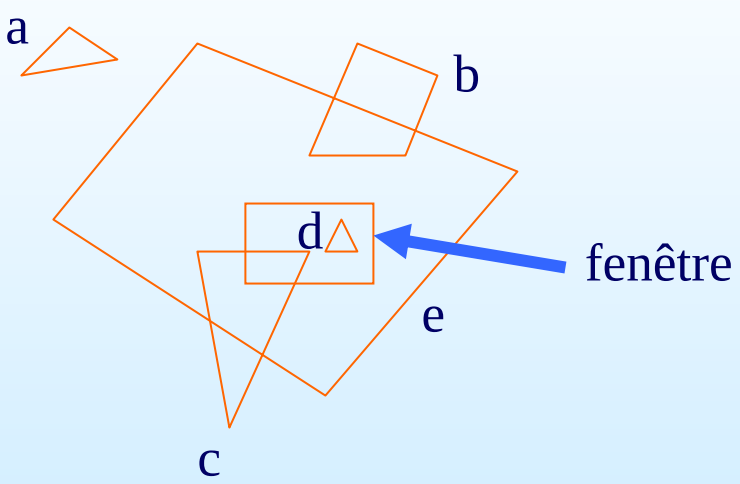
↓ Autrement, on subdivise la fenêtre à nouveau.

ALGORITHME DE WARNOCK

Scène constituée de polygones

La subdivision de fenêtres en sous-fenêtres nous amène à tenter de réduire le plus possible la scène associée à chaque fenêtre.

CLASSIFICATION DES POLYGONES DANS UNE FENÊTRE



I. cas a, b : aucune partie du polygone ne touche la fenêtre.

II. cas c,d : une partie ou la totalité du polygone est dans la fenêtre.

III. cas e : la fenêtre est recouverte.

1er groupe : À éliminer

3e groupe : On veut éliminer les polygones plus loin que ceux-ci.

ALGORITHME DE WARNOCK

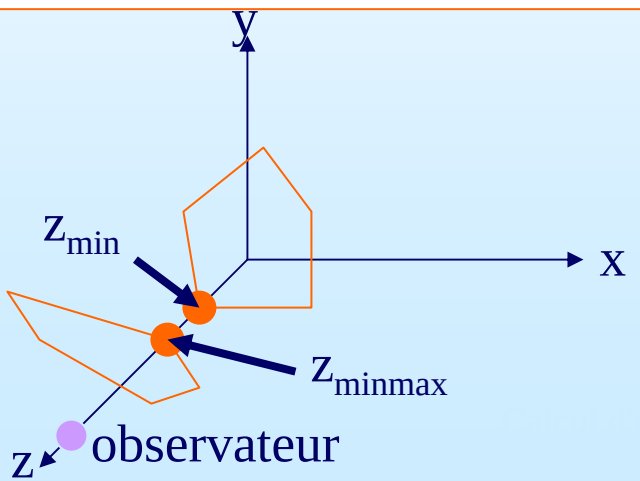
Scène constituée de polygones

3e groupe : La fenêtre est recouverte par un polygone A.
On veut éliminer les polygones plus loin que A.

(a) Trier la liste des polygones associés à la fenêtre selon z_{min}
(la profondeur du point le plus proche du polygone)

(b) Pour tout polygone du 3e groupe qui recouvre la fenêtre, calculer z_{minmax}
(la profondeur du point le plus éloigné du polygone)

(c) Éliminer tout polygone associé à la fenêtre tel que $z_{min} < z_{minmax}$.

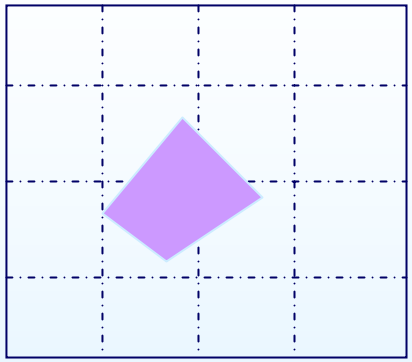


ALGORITHME DE WARNOCK

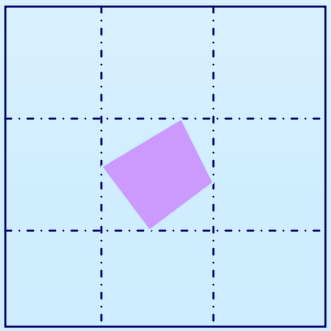
Scène constituée de polygones

Processus de subdivision d'une fenêtre en « sous-fenêtres »

A) Approche originale



B) Subdivision basée sur le rectangle d'aire minimale renfermant un polygone quelconque inclus dans la fenêtre.



