Synthèse d'Images

Visibilité, Apparence, Texture

Tamy Boubekeur

RAPPEL : SCÈNE ET GÉOMÉTRIE

Modèles de Scène 3D

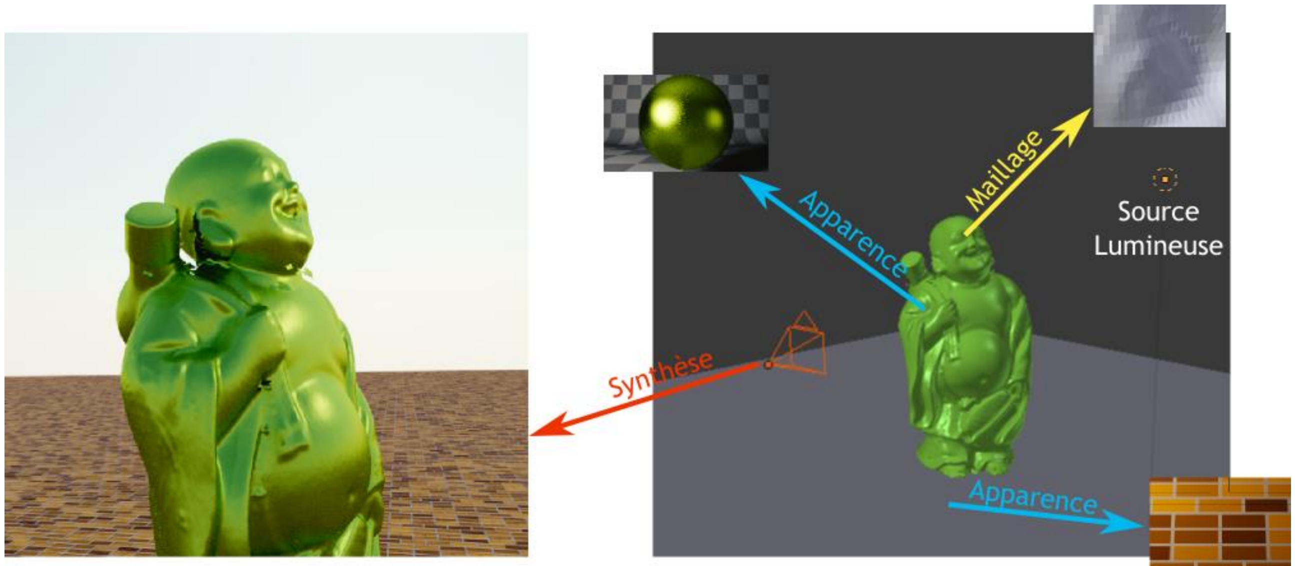


Image Numérique Scène 3D

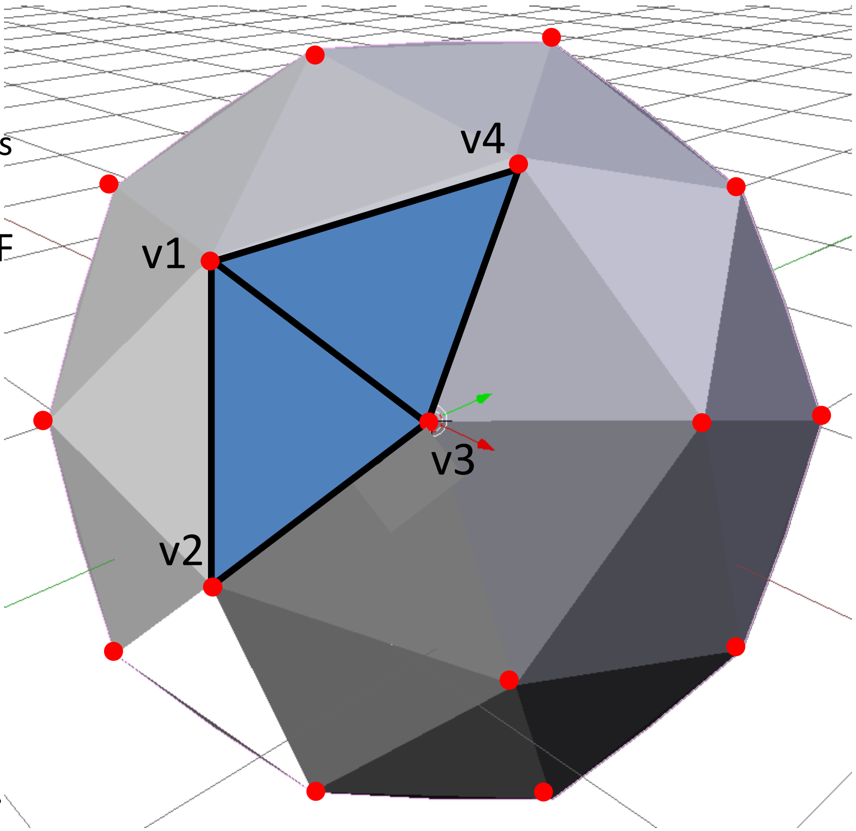
- Une collection de modèles :
  - Capteur (caméra)
  - Géométries
    - Maillages, particules, iso-surfaces, etc
  - Apparence
    - Matériaux, textures
  - Lumières
  - Animation
    - Évolution temporelles des paramètres

des autres modèles

    - Physique solide, fluides, corps déformables
    - Interactivité et actuators
- Une structure entre ces modèles
  - Appartenance et hiérarchie
  - Données et instances

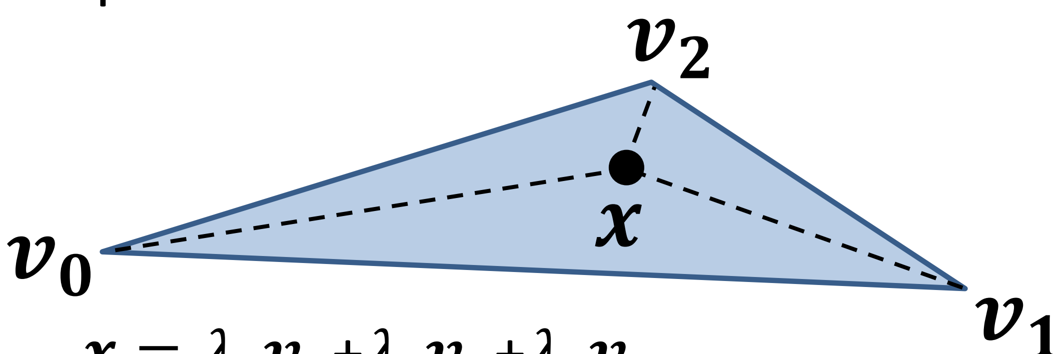
Surface Maillée

- Maillage:
  - modèle géométrique dominant en rendu
  - Génération possible à partir des autres modèles
    - Cf cours de Modèlisation Géométrique
- Définition: un ensemble de faces polygonales F indexant un ensemble de sommets V.
- V: Ensemble de sommets (géométrie)
  - v1 (x, y, z)
  - v2 (x, y, z)
  - v3 (x, y, z)
  - v4 (x, y, z)
- F: Ensemble de faces (topologie)
  - (v1, v2, v3)
  - (v1, v3, v4)
- Outre la position, chaque sommet peut porter d'autres attributs:
  - vecteur *normales,* critiques en rendu.
  - *Couleur par sommet*
  - *Coordonnées de textures (UV)*



Coordonnées barycentrique

- Coordonnée d’un point dans l’espace d’un polygone
- Simple pour un triangle
- Permet d’interpoler linéairement tout attribut de sommet



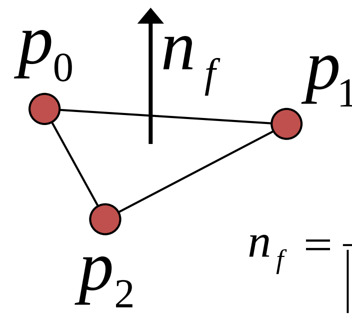
$$\boldsymbol{x} = \lambda_0 \boldsymbol{v_0} + \lambda_1 \boldsymbol{v_1} + \lambda_2 \boldsymbol{v_2}$$

$$\begin{array}{r}\text{Coordonnée}\\ \text{barycentriques de x}\\ \text{dans le triangle}\end{array}\left\{\begin{aligned}
\lambda_0 &= \left[(\boldsymbol{v_2} - \boldsymbol{v_1}) \otimes (\boldsymbol{x} - \boldsymbol{v_1})\right] \cdot \boldsymbol{a} \\
\lambda_1 &= \left[(\boldsymbol{v_0} - \boldsymbol{v_2}) \otimes (\boldsymbol{x} - \boldsymbol{v_2})\right] \cdot \boldsymbol{a} \\
\lambda_2 &= \left[(\boldsymbol{v_1} - \boldsymbol{v_0}) \otimes (\boldsymbol{x} - \boldsymbol{v_0})\right] \cdot \boldsymbol{a}
\end{aligned}\right.$$

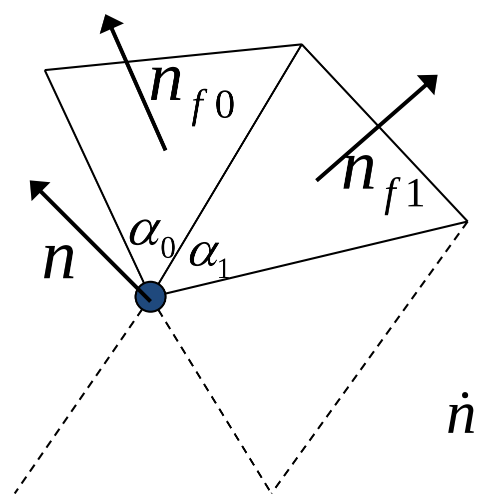
$$\boldsymbol{a} = \frac{(\boldsymbol{v_1} - \boldsymbol{v_0}) \otimes (\boldsymbol{v_2} - \boldsymbol{v_0})}{\|(\boldsymbol{v_1} - \boldsymbol{v_0}) \otimes (\boldsymbol{v_2} - \boldsymbol{v_0})\|^2}$$

Normales

- **Essentielles pour le rendu**
  - Alignement de la BRDF
- Stockage aux sommets ou par cartes (normal maps)
- Calculs possibles:
  - Moyennes des normales des faces incidentes
  - Moyennes pondérée par les angles des arêtes incidentes
    - Plus robustes pour les distributions de triangles non uniformes
  - Moyenne pondérée par l'aire de l'intersection du triangle et de la cellule de voronoI du sommet.



$$n_f = \frac{(p_1 - p_0) \wedge (p_2 - p_0)}{\left\|(p_1 - p_0) \wedge (p_2 - p_0)\right\|}$$



$$\dot{n} = \frac{\sum_i \alpha_i n_i}{\sum_i \alpha_i}$$

$$n = \frac{\dot{n}}{\left\|\dot{n}\right\|}$$

Interpolation de Normales

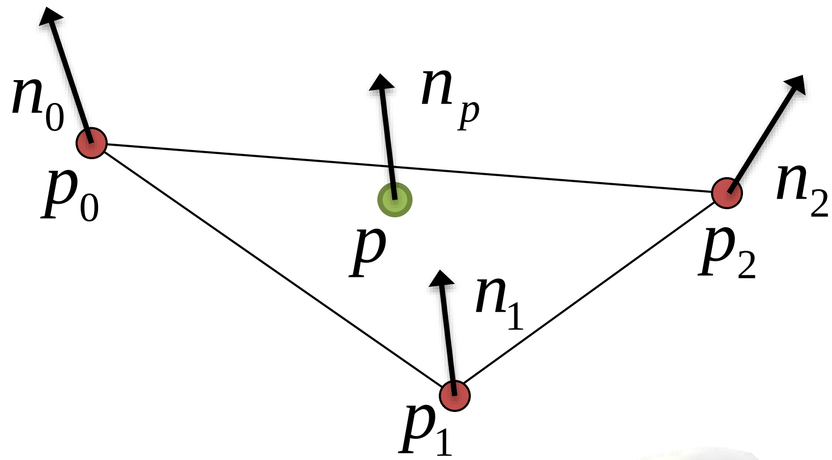
- Une normale en chaque point d'un maillage à partir des normales de ses sommets [Phong 75].
- Soit un point p sur un triangle t tel que :

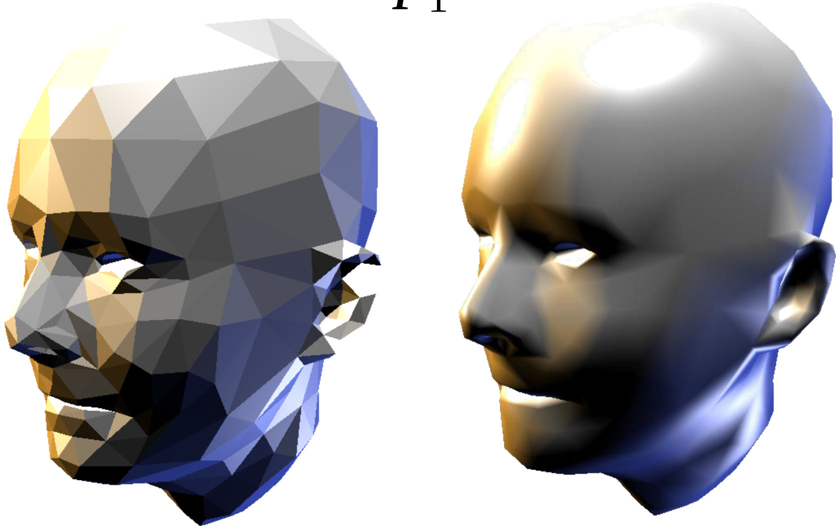
$$p = \lambda_0 p_0 + \lambda_1 p_1 + \lambda_2 p_2$$

Avec $(\lambda_0, \lambda_1, \lambda_2)$ les **coordonnées barycentrique** de p dans t.

Alors on définit la normale interpolée de Phong en p comme :

$$n_p = \frac{\lambda_0 n_0 + \lambda_1 n_1 + \lambda_2 n_2}{\|\lambda_0 n_0 + \lambda_1 n_1 + \lambda_2 n_2\|}$$



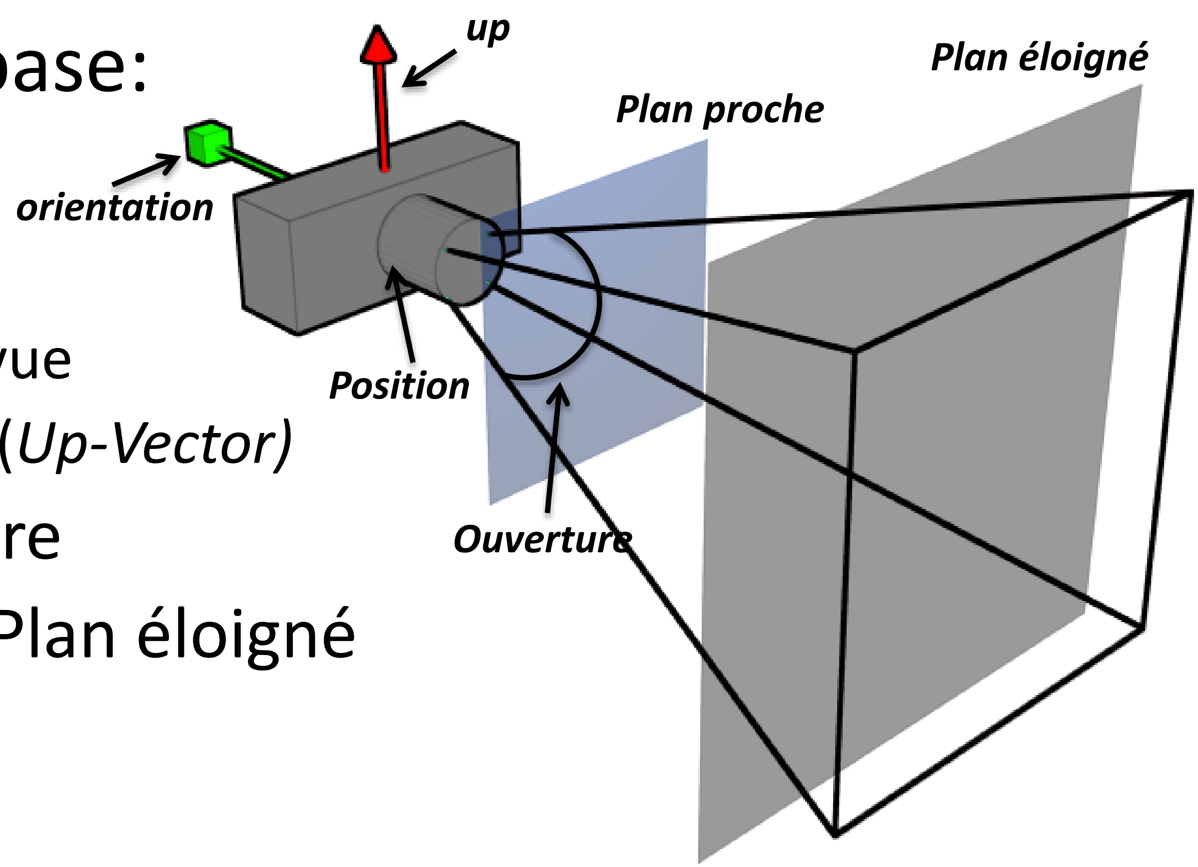


Normale de face Normale de Phong

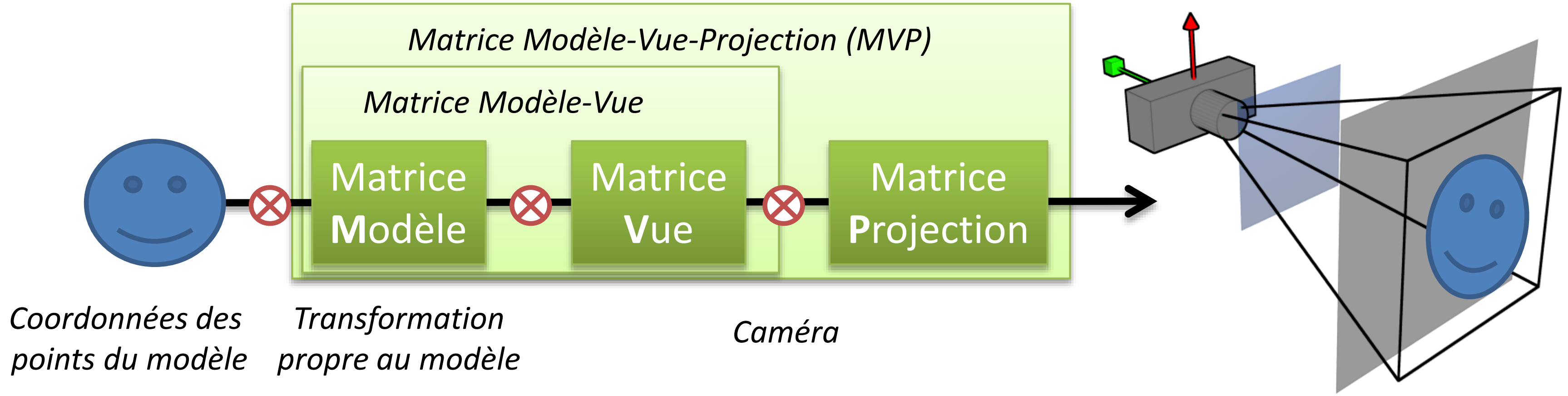
RAPPEL : CAMÉRA ET TRANSFORMATIONS

Modèle de Caméra

- En général: *pinhole* camera
- Projection perspectivique
- Paramètre de base:
  - Position
  - Orientation
    - Direction de vue
    - Vecteur haut (*Up-Vector)*
  - Angle Ouverture
  - Plane proche/Plan éloigné



Transformation et Projections

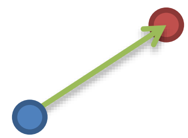


- Représentation par une matrice 4x4
- Transformation rigide
  - Translation
  - Rotation
  - Echelle
- Utilisation: changement de repère pour le placement des géométries dans le repère de la caméra et leur projection

Transformation Affine

Translation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



Rotation

$$R_X(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \quad R_Y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \quad R_Z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Scaling

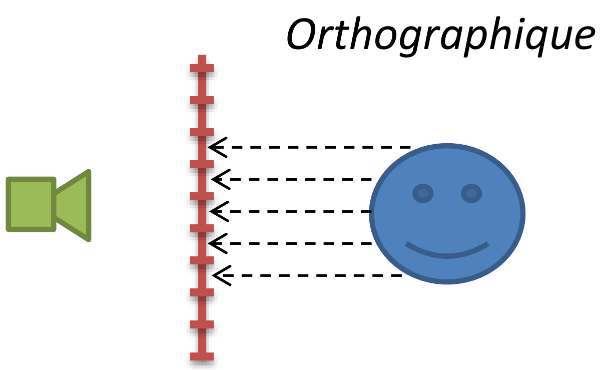
$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

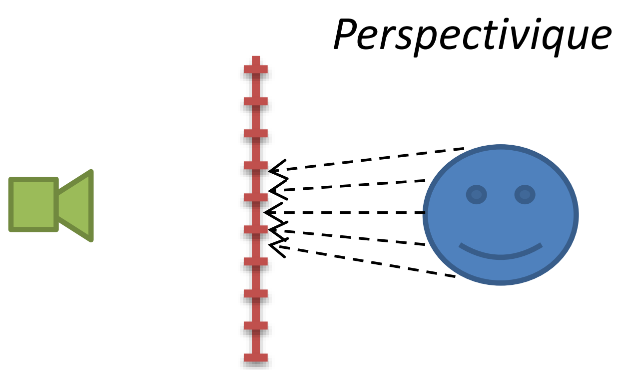
Transformations appliquées pour :

- *Déplacer les sommets des polygones en 3D*
- *Les placer dans le repère de la caméra*

Projection

- Projeter les sommets des polygones (transformés) dans le plan de l'image.
- 2 types de projections:





- Encore une fois exprimable à l'aide d'une matrice 4x4 : la **matrice de projection**

Géométrie Projective

- Raisonner dans l'espace des droites
- Projection en perspective

Point xyz > xyz**w : coordonnées homogènes**

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ w=1 \end{pmatrix} \rightarrow ...transformations... \rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} \rightarrow \begin{pmatrix} x'/w' \\ y'/w' \\ z'/w' \end{pmatrix}$$

Projection Homogène

Rasterization & Lancer de Rayon

ALGORITHMES DE VISIBILITÉ

Visibilité

- **Objectif:** déterminer quels primitives sont visibles/cachées depuis un point donné, comme par exemple depuis :
  - une caméra > formation d'une image
  - une source de lumière > éclairage
- Détermine les 2 grandes classes d'algorithmes de rendu:

Projection

Rasterization

- Alg. du peintre
- Z-Buffer
- Ombrage Différé

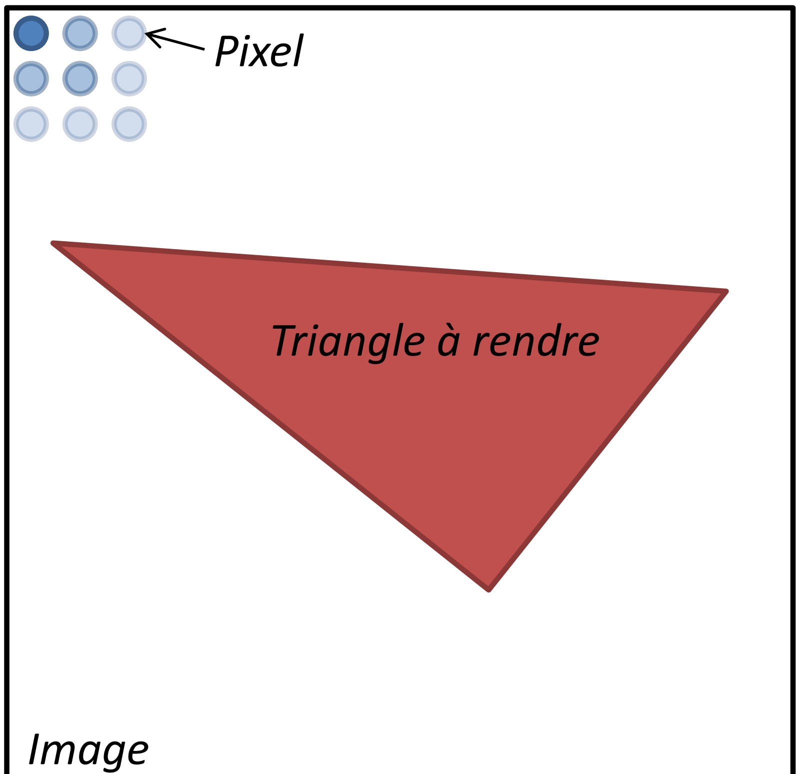
Lancer de rayon

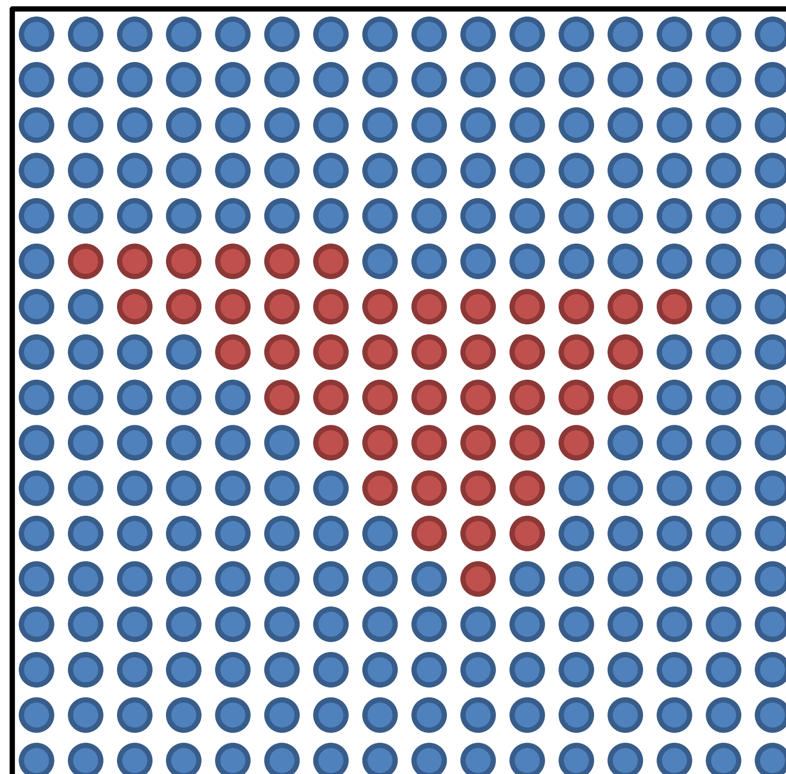
Ray casting/tracing

- Intersection rayon/triangle
- Récursion

Rasterisation

- Discrétisation d'un polygone dans une grille (image)
- Plusieurs algorithmes alternatifs
  - 1 triangle > n **fragments**
  - **Fragment = {x,y}**
  - **Pixel** = fragment visible
    - Plusieurs triangles peuvent couvrir les même pixel
    - Visibilité déterminé plus tard



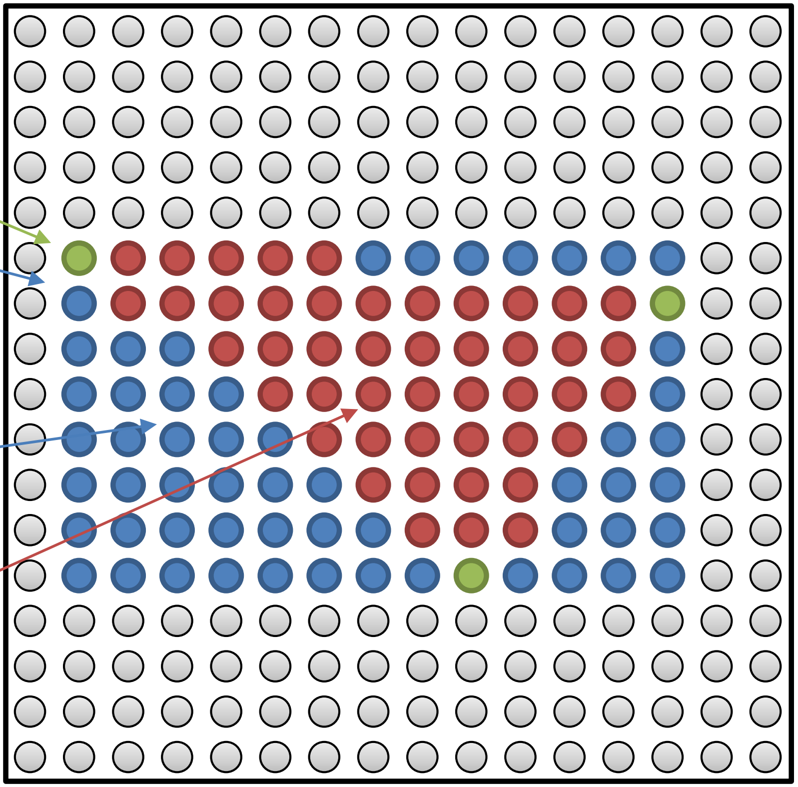


Rasterisation

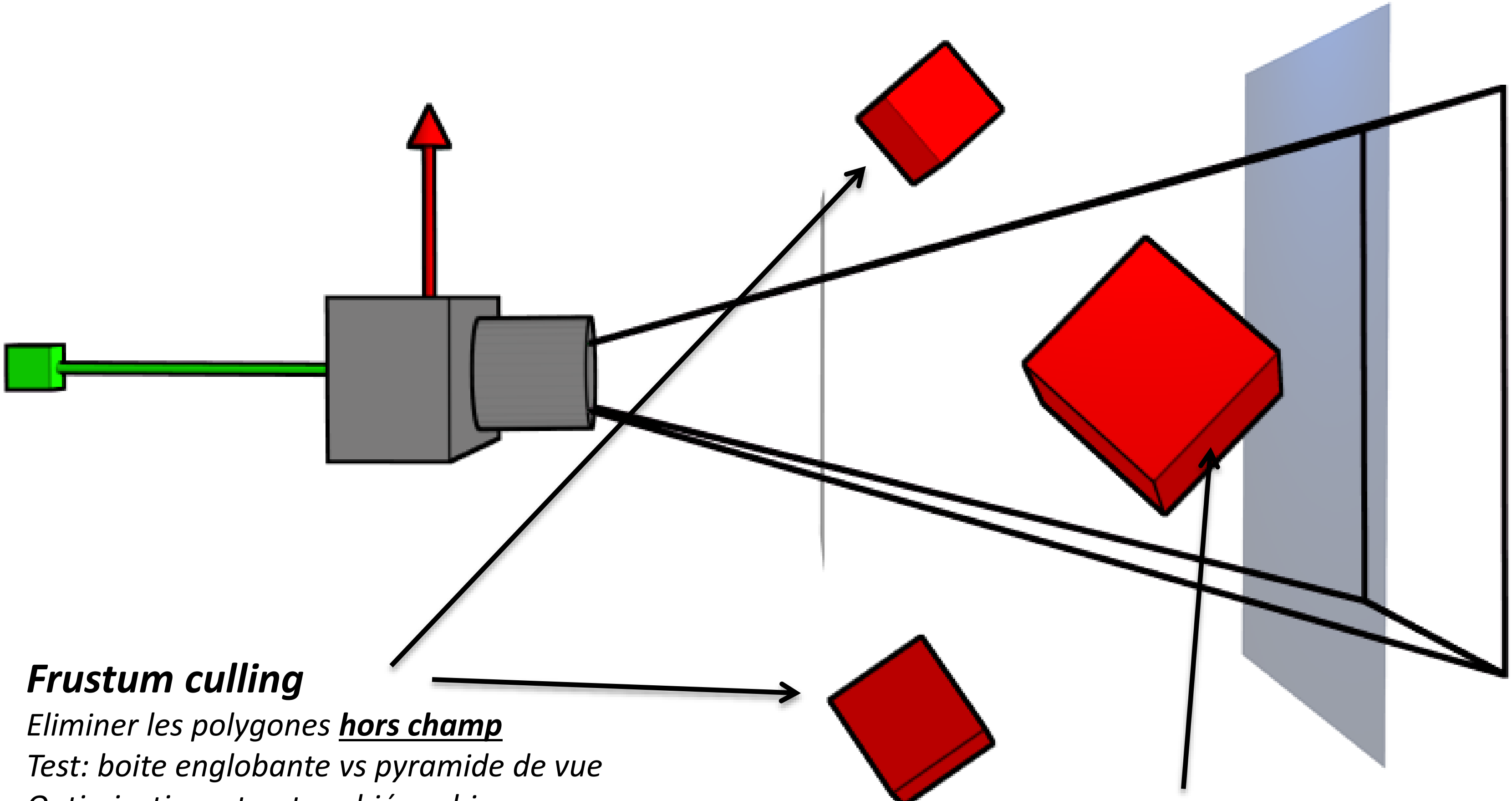
- Discrétisation d'un polygone dans une grille (image)
- Plusieurs algorithmes alternatifs
  - Pixel = fragment visible
- Sans précaution, tous les polygones de la scène sont *traités*

  > Elimination de primitives

1. Projection des **sommets 3D** du triangle dans le plan de l'image.
2. Calcul de la boîte englobante {{minX, minY},{maxX, maxY}} des pixels candidats
3. Calcul des coordonnées barycentriques {b0, b1, b2} de ceux-ci en fonction du triangle
4. Classification des pixels effectivement couverts
   - b0,b1,b2 >=0 et b0+b1+b2 = 1
5. Emission des fragments correspondant



Elimination de primitives (Culling)



Frustum culling

*Eliminer les polygones **<u>hors champ</u>***

Test: boite englobante vs pyramide de vue

Optimisation: <u>structure hiérarchique</u>

Backface culling

Elimination des faces arrières

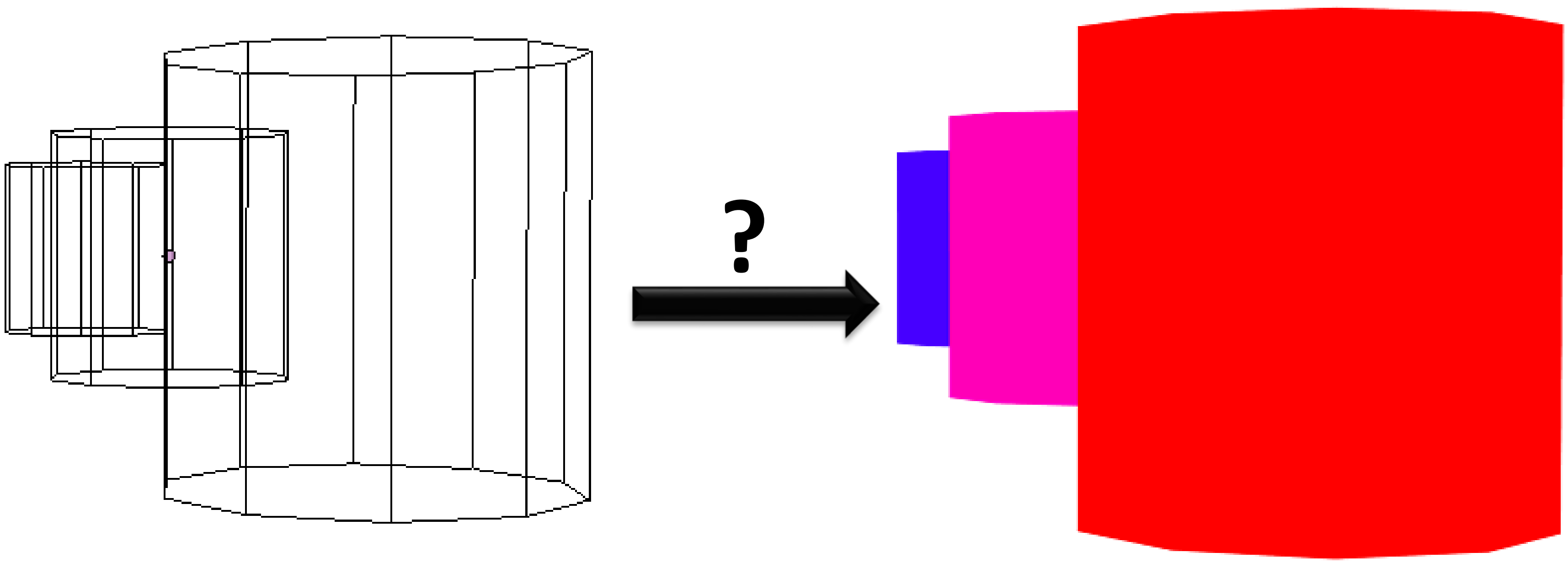
Test: normal vs caméra orientation

Occlusion culling:

- *éliminer les objets occultés dans la pyramide de vue*
- *Compliqué > peu utilisé*

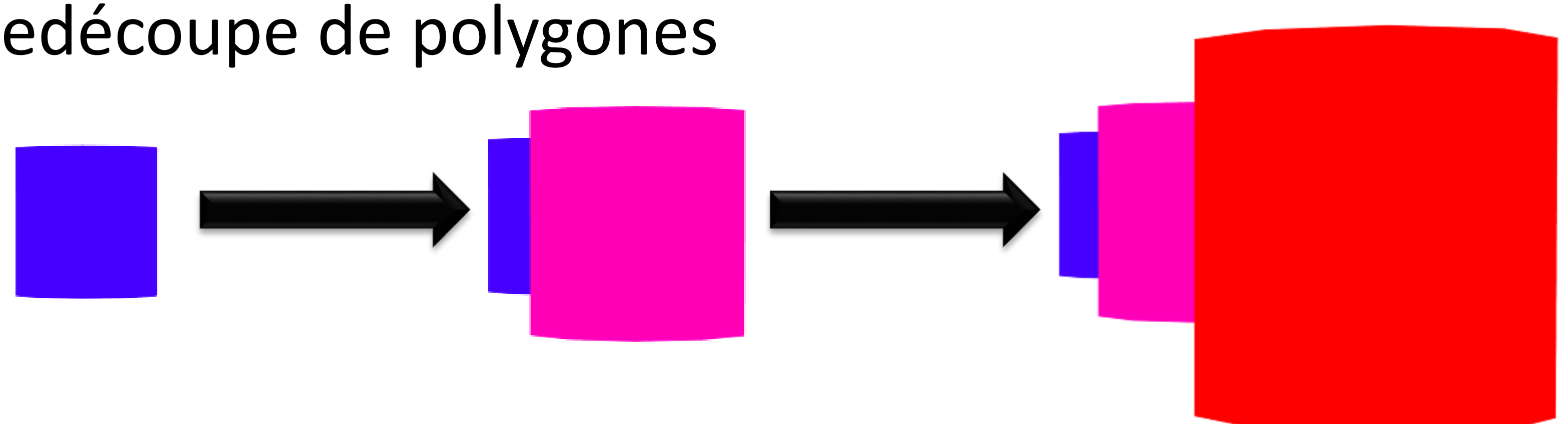
Visibilité dans le champ de vue

- Comment déterminer les parties visibles et les parties cachées de la géométrie des objets depuis un point donné (e.g. position de la caméra) ?



Algorithme du peintre

- Ordonnancement général des polygones le long de l’axe de vue
- Dessin de *loin en proche* de la liste ordonnée
- Lent > Optimisé via un **BSP-Tree**, mais *statique*
- Cas ambigus : intersection de polygones
  - Redécoupe de polygones

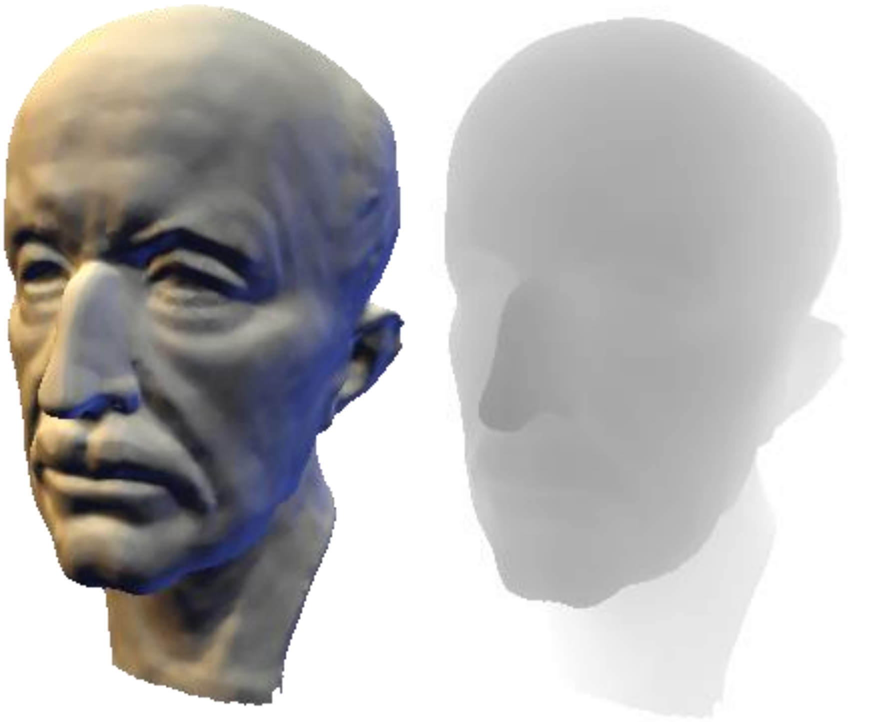


Rasterization avec Z-Buffer

Idée: maintenir un tampon (buffer) ZB de la même taille que le tampon couleur FB de l'écran, mais stockant pour chaque pixel la **profondeur** de la géométrie le recouvrant

Algorithme

```
Pour chaque polygone t :
    Si t hors-champ ou t non face caméra
        Ignorer t
    Rasterizer t
    Pour chaque fragment (x,y) de t :
        c := couleur de de t en (x,y)
        z := distance fragment-caméra
        Si ZB(x,y) > z alors :
                FB(x,y) := c
                ZB(x,y) := z
        Sinon
                Ignorer (x,y)
```



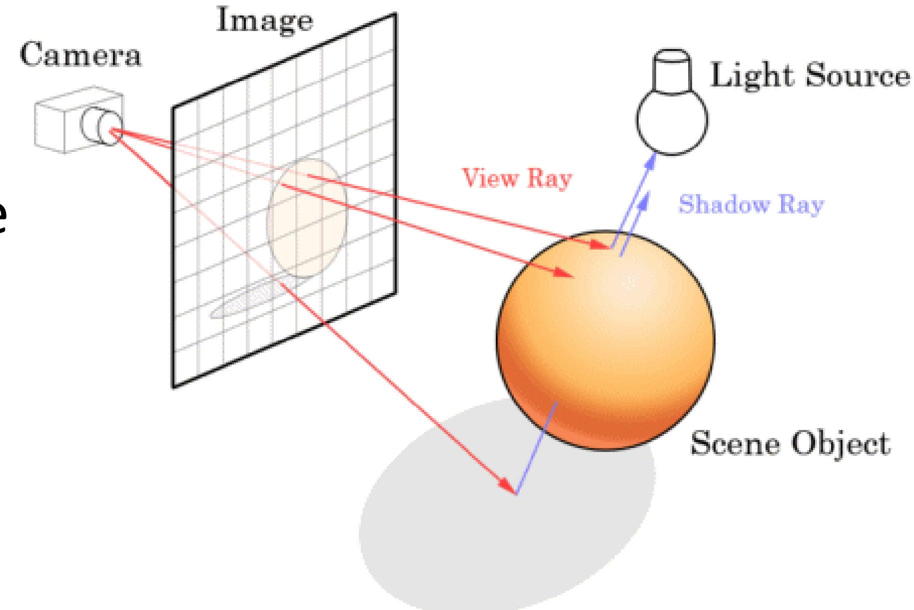
Frame-buffer (FB) ***Z-Buffer (ZB)***

☺ Rapide, support GPU, linéaire en temps

☹ Plusieurs polygones par pixel / aliasing

Lancer de Rayon (Ray Tracing)

Idée: partir du point de vue et chercher pour chaque pixel le premier objet intersectant la ligne de vue au travers du pixel



Algorithme

```
Pour chaque pixel (x,y)
    r := rayon caméra-pixel
    e = +∞
    FB (x,y) = (0,0,0)
    Pour chaque polygone t
        x := intersection (r, t)
        Si x != null
            d = distance camera-x
            Si d < e
                e = d
                FB (x,y) = couleur de X
```

☺ Simple, facilement généralisable

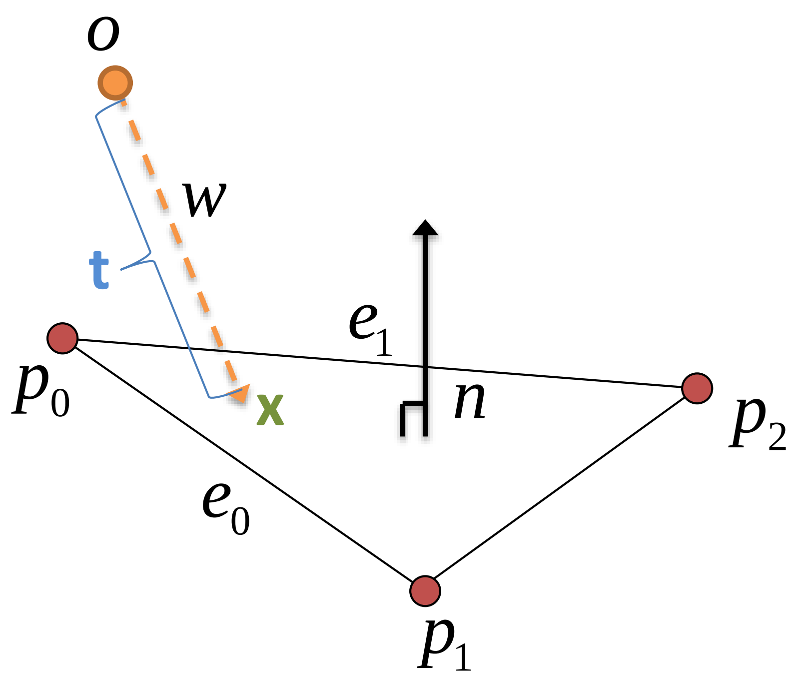
☹ Couteux

Accélération:

- Ranger les polygone dans un kd-tree
  - Pré-calcul en $O(n \log n)$
- Utiliser pour le kd-tree pour la recherche d'intersection
  - Coût par pixel: $O(\log n)$

Intersection Rayon-Triangle

```
IntersectionRayonTriangle (o, w, p0, p1, p2) :
        e0 = p1 – p0
        e1 = p2 – p0
        n = e0^e1 / |e0^e1|
        q = w^e1
        a = e0.q
        si n.w >= 0 ou |a| < epsilon
                retourner null
        s = (o – p0)/a
        r = s^e0
        b0 = s.q
        b1 = r.w
        b2 = 1 – b0 – b1
        si b0 < 0 ou b1 < 0 ou b2 < 0
                retourner null
        t = e1.r
        si t >= 0
                retourner [b0, b1, b2, t]
        retourner null
```



Intersection exprimée :

- selon le triangle, en coordonnées *barycentriques* x=b0*p0+b1*p1+b2*p2
- Selon la forme paramétrique du rayon : x = o + t*w

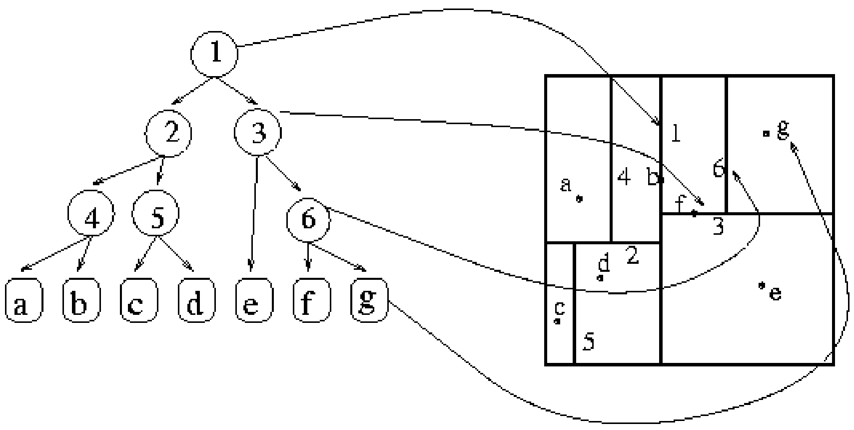
Accélération du lancer de rayon

- Pour chaque rayon > éviter d'inspecter les polygones de la scène un à un
- Ranger les polygones au préalable dans une structure de données
  - Hiérarchique
  - Ajustée à la scène
  - Permettant de rapidement éliminer des pans entier de la scène lorsqu'on qu'on cherche l'intersection d'un rayon (ou d'un ensemble de rayon) avec celle-ci
- Solutions classiques
  - kD-Tree
  - BVH

kD-Tree

- Structure de partitionnement ***orthogonale*** d'échantillons
- Arbre binaire. A chaque niveau:
  - Calculer la boite englobante de **P**
  - Diviser **P** le long du plus grand axe (X, Y ou Z)
- Algorithme de construction:

```
KDNode buildKDTree (PointList P) {
        BBox B = computeBoundingBox (P);
        Point q = findMedianSample (B,P);
        Node n;
        Plane H = plane (q, maxAxis (B))
        n.data = <q,H>;
        PointList Pu = upperPartition (P, H);
        PointList Pl = lowerPartition (P, H);
        n.leftChild = buildKDTree (Pu);
        n.rightChild = buildKDTree (Pl);
        return n;
}
```



Propriétés du kD-Tree

Générique

- Ordonnancement spatial en dimension arbitraire
- Robuste et constructible sur n'importe quel ensemble de points

Accélération de la recherche des plus proches voisins (NN)

- Recherche par distance: tous les points à située dans une boule de rayon r centrée sur l'origine de la recherche
  - Basées sur le test d'intersection sphère/boite
- Recherche par cardinal: trouver les k plus proches voisins (kNN)
  - via une file à priorité de taille maximum pour ordonner les points (O(k log N) pour une arbre équilibré

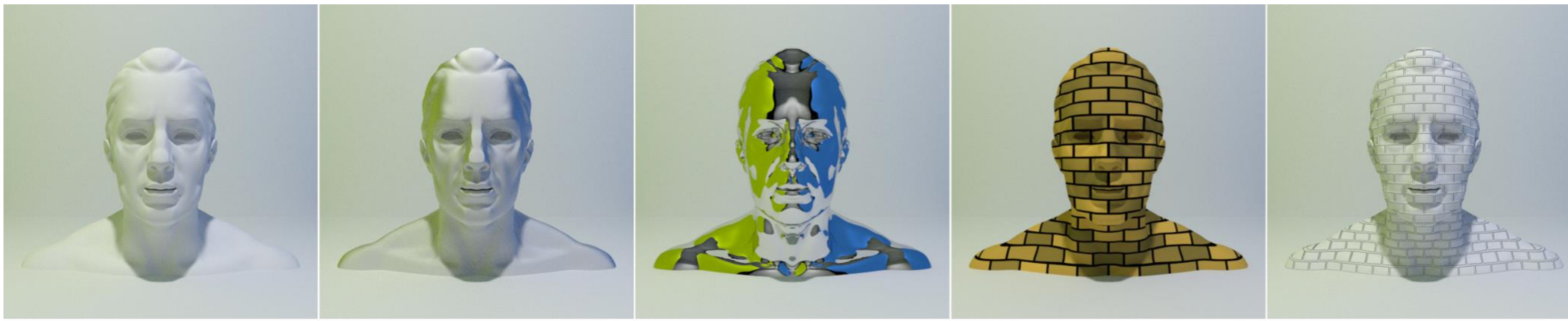
Accélération des tests d'intersection

- Test récursif
- Traitement de « paquets de rayons »
- Possibilité d'ajouter un biais géométrique
  - e.g. *Surface Area Heuristic* ou SAH

Lumière, réflectance, couleur.

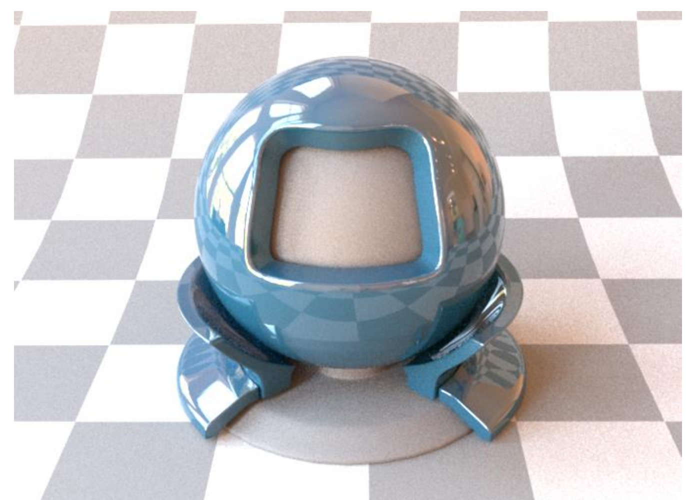
APPARENCE

Apparence

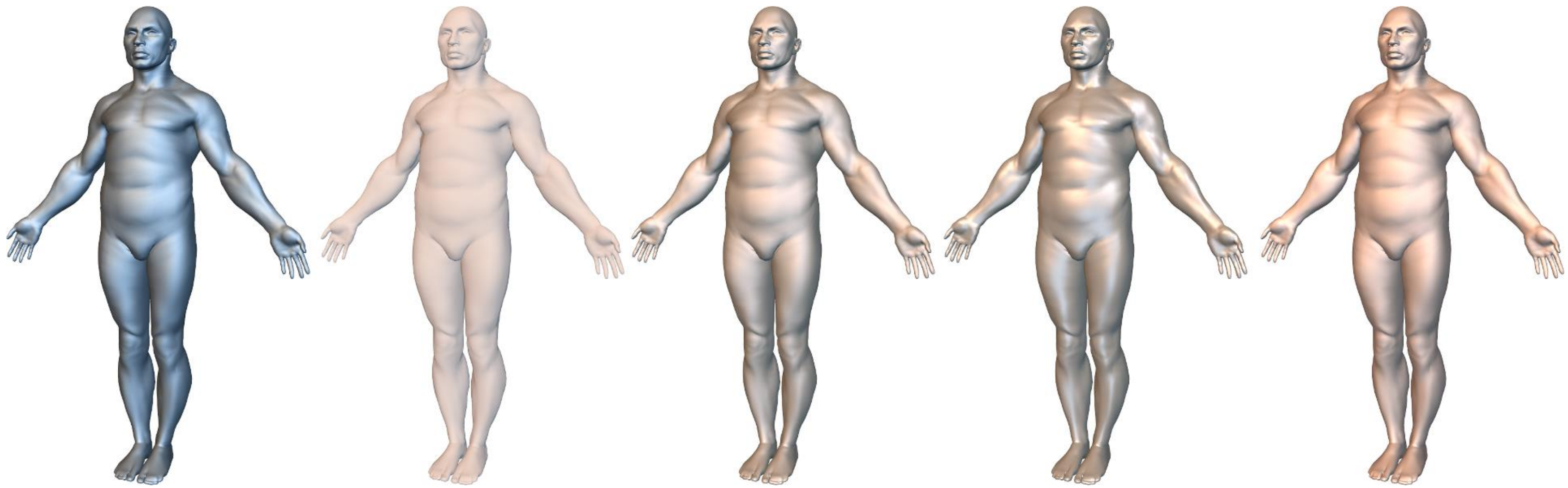


Surface Diffuse | Surface *Glossy* | Surface Spéculaire | Carte Couleur | Carte Relief

- Matériaux
- Textures
  - Variation des paramètres des matériaux sur la surface
- *Meso-* et *micro-structure* de la surface



Apparence

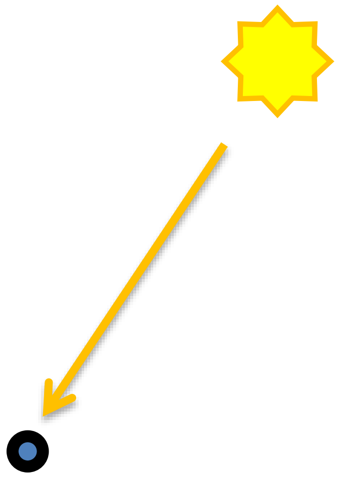


- Couleur en un point p :
  - depuis un point de vue donné
  - pour une scène donnée
- Fonction de:
  - L'**éclairage** (illumination) en p
  - La **réflectance** du matériau en p

Eclairage

Radiance en un point:

- Mesure radiométrique décrivant la quantité d’énergie passant en un point pour une direction donnée.
- Exprimée en Watt par Stéradian par Mètre-carré (W /m²sr)

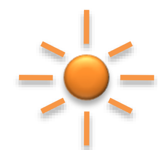


Champ de Lumière

- Ou *Light Field*
- $L(\boldsymbol{p}, \boldsymbol{\omega}), \boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{\omega} \in S^2$
- ***<u>Radiance</u>*** *(éclairement) au point* $\mathbf{p}$*, dans la direction* $\boldsymbol{\omega}$
- Résulte
  - *des sources primaires (surfaces émissives, source virtuelles),*
  - *du transport lumineux global dans la scène*
    - *éclairage indirect*

Sources Virtuelles de Lumière

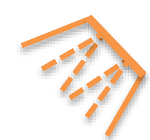
- Intensité
- Couleur:
  - En général: un triplet RVB attaché à la source
  - Modélisation physique: **spectre complet**
- Type :
  - Sources **ponctuelle** : définit par une position
    - Emet de l'énergie dans toutes les directions



  - Source **directionnelle** : une position + une direction
    - Rayons parallèles
    - Souvent utilisée pour modéliser la lumière du soleil

  - **Spot**: portion angulaire d'une source ponctuelle

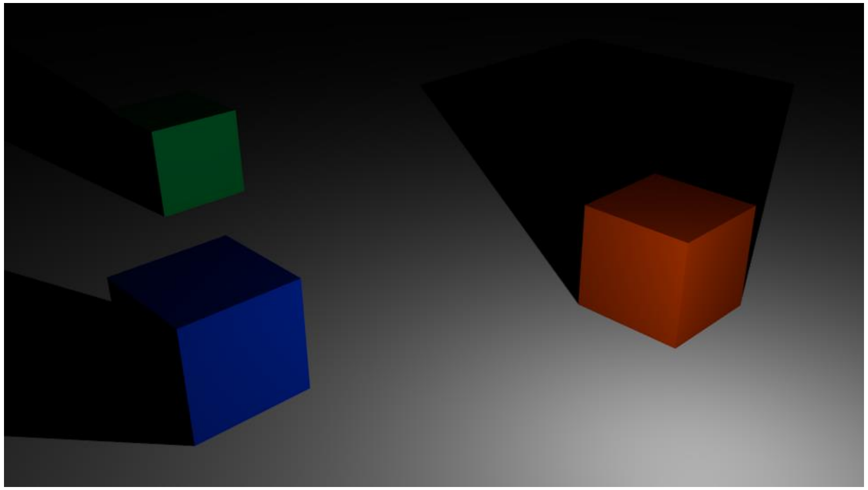
  - **Source Etendue** (*Area Light*) : un morceau de surface émettant de la lumière
    - Ombres douces
    - Peut-être défini à partir de n'importe quelle géométrie de la scène



Atténuation Lumineuse

- Modélise l'énergie reçu en un point à une distance d de la source
- Typiquement caractérisé en information par un triplet de valeurs:
  - $a_c$le coefficient d'atténuation constante
  - a_l le coefficient d'atténuation linéaire
  - a_q le coefficient d'atténuation quadratique

$$L(d) = \frac{L}{a_c + a_l \cdot d + a_q \cdot d^2}$$



Note : à l'air libre, l'atténuation d'une source ponctuelle se modélise avec $a_c = 0, a_l = 0, a_q = 1$

Environnement Lumineux

Analytique

- Ensemble de sources lumineuses

Echantillonné

- Exemple: fonction (hémi)-sphérique décrivant l'éclairage (supposé) infiniment distant
  - Capturée via un *lightprobe* ou un panorama de photos
  - En général en haute dynamique (imagerie HDR)
  - Analyse fréquentielle

Carte d’Environnement HDR

- Image sphérique à haute dynamique (HDR), approximant l’éclairage distant.
- Evaluation explicite
  - Séquences quasi-aléatoire d’échantillons,
  - Eventuellement modulée par une fonction d’importante (énergie, réflectance au point éclairé),
  - Chaque échantillon agit comme une source directionnelle.
- Projection dans une base de fonction sphérique
  - Peu de coefficients, grands composantes capturées rapidement
  - E.g., harmoniques sphériques

Equation du rendu

Intensité renvoyée

Emission intrinsèque

