Infographie : Fiche

Sommaire

---

1. Modélisation géométrique

1.1. Introduction

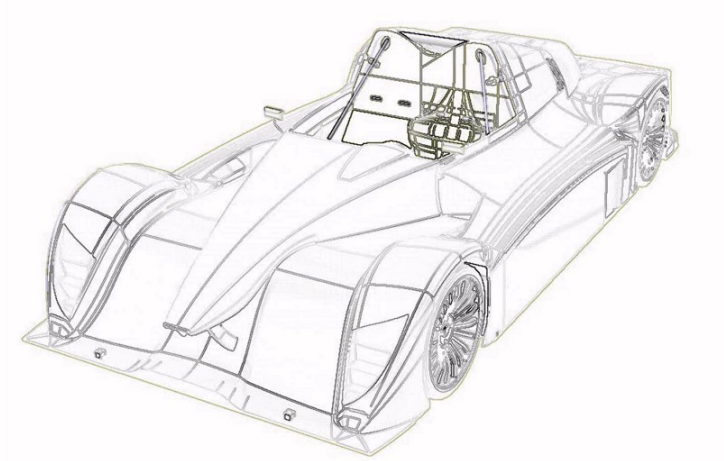
> Le but est de fabriquer des objets
>
> **Modèle :** simplification du monde réel.

1.2. Modèles explicites

> Pas de calcul intermédiaire avant l'exploitation

- ***B-Rep*** *(Boundary REPresentations)*
    - Modèle à frontières / à enveloppe
    - Vectoriel : (2D) courbe - (3D) surface
- ***Voxel*** *(VOlume ELement map)*
    - Modèle à évolution spatiale
    - Map : (2D) pixel - (3D) voxel
- Contraintes de choix et d'utilisation :
    - Précision de la représentation
    - Domaine de représentation
    - Efficacité mémoire / temps de calcul

1.2.1. Modèle B-Rep



1.2.1.1. Principe

> Un objet = Une liste (tableau, etc) de **carreaux de surface** (= *patch*).

- **patch** = restriction paramétrique d'une modélisation mathématique de surface
  - Modélisation mathématique de surface explicite : $z = f(x, y)$
  - Modélisation mathématique de surface implicite : $f(x, y, z) = 0$
  - Modélisation mathématique de surface paramétrique (ex : Bézier / Hermite → NURBS) : $\begin{cases} P_x = Q_x(u, v) \\ P_y = Q_y(u, v) \\ P_z = Q_z(u, v) \end{cases}$
- Stockage : modèle point-face
- Visualisation : modèle point-arrête-face (table de points, table d'arrêtes, table de faces)

1.2.1.2. Modèle windged-edge

- **Point :** $x \;\; y \;\; z$
- **Arrêtes :**
  - $p_{départ} \quad p_{fin}$
  - $f_{gauche} \quad f_{droite}$
  - $a_{gauche\ procédurale} \quad a_{gauche\ suivante}$
  - $a_{droite\ procédurale} \quad a_{droite\ suivante}$
- **Face :** Un numéro d'arrête

1.2.1.3. Modèle half-edge

- **Point :** $x \;\; y \;\; z$
- **Arrêtes :**
  - $p_{départ} \quad p_{fin}$
  - f
  - $a_{jumelle}$
  - $a_{précédente}$
  - $a_{suivante}$
- **Face :** Un numéro d'arrête

1.2.1.4. Subdivision d'un triangle

Procédure classique : diviser un traingle en 4 triangles.

> **Attention :** On coupe les arrêtes, il faut donc subdiviser les faces voisines ou créer des arrêtes fictives

1.2.2. Les énumérations spatiales - Voxel

1.2.2.1. Principe

> Enumérations spatiales $\rightarrow$ tableau 3D. (Version 3D des Pixmaps)

- Souvent issus de processus de mesure (scan)
    - Information de densité :
        - 0 si vide
        - 1 si plein
        - Possibilité de réaliser des opérations de filtrage *("Telle gamme de densité correrspond aux os, telle gamme de densité correspond aux muscles")*.
    - Structures de données : 1 tableau 3D $[i][j][k]+$ taille et quantité des voxels.

> **Domaine de représentation le plus complet mais par conséquent le plus lourd en mémoire**.

- Permet de représenter des objets issus de **scan** ou des objets "**naturels**" (ex. nuages).
- Utilisé en impression 3D.

1.2.2.2. Formats de fichier

Exemple : Un objet de $1\ m^3$ avec des voxels de $1\ mm^3 \Rightarrow 10^9$ voxels. C'est énorme.

- **DICOM (Digital Imaging and COmmunications in Medicine)** : Utilisé par les médecins pour archiver les scans.
    - Pile d'images 2D *(slices)*

- Formats pour artistes → spécifiques à chaque logiciel (ex. Magivoxel).

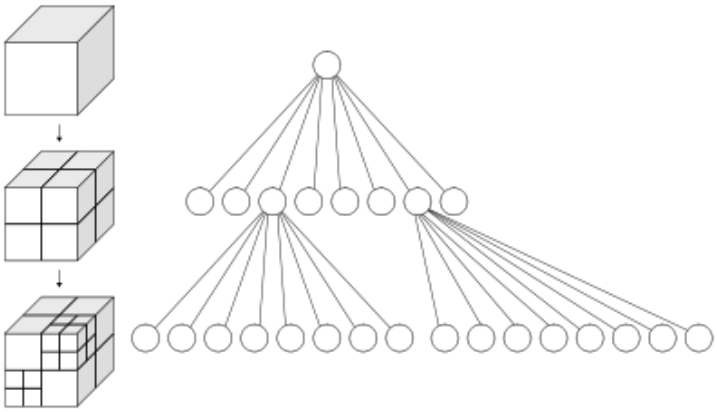
1.2.2.3. Jeux

- **Minecraft** (MCEdit)

1.3. Modèles implicites

> Calcul intermédiaire avant l'exploitation

1.3.1. Octree



1.3.1.1. Principe

> **Octree** = arbre de subdivision de l'espace (d'ordre **8**).