But : Minimiser le nombre de subdivisions.

ALGORITHME DE WARNOCK

Scène constituée de polygones

Processus de subdivision d'une fenêtre en « sous-fenêtres »

C) Algorithme de Weiler-Atherton (1977)

But : Minimiser le # de subdivisions en considérant des sous-fenêtres polygonales.

Construire 3 listes S, A et B : la première renferme la scène et les autres sont vides.

1e étape :

En supposant l'observateur sur l'axe des z positifs à l'infini,
trier en ordre décroissant la liste de polygones S selon la coordonnée en z du sommet le plus proche de l'observateur.

2e étape : Le 1er polygone de la liste ordonnée S joue le rôle de fenêtre.

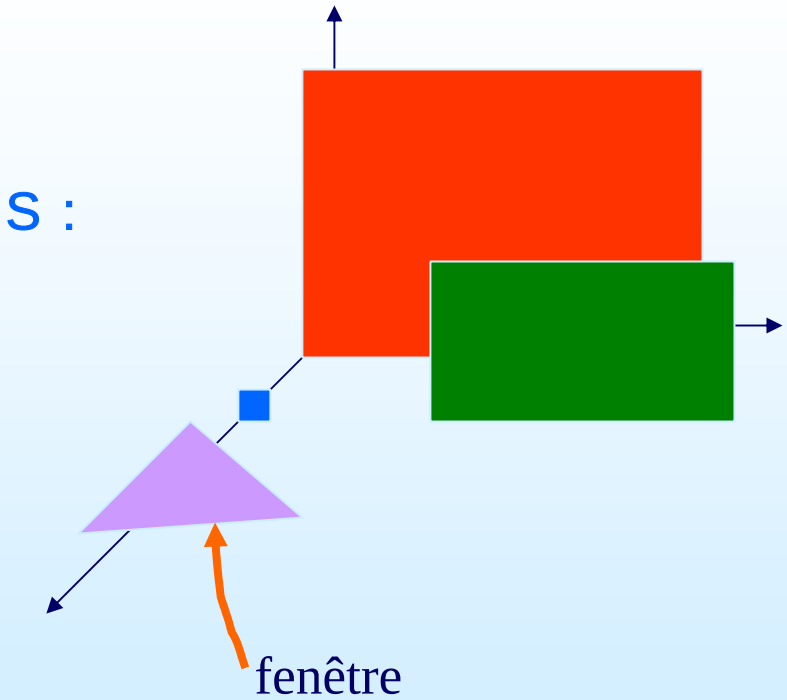
Effectuer le découpage selon cette fenêtre (après projection).

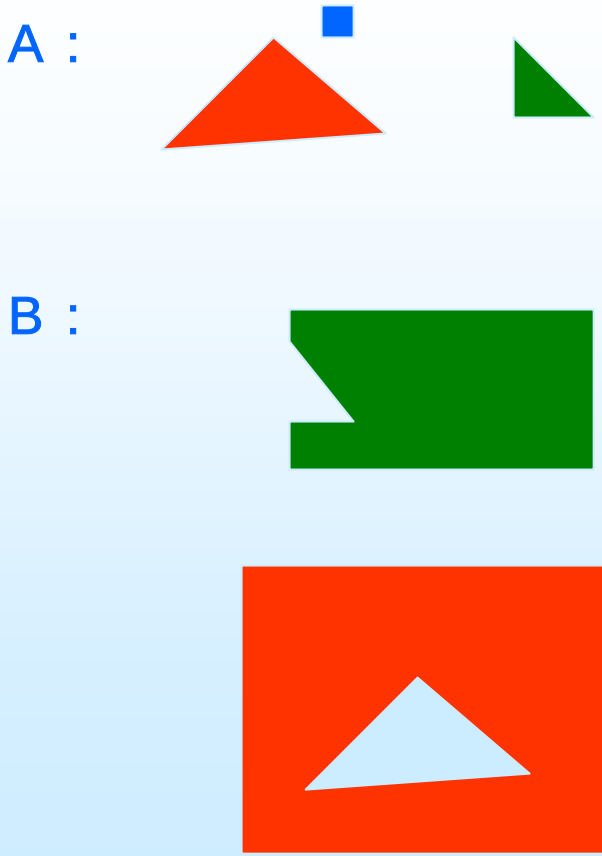
A : Contient chaque partie de polygone de S à l'intérieur de la fenêtre, incluant la fenêtre elle-même.

