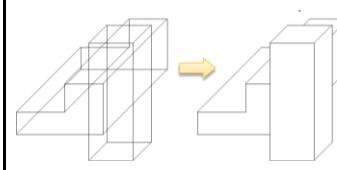
Visibilité

Visibilité

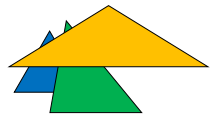
- Calcul des **primitives visibles**
- Remplissage du **frame buffer** avec le bon format de couleur



- Modeling Transformations
- Illumination (Shading)
- Viewing Transformation (Perspective / Orthographic)
- Clipping
- Projection (to Screen Space)
- Scan Conversion (Rasterization)
- **Visibility / Display**

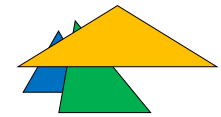
Élimination des parties cachées

- **Une scène = un ensemble de primitives** : en général un ensemble de polygones (triangles)
- **Détermination des surfaces visibles** : **Combien ça coute ???**



Élimination des parties cachées

- **Une scène = un ensemble de primitives** : en général un ensemble de polygones (triangles)
- **Détermination des surfaces visibles** : équivalent au tri ⇨ **O(*n* log *n*)**



Comment on fait alors ???

Algorithme à précision objet

- Masquage sur le modèle de données physiques
  - ⇨ Calcul en coordonnées du monde

\+ : masquage valable quelle que soit l'échelle du dessin
\- : plus cher

Algorithme à précision image

- Masquage sur les pixels écran
  - ⇨ Calcul en coordonnées fenêtre

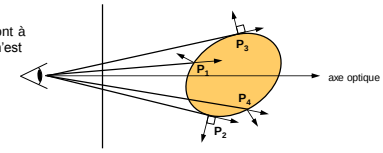
+ : peut utiliser la cohérence, plus rapide en moyenne
− : précision limitée au nombre de pixels

7

Backface culling

- Eliminer les parties de la surface pour lesquelles la normale pointe dans la direction opposée au point d'observation. Parties occultées par l'objet lui-même.

P_3 est visible, P_2 et P_3 sont à la limite de visibilité, P_4 n'est pas visible

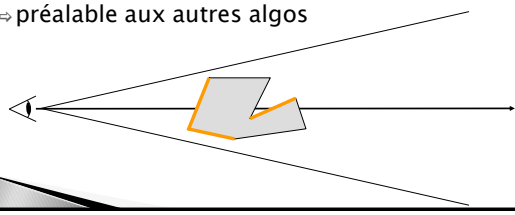


- La suppression des faces arrières suffit
  1. si l'objet est seul dans la scène (il ne faut pas qu'un objet puisse en masquer un autre)
  2. si l'objet est convexe (dans un objet concave, des faces avant peuvent être masquées par d'autres)

8

Backface culling

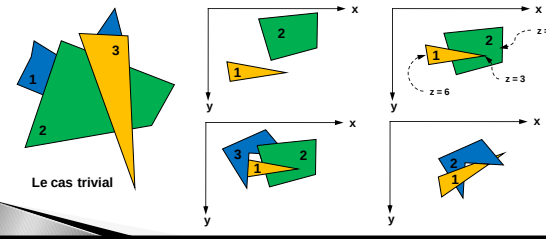
- **Calcul** : produit scalaire
  - (Sommet − PointDeVue) * normale
  - > 0 : on garde le polygone
  - < 0 : on l'élimine
- Économise 50 % du temps de calcul
  - ⇨ préalable aux autres algos



9

Algorithme du peintre

- Peindre les facettes polygonales dans la mémoire vidéo suivant un ordre de distance décroissante au point d'observation.
  1. Trier les facettes suivant les z décroissants dans le repère de la camera.
  2. Résoudre les ambiguïtés dans la liste lorsque les facettes se recouvrent.
  3. Projeter les facettes et remplir les polygones suivant la liste.



10

Algorithme du peintre : pour ou contre ?

- Le plus intuitif des algorithmes
- Coût en mémoire :
  - Affichage direct à l'écran : O(*p*)
  - Il faut trier les polygones : O(*n* log *n*)
- Temps de calcul :
  - On affiche toute la scène
  - Efficace surtout sur des petites scènes

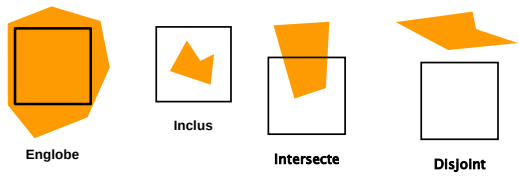
11

Area Subdivision (Warnock)

- On utilise la **cohérence spatiale**
- On divise l'écran en zones de travail
- Pour chaque zone, on ne considère que les polygones qui l'intersectent
- Si la visibilité est connue, on s'arrête
- Si la visibilité est inconnue, on subdivise
- On interrompt la subdivision quand on atteint une taille limite

12

Polygones/zone de travail

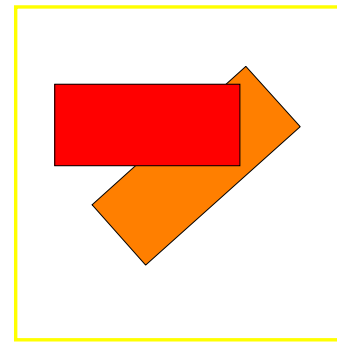


Quand est-ce que la visibilité est connue ?

