Synthèse d'image

CHAPITRE 05:

Cours01:

ALGORITHMES DE BASE POUR L'AFFICHAGE

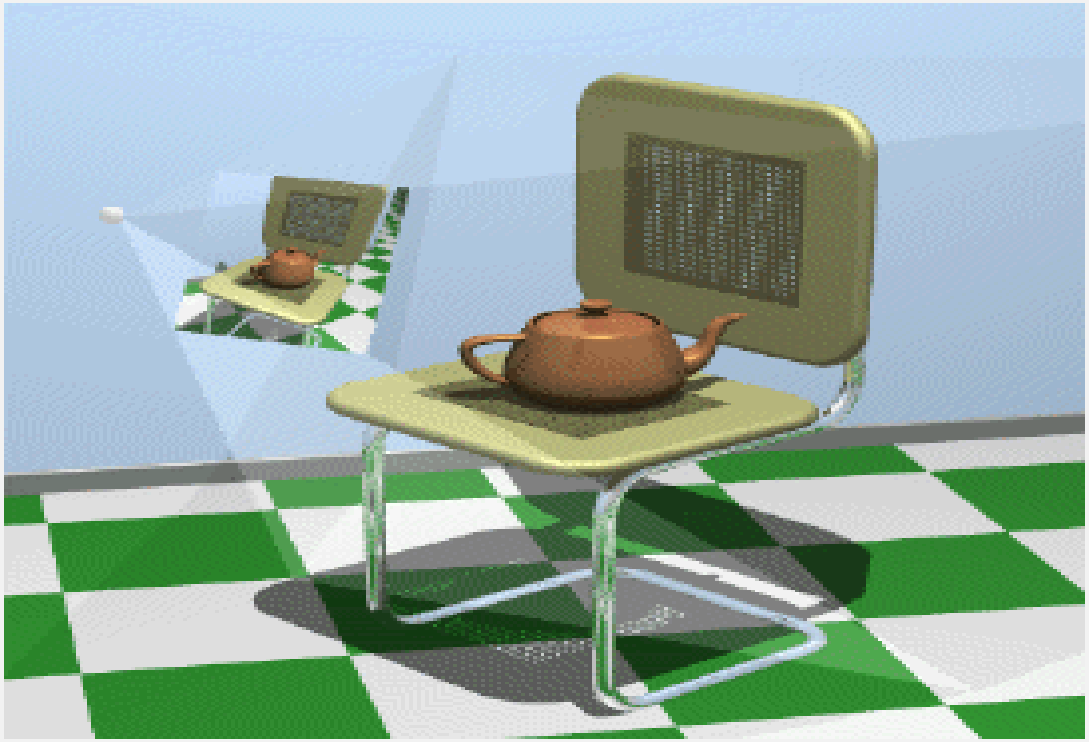
Introduction

- Après modélisation et représentation des objets de la scène. Il y'a affichage de ces objets sur un plan image
- Les effets suivants se produisent:
  - Il y'a des objets qui ne sont pas affichés sur l'image.
  - Il y'a des objets qui sont partiellement affichés sur l'image.
  - Il y' a des objets qui cachent d'autres objets.
- Raisons
  - Les objets ne sont pas (ou partiellement) dans le champs de vision de la caméra (**les parties hors champs de vision sont coupées**): Le clipping
  - Des objets sont cachés par d'autres (**les parties cachées sont éliminées**): élimination des faces cachées

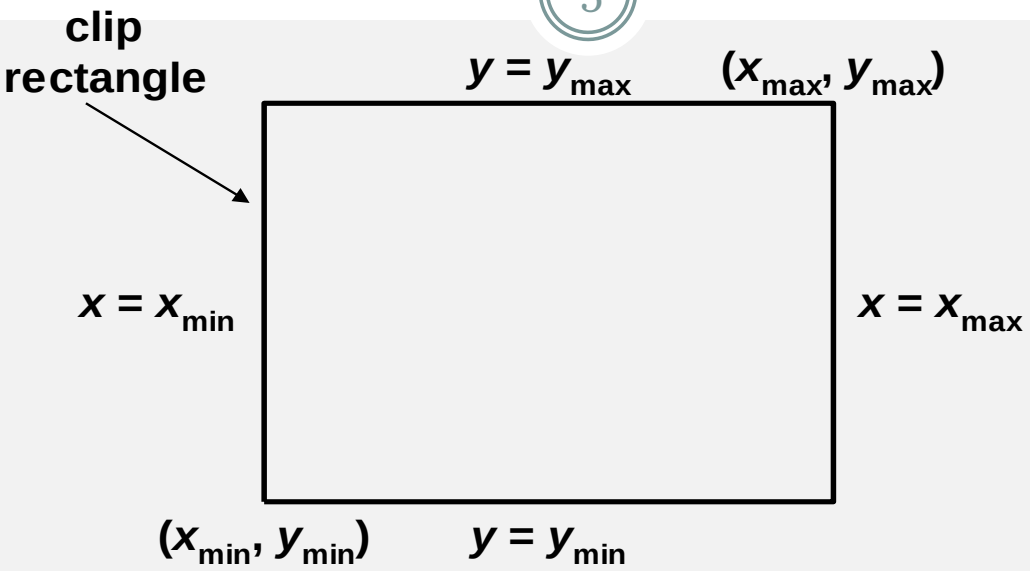
Le clipping (coupage)

Le Clipping (coupage)

- **Clipping** : Enlever les objets en dehors du volume de vue (frustum).



Principe en 2D



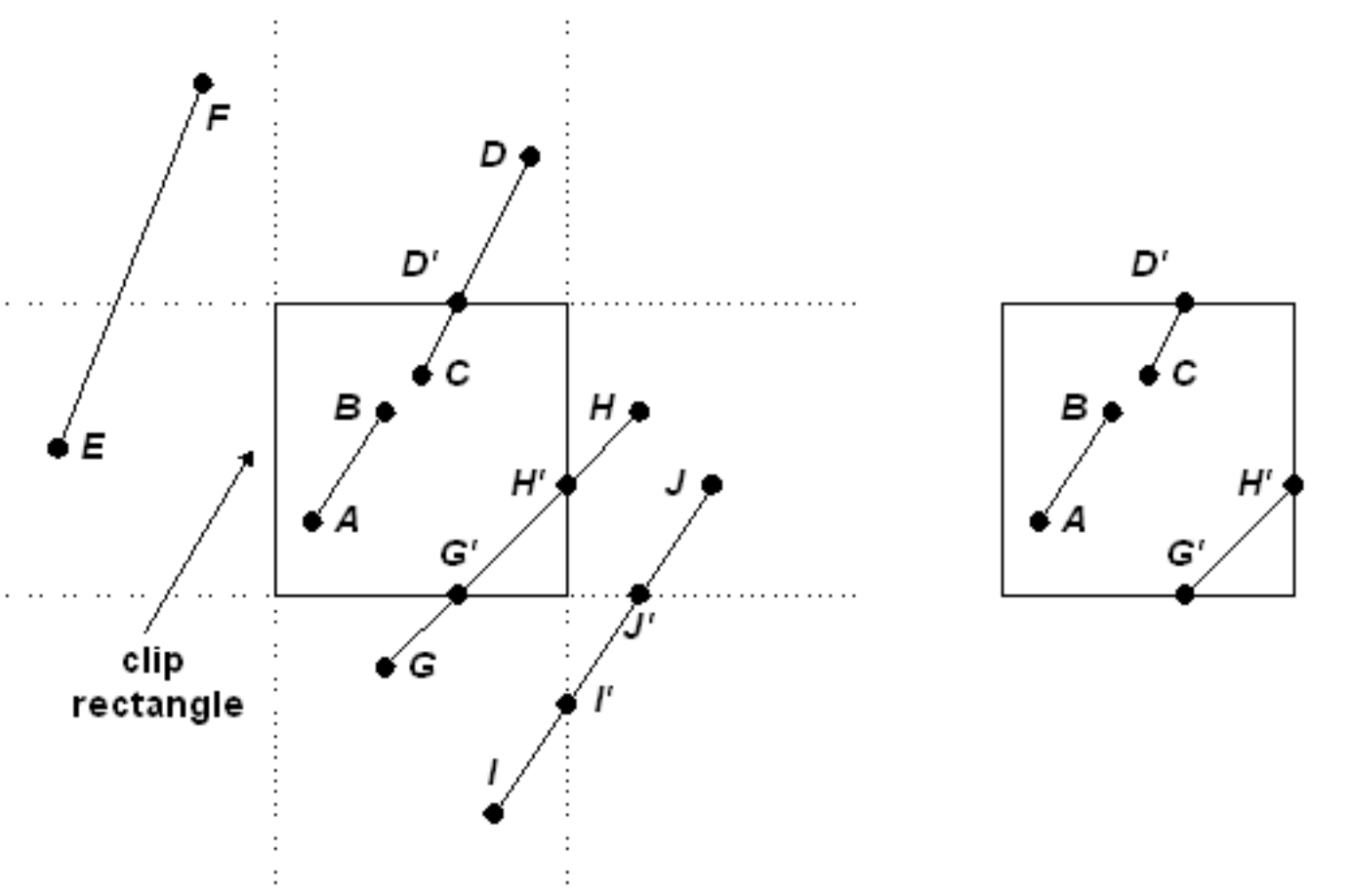
Pour qu'un point (x,y) soit dans le rectangle de coupage (clipping) :