B : Ajoute à la liste B chaque partie de polygone de S à l'extérieur de la fenêtre.

ALGORITHME DE WARNOCK

Scène constituée de polygones





ALGORITHME DE WARNOCK

Scène constituée de polygones

3e étape : Comparaison de la profondeur des sommets des polygones de la liste A avec celle de la fenêtre.

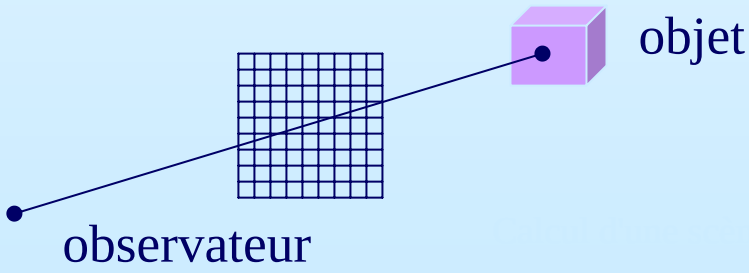
- z_{min} désigne la profondeur du sommet le plus éloigné de la fenêtre.
- ∀ polygone de la liste A,
  ∀ sommet (x, y) de ce polygone,
  Déterminer la profondeur z.
  Si $z > z_{min}$ // Le polygone est en avant de la fenêtre.
  alors Ce polygone plus proche de l'observateur devient la nouvelle fenêtre; il devient le 1er de la liste A.
  On poursuit à l'étape 2 où S devient A, A devient vide et B demeure inchangée.
- // Tous les polygones de A sont cachés par la fenêtre;
  // la fenêtre est affichée et A devient vide.

On poursuit à l'étape 1 avec B après avoir décomposé les polygones en formes simples le cas échéant : S ← B tandis que A et B sont vides.

ALGORITHME DE TRACÉ DE RAYONS

PRINCIPES DE BASE :

- Un observateur aperçoit un objet grâce à des rayons lumineux (provenant d'une source) qui interceptent l'objet et parviennent à l'observateur.
- Les rayons lumineux parviennent à l'observateur par
  RÉFLEXION, RÉFRACTION, TRANSPARENCE
- Caractéristiques : Peu de rayons émis parviennent à l'observateur.
  → Très peu efficace.
- [A. Appel, 1968]
  Suggère de tracer les rayons dans la direction opposée, i.e. de l'observateur à l'objet.



ALGORITHME DE TRACÉ DE RAYONS

DESCRIPTION DE L'ALGORITHME :

- On se ramène au système de coordonnées écran.
- L'observateur est à $(0, 0, \infty)$ ⟹ tous les rayons sont parallèles à l'axe des z.
- Chaque rayon lumineux traverse une position à l'écran.
- Il s'agit de déterminer pour chaque rayon lumineux tous les points d'intersection avec tous les objets de la scène.
- On considère alors le point d'intersection de profondeur z maximale, l'objet auquel appartient ce point permet de déterminer les attributs de ce « pixel ».

Note :

- La scène renferme n'importe quel type d'objets en autant que l'on puisse calculer l'intersection entre un rayon lumineux et l'objet.
- L'efficacité de l'algorithme dépend de la procédure utilisée pour calculer ces points d'intersection.