- Tous les polygones sont ***disjoints***
- Un seul polygone ***intersecte*** ou ***inclus***
- Un polygone ***englobant*** qui est devant les autres

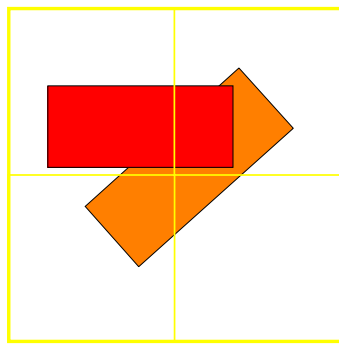
13

Warnock : exemple (1)



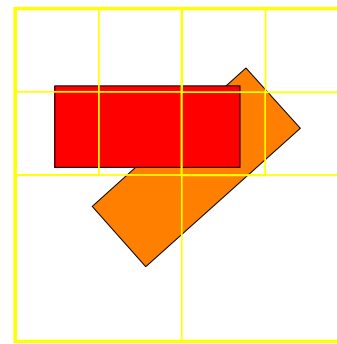
14

Warnock : exemple (2)



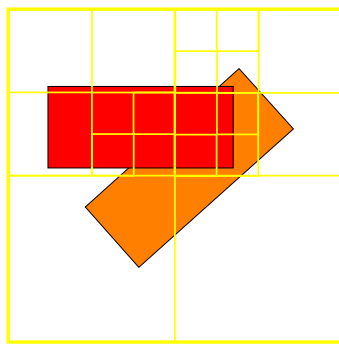
15

Warnock : exemple (3)



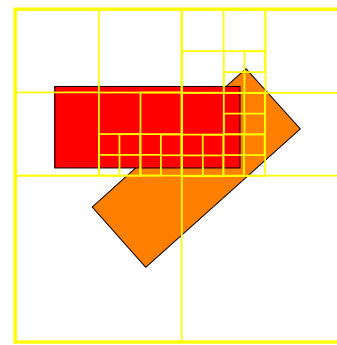
16

Warnock : exemple (4)



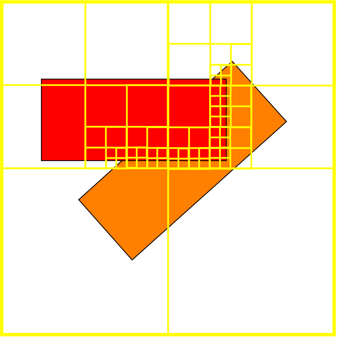
17

Warnock : exemple (5)



18

Warnock : exemple (6)



Warnock : pour ou contre ?

- Utilise la cohérence spatiale
- Plus efficace avec des grands polygones
- Coût mémoire parfois élevé
- Implémentation facile : appels récursifs à la même fonction

Z-buffer

- Un tableau, de la taille de l'écran
- On stocke la valeur maximale de z pour chaque pixel
  - *z* est la direction de visée, exprime la distance à l'œil
- Initialisation : tous les pixels à moins l'infini
- Projection de tous les polygones
  - On met à jour les pixels de la projection du polygone

Z-buffer : algorithme

- Pour chaque polygone :
  - Projeter le polygone sur le plan image
  - Pour chaque pixel dans la projection du polygone
    - Calculer la valeur de *z* pour ce pixel
    - Si z > à la valeur courante de z max
      - Changer z max
      - Afficher le pixel à l'écran, de la couleur du polygone

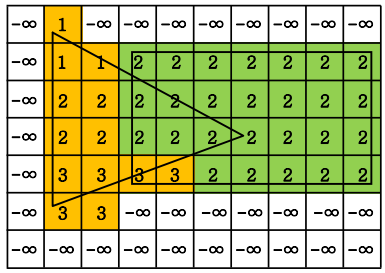
Z-buffer (1)

$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Z-buffer (2)

$-\infty$	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	1	1	1	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	2	2	2	2	2	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	2	2	2	2	2	2	$-\infty$	$-\infty$	$-\infty$
$-\infty$	3	3	3	3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	3	3	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$

Z-buffer (3)



25

Z-Buffer : pour ou contre

- **Pour :**
  - Facile à implémenter, scan-line
  - Travaille dans l'espace image
    - Rapide
- **Contre :**
  - Coût en mémoire
  - Travaille dans l'espace image
    - aliasing
    - artefacts

26

Z-Buffer

- Combien d'information ?
  - Combien de bits pour z max ?
  - Limité par la mémoire :
    - 8 bits, 1024x1280: 1.25 Mb
    - 16 bits, 1024x1280: 2.5 Mb
  - Nécessaire pour la séparation des objets proches :
    - 8 bits, distance minimale entre objets de 0.4 % (4mm pour 1m)
    - 16 bits, distance minimale de 0.001 % (1mm pour 1km)
    - Que se passe t-il en dessous de cette limite ?

27