$$x_{\min} \leq x \leq x_{\max}$$

$$y_{\min} \leq y \leq y_{\max}$$

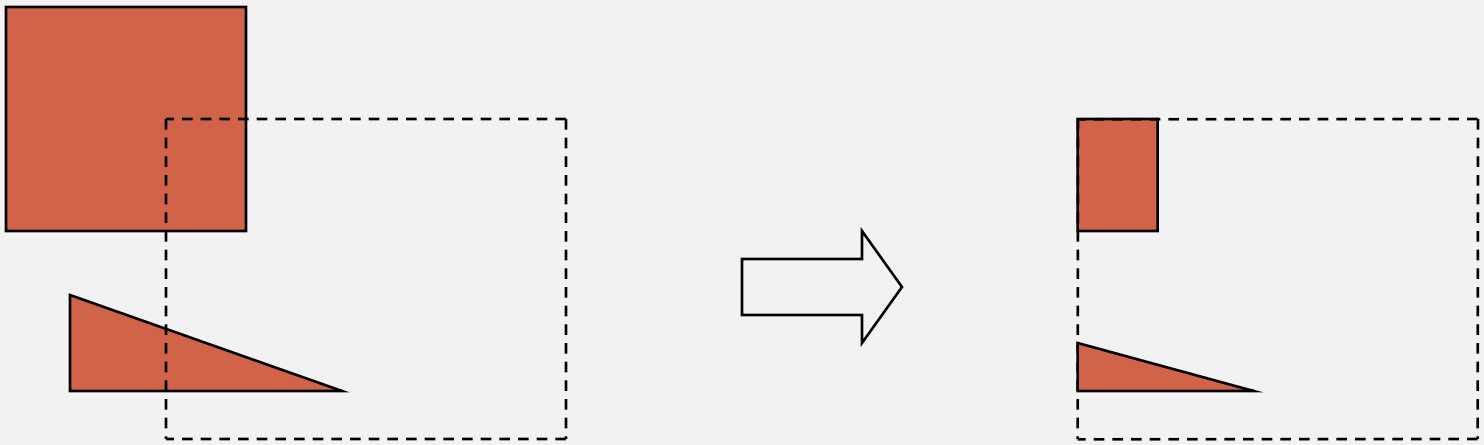
Principe en 2D

- Exemple en 2D: Clipping de lignes



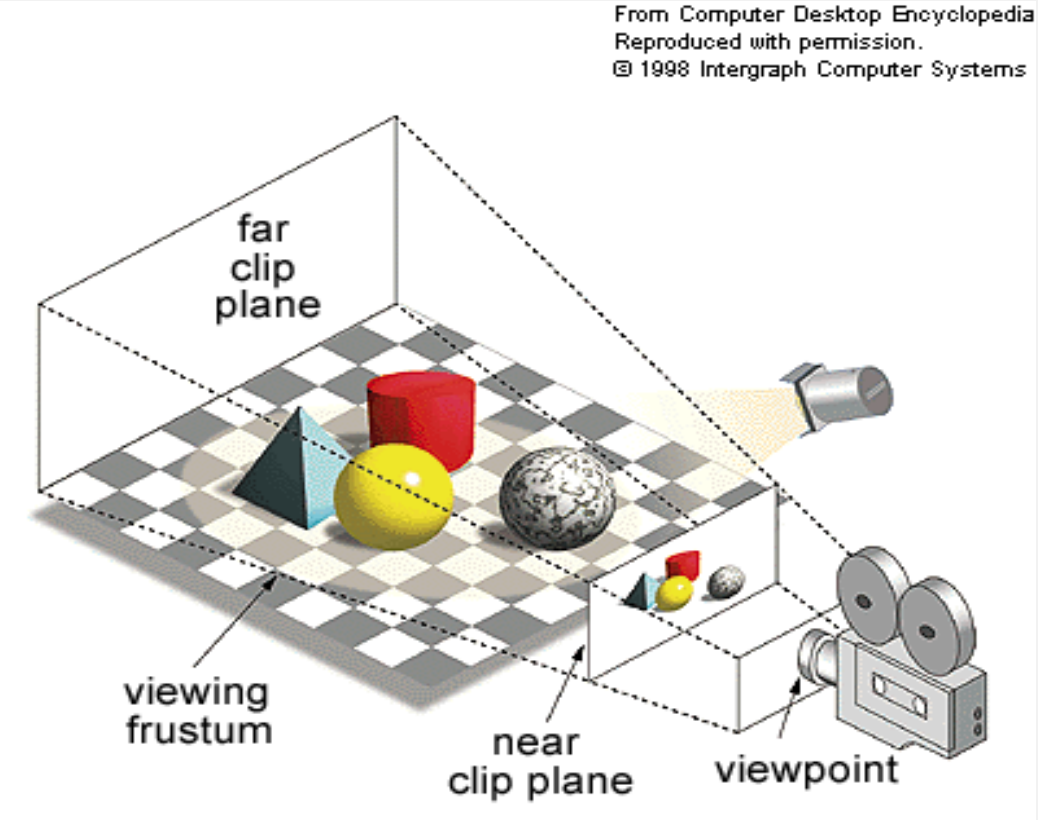
Principe en 2D

- Exemple en 2D: Clipping de polygones



Principe en 3D

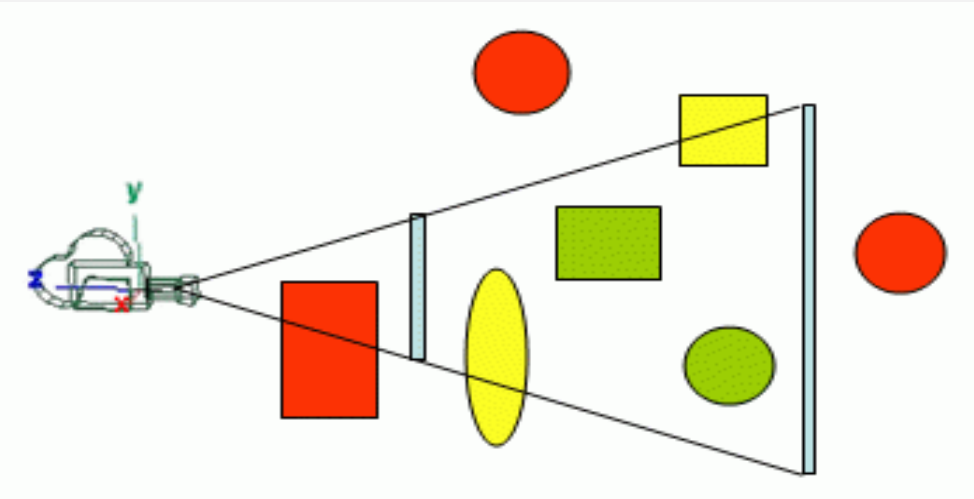
- Clipping en 3D:



Principe en 3D

En 3D :

- Les parties en dehors de la cône de vision sont coupés
- Les parties plus loin que le « plan loin (far)» ne son pas affichées.
- Les parties plus proche que le « plan proche (Near plan) » ne son pas affichées.