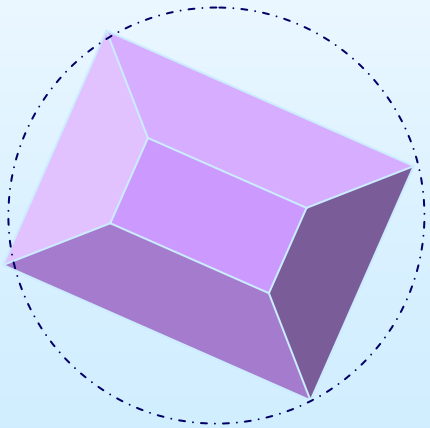
ALGORITHME DE TRACÉ DE RAYONS

Stratégie pour réduire les temps de calculs des points d'intersection

A. Considérer l'intersection d'un rayon avec un volume « enveloppant » de l'objet.

Si le rayon n'intercepte pas le volume « enveloppant »
alors l'objet n'est plus considéré pour ce rayon
sinon le calcul exact est effectué.

1°) Choix d'une sphère comme volume enveloppant



Peut être non efficace.

ALGORITHME DE TRACÉ DE RAYONS

Le calcul d'intersection est simple.

Soit le rayon lumineux $P(t) = P_1 + t\,(P_2 - P_1)$ où $P_1 = (x_1, y_1, z_1)$

$P_2 - P_1 = (a, b, c)$

soit $C(x_0, y_0, z_0)$ le centre de la sphère,

R le rayon de la sphère,

alors

d = distance minimale entre le rayon lumineux et le centre C de la sphère

$d^2 = (P(t) - C) \cdot (P(t) - C)$

où $t = - \{\dfrac{a(x_1 - x_0) + b(y_1 - y_0) + c(z_1 - z_0)}{a^2 + b^2 + c^2}\}$

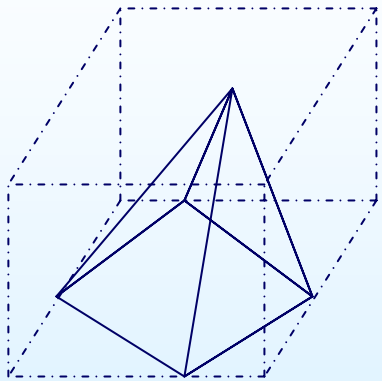
Si $d^2 > R^2$ alors le rayon n'intercepte pas la sphère.

ALGORITHME DE TRACÉ DE RAYONS

2°) Choix d'une boîte comme volume enveloppant

Calcul d'intersection plus dispendieux

1er méthode : calcul direct



Intersection du rayon avec au moins 3 plans, en général.

2e méthode :

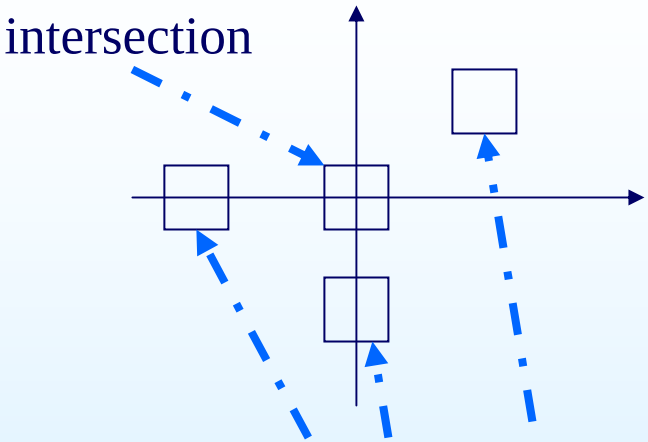
- Appliquer des transformations élémentaires (translation et rotation) à l'objet afin de faire coïncider le rayon lumineux avec l'axe des z.
- Appliquer les mêmes transformations à cette boîte.

➡ Le rayon intercepte la boîte si dans le nouveau système de coordonnées,

$x_{min} * x_{max} < 0$ et $y_{min} * y_{max} < 0$.

ALGORITHME DE TRACÉ DE RAYONS

2°) Choix d'une boîte comme volume enveloppant



Le rayon n'intercepte pas la boîte.

CAS PARTICULIER :

Soit la surface quadrique $a_1x^2 + a_2y^2 + a_3z^2 + b_1yz + b_2\,xz + \ldots + c_3z + d = 0$

après transformation, $x = y = 0$ $\Rightarrow$ $a_3'z^2 + c_3'z + d' = 0$

Nouveau système de coordonnées

Si $c_3'^2 - 4a_3'd' < 0$ alors aucune intersection.