- Version 2D : **Quadtree** (d'ordre **4**).
- **Noeud** (cube) → plein / vide / complexe. Si complexe, on subdivise.

> Format compact des énumérations spatiales.

- Utilise la **cohérence spatiale** des objets.

1.3.2. Instanciation de primitives

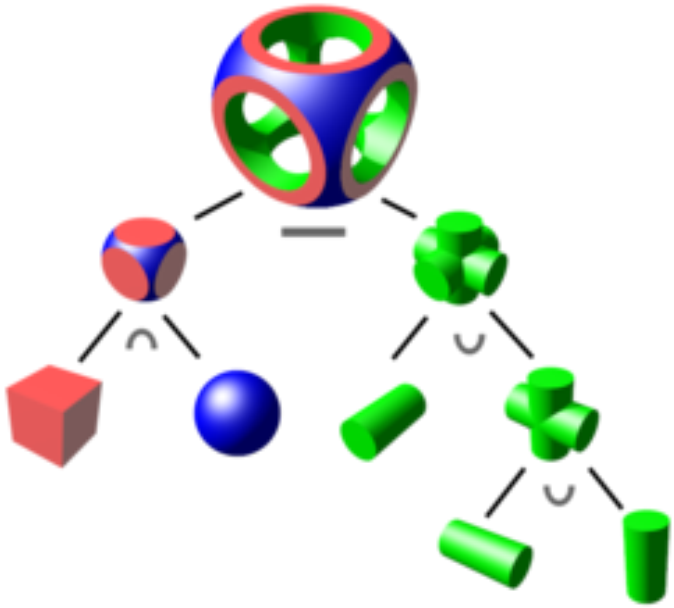
1.3.2.1. Principe

> **Instanciation** = répétition d'un objet.
>
> **Primitive** = Catalogue de composants (ex. cube, sphère, cylindre, etc).

Modèle = liste de primitives paramétrées.

- **Paramètres :** transformations géométriques et autres paramètres (ex. nombre de trous dans un cube).
- **Domaine de représentation** : dépend du catalogue.
- **Précision** : dépend du catalogue.
- **Encombrement** : réduit.

1.3.3. Constructive Solid Geometry (CSG)

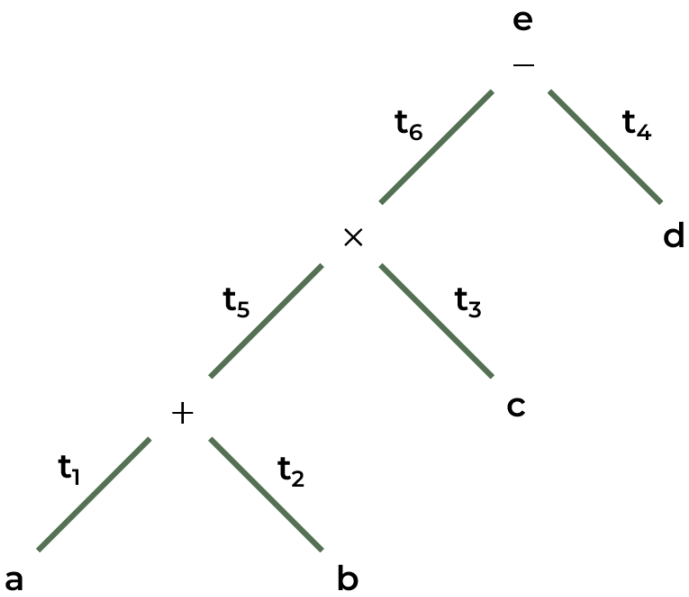


1.3.3.1. Principe

Modèle : Expression de calcul d'arbre d'ordre 2.

- **Racine** : Objet complet.
- **Feuilles** : primitives (plan/boîte, sphère, cylindre, cone).
- **Noeud** : opération (union, intersection, différence, etc).

1.3.3.2. Equation



$$e = t_6(t_5(t_1(a) + t_2(b)) \times t_3(c)) - t_4(d)$$

Avec t_i = transformation géométrique.

1.3.4. Surfaces isoparamétriques

1.3.4.1. Principe

> **Surfaces isoparamétriques** = surfaces paramétriques.

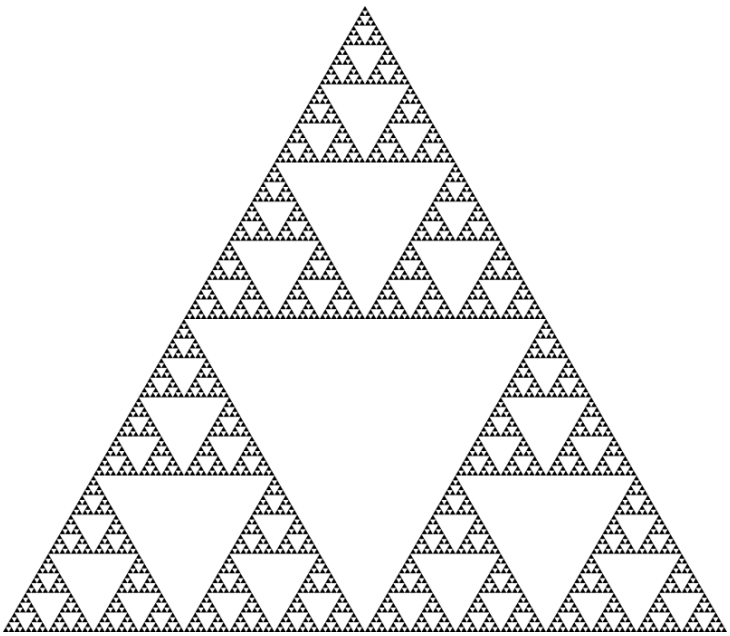
- Surface de valeur : $f(x, y, z) =$ seuil
    - Ex. par rapport à un point : $f(x, y, z) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} =$ seuil

1.3.4.2. Métaballes.

- **Métaballe** *(blob)* : permet de créer des formes organiques ou de représenter des fluides.

> **Fonctions génératrices** (distance) : $\sum_n f_n(x, y, z) <$ seuil
>
> Métablle + Valeur aléatoire + Seuil.