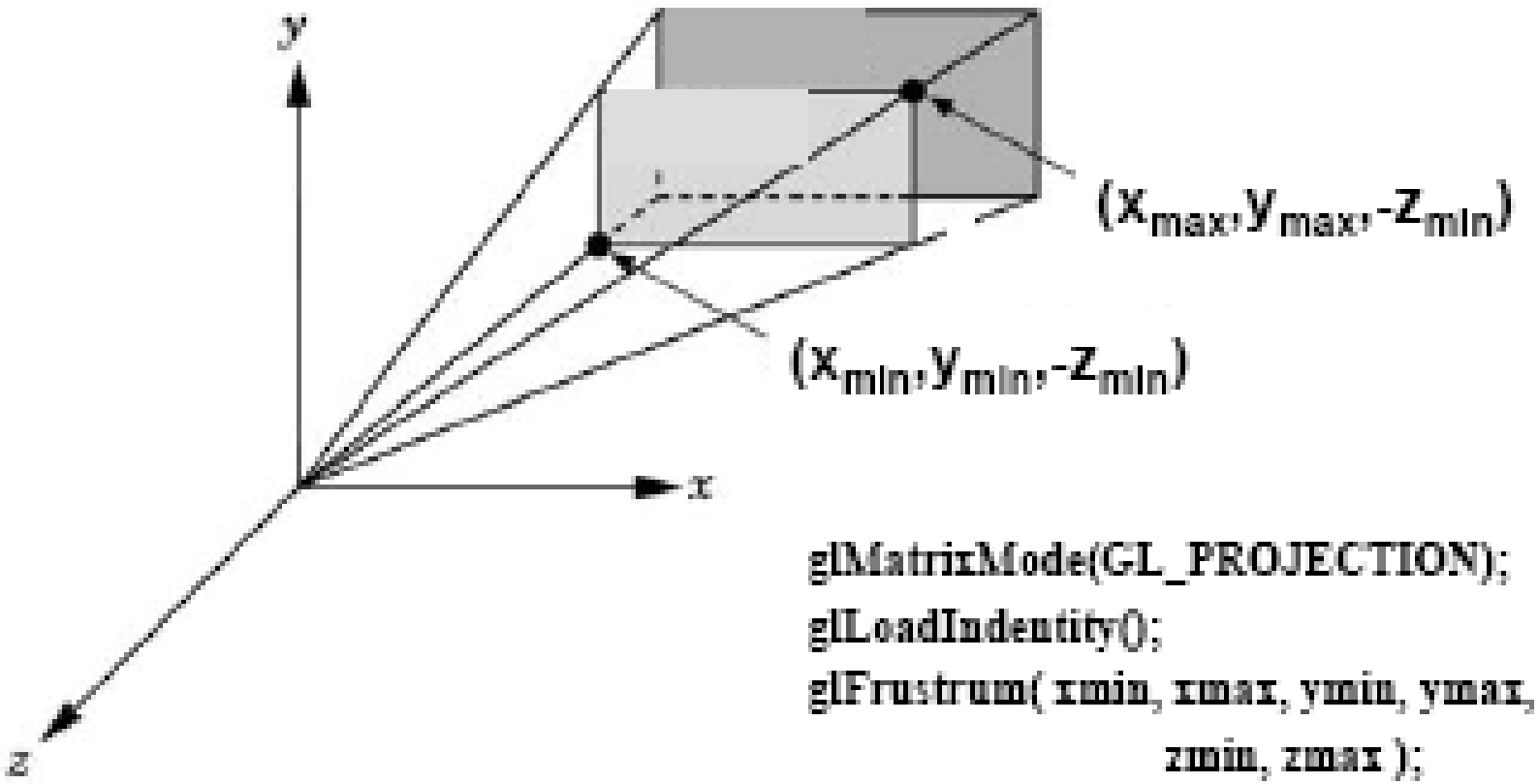
Clipping en OpenGL

En OpenGL

- Perspective projection
  - **gluPerspective(FOVy, aspect, zNear, zFar)**
  - **glFrustum(left, right, bottom, top, zNear, zFar)**
- Orthographic parallel projection
  - **glOrtho(left, right, bottom, top, zNear, zFar)**
  - **gluOrtho2D(left, right, bottom, top).**
- Position et orientation de la caméra gluLookAt();

Clipping en OpenGL

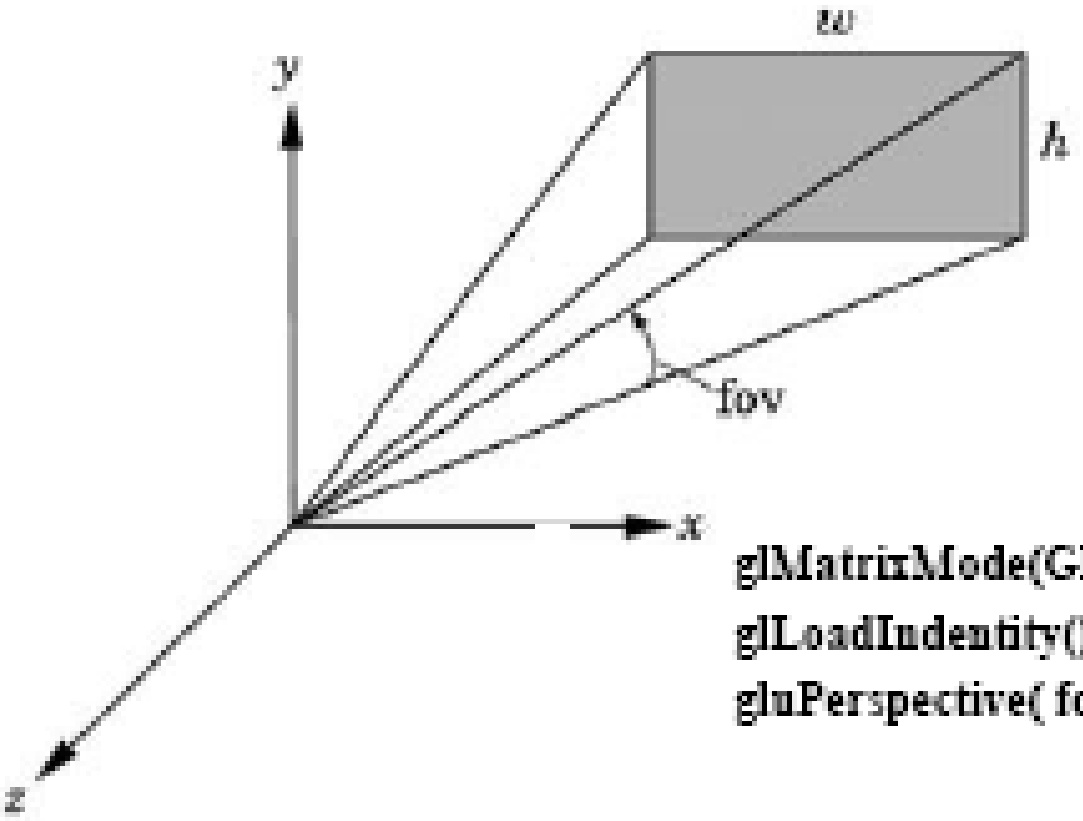
glFrustum(xmin, xmax, ymin ymax, near, far)




```
glMatrixMode(GL_PROJECTION);
glLoadIndentity();
glFrustrum( xmin, xmax, ymin, ymax,
                    zmin, zmax );
```

Clipping en OpenGL

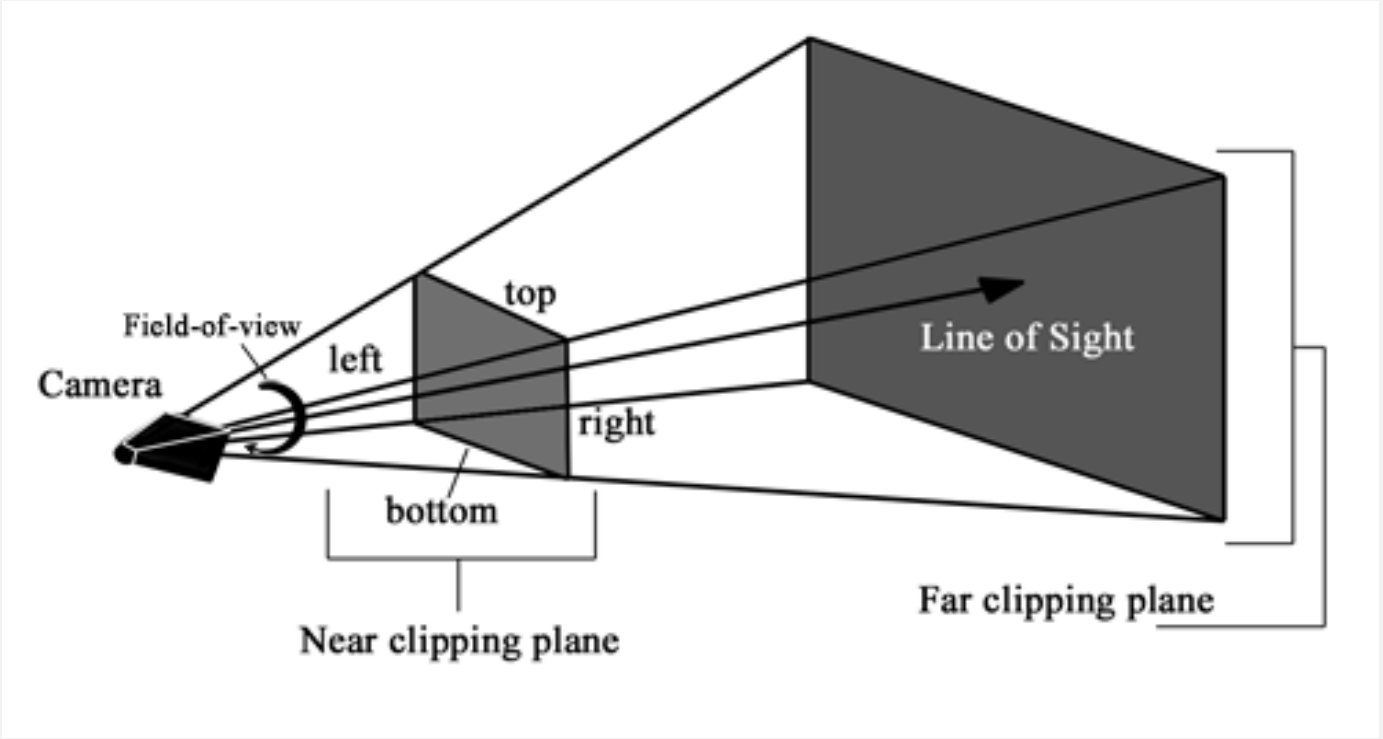
gluPerspective(fovy, aspect , near, far)




```
glMatrixMode(GL_PROJECTION);
glLoadIndentity();
gluPerspective( fovy, aspect,
                zmin, zmax );
```