ALGORITHME DE TRACÉ DE RAYONS

B. Accélérer les temps de calculs des points d'intersection grâce à une technique de subdivision récursive [Whitted, 79]

- Scène formée de surfaces courbes (Bézier, B-splines, …).

- Si un rayon lumineux intercepte la sphère englobante de l'objet,
  l'objet est subdivisé en « sous-surfaces ».
  On teste de nouveau avec chaque « sous-surface » obtenue.
  S'il n'y a aucune intersection avec les sphères englobantes,
  on poursuit avec un autre objet.
  Si une sphère est interceptée, on subdivise de nouveau.

- Le processus arrête
  - lorsque la sphère n'est pas interceptée ou
  - son volume est inférieur à un seuil
  ce qui correspond au point d'intersection.

ALGORITHME DE TRACÉ DE RAYONS

NOTE :

- Les volumes englobants sont obtenus à l'aide des points de contrôle des surfaces.
- En faisant coïncider le rayon lumineux avec l'axe des z, on peut choisir des boîtes comme volume englobant ➡ peut réduire le # de subdivisions
- Surfaces de Bézier et B-Spline : l'enveloppe convexe de chaque surface est considérée.

Calcul du point d'intersection plus complexe

ALGORITHME DE TRACÉ DE RAYONS

C. Volume englobant à un groupe d'objets

- Si un rayon n'intercepte pas le volume englobant, il n'intercepte pas les objets à l'intérieur de ce volume.
- Sinon, on réitère avec les volumes englobants de ces objets.
- Dès qu'un volume englobant contenant un objet unique est intercepté, l'objet est placé dans la liste des objets actifs.

D. Trier la liste des objets actifs

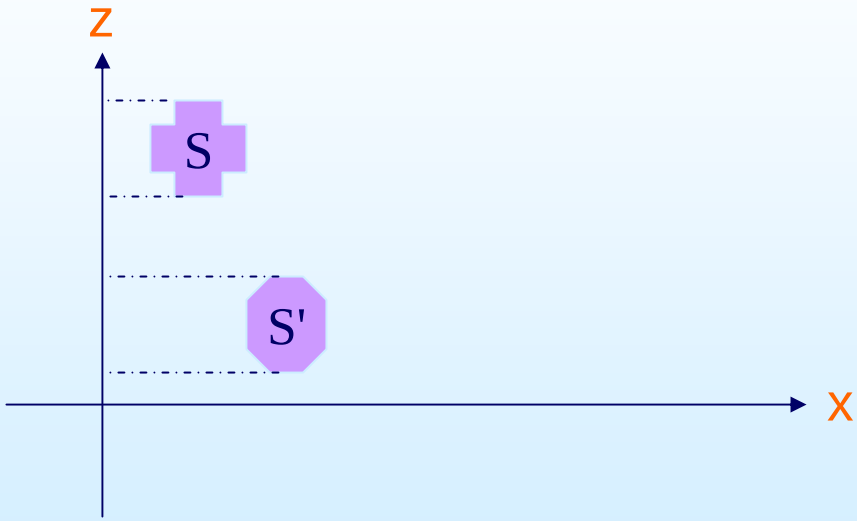
Avant de calculer les points d'intersection avec les objets de la scène,

- Trier la liste des objets actifs selon la profondeur.
- Adapter l'algorithme du peintre.

Le centre de la sphère ou la composante max. de z de la boîte est utilisé.

ALGORITHME DU PEINTRE

- Trier en ordre décroissant de profondeur les objets de la scène
- Chaque surface S est comparée aux autres surfaces S' à la suite de S.
  S'il n'existe aucun recouvrement en profondeur, on affiche S.