1.3.5. Fractales

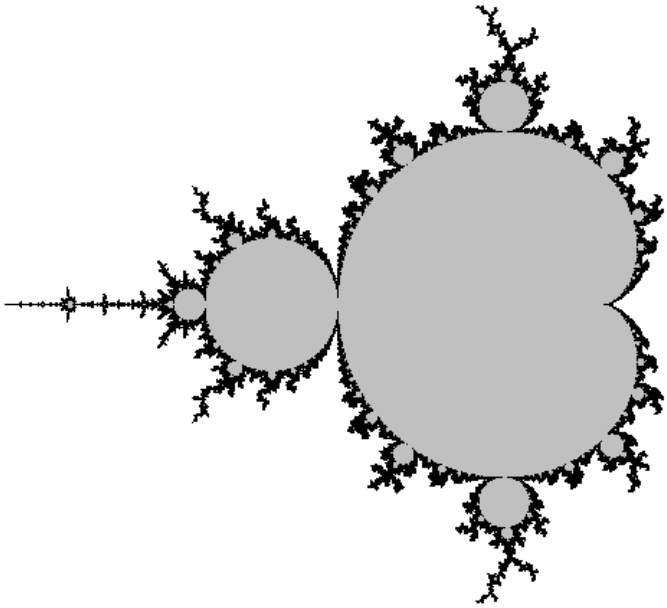


1.3.5.1. Principe

> **Fractale** = objet dont la structure est répétitive à différentes échelles.
>
> **Modèle** = objet initial + procédure de transformation.

- Volume nul, surface infinie.
- Construction itérative/récursive.

1.3.5.2. Cas particulier : ensemble de Mandelbrot



- Etude de la suite complexe $\begin{cases} z_0 = 0 \\ z_{n+1} = z_n^2 + c \end{cases}$ avec c l'objet initial.
- Divergent si $\exists i \in \mathbb{N}$ tel que $|z_i| \geq 2$.

1.3.5.3. Modélisation : Grammaire

1. alphabet
2. règles de transformation

Exemple :

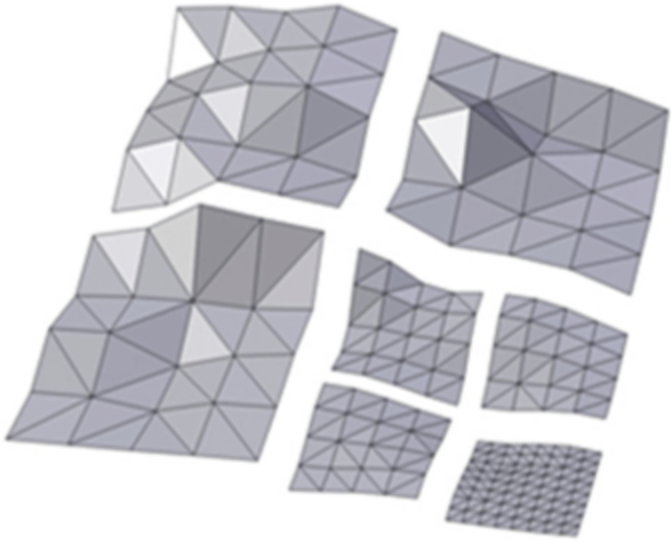
1. alphabet :

- $\Sigma = \{A, B, [\ ,\ ]\}$

2. règles de transformation :
   - $A \rightarrow AB$
   - $B \rightarrow A[AB]A[AB]$
   - $[\rightarrow [BB$
   - $]\ \rightarrow]AB$
3. exemple de procédé :
   1. $ABBA$
   2. $ABA[AB]A[AB]A[AB]A[AB]AB$
   3. $ABA[AB]A[AB]AB[BB...$
   4. ...

1.3.5.4. Terrains par subdivision de surface

> Subdivision de surface + hauteur aléatoire à chaque division.



- Utilisé pour crée des terrains de synthèse.
- Procédé que nous pouvons utiliser pour obtenir en 3D un **B-Rep** ou en 2D une image en niveau de gris.

1.4. Outils de modélisation

1.4.1. Opérateurs ensemblistes

> Opérations sur des volumes.

- $+$: union
- $\times$: intersection
- $-$: différence
  - $a - b = a \times -b$

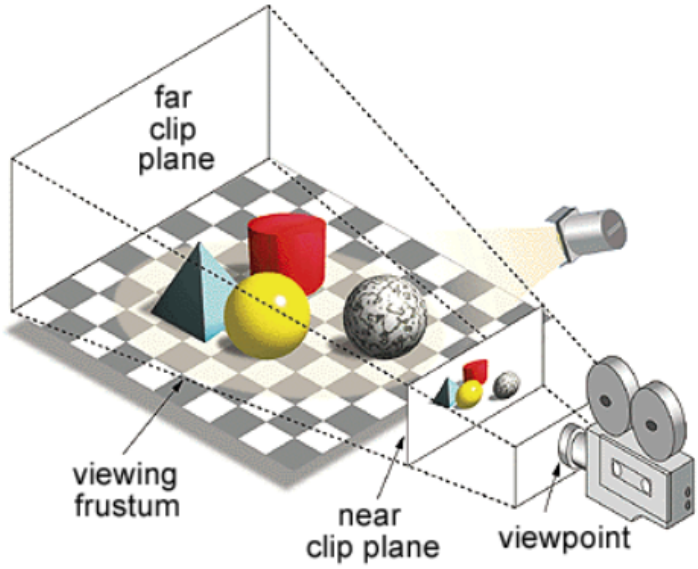
> Intérieur et extérieur d'un objet.

- **Voxel** : pas de problème.

- **B-Rep** : calculs d'intersection sur les enveloppes.

2. Algorithmes de visualisation

> **Principal problématique** : élimination des parties cachées.

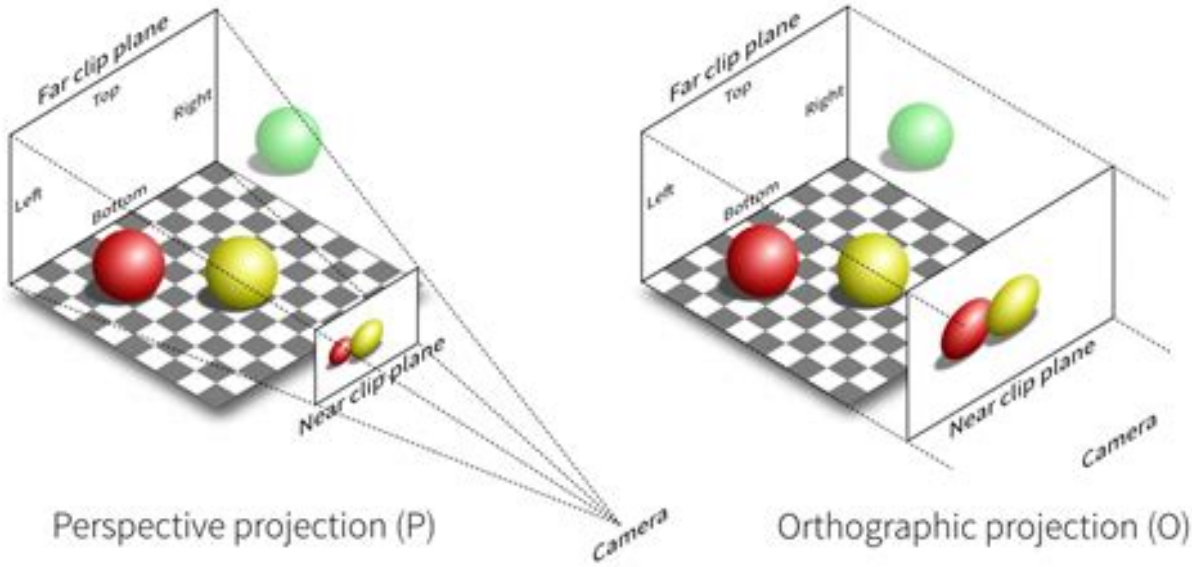


2.1. Principe

2.1.1. Etape 1 : Changement de repère et transformation de projection

> La première étape est de **changer de repère**.

- Repère scène → repère caméra.
- Puis transformation de projection.
  - **Perspective** :
  - **Orthographique** :



2.1.2. Etape 2 : Clipping

> La deuxième étape est le **clipping** (élimination des parties cachées).

- Haut, bas, gauche, droite → rectangle image + avant/arrière.

2.2. Algorithme naïf

- Visualisation (caméra, scène). n objets, kn points.

    1. Changement de repère et transformation de projection.

        > **Complexité** : $O(n)$

    2. Clipping.

        > **Complexité** : $O(n)$

    3. Comparaison entre objets (qui est devant, qui est derrière) = **Fenêtrage** :

        > **Complexité** : $O(n^2)$

        - **Ligne / Contour** : produit les segments visibles. **Pseudo-intersection** entre la ligne et le contour, on se réfère à l'axe $\vec{z}$ pour savoir qui est devant et qui est derrière.
        - **Contour / Contour** : produit les contours (surfaces) visibles. Calcul d'intériorité, une ligne $\rightarrow$ une liste de pseudo-intersections occultatntes.
        - Algorithme de fenêtrage :

```
début
  pour chaque objet i
    pour chaque objet (j != i)
      clipping(Oi,Oj))
      visibilité des extrémités
    fin pour
  fin pour

  pour chaque objet i
    afficher les parties visibles de Oi
  fin pour
fin
```

- **Algorithme d'Appel** *(1967)*. Très utile pour le dessin technique.

    > Permet de faire une visualisation en lignes cachées ou partiellement cachées.

    - **Visibilité quantitative**
        - $\rightarrow$ Pseudo-intersection entrante ou sortante
        - $\rightarrow$ Nombre de surfaces occultantes aux extrémités
    - Compteur d'occultation $\rightarrow$ style de ligne (traits plus ou moins visibles et/ou pointillés).

2.3. Optimisation des calculs

> **Moins d'objets**.

2.3.1. Etape 1 : Boites englobantes

> **Boite englobante** : boite la plus petite qui contient l'objet.

- Les boîtes de la scène sont disjointes $\rightarrow$ pas d'occulations $\Rightarrow$ pas de calcul. (comparaison des coordonnées min/max)

2.3.2. Etape 2 : Elimination des faces arrières *(Backface culling)*

> **Face arrière** : face qui n'est pas visible, ces faces ont un **vecteur normal** avec $z > 0$ (dans le repère caméra).

- On élimine simplement les faces avec un **vecteur normal** où $z > 0$.
- **Objet avec une courbe** : subdiviser la courbe en segments et éliminer les faces avec un **vecteur normal** où $z > 0$.

2.3.3. Etape 3 : Occlusion totale *(Occlusion culling)*

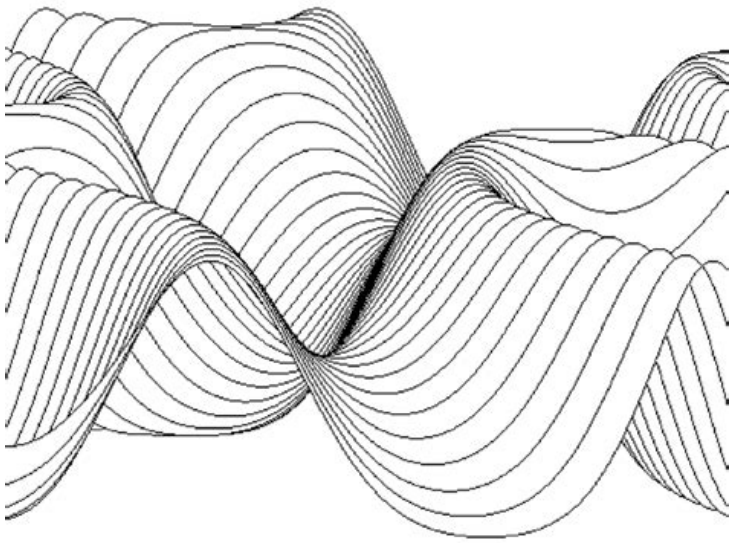
> **Occlusion totale** : si un objet est entièrement occulté par un autre objet, on peut l'éliminer.

- **Calcul d'intériorité** : Pour tous les sommets de l'objet, on regarde s'il est dans l'objet occultant.
- **Arêtes réelles / virtuelles** :
  - **Arête réelle** : arête que existe dans l'objet.
  - **Arête virtuelle** : arête qui n'existe pas dans l'objet, mais qui est créée pour pour la construction. Une sphère n'est constituée que d'arêtes virtuelles. On ne va conserver que les arêtes virtuelles qui servent à la **silhouette**. Pour savoir si une arête est une ligne de silhouette, on regarde si elle est partagée par une face avant et une face arrière. Si on a déjà supprimé les faces arrières, on peut directement regarder si l'arête est partagée par une seule face.

2.3.3.1. Algorithme de l'horizon flottant *(Floating horizon)*

> Simplifier le calcul de visibilité.

- Déterminer un axe majeur puis tracer des lignes perpendiculaires à cet axe en s'éloignant de l'observateur.



2.3.3.2. Lignes d'expression

> **Ligne d'expression** : Eclairage simulé par des lignes et lignes de courbure.

- Pour calculer l'éclairage, on utilise le *cosinus de Lambert* : $\vec{n}.\vec{l} = ||\vec{n}|| \times ||\vec{l}|| \times \cos(\vec{n}, \vec{l})$

2.3.3.3. Visualisation en atténuation de profondeur

> Coloration proportionnelle à z *(depth cueing)*.

2.3.3.4. Halo

> Tri en profondeur décroissante des lignes. Dessin avec zone d'occultation.

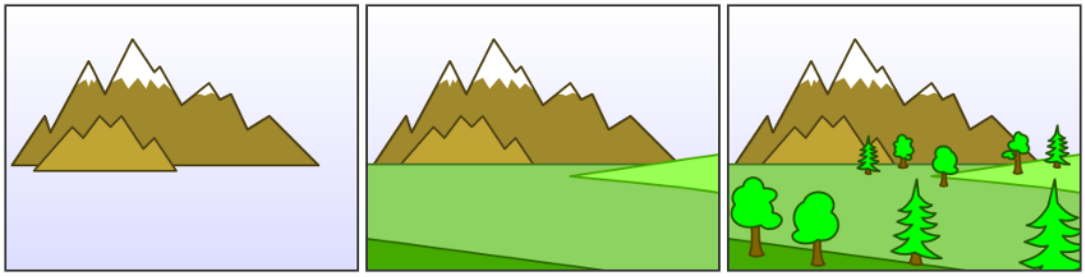
2.3.3.5. Style/z

> Changement des styles des lignes suivant z (plus épais, etc).

2.4. Elimination des surfaces cachées

> Tri en profondeur (en z décroissant), **algorithme du peintre**.

Algorithme du peintre :



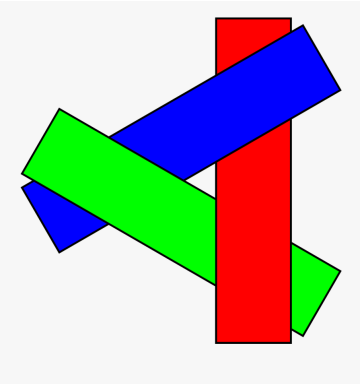
- Objets simples et petits :
  - **B-Rep** (Triangulaires)
  - **Voxels**

2.4.1. Tri en profondeur des voxels :

- Axe dominant opposé à l'axe de plus petit angle par rapport à la direction d'observation.

2.4.2. Tri en profondeur des triangles :

1. Boîtes englobantes en profondeur.
2. Projection des points d'un objet sur le plan support de l'autre. Cela nous fournira
   - Soit le classement
   - Soit si nécessaire, une découpe de l'objet :



2.4.3. Tampon de profondeur *(Z-buffer)*

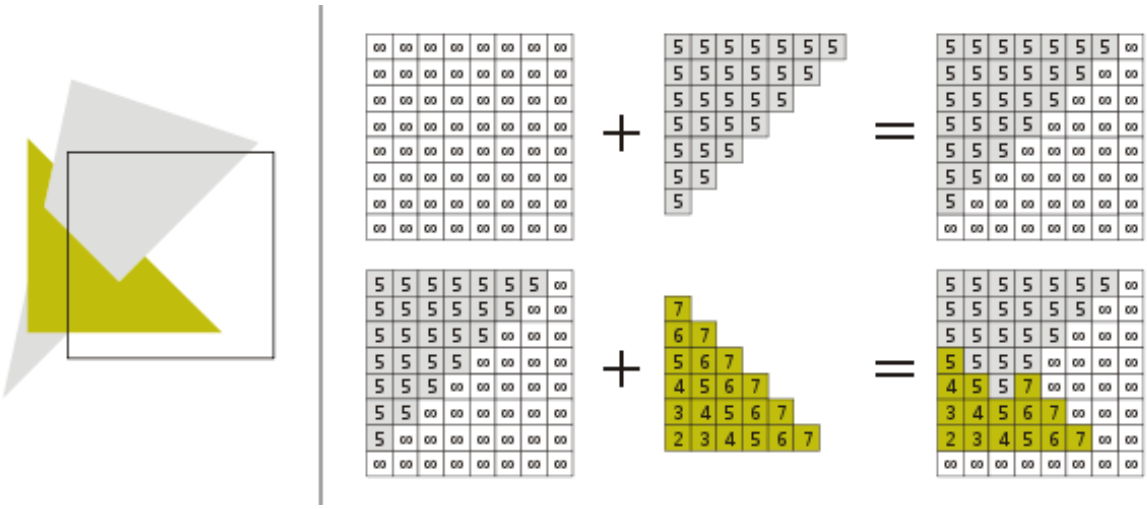
Tampon de profondeur : tableau de valeurs de profondeur.

- Pour chaque pixel, stocker le z de la surface génératrice :

R	G	B	α	Z

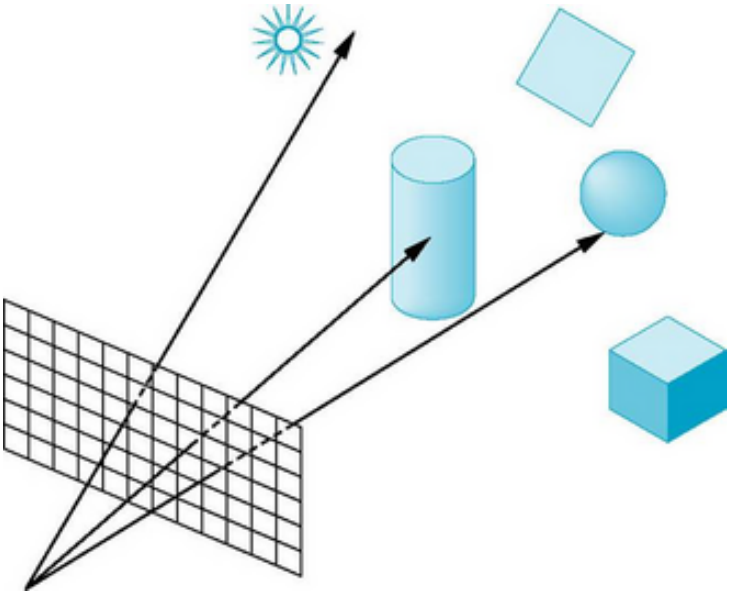
2.4.3.1. Dessin (observateur, scène)

1. Changement de repère **scène** $\rightarrow$ **observateur**.
2. Pour chaque objet de la scène, le dessiner.



2.4.4. Ray-casting

- Pour chaque pixel, on calcule un rayon qui part de l'observateur et qui passe par le pixel et on calcule l'intersection avec les objets de la scène. Ces intersections fournissent une distance par rapport à l'observateur. On ne conserve que l'intersection la plus proche de l'observateur sur un pixel.



3. Modèles d'éclairement

3.1. Lumière

Acteurs :

- Surfaces réceptrices
- Milieu de propagation
- Observateur

La **fréquence** des ondes joue sur la **teinte**.

Photon : Paquet d'énergie $\rightarrow$ spectre.

Spectre (énergie)

- en RVB (3 échantillons)
- Ou plus d'échantillons (10 ou 30)

3.1.1. Unités

- **Energie transportée** : $\text{Joule} \rightarrow \text{lumen.s}^{-1}$
- **Puissance** : $\text{Watt} \rightarrow \text{lumen}$
- **Intensité** (flux par unité d'angle solide) : $\text{Watt.sr}^{-1} \rightarrow \text{candela}$

> sr : **steradian** (unité d'angle solide)

- **Eclairement** (flux émis/reçu par unité de surface) : $\text{Watt.m}^{-2} \rightarrow \text{lux}$
- **Luminance** (flux émis/reçu par unité de surface apparente) : $\text{Watt.m}^{-2}\text{.sr}^{-1} \rightarrow \text{nit}$

> Apparente : Angle formé par les rayons visuels qui vont de l'observateur à chacun des astres ou objets.

> nit : **nit** (unité de luminance)

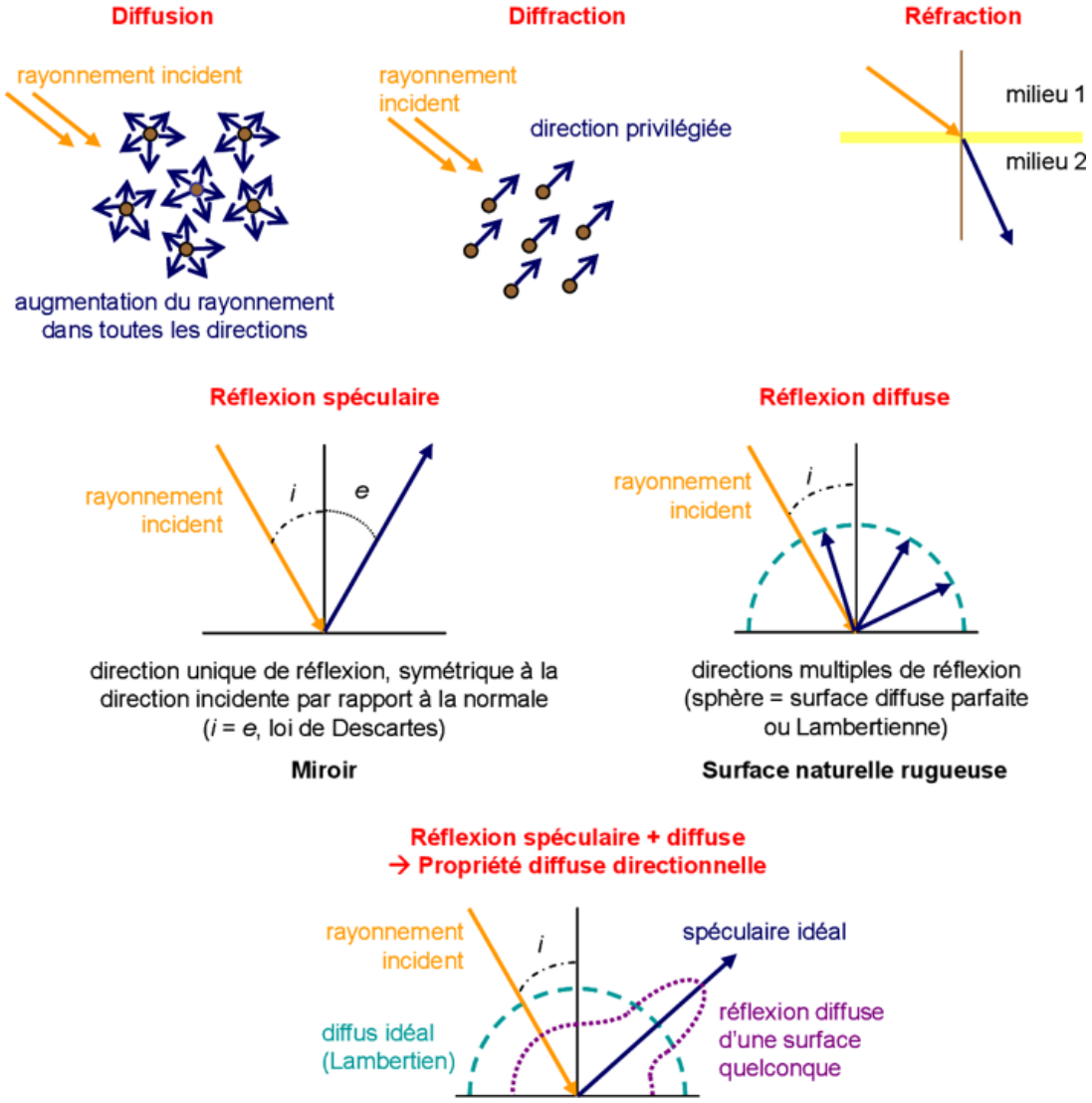
3.2. Sources de lumière

Photon + Direction de propagation + Spectre *(= Tableau d'énergie/fréquence)*

3.3. Milieu de propagation

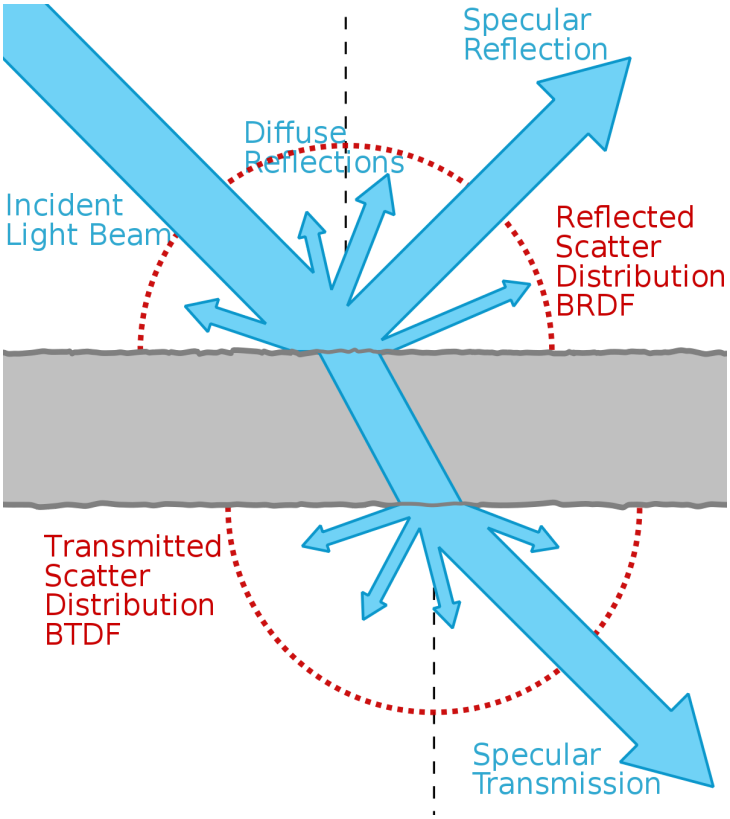
Changement de milieu ⇒ densités différentes.

> Loi de Snell-Descartes : $\frac{\sin(\theta_1)}{\sin(\theta_2)} = \frac{n_1}{n_2}$
>
> n_i : **Indice de réfraction**
>
> $\frac{n_1}{n_2}$: **Densité relative**



3.4. Surfaces réceptrices

- *Bidirectionnal Refection Distribution Function* ***(BRDF)***
- *Bidirectionnal Transmission Distribution Function* ***(BTDF)***



3.5. Equation de rendu (Kajuja, 1986)

$$L_0(x, \omega_0, \lambda, t)$$

- x : position
- ω_0 : direction
- λ : longueur d'onde
- t : temps

$$L_e(x, \omega_0, \lambda, t) + \int_\Omega f_s(x, \omega_0, \omega_i, \lambda, t) L_i(x, \omega_i, \lambda, t).(\vec{\omega_i}.\vec{n}) d\omega_i$$

3.6. Modèles locaux d'éclairement

- Surface à éclairer + observateur + sources de lumière (ambiant + diffus + spéculaire) + effets divers (micro-relief par perturbation de la normale)
- Uniquement les sources primaires

⇒ On ajoute un terme qui résume l'éclairement des sources secondaires. (éclairage ambient).

$$I = I_a \times k_a + \sum_i v(x_i) f(d_i) I_i [k_d(-\vec{\omega_i}.\vec{n}) + k_s(\vec{\omega_i}.\vec{h}.\vec{n})^r]$$

- I_a : intensité lumineuse ambiante
- k_a : coefficient de réflexion de la lumière ambiante
- $v(x_i)$: visibilité de la source
- I_i : intensité lumineuse de la source primaire
- $f(d_i)$: atténuation par la distance

- $v(x_i).f(d_i).I_i$: quantité de lumière reçue par la surface
- k_d : coefficient de réflexion diffuse
- k_s : coefficient de réflexion spéculaire
- $k_d.(-\vec{\omega_i}.\vec{n})$: réflexion diffuse $\rightarrow$ **modèle de Lambert**
    - Soit $(k_{d_r}, k_{d_v}, k_{d_b})$ ou $k_d \times (T_r, T_v, T_b)$
- $\vec{h} = \frac{1}{2}(-\vec{\omega_i}, -\vec{\omega_0})$: half-vector

3.6.1. Textures

> Modification des paramètres de surface ou de volume

- Couleur
- Coefficient de réflexion (k_a, k_d, k_s)
- Normale $(\vec{n})$
- Position

> **Habillage** : mise en correspondance de l'**espace texture** et de l'**espace modèle géométrique**.

Exemple :

$$\begin{array}{ccccc} \text{texture 2D} & \longleftrightarrow & \text{surface objet 2D} & \text{mapping manuel} + \text{mise à plat} \\ & & & u_0, v_0 \rightarrow r_0, v_0, b_0 \\ (x_t, y_t) & \longleftrightarrow & (u, v) & \rightsquigarrow u_1, v_1 \rightarrow r_1, v_1, b_1 \\ & & & \text{etc.} \\ & f & & \\ & & & x_0, y_0, z_0 \rightarrow r_0, v_0, b_0 \\ & \longleftrightarrow & \text{ou } (x, y, z) & \rightsquigarrow x_1, y_1, z_1 \rightarrow r_1, v_1, b_1 \\ & & & \text{etc.} \end{array}$$

3.6.1.1. Méthodes projectives

- **Projection droite** (orthographique ou perspective)
- **Projection cylindrique** et **sphérique** (changement de coordonnées $\text{cartésien} \rightarrow \text{polaire}$)

3.6.1.2. Textures procédurales

> 2D ou 3D (volumique) **bruit de Perlin**.

Variations (filtrage) sur une fonction de bruit :

Exemple :

$$\left.\begin{array}{c} \text{point en espace objet} \\ \end{array} \begin{array}{c} (u, v) \\ \text{ou } (x, y, z) \end{array}\right\} \rightarrow < r, v, b >$$

$$\text{avec } r, v, b \text{ aléatoires}$$

- précalcul d'une texture de bruit échantillonnée + exploitation par interpolation

Textures procédurales non aléatoire :

Exemple :

$$f(x, y, z) = \text{parité}(\text{int}(\text{x}), \text{int}(\text{y}), \text{int}(\text{z}))$$

en **2D** :

$$\begin{array}{cccccc} y & & & & & \\ \uparrow & 0 & 1 & 0 & 1 & \\ & 1 & 0 & 1 & 0 & \\ & 0 & 1 & 0 & 1 & \\ & & & & \rightarrow & x \end{array}$$

3.7. Modèles globaux d'éclairement

> **Tous** les objets de la scène contribuent à l'éclairement (éclairement indirect).

1. Modèles **strictements lambertiens** (mats) $\Rightarrow$ **"radiosité"**
2. Objets **strictements spéculaires** (miroirs) $\Rightarrow$ **"photons"**

Soient :

- B_i : Luminance de l'élément de surface
- E_i : Emmission propre
- ρ_i : Réflexion propre

On a :

$$B_i = E_i + \rho_i \sum_j F_{ij} B_j$$

Que nous pouvons écrire :

$$\begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_n \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -\rho_1 F_{12} & -\rho_1 F_{13} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1 & -\rho_2 F_{23} & \cdots & -\rho_2 F_{2n} \\ -\rho_3 F_{31} & -\rho_3 F_{32} & 1 & \cdots & -\rho_3 F_{3n} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \cdots & \cdots & 1 \end{bmatrix}}_{\text{matrice } n \times n} \cdot \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \vdots \\ B_n \end{bmatrix}$$

> n : nombre d'éléments de surface de la scène complète.

$$\Rightarrow E = F \times B \Rightarrow F^{-1} \times E = B$$

Calcul de F^{-1} : $O(n^3)$

3.8. Algorithmes d'éclairement

Modèle d'éclairement **local** (ambient, diffuse, spéculaire)

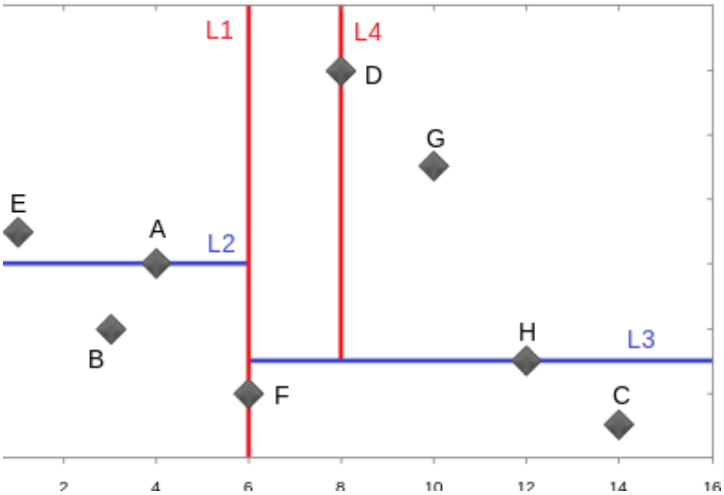
Modèle d'éclairement **global** :

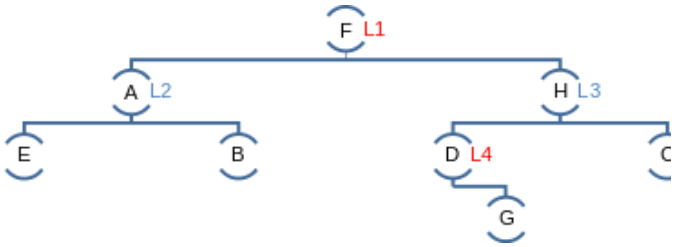
- **i) radiosité** : seulement diffus
- **ii) placage de photons** *(photon mapping)* : seulement spéculaire

> Algorithme de visualisation, lancer de rayons récursifs (en partant des sources lumineuses) :

- Ce que nous voyons est le résultat de l'éclairement direct + le rayon réfléchi + le rayon réfracté.

1. Carte d'éclairement *(kd-tree)*. Photon (x, y, z).





2. Exploitation de la carte

- **iii) objets translucides** :
  - ***SSLT***: *SubSurface Light Transport* (modèle de transport de lumière dans les objets translucides)
  - a) Calculer une carte d'éclairement direct objet par objet, mettre à plat les cartes flou.
  - b) Transmission/distance.

4. Animation

> Perception du mouvement :
>
> - oeil + nerf optique : persistance "rétinienne" *(persistance of vision)* $\Rightarrow$ image résiduelle pendant $1/20s$
> - cerveau :
>   - mouvement B
>   - phénomène φ

Annexe

Logiciels de modélisation géométrique

- **AutoCAD :** (CAD $\rightarrow$ Computer Aided Design) dessin technique (2D) est devenu **3DS** (3D artistique)
- **Maya** $(\sim 3\ 500€)$
- **Blender** (gratuit)