Clipping en OpenGL

- **glOrtho(left, right, bottom, top, zNear, zFar)**



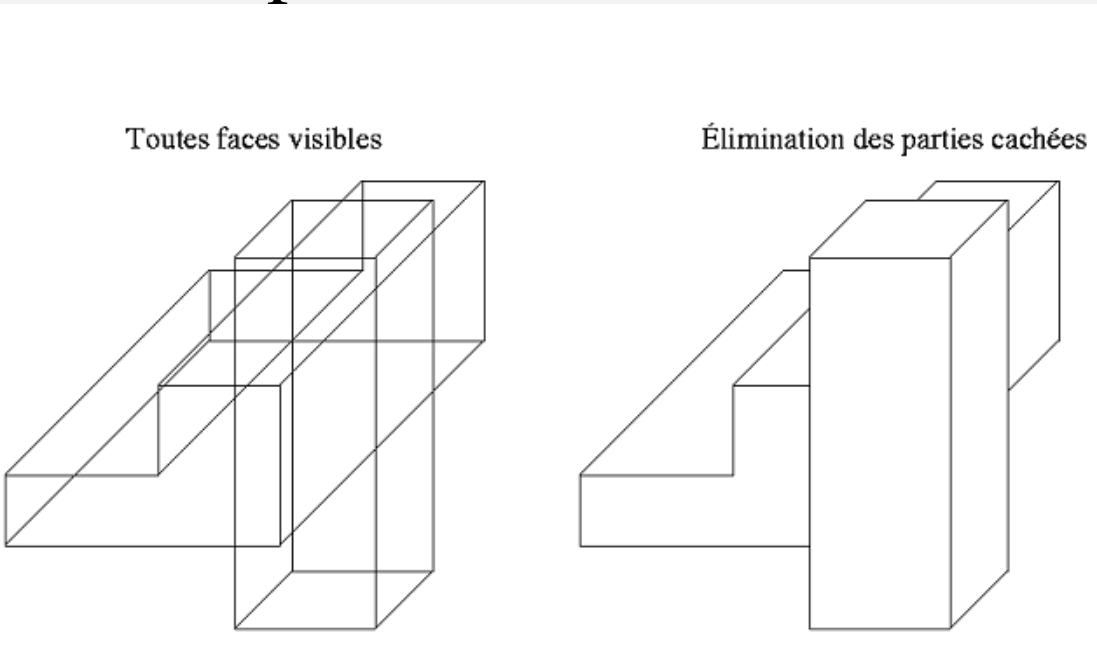
Les algorithmes de clipping

- Algorithme de :
  - Cohen Sutherland (Line)
  - Cyrus-Back (Line)
  - Sutherland-Hodgeman (Polygon)
  - Cohen Sutherland (3d)

Elimination des parties cachées

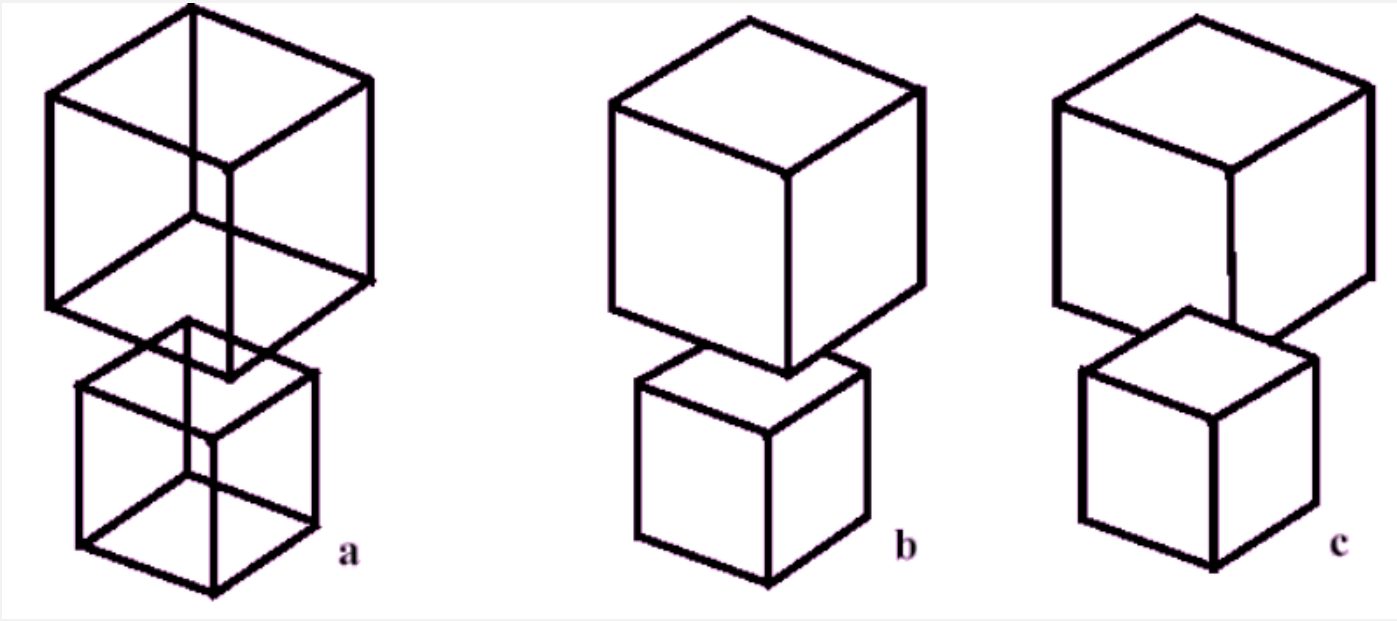
Principe

- Pour afficher une scène composée d'objets sur un dispositif d'affichage essentiellement **bidimensionnel**, il faut recréer par divers artifices l'impression de profondeur.



Principe

- Les lignes cachées dans l'image servent à donner l'illusion 3D.
- Elles permettent de voir les objets devant ou derrière.