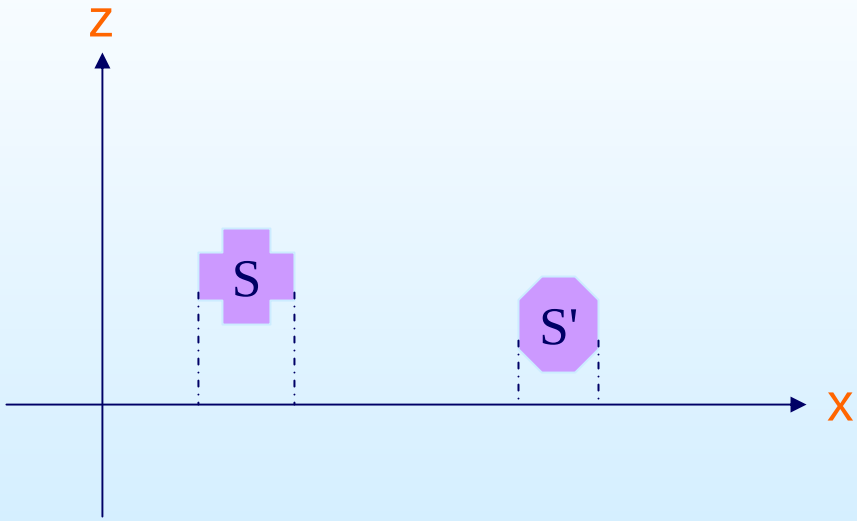


S'il existe un recouvrement en profondeur entre S et S',
il y a possibilité de réordonnancement.

ALGORITHME DU PEINTRE

- On ne réordonne pas la liste si l'un des tests suivants est vérifié (i.e. S' n'est pas derrière S) :

A) les 2 surfaces ne se recouvrent pas dans la direction x



B) les 2 surfaces ne se recouvrent pas dans la direction y

ALGORITHME DU PEINTRE

C) S est derrière le plan coïncidant avec S'.

TEST :

Soit $a_1x + b_1y + c_1z + d_1 = 0$ l'équation du plan coïncidant avec S',
soit (x_0, y_0, z_0) la position de l'observateur,
si $a_1x_0 + b_1y_0 + c_1z_0 + d_1 < 0$
alors $a_1x + b_1y + c_1z + d_1 > 0$ $\forall$sommet (x, y, z) de S

D) S' est en avant du plan coïncidant avec S.

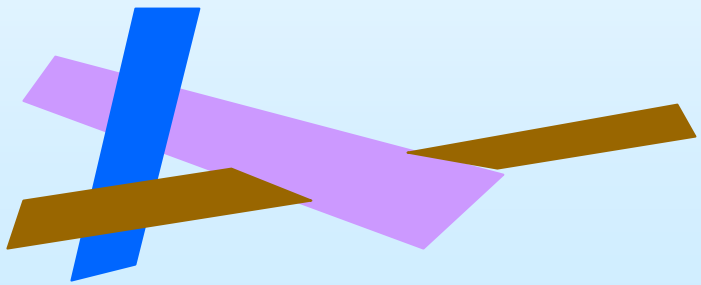
TEST :

Soit $ax + by + cz + d = 0$ l'équation du plan coïncidant avec S,
soit (x_0, y_0, z_0) la position de l'observateur,
si $ax_0 + by_0 + cz_0 + d > 0$
alors $ax + by + cz + d > 0$ $\forall$sommet (x, y, z) de S'.

E) Les projections des 2 surfaces sur le plan de vue ne se recouvrent pas.

ALGORITHME DU PEINTRE

- 2 situations peuvent intervenir :
  - aucun réordonnancement de la liste a été effectué
    - La surface S est affichée.
    - On passe à la surface suivante.
  - Dès qu'un réordonnancement a lieu, on recommence le tout avec la nouvelle surface.



L'algorithme peut cycler.

ALGORITHME DU PEINTRE

1 Trier la scène en ordre décroissant de profondeur.

2 Soit S le premier objet de la scène triée, S jouera le rôle de fenêtre,

2.1 ∀ objet S' ≠ S

∀ test i

Si le test i est vérifié, on poursuit avec une autre surface S'
sinon on va au test suivant.

Puisqu'aucun test n'est vérifié, un réordonnancement de la liste a lieu.

On recommence au début de 2 avec comme fenêtre S'.

2.2 Puisqu'aucun réordonnancement n'a été effectué,

la surface S est affichée,

la surface S est enlevée de la scène triée.

On poursuit au début de l'étape 2 avec la scène réduite.

FIN