La synthèse d'images

- Rendu Temps réel -

Jonathan Fabrizio

http://jo.fabrizio.free.fr

Version : Tue Mar 9 10:13:12 2021

Le Rendu

Rendu temps réel vs Rendu photoréaliste

- Rendu photoréaliste
  - Objectif :
    - Génération d'images réalistes
    - Contrainte de temps faible...
  - Stratégies :
    - Object-based rendering algorithms
      Illumination globale calculée indépendamment du point de vue
    - Image-based rendering algorithms
      Illumination calculée partiellement, en fonction du point de vue
    - Deterministic rendering algorithms
    - Monte Carlo rendering algorithms
- Rendu temps réel
  - Objectif :
    - Génération rapide d'images

Le Rendu Temps Réel

Le Rendu Temps Réel

- Principe général
- Algorithmes 3D fondamentaux
- Algorithmes 2D fondamentaux

Le Rendu Temps Réel

Principe général

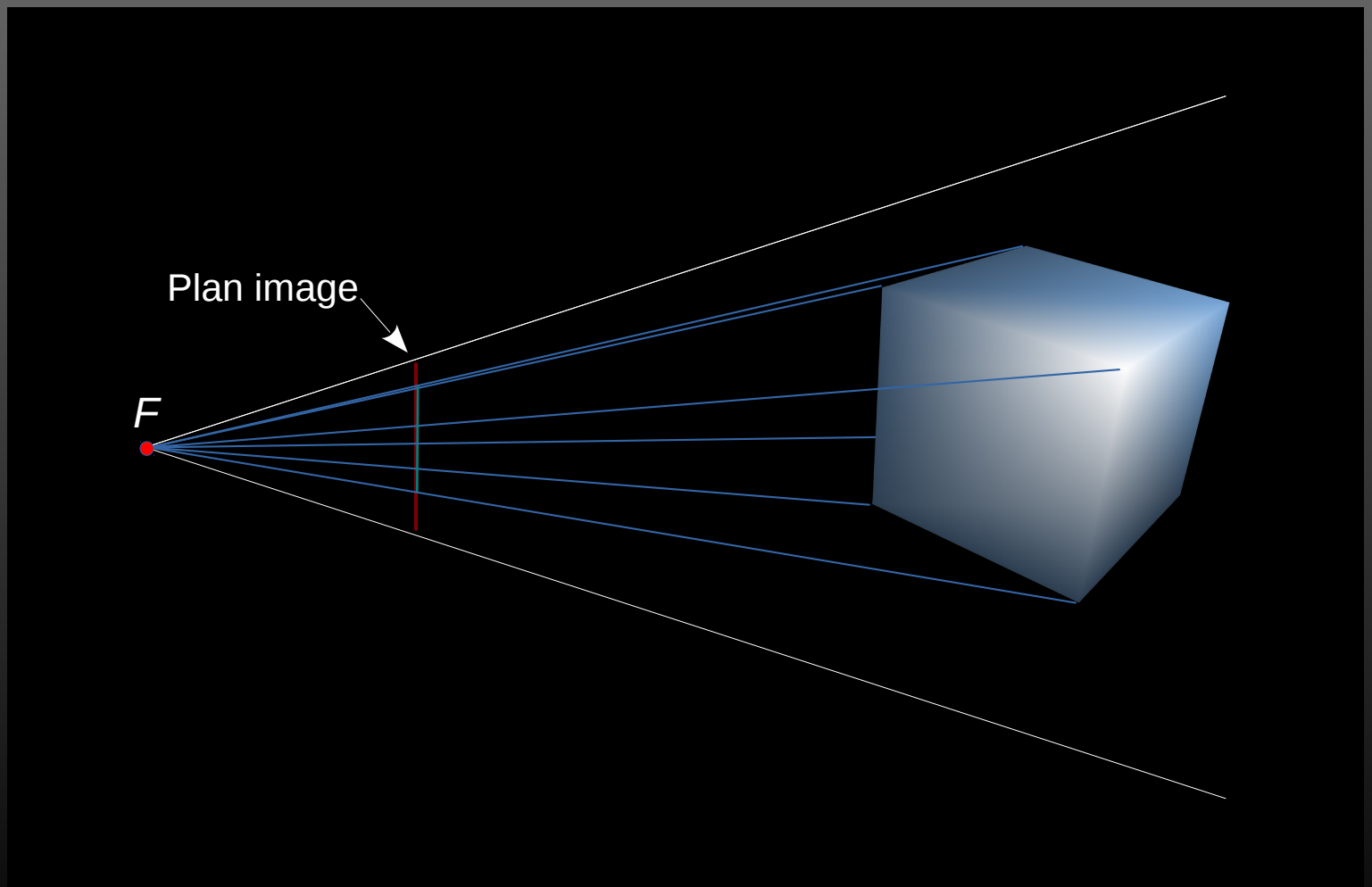
- Modélisation des objets dans un repère local
- Modélisation de la scène dans un repère global
- Projection de la scène sur le plan image
  - passage repère global au repère caméra
  - projection sur le plan image (+dessin 2D)

Le Rendu Temps Réel

Algorithmes 3D fondamentaux

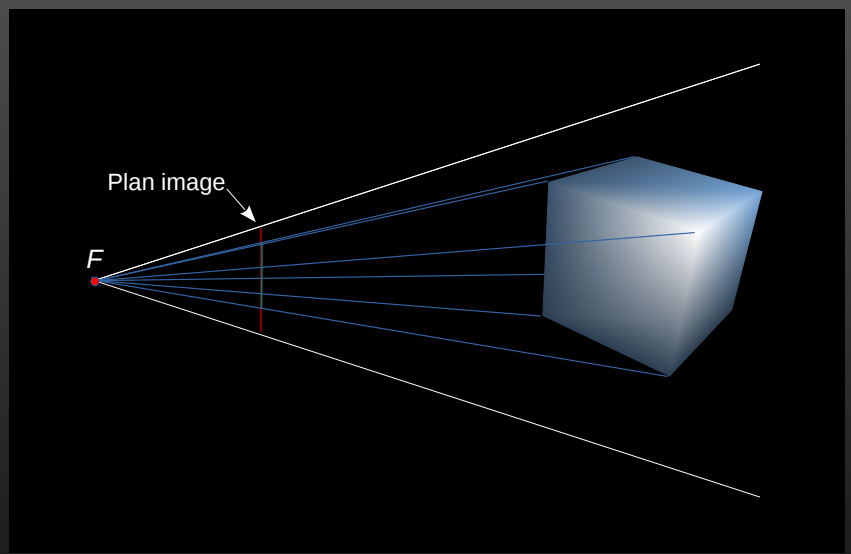
Algorithmes 3D fondamentaux

Projection des objets sur le plan image.



Algorithmes 3D fondamentaux

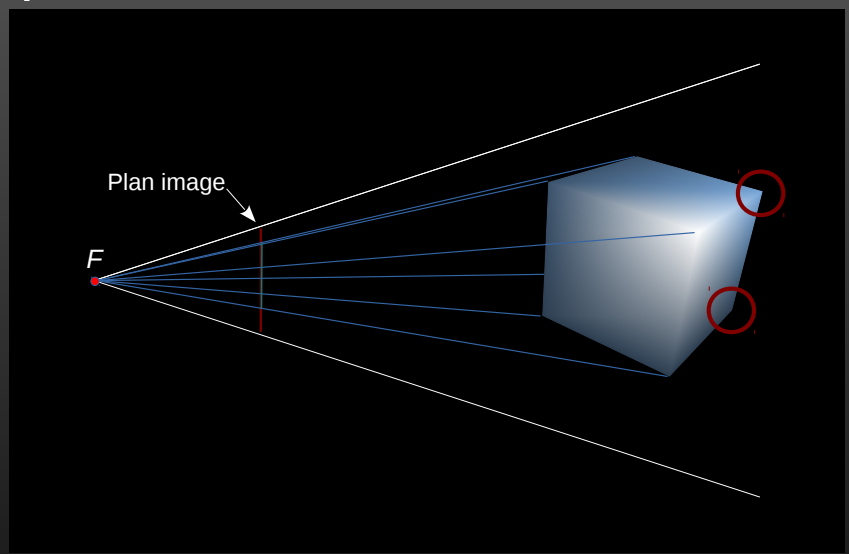
Projection des objets sur le plan image.
Il faut identifier les problèmes !



Algorithmes 3D fondamentaux

Projection des objets sur le plan image.
Il faut identifier les problèmes !

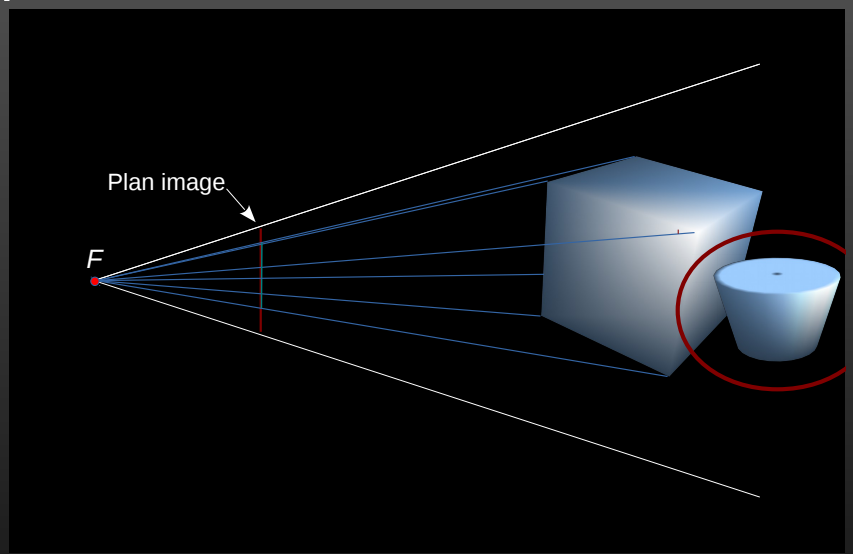
- Comment déterminer les sommets/faces non visibles ?



Algorithmes 3D fondamentaux

Projection des objets sur le plan image.
Il faut identifier les problèmes !

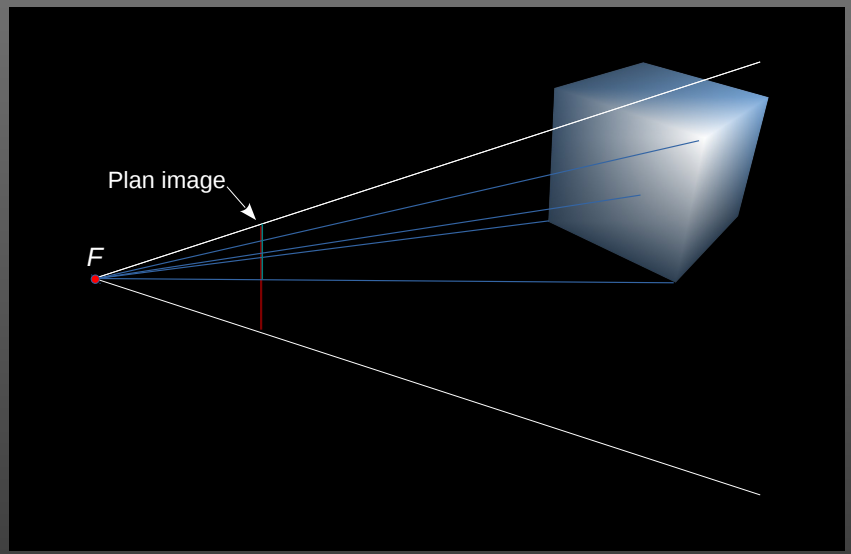
- Comment déterminer les sommets/faces non visibles ?
- Comment déterminer les objets cachés (ou partiellement cachés) ?

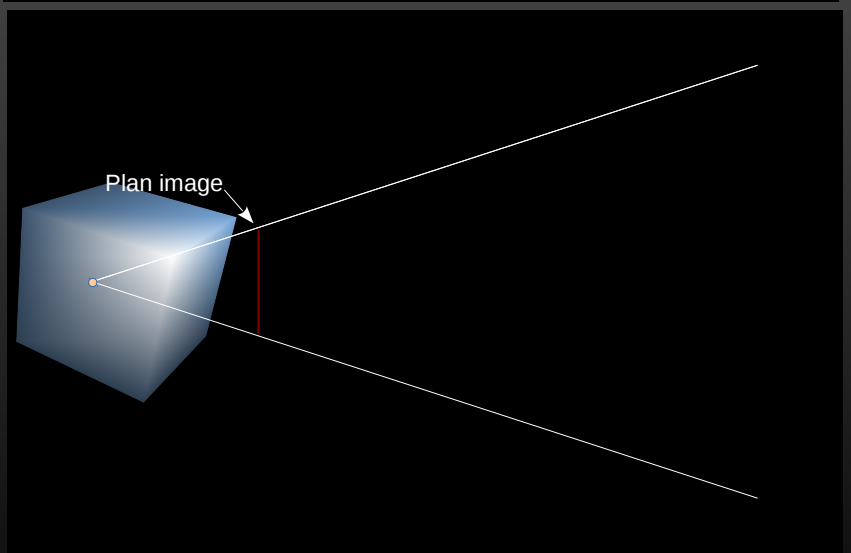


Algorithmes 3D fondamentaux

Projection des objets sur le plan image
Il faut identifier les problèmes !

- Comment déterminer les sommets/faces non visibles ?
- Comment déterminer les objets cachés (ou partiellement cachés) ?
- Comment déterminer les objets qui sont hors du champ (ou partiellement hors du champ/derrière le plan image) ?





Algorithmes 3D fondamentaux

Projection des objets sur le plan image.
Il faut identifier les problèmes !

- Comment déterminer les sommets/faces non visibles ?
- Comment déterminer les objets cachés (ou partiellement cachés) ?
- Comment déterminer les objets qui sont hors du champ (ou partiellement hors du champ/derrière le plan image) ?

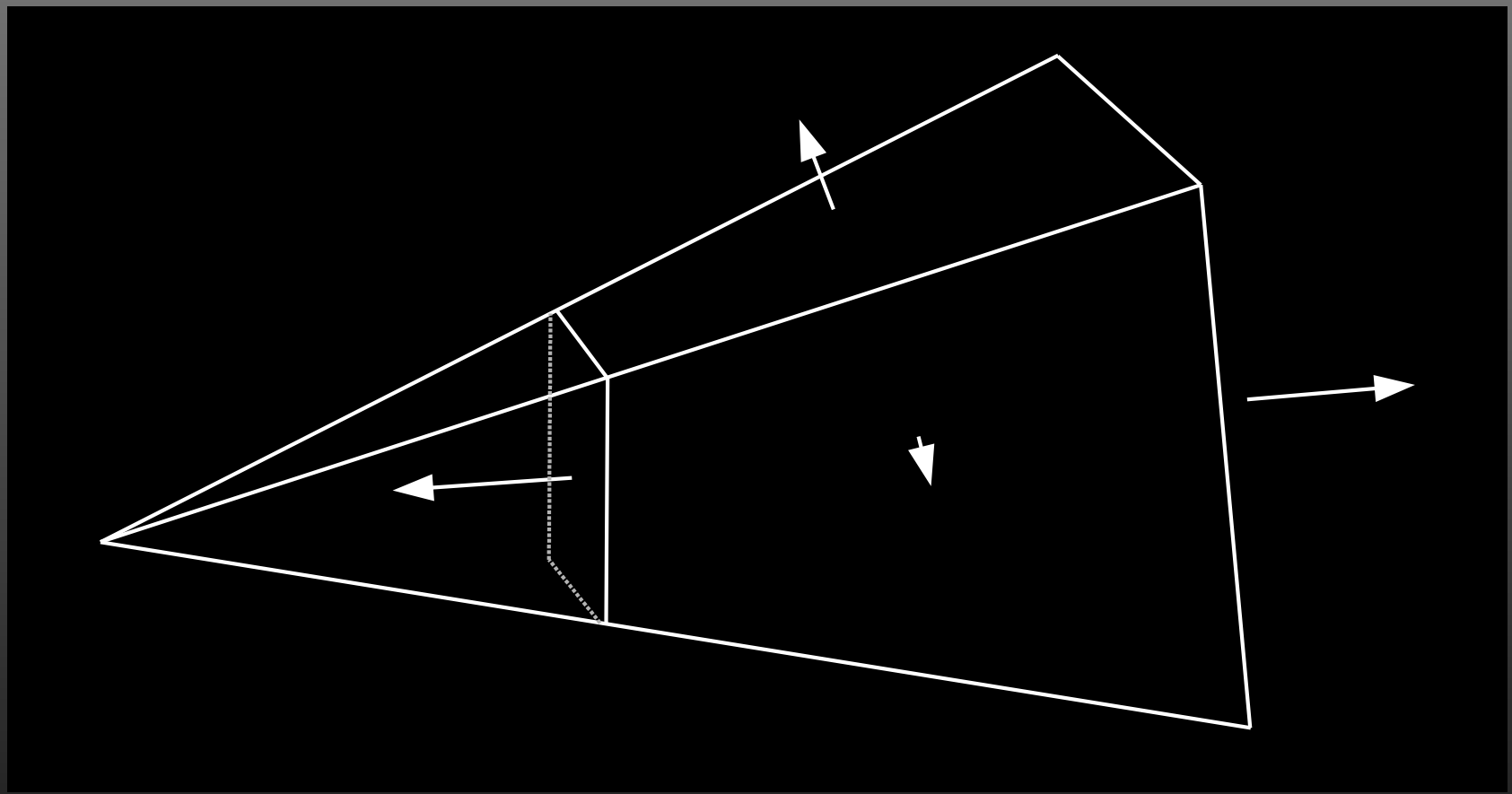
afin de les résoudre et :

- avoir une projection correcte
- être efficace

Algorithmes 3D fondamentaux : *Clipping*

- Comment déterminer les objets qui sont hors du champ (ou partiellement hors du champ) ?
- Comment déterminer les objets qui sont derrière le plan image ou partiellement visible ?

Algorithmes 3D fondamentaux : *Clipping*



- Estimation de la position d'une face par rapport aux plans
- Élimination des faces à l'extérieur
- Découpage des polygones à cheval (équations paramétriques)

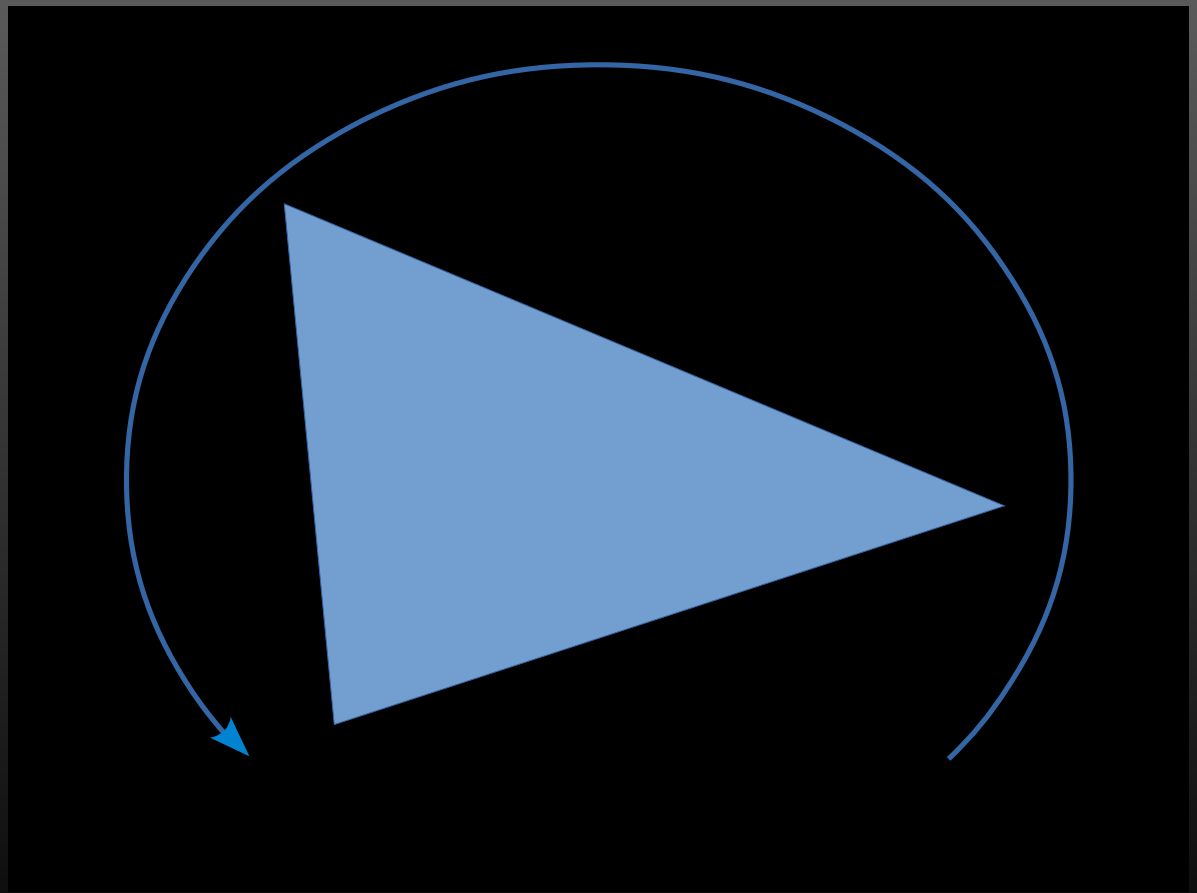
Algorithmes 3D fondamentaux

Comment déterminer les faces non visibles ?

Algorithmes 3D fondamentaux : *Backface culling*

Comment déterminer les faces non visibles ?

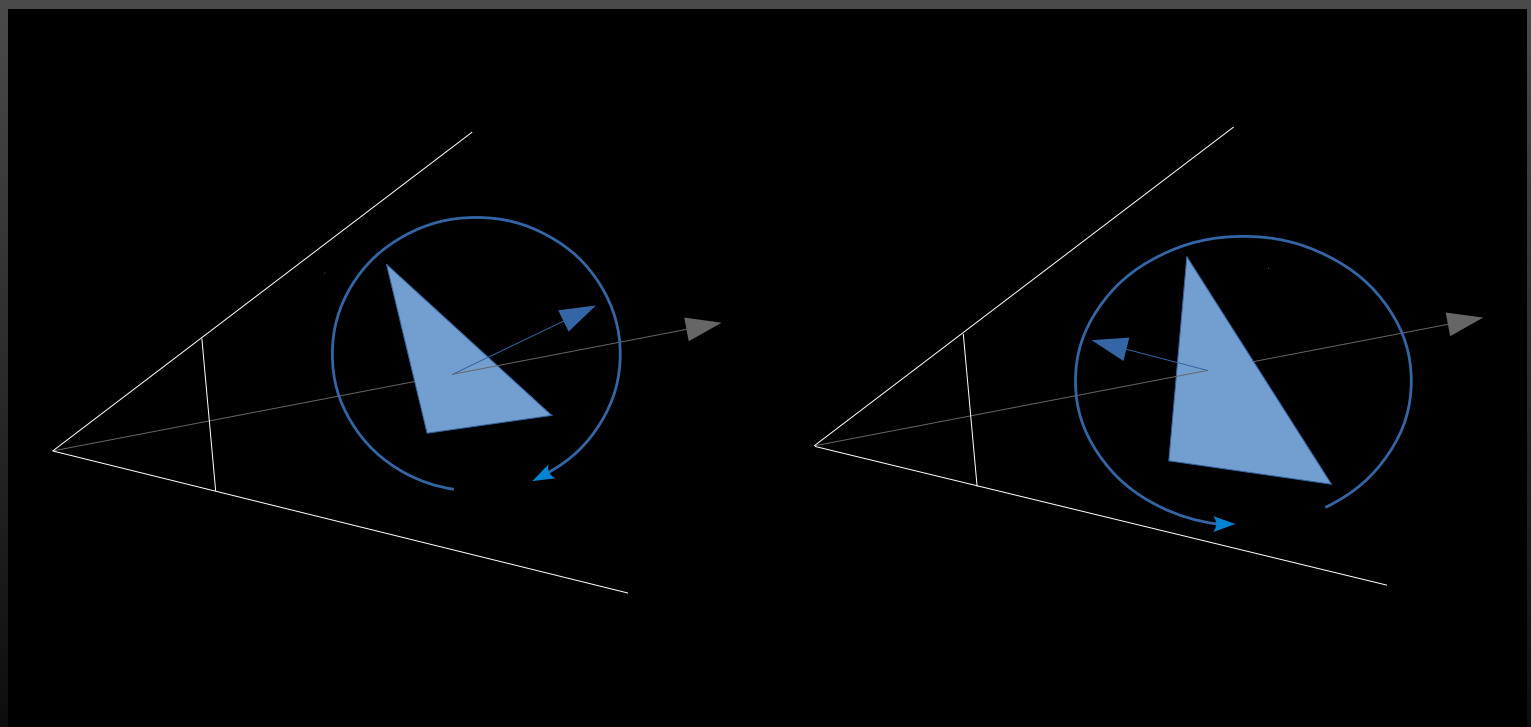
- énumérer les sommets toujours dans le même sens



Algorithmes 3D fondamentaux : *Backface culling*

Comment déterminer les faces non visibles ?

- déterminer l'orientation de la face par rapport à l'axe optique :
  - Calculer le vecteur normal à la surface (produit vectoriel)
  - Déterminer l'angle entre le vecteur normal à la surface et le vecteur directeur de l'axe optique (produit scalaire)



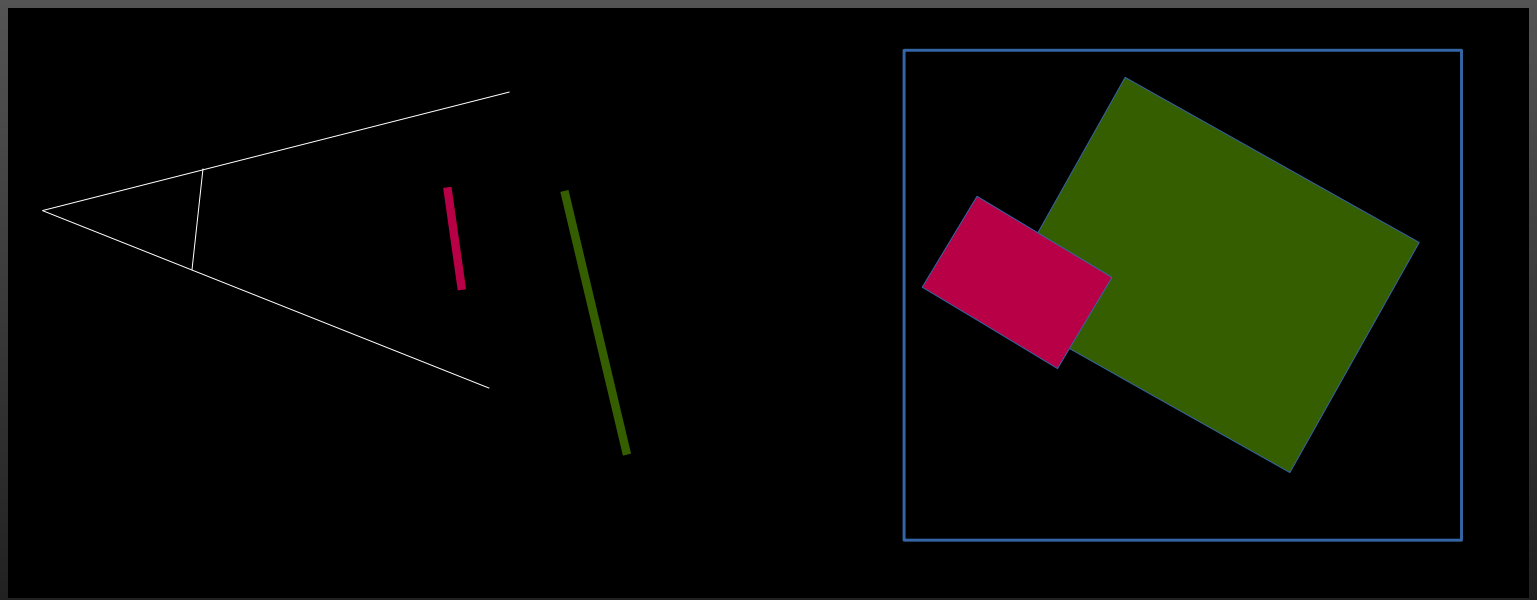
Algorithmes 3D fondamentaux

Comment déterminer les objets cachés (ou partiellement cachés) ?

Algorithmes 3D fondamentaux

Comment déterminer les objets cachés (ou partiellement cachés) ?

- Trier les objets et les dessiner dans l'ordre

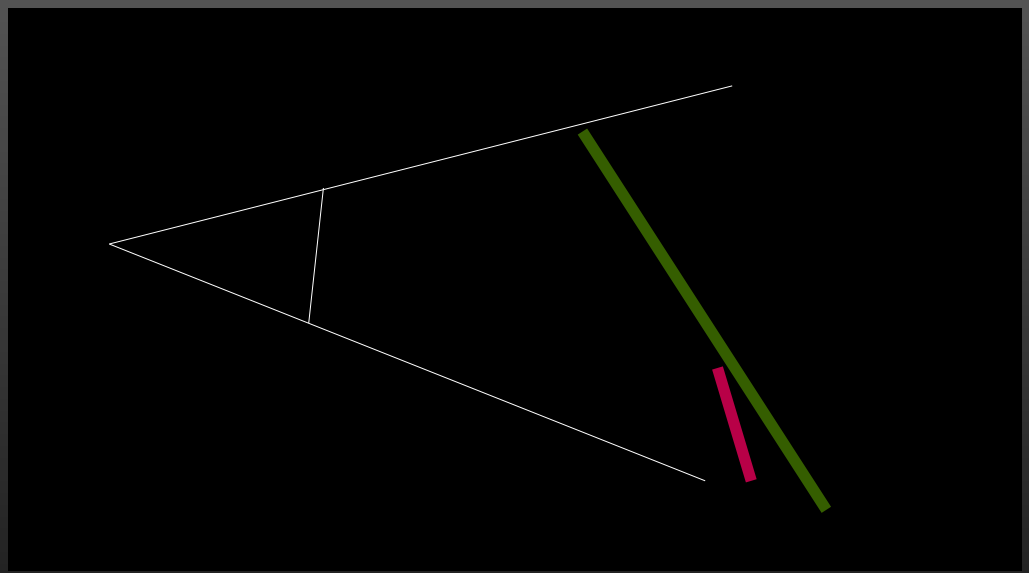


- Comment déterminer cet ordre ?

Algorithmes 3D fondamentaux

Comment déterminer les objets cachés (ou partiellement cachés) ?

- Trier les objets et les dessiner dans l'ordre



- Ne fonctionne pas dans tous les cas !

Algorithmes 3D fondamentaux

Comment déterminer les objets cachés (ou partiellement cachés) ?

- Utilisation d'un arbre B.S.P. (Binary Space Partitionning Tree)
  - Chaque nœud représente un hyperplan (déduit d'une face F)
  - Le premier fils contient les faces du demi-espace derrière F et le second fils contient les faces du demi-espace devant F.
  - Lorsque l'hyperplan intersecte une face, la face est coupée en deux

Algorithmes 3D fondamentaux : *B.S.P.*

Comment déterminer les objets cachés (ou partiellement cachés) ?

- Utilisation d'un arbre B.S.P. (Binary Space Partitionning Tree)
  - On peut déduire un ordre de parcours des polygones pour les dessiner du plus éloigné au plus proche
  - idem, du plus proche au plus éloigné
- Efficacité :
  - Compromis entre arbre équilibré et nombre de polygones (fragmentation des polygones)

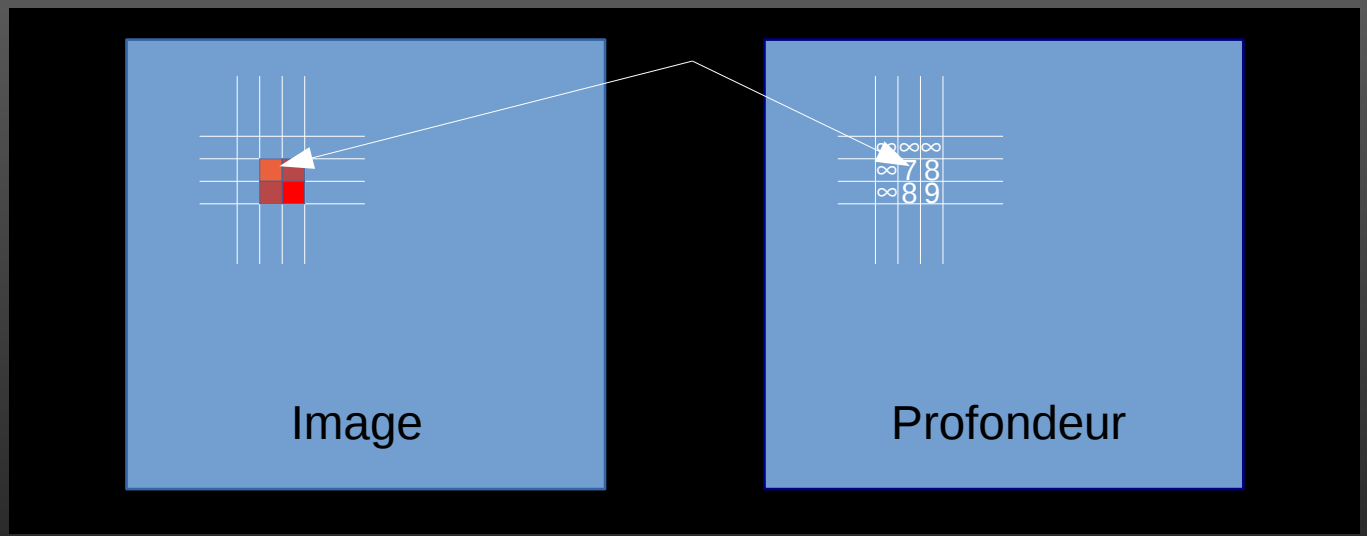
Algorithmes 3D fondamentaux :

Comment déterminer les objets cachés (ou partiellement cachés) ?

Algorithmes 3D fondamentaux : *Z-buffer*

Comment déterminer les objets cachés (ou partiellement cachés) ?

- Utilisation du Z-buffer
  - Sauvegarde de la profondeur pour chaque pixel dessiné

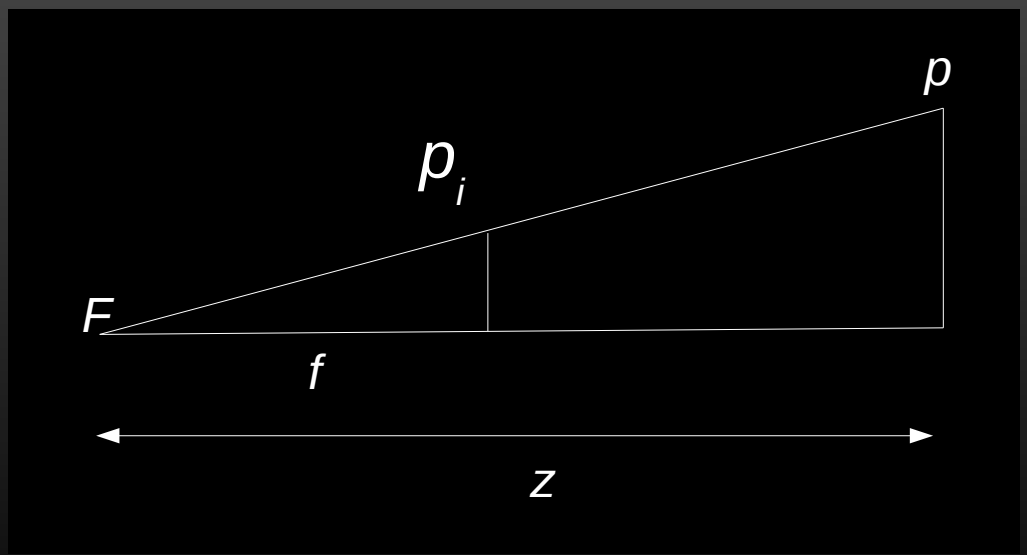


- Avantages
  - Simple
- Inconvénients
  - Oblige à projeter l'ensemble des polygones
  - Problème de résolution lors de l'encodage du Z

Algorithmes 3D fondamentaux : Projection

- Une fois que l'on a éliminé les éléments hors du champ de la caméra, les éléments qui ne sont pas de face
  - On projette les sommets et on dessine (et rempli) le polygone en tenant compte de la profondeur

$$p_i = \frac{fp}{z} \tag{1}$$



Algorithmes 2D fondamentaux

Algorithmes 2D fondamentaux : Remplissage de polygones

Algorithmes 2D fondamentaux : Remplissage de polygones

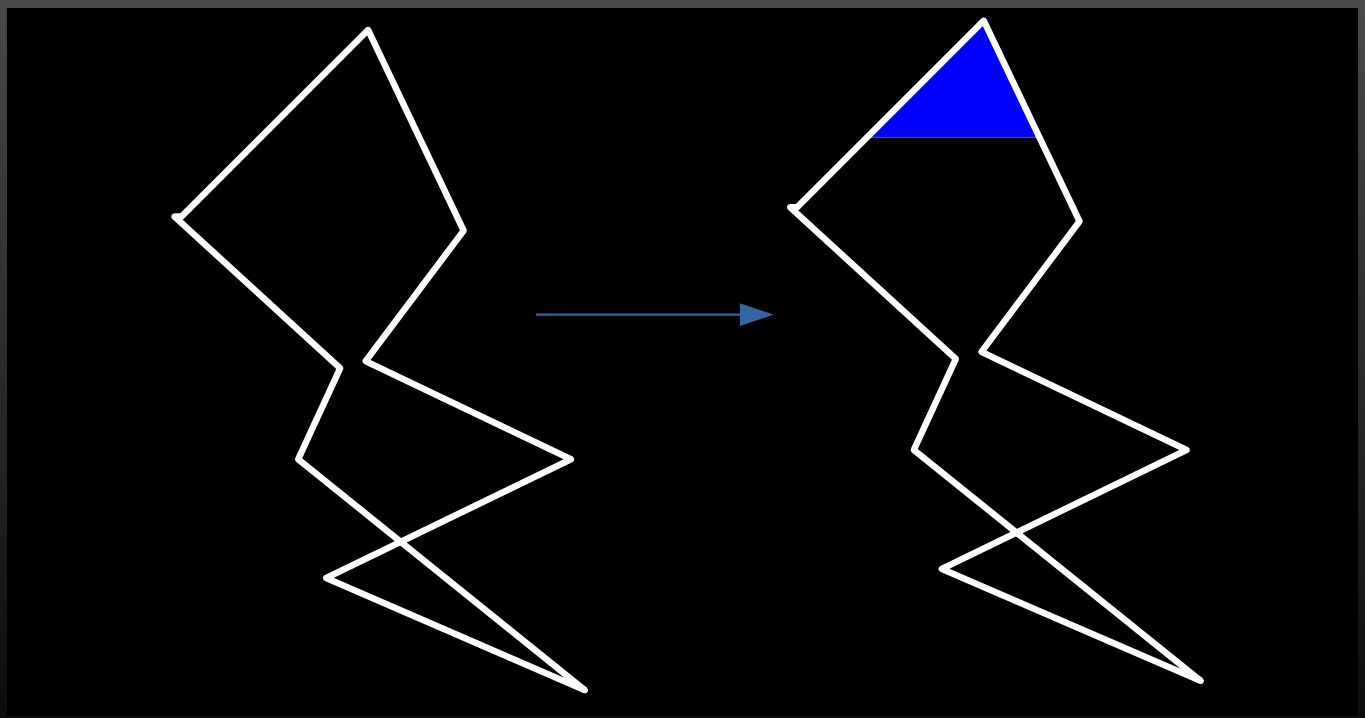
- Suivant la projection des polygones, il faut dessiner/remplir le polygone
  - déterminer si une partie n'est pas visible
  - déterminer la couleur et l'éclairage
    - éventuellement plaquer une texture
  - ...
- Les données sont la liste des sommets

Algorithmes 2D fondamentaux : Remplissage de polygones

- Types de polygone
  - Triangle ?
  - Convexe ?
  - quelconque... ?
- Données :
  - liste de sommets
- Approches :
  - Triangulation
  - Remplissage direct
  - Inondation

Algorithmes 2D fondamentaux : Remplissage de polygones

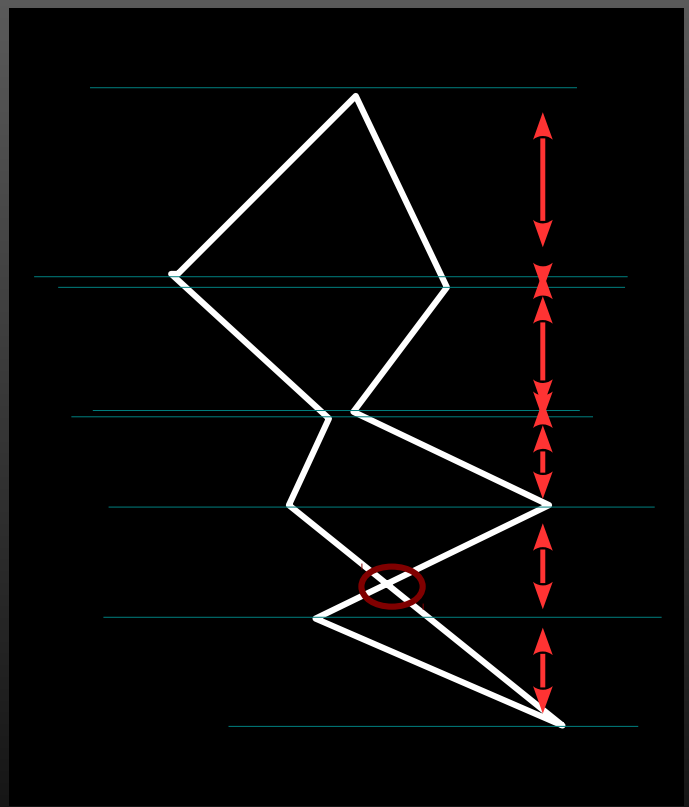
- Données :
  - liste de sommets
- Algorithme :
  - Parcourir toutes les arrêtes de haut en bas et remplir horizontalement



Algorithmes 2D fondamentaux : Remplissage de polygones

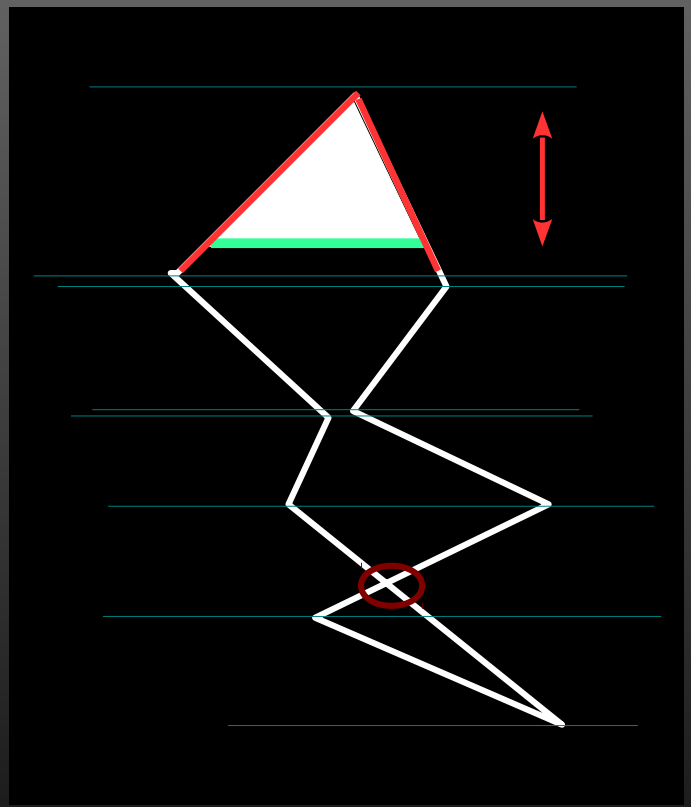
Algorithme :

- Trier les sommets pour définir des sections



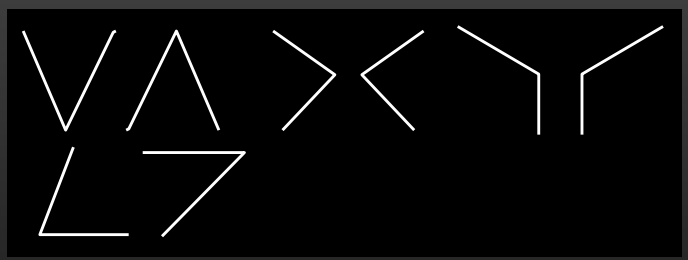
Algorithmes 2D fondamentaux : Remplissage de polygones

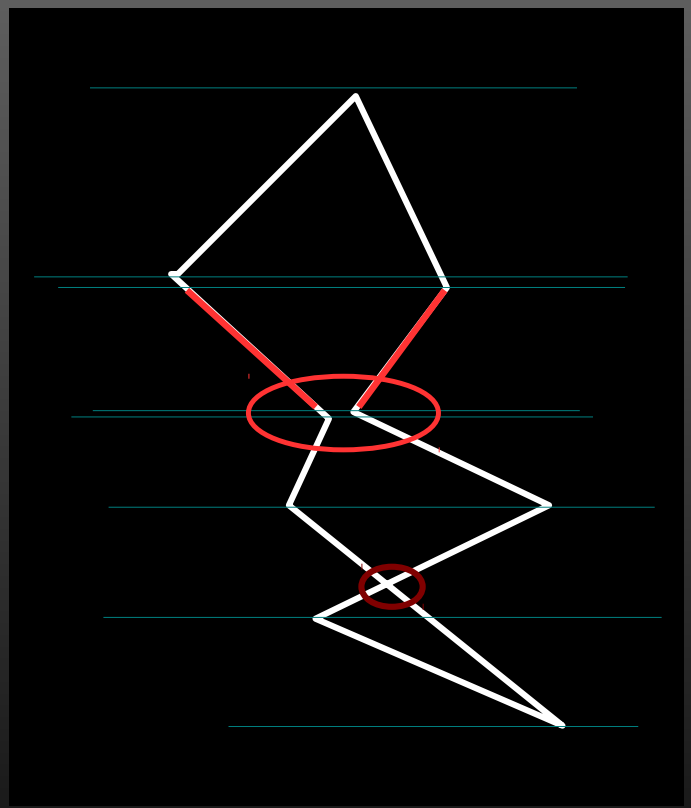
- Remplir les sections les unes après les autres
- Déterminer les arrêtes actives (dans la section)



Algorithmes 2D fondamentaux : Remplissage de polygones

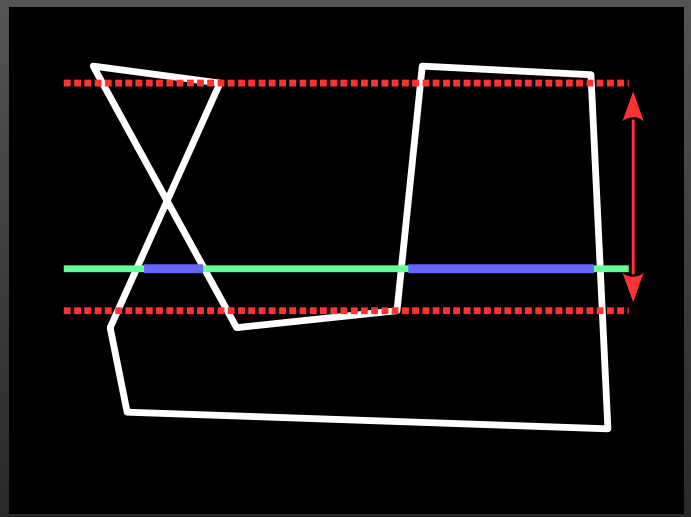
- A chaque transition, il faut remettre à jour la liste des arrêtes actives





Algorithmes 2D fondamentaux : Remplissage de polygones

- A chaque niveau il faut tracer des segments horizontaux



Algorithmes 2D fondamentaux : Remplissage de polygones

- Simplifications pour les polygones convexes (ou même dans le cas du triangle).

Algorithmes 2D : Tracé de segments

- Tracé rapide de segments
  - Affichage de segments
  - Suivi des arrêtes actives
- Comment faire ?

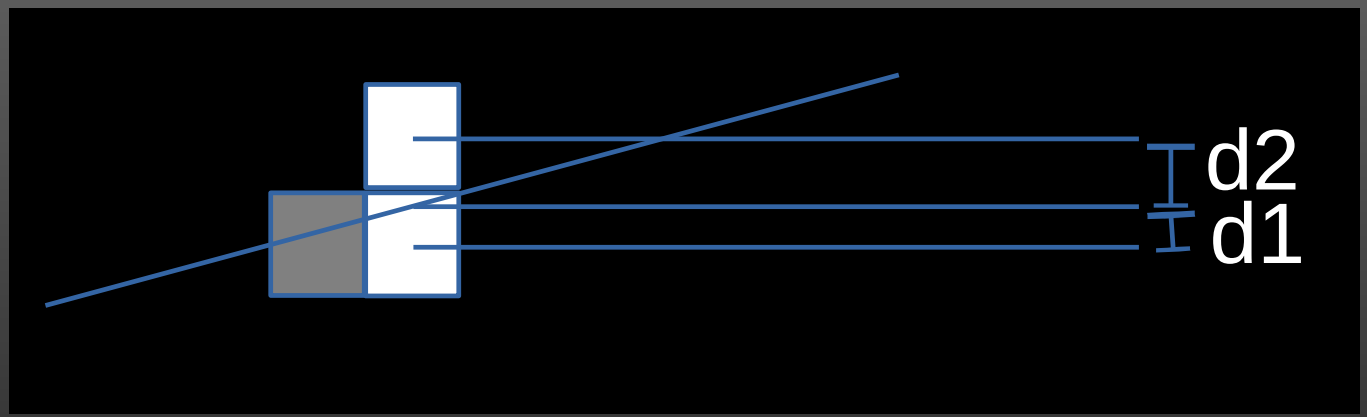
Algorithmes 2D : Tracé de segments

Algorithme naïf

- Repose sur l'utilisation des nombres à virgule flottante (un peu lent).

Algorithmes 2D : Tracé de segments

- Bresenham (65 ?)
- Uniquement avec des additions d'entiers



- Critère :
  - $y = mx + p$ avec $m = d_y/d_x$
  - $D = d1 - d2 = (m(x_p + 1) + p - y_p) - (y_p + 1 - m(x_p + 1) - p)$
  - $D = d1 - d2 = 2d_y(x_p + 1) - 2d_x y_p - d_x + 2d_x p$
  - $D < 0 \Rightarrow (x_{p+1}, y_p)$ inc : $2d_y$
  - $D > 0 \Rightarrow (x_{p+1}, y_{p+1})$ inc : $2d_y - 2d_x$
- Problème d'*aliasing*

Algorithmes 2D : Tracé de cercle

Algorithmes 2D : Tracé de cercle

Algorithme naïf

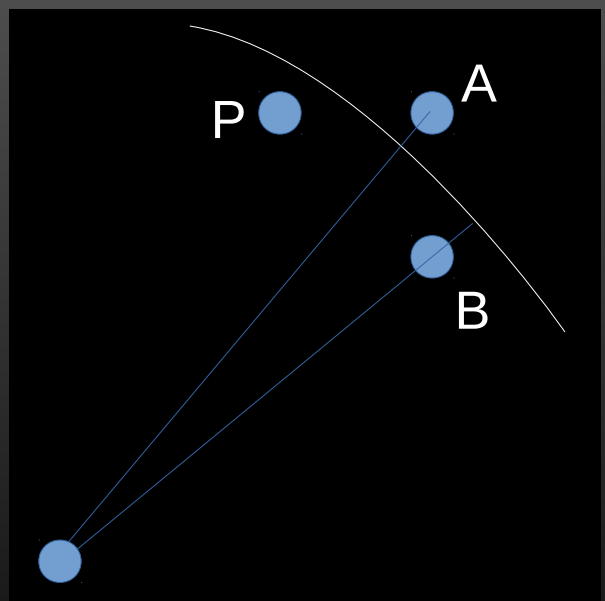
- Repose sur l'utilisation des nombres à virgule flottante (un peu lent).
- Utilisation des symétries
- Precalcul des fonctions trigos.

Algorithmes 2D : Tracé de cercle

Algorithme de Bresenham

- Même esprit que pour les segments

- $D(P) = x^2 + y^2 - r^2$
- $D(A) = (x+1)^2 + y^2 - r^2$
- $D(B) = (x+1)^2 + (y-1)^2 - r^2$
- $S = D(A) + D(B)$
- $S >= 0 \Rightarrow B$
- $S < 0 \Rightarrow A$
- Calcul de S incrémental



Algorithmes 2D

Clipping

Algorithmes 2D : *Clipping*

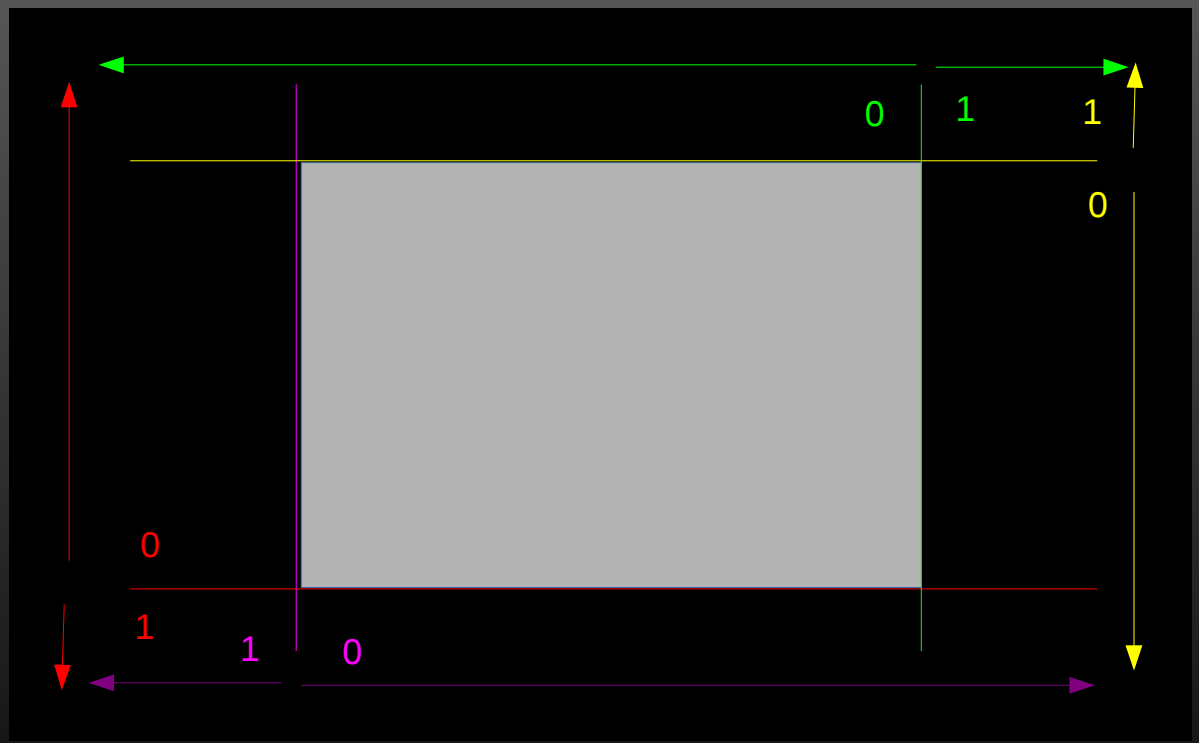
Même s'il est possible de fenêtrer les polygones dans l'espace, on peut aussi le faire dans le plan (après projection)

- Fenêtrage rectangulaire de segments :
  - Cohen-sutherland
- Fenêtrage d'un polygone à partir des segments :
  - Weiler – Atherton

Algorithmes 2D : *Clipping*

Fenêtrage rectangulaire de segments :

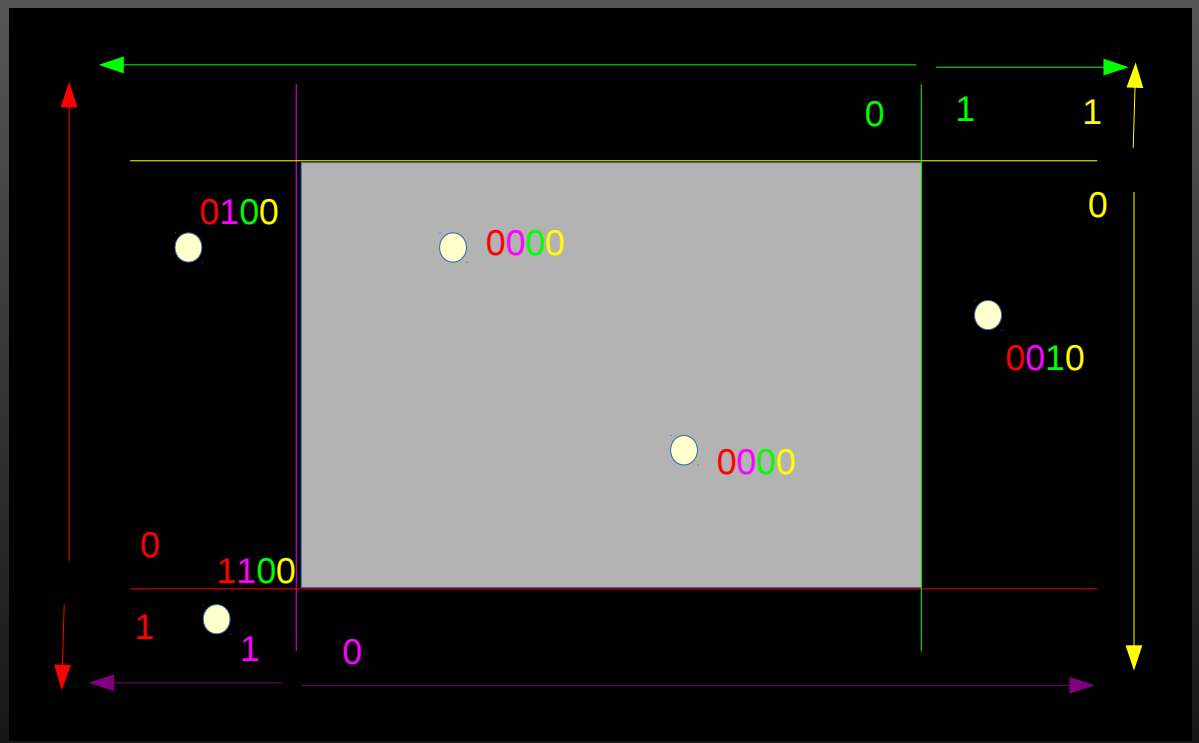
- Cohen-sutherland



Algorithmes 2D : *Clipping*

Fenêtrage rectangulaire de segments :

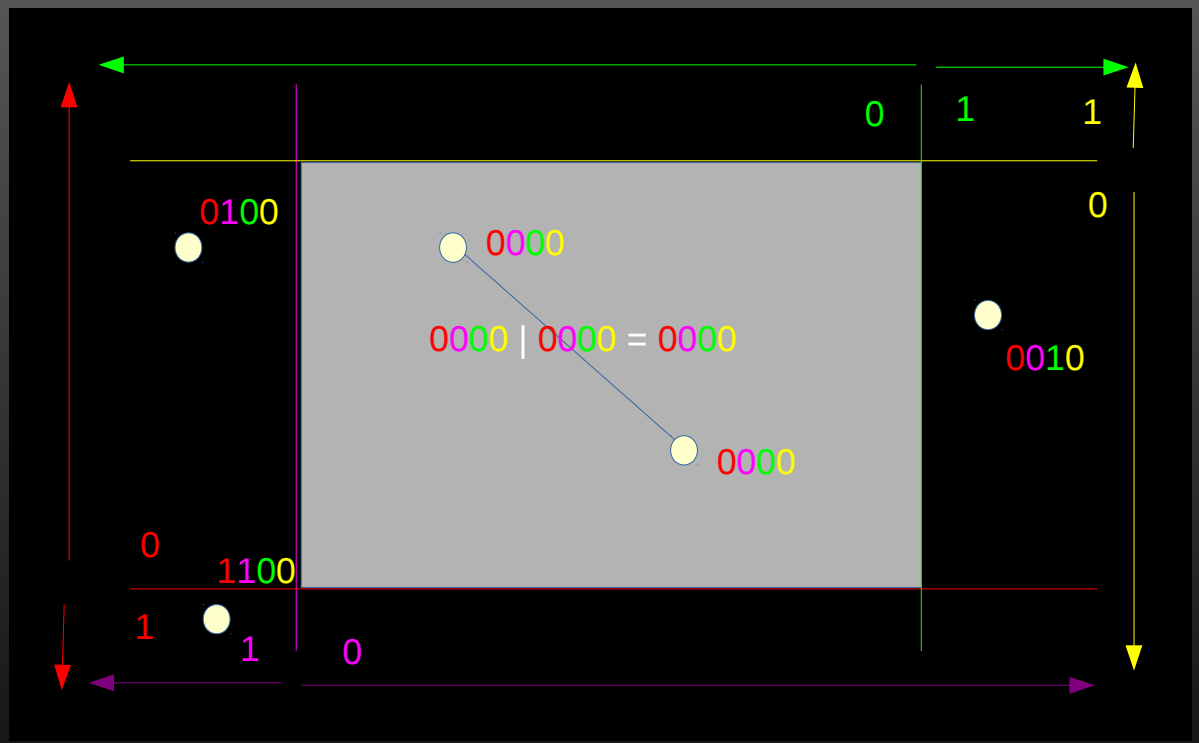
- Cohen-sutherland



Algorithmes 2D : *Clipping*

Fenêtrage rectangulaire de segments :

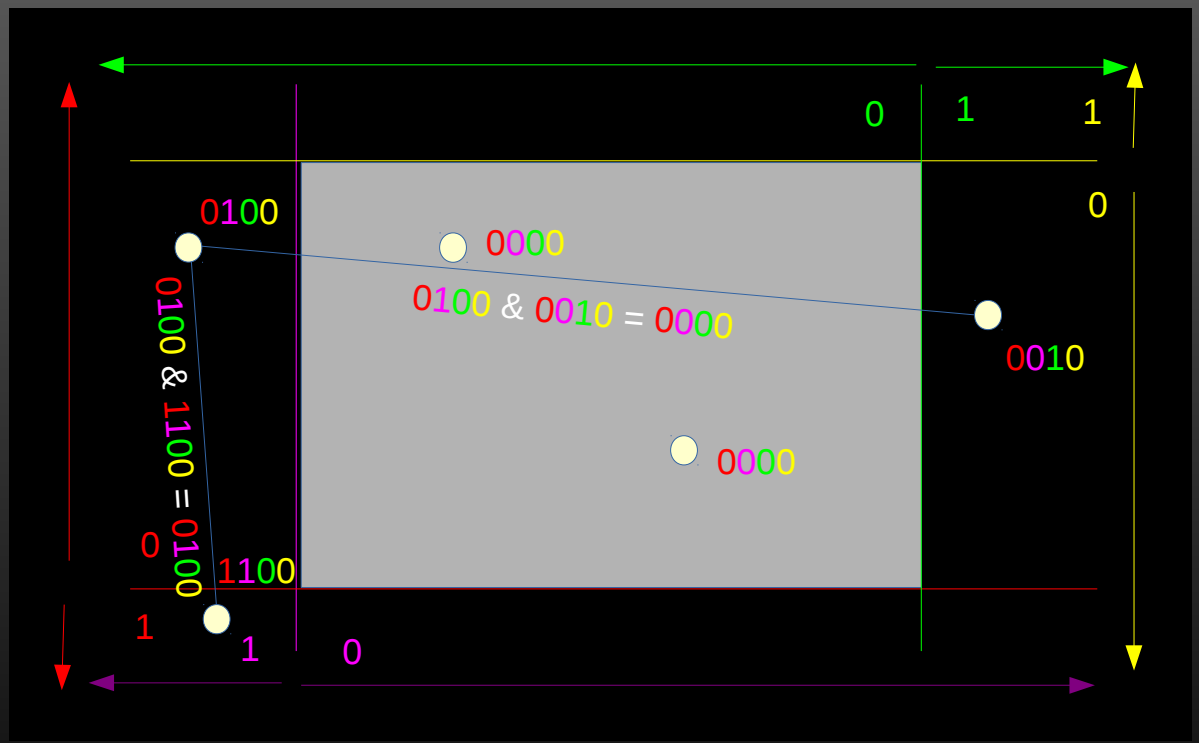
- Cohen-sutherland



Algorithmes 2D : *Clipping*

Fenêtrage rectangulaire de segments :

- Cohen-sutherland

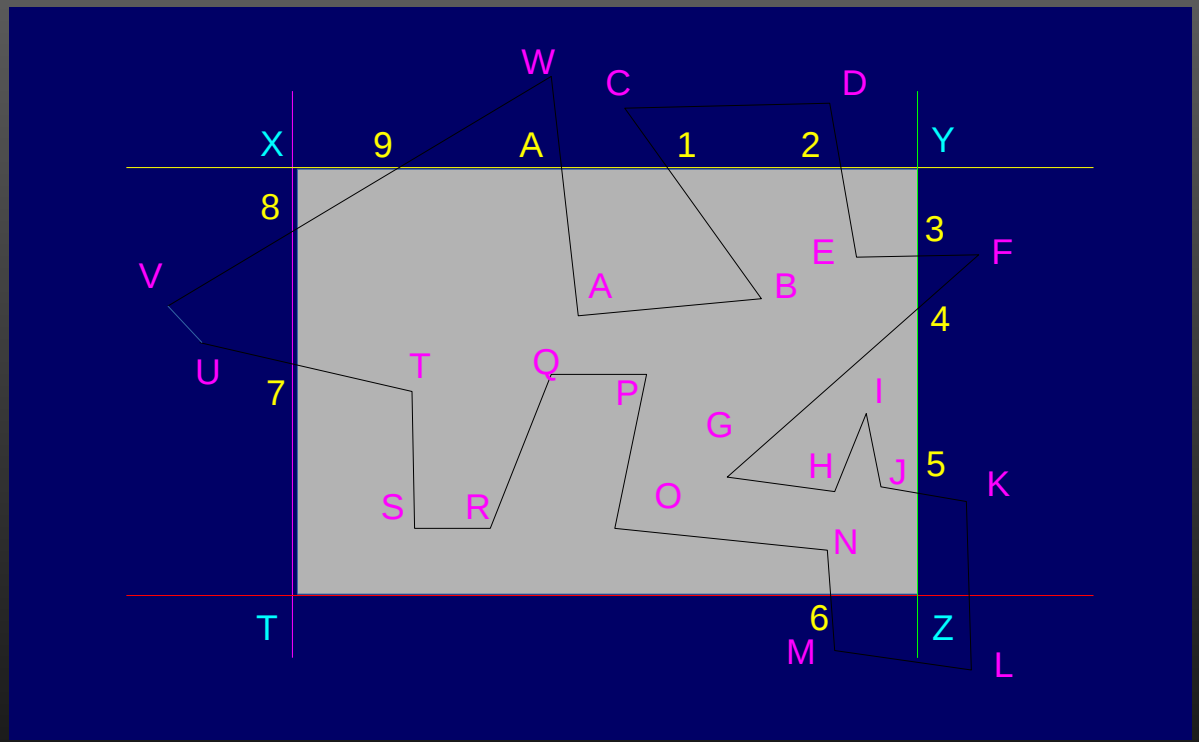


Algorithmes 2D : *Clipping*

Clipping

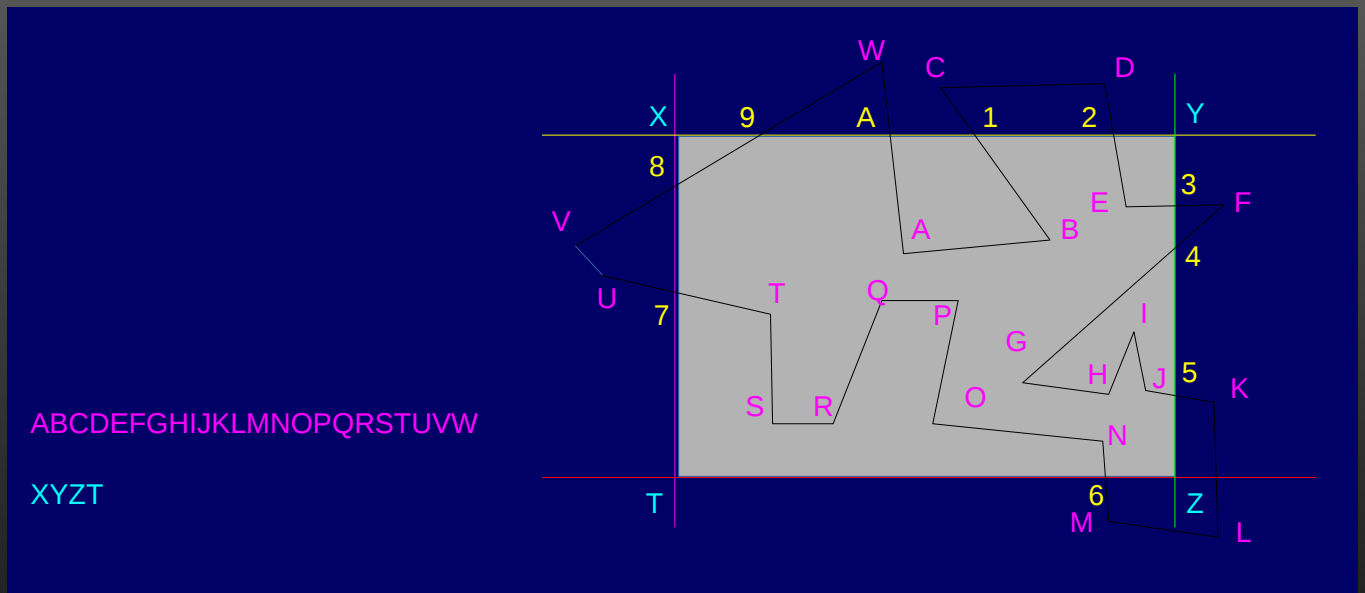
Algorithmes 2D : *Clipping*

Weiler-Atherton



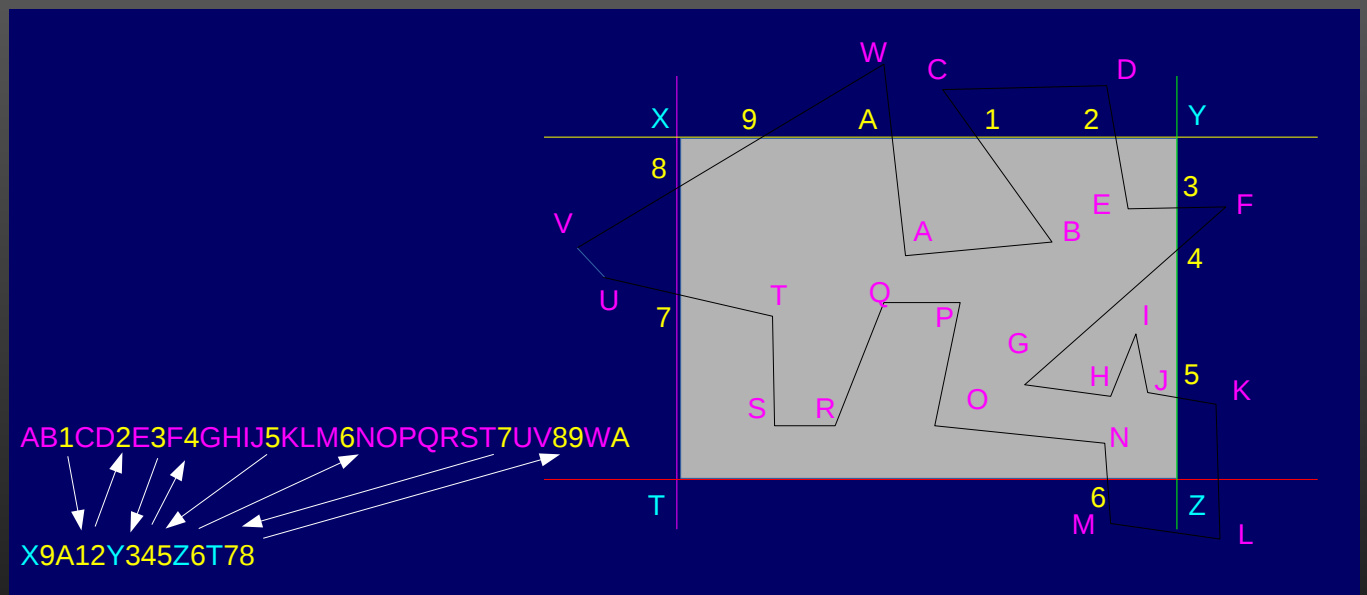
Algorithmes 2D : *Clipping*

Weiler-Atherton



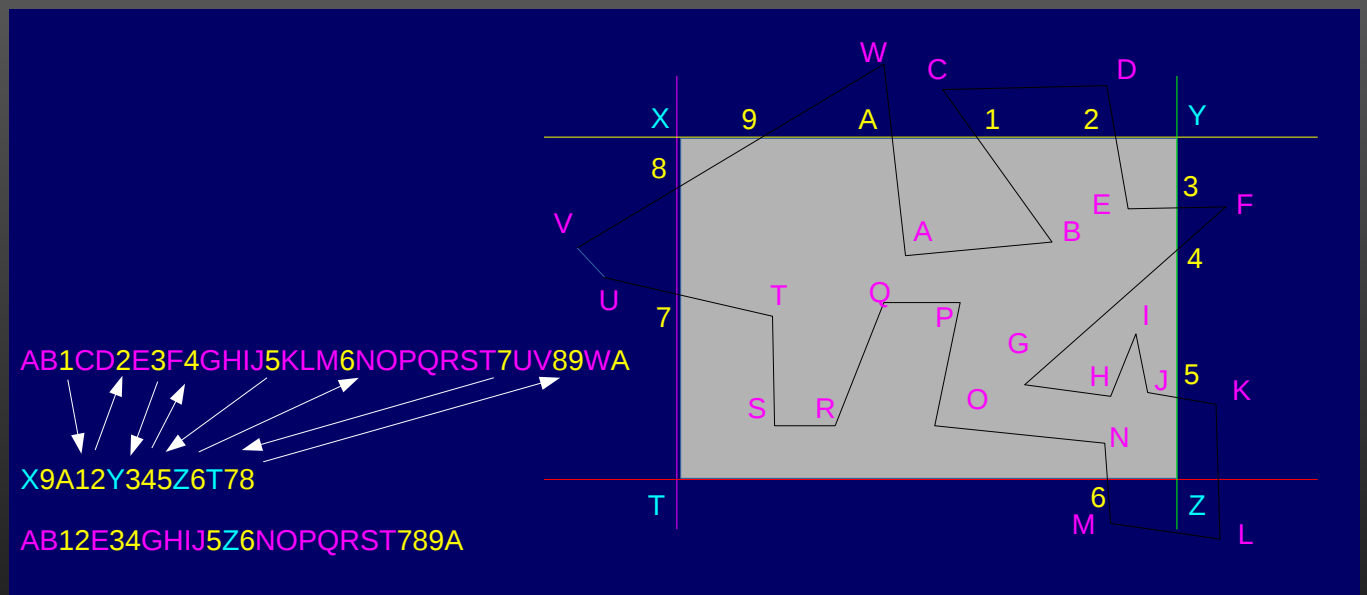
Algorithmes 2D : *Clipping*

Weiler-Atherton



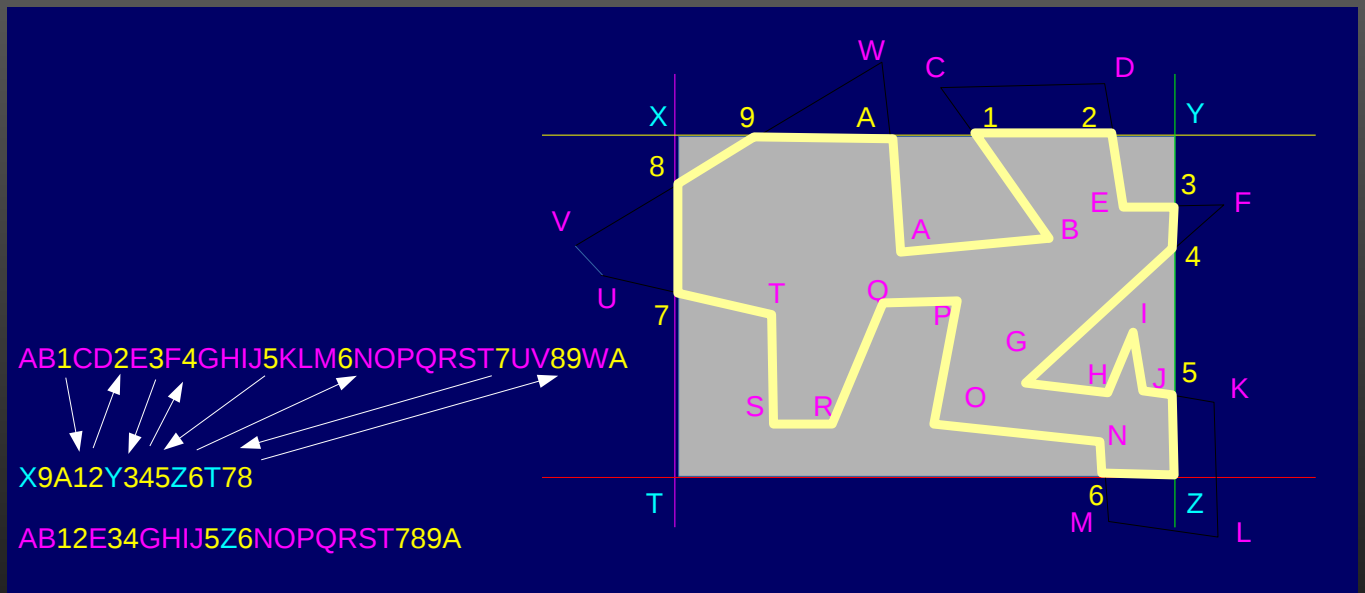
Algorithmes 2D : *Clipping*

Weiler-Atherton



Algorithmes 2D : *Clipping*

Weiler-Atherton



Algorithmes 2D : *Clipping*

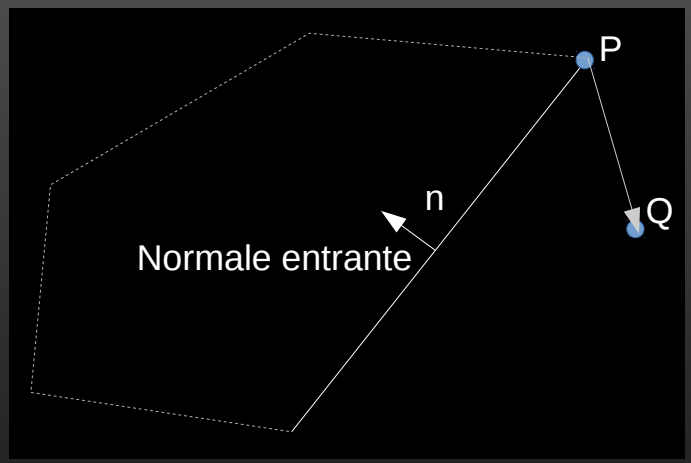
Fenêtrage entre deux polygones

- Cyrus-Beck (Pour deux polygones convexes)

Algorithmes 2D : Cyrus-Beck

Connaître la position d'un point Q par rapport à un coté de la fenêtre ?

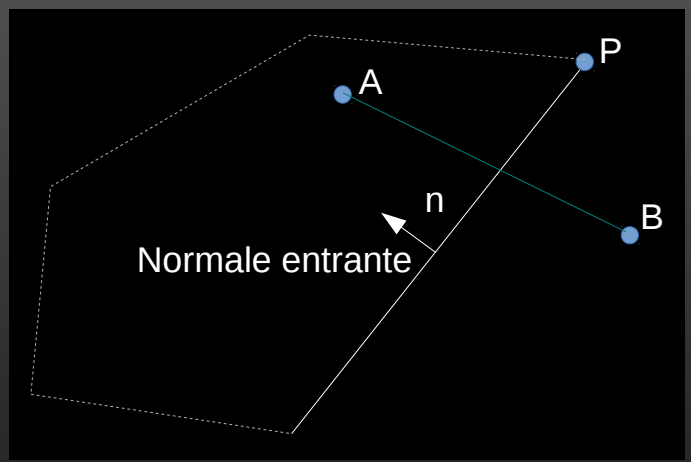
$$I(Q) = (Q - P).n \begin{cases} = 0 \\ < 0 \\ > 0 \end{cases} \quad (2)$$



Algorithmes 2D : Cyrus-Beck

Fenêtrage d'un segment ?

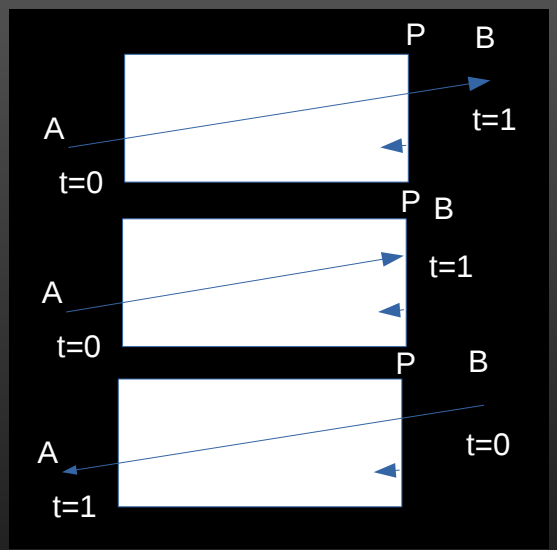
- $L(t) = A + (B - A)t$
- $I(Q) = (Q - P).n$
- $I(L(t)) = (L(t) - P).n$
- Intersection :
  - $(L(t) - P).n = 0 \Leftrightarrow t = (A - P).n/(A - B.n)$



Algorithmes 2D : Cyrus-Beck

Fenêtrage d'un segment ?

- $D = (B - A)$
  - Cas 1 : $D.N < 0 => t = t_{sup}$
  - Cas 2 : $D.N = 0$
  - Cas 3 : $D.N > 0 => t = t_{inf}$
- On recommence pour tout les segments de la fenetre
  - puis si $t_{inf} > t_{sup}$ segment non visible
  - sinon segment compris entre $[t_{inf}..t_{sup}]$



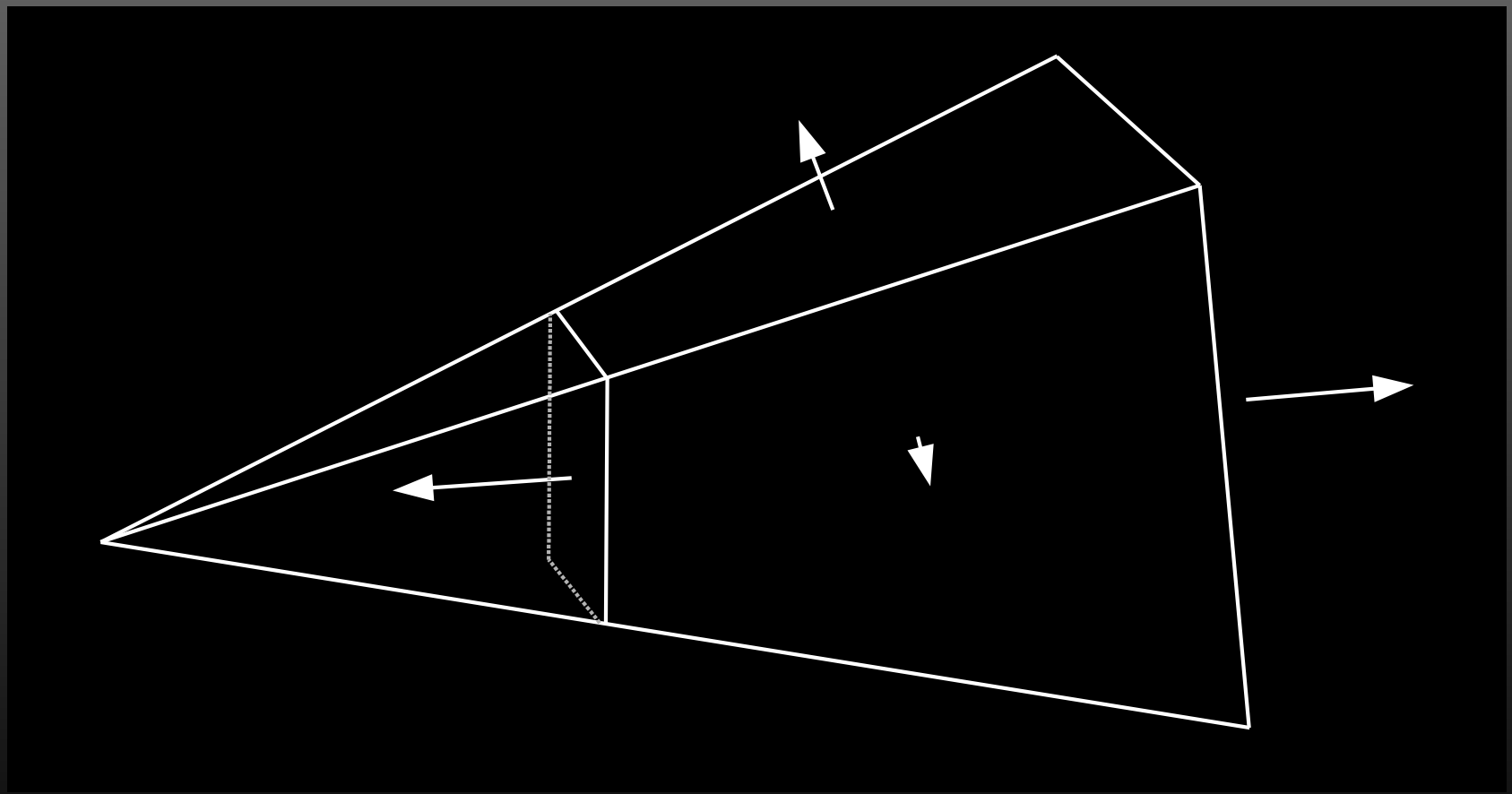
Algorithmes 2D

Polygônes non convexes

- Découpage en polygones convexes
- Triangulation de Delaunay

Algorithmes 3D : Retour sur le *clippling*

Application de Cohen-Sutherland sur la pyramide 3D



Algorithmes 2D : Détermination de la couleur des pixels durant le remplissage

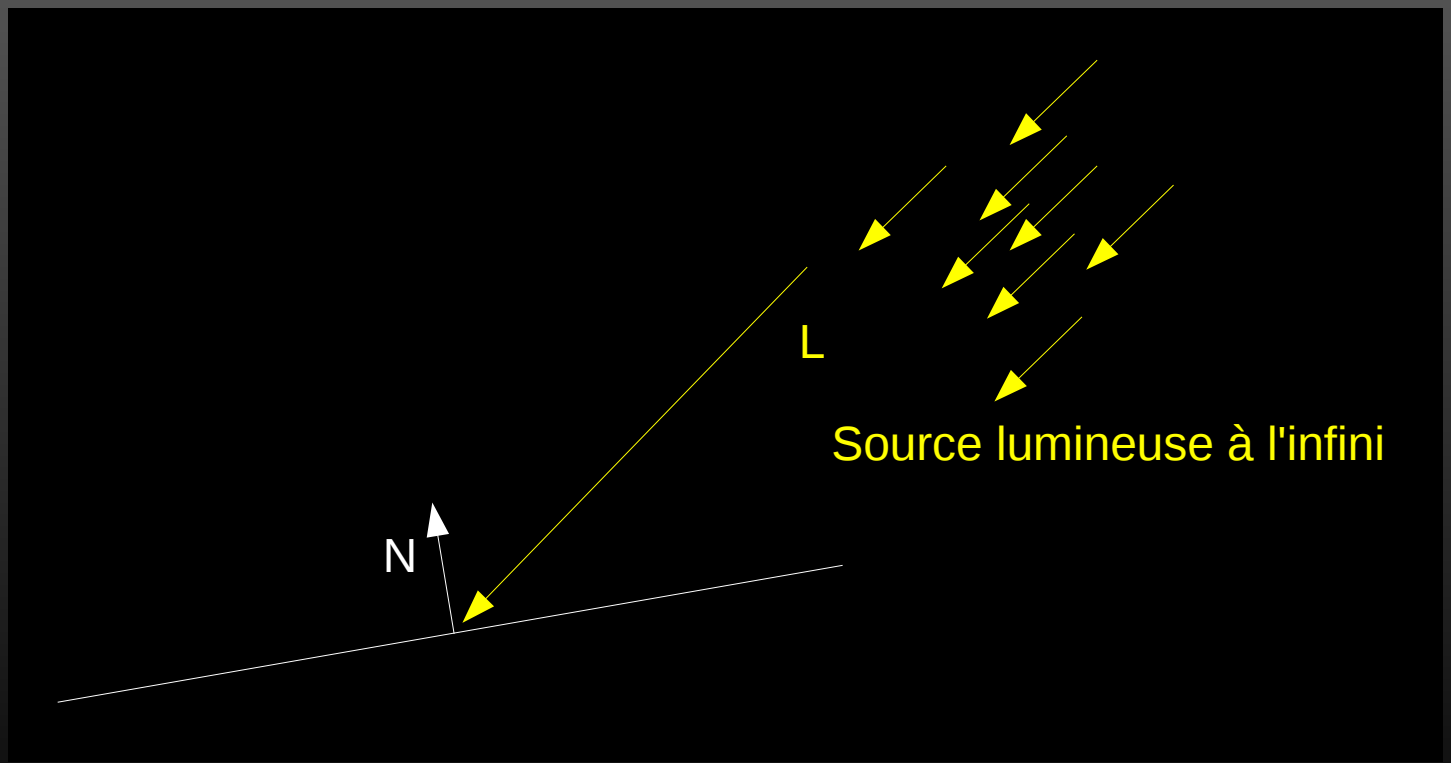
Remplissage en fonction de :

- La couleur de la face ?
- L'éclairage ?

Algorithmes 2D : Éclairage

Détermination de la couleur

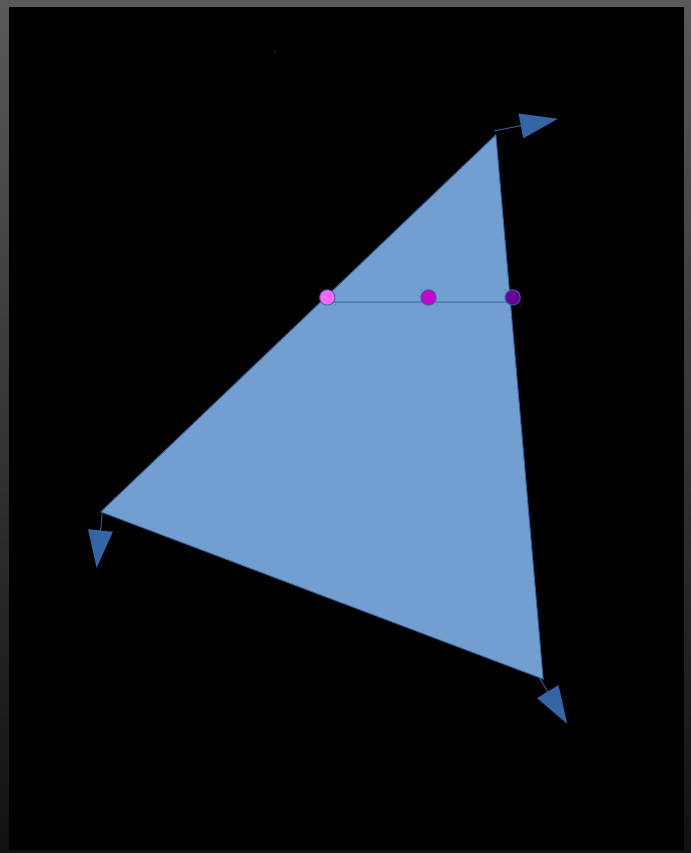
- Modèle de Lambert
  - $Id = k * N.L/||N||||L||$



Algorithmes 2D : Éclairage

Détermination de la couleur

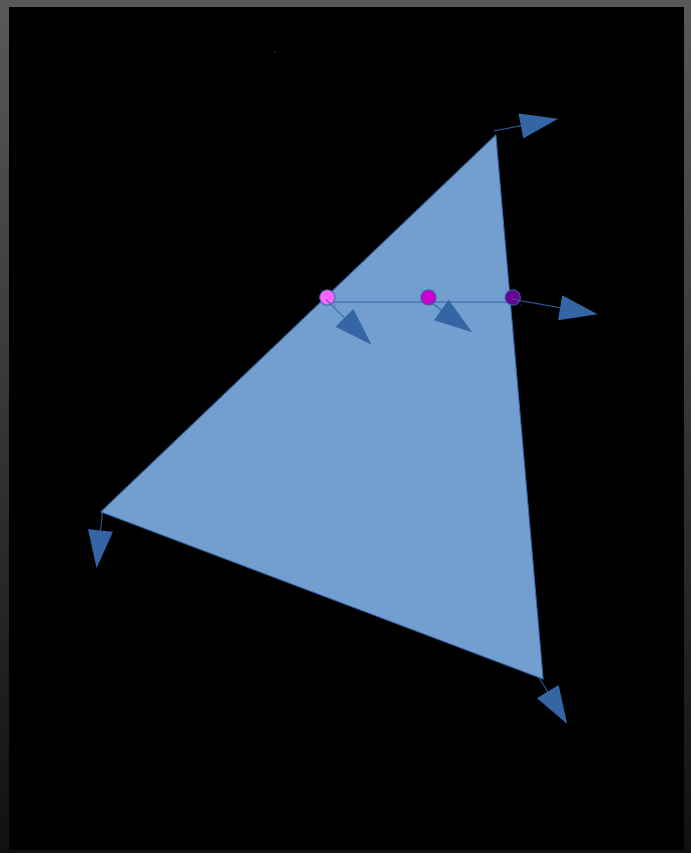
- Modèle de Gouraud



Algorithmes 2D : Éclairage

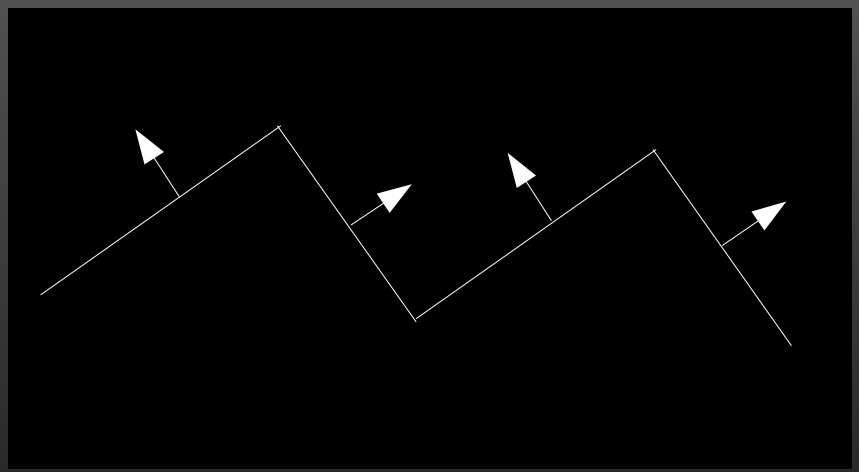
Détermination de la couleur

- Modèle de Phong



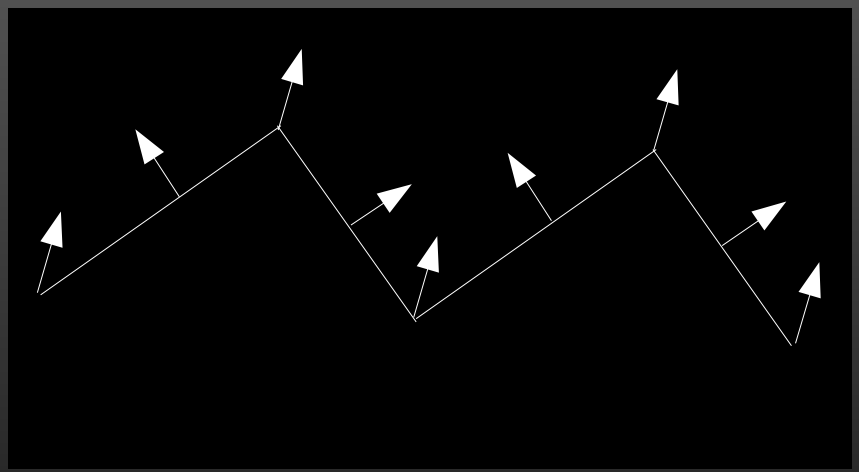
Algorithmes 2D : Éclairage

Attention au calcul des normales



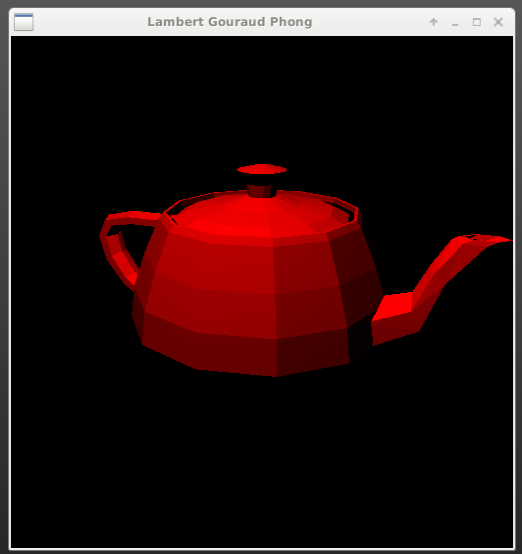
Algorithmes 2D : Éclairage

Attention au calcul des normales

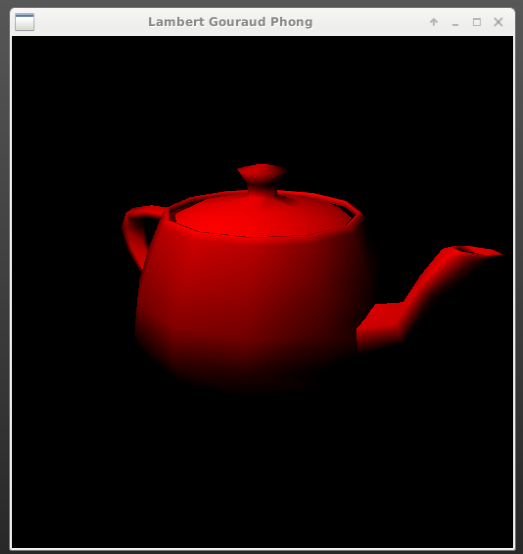


Algorithmes 2D : Éclairage

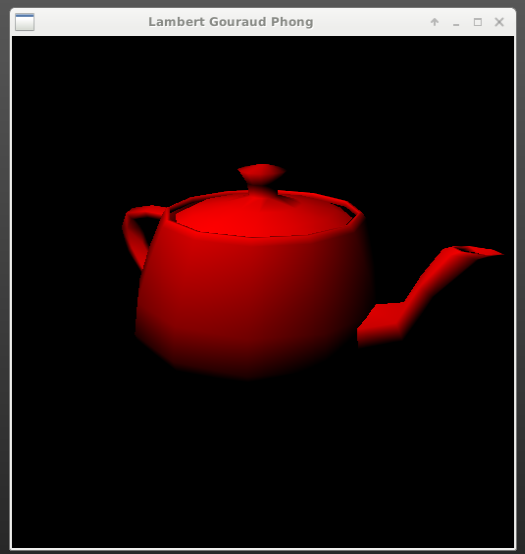
Résultats



Lambert



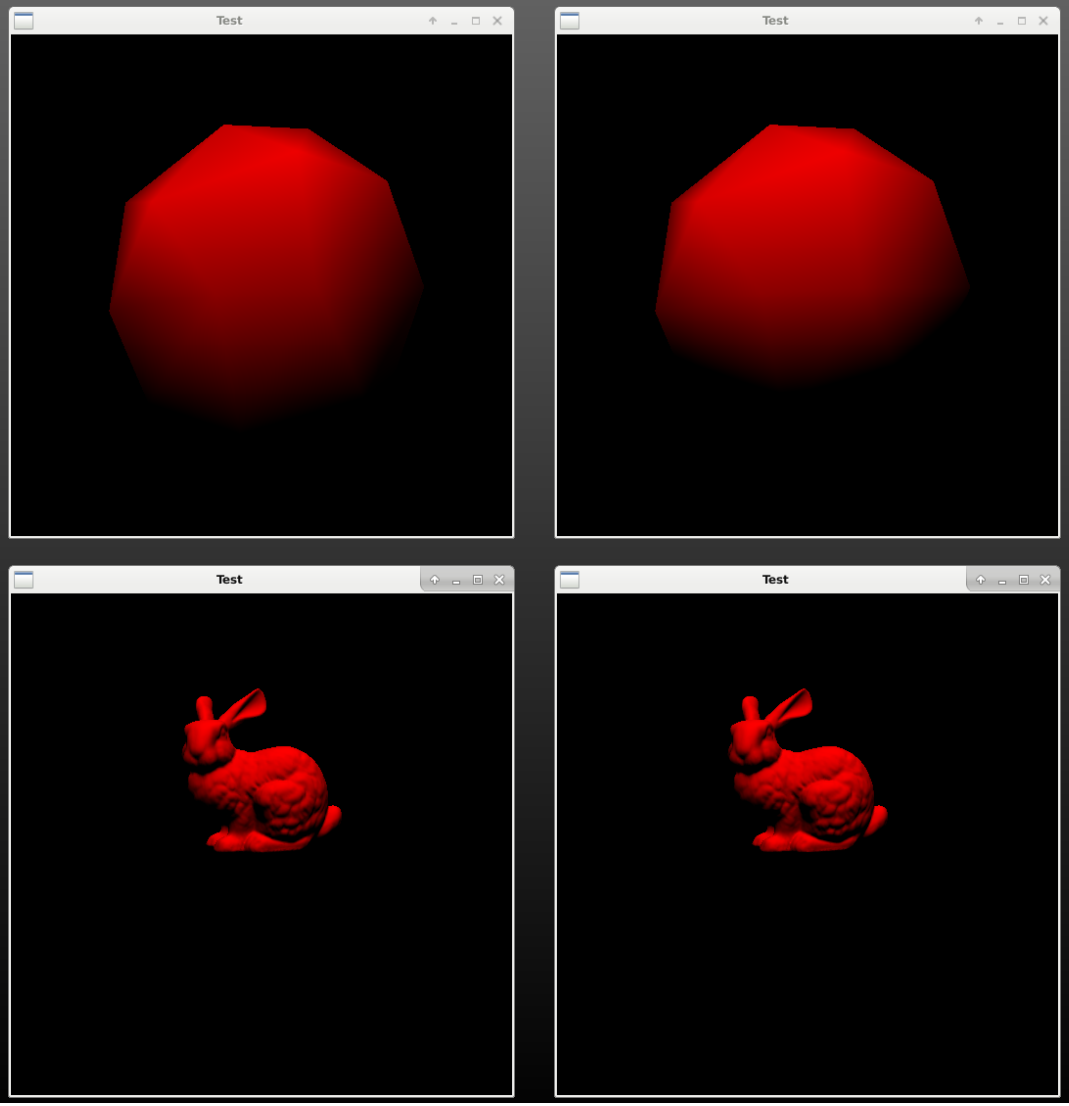
Gouraud



Phong

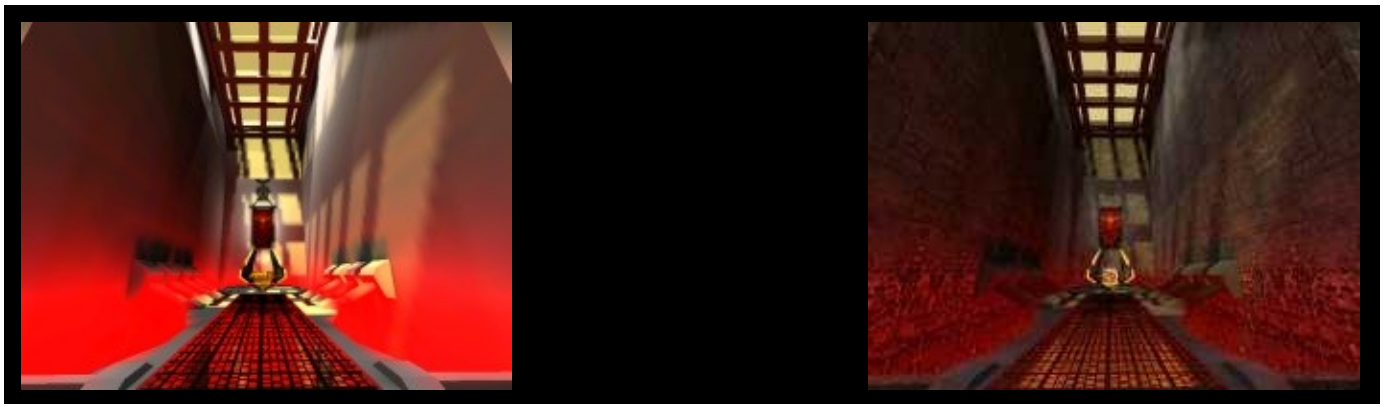
Algorithmes 2D : Éclairage

- La différence entre Phong et Gouraud se creuse lorsque le modèle est pauvre.



Algorithmes 2D : *lightmap*

Éclairages plus évolués



source : http ://www.photo-sport.ch/diverts/quark/jk2/shadertut.htm

Conclusion

- beaucoup d'algorithmes
- implémentés dans les moteurs
  - A connaitre pour inter-argir avec ces moteurs
- de nouvelles technologies emergent (RTX...)