Les algorithmes

Algorithmes d'élimination des parties cachées dans l'espace image et dans l'espace objet (scène)

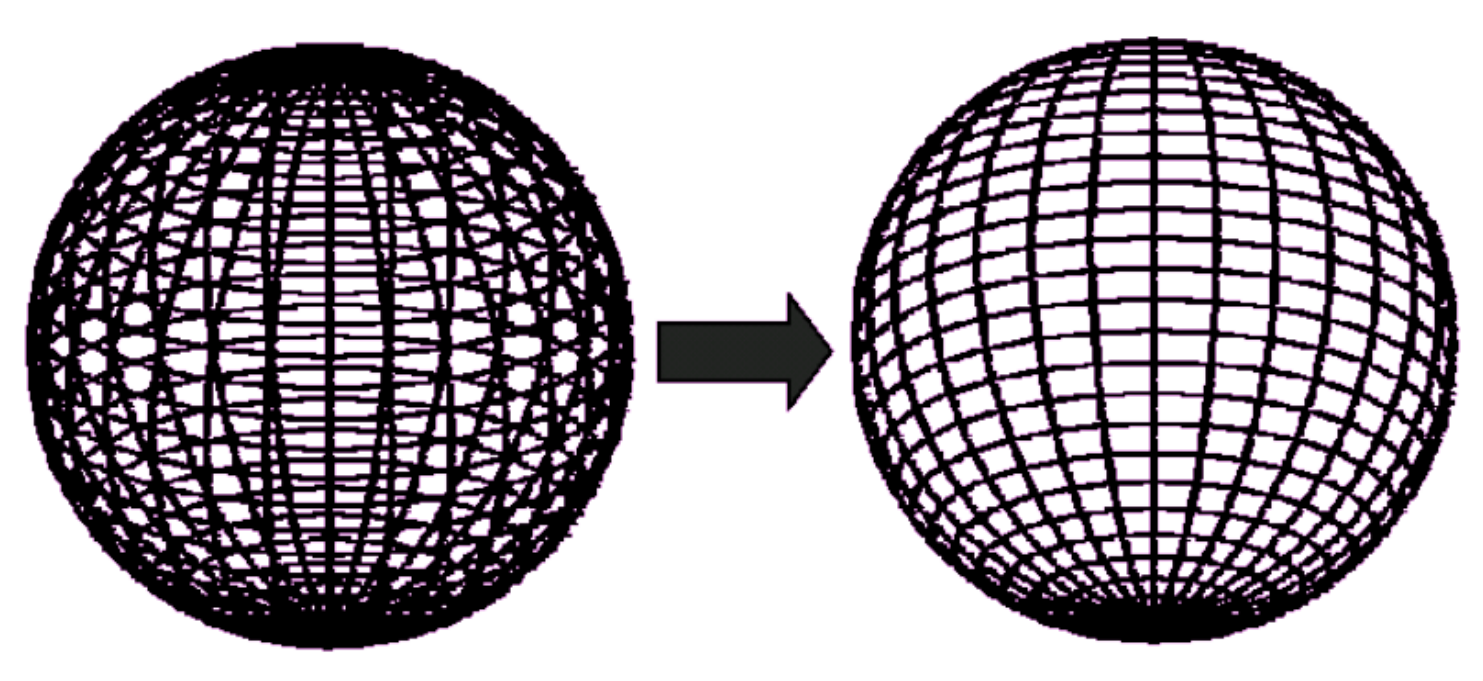
- Algorithmes dans l'espace objet
  - Backface culling
  - Algorithme du peintre (tri par la profondeur)
- Algorithmes dans l'espace image
  - Warnock
  - Z-buffer
  - Balayage de lignes (Scan line)

Les algorithmes

- Un algorithme qui travaille dans l'espace objet, effectue la majorité de ses calculs dans le système de coordonnées où la scène est définie. La caractéristique commune à cette famille réside dans la précision des résultats obtenus; elle n'est limitée que par la précision de la machine utilisée.
- Un algorithme travaillant dans l'espace image effectue pour sa part la majorité de ses calculs dans le système de coordonnées du plan de vision. La précision des résultats est alors limitée par le niveau de résolution du support d'affichage utilisé, généralement le nombre de pixels à l'écran.

Algorithmes dans l'espace objet

Principe : Détermination des surfaces visibles par élimination de tous les polygones qui ne sont pas tournés vers la caméra.

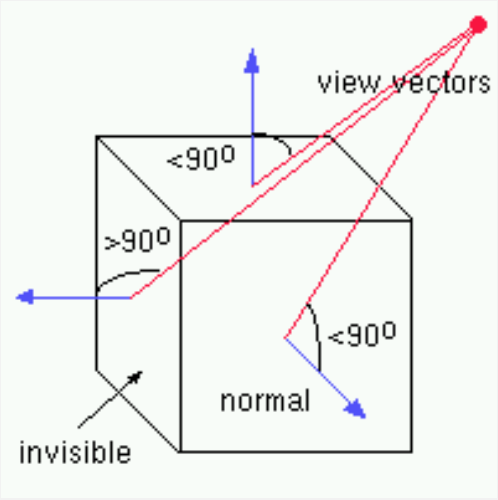


Algorithmes dans l'espace objet

- **Backface culling** (Test de visibilité) :
  - Pour chaque polygone, effectuer le produit scalaire entre le vecteur normal au polygone et le vecteur de projection à l'écran.

Produit scalaire > 0 : polygone non-visible (caché)

Produit scalaire < 0 : polygone visible (affiché)



Algorithmes dans l'espace objet

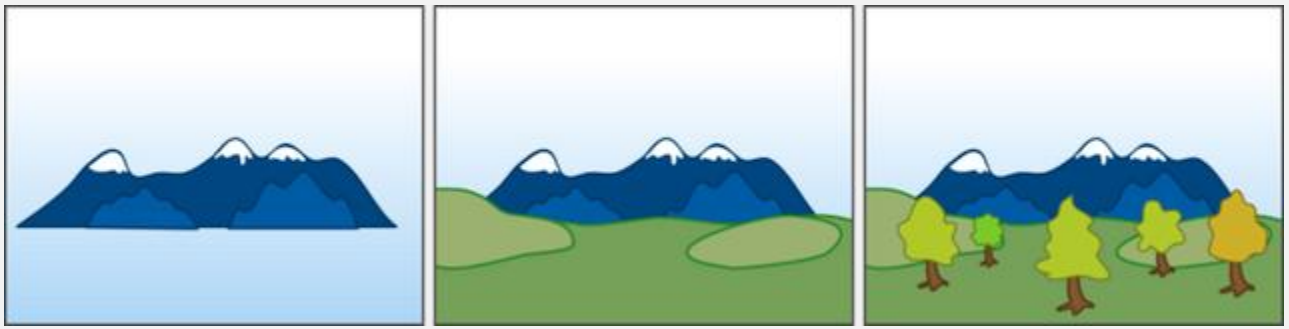
Evaluation de l'algorithme Backface culling

- Cette algorithme est une étape préliminaire pour les autres algorithmes. Economise 50 % du temps de calcul en moyenne.
- Suffisant pour un seul objet convexe.

Algorithmes dans l'espace objet

Algorithme du peintre:

- Analogie avec un peintre qui va peindre d'abord le ciel, puis les nuages et puis les oiseaux
- Affichage des objets en partant du plus lointain et en allant vers le plus proche.
- Remarque : Autre nom de cette algorithme: Tri par la profondeur (Depth-sort searching)



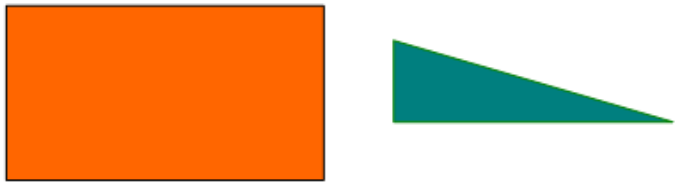
Algorithmes dans l'espace objet

- Algorithme :
  - Trier les objets par profondeur
  - Affichage des objets en partant du plus lointain et en allant vers le plus proche.
- Un polygone a une profondeur minimum et une profondeur maximum
  - Tri selon profondeur maximum
  - Comparaison des polygones sur l'intervalle (minimum-maximum).
  - Problème si chevauchement des profondeurs.

Algorithmes dans l'espace objet

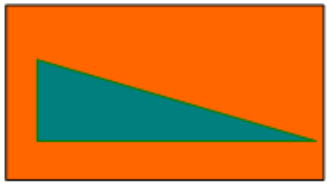
- **Différents cas:**

A) Polygones disjoints en X,Y : aucun problème



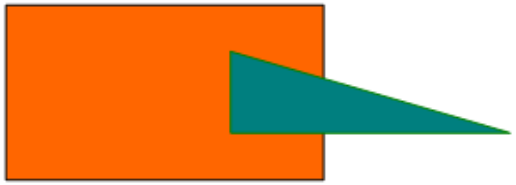
Aucun problème

B) Un polygone est inclus dans l'autre
- comparer les profondeurs Z



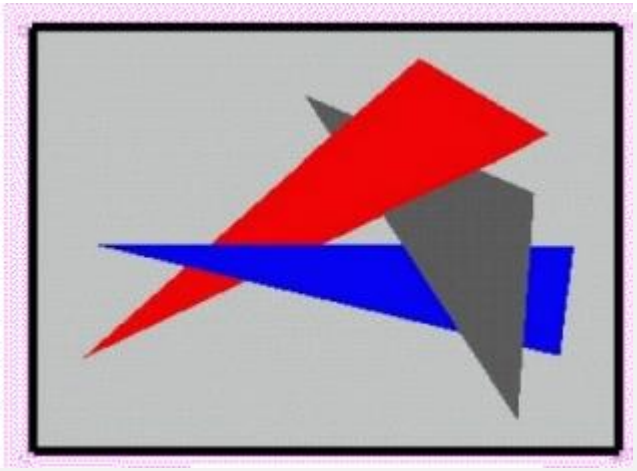
Aucun problème

C) Les deux polygones s'intersectent partiellement
- faire plus de tests sur Z



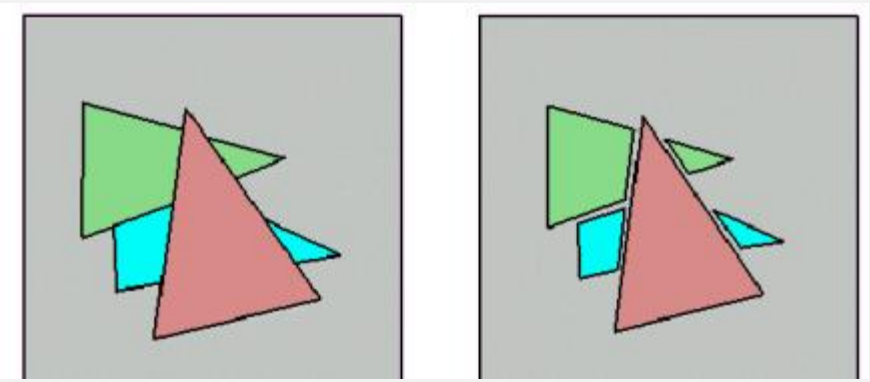
Possibilités de problèmes !

Algorithmes dans l'espace objet



- Problèmes :
  - Objets sur plusieurs profondeurs
  - Objets entrelacés.
- Solution :
  - découpage en polygones plus petits

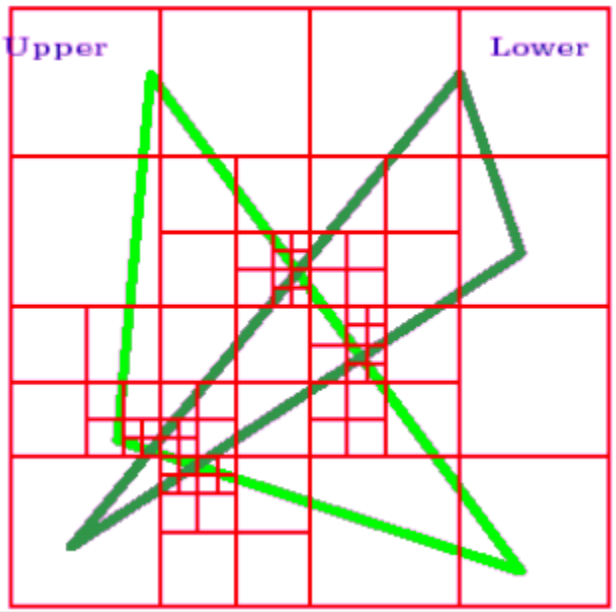
(et on recommence le tri)



Algorithmes dans l'espace image

Algorithme de Warnock

- Subdiviser l'image en quatre zones carrées.
- Si aucun polygone dans une zone
  - Remplir la zone avec la couleur du fond
- Si un seul polygone dans la zone
  - Remplir avec la couleur du fond
  - Dessiner le polygone
- Si plusieurs polygones dans la zone
  - -Subdiviser la zone en quatre zones carrées



Algorithmes dans l'espace image

Evaluation de l'algorithme Warnock

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

Algorithmes dans l'espace image

Algorithme du Z-buffer

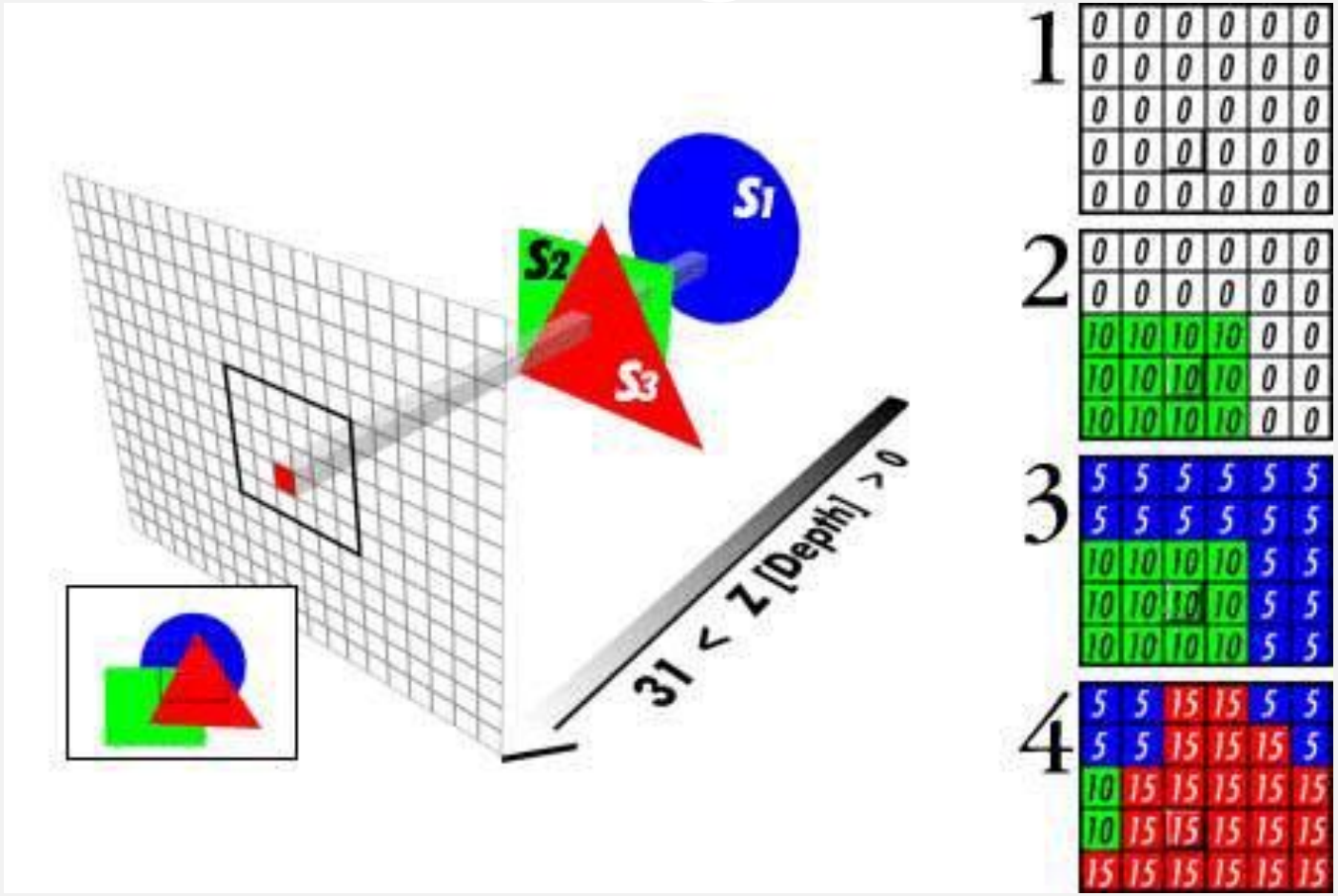
- L'algorithme du z-buffer [1974] consiste à stocker, pour chaque pixel de l'écran, non seulement les valeurs couleurs (framebuffer) mais aussi la profondeur (z-buffer).
- Le z-buffer est initialisé à l'infini. Durant le processus d'affichage d'un polygone, si le point traité a un z inférieur à celui correspondant dans le z-buffer, alors ce point est affiché et sa profondeur z remplace l'ancienne valeur dans le z-buffer.

Algorithmes dans l'espace image

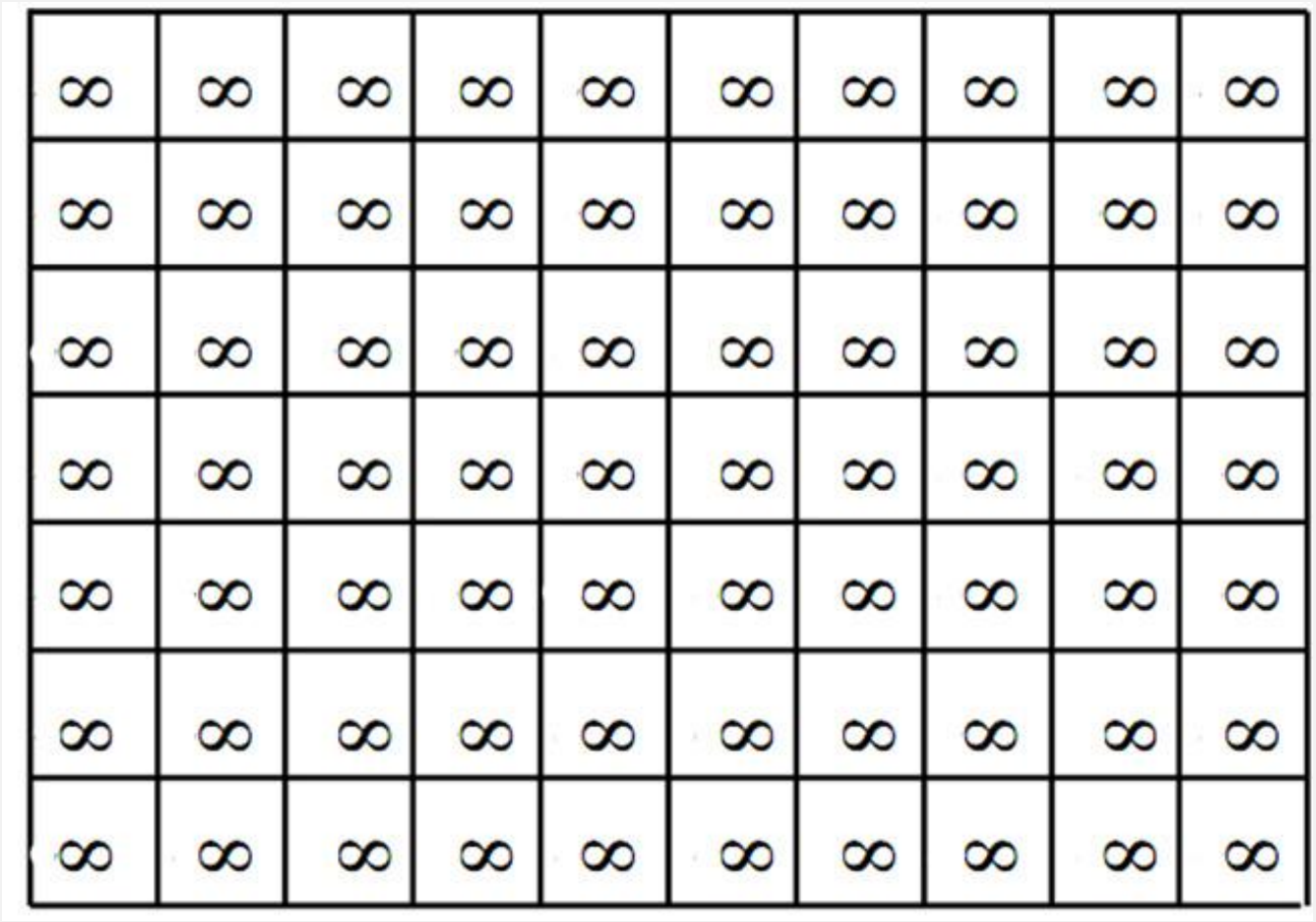
L'algorithme du Z-buffer

- **Pour** chaque polygone
  - **Pour** chaque pixel (u,v) du polygone
    - z = profondeur du point du polygone correspondant au pixel (u,v)
    - **Si** z < z_buffer(u,v) **alors**
      - afficher(u,v,couleur) /* framebuffer(u,v) = couleur */
      - z_buffer(u,v) = z
    - **finSi**
  - **finPour**
- **finPour**

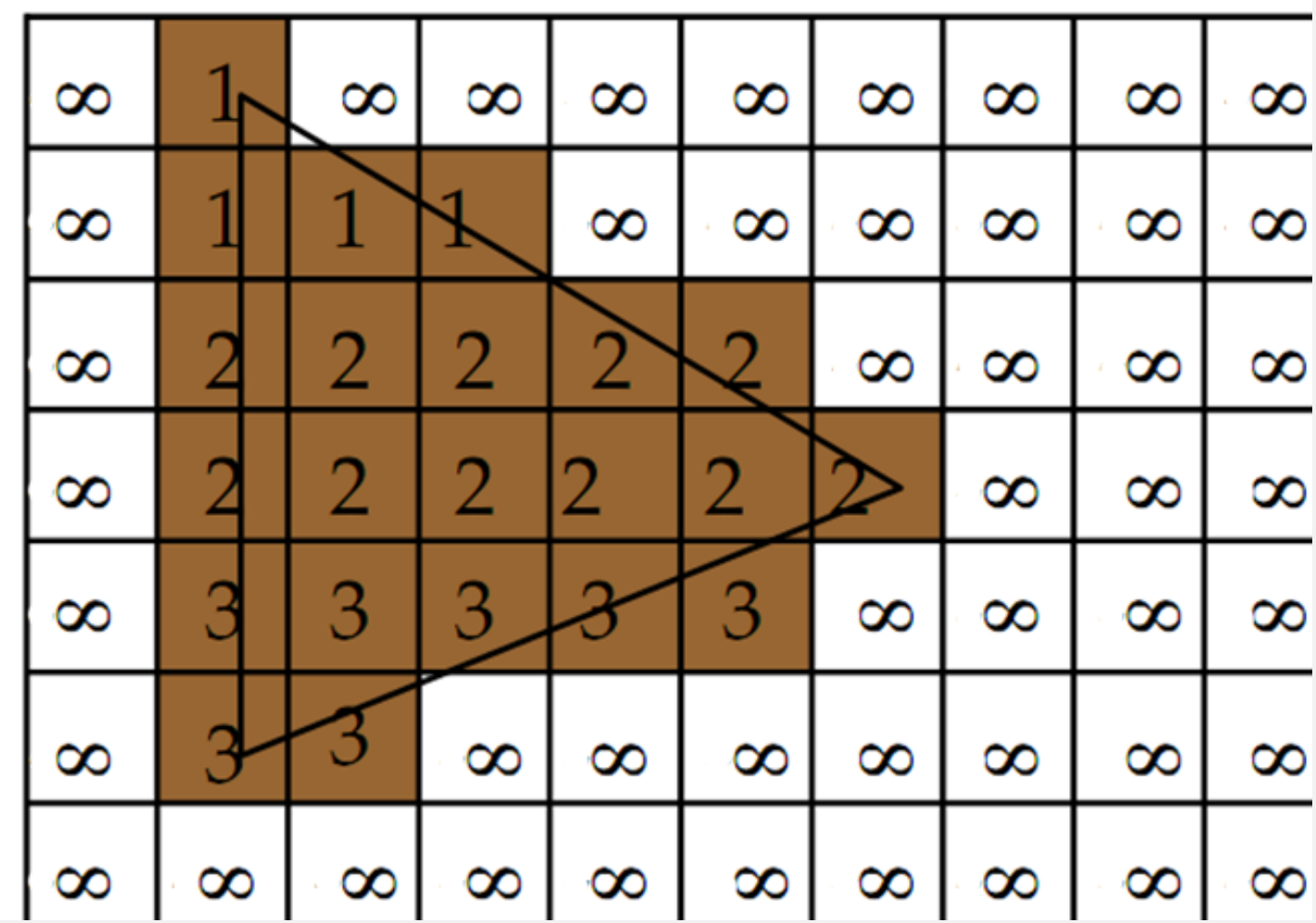
Algorithmes dans l'espace image



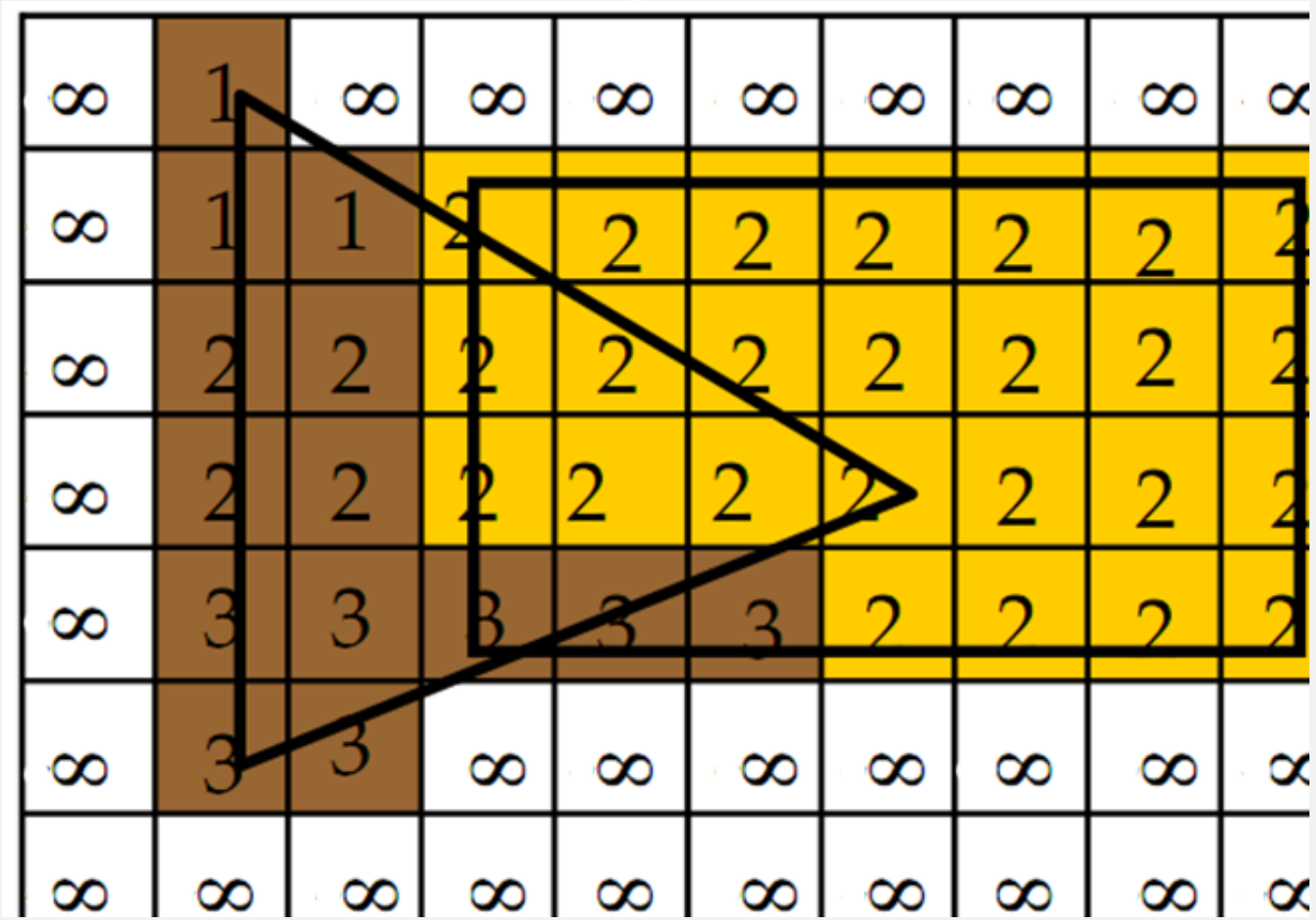
Algorithmes dans l'espace image



Algorithmes dans l'espace image



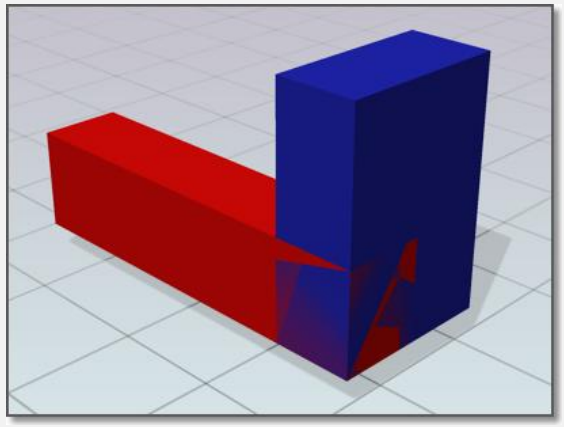
Algorithmes dans l'espace image



Algorithmes dans l'espace image

Evaluation de l'algorithme du Z-buffer

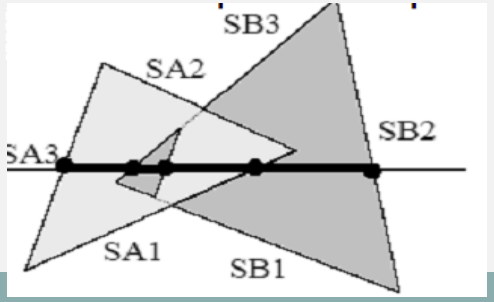
- Avantages
  - Le meilleur choix pour une implémentation matérielle.
  - Facile à implémenter,
  - Rapide et précis
  - Pas de pré-traitement
- Problèmes :
  - Coût en mémoire
  - Aliasing, artefacts



Algorithmes dans l'espace image

Balayage de ligne (Scan Line): Amélioration de l'algorithme z-buffer.

- Le z-buffer doit visiter chaque pixel couvert par un objet, même s'il est caché.
- Le balayage de lignes ne calcule la visibilité qu'entre les polygones.
  - Traiter plusieurs polygones à la fois par balayage,
  - La profondeur n'est calculée que lorsque plus d'un polygones s'entrecoupent.



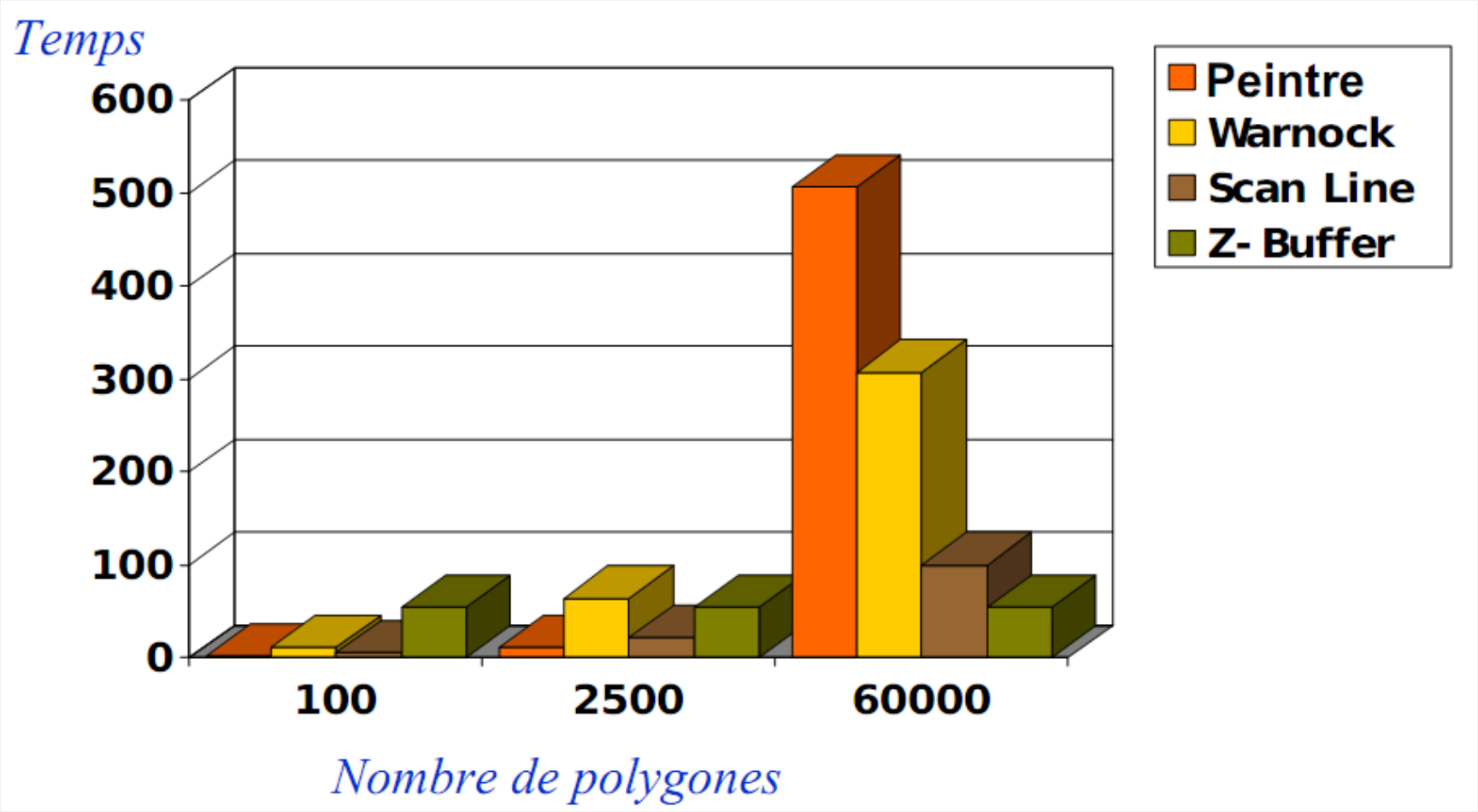
Algorithmes dans l'espace image

Evaluation de l'algorithme de Balayage de ligne

- Faible coût mémoire
- Fonctionne bien sans utiliser la carte graphique

Algorithmes dans l'espace image

Coûts comparés



Fin du Cours
Question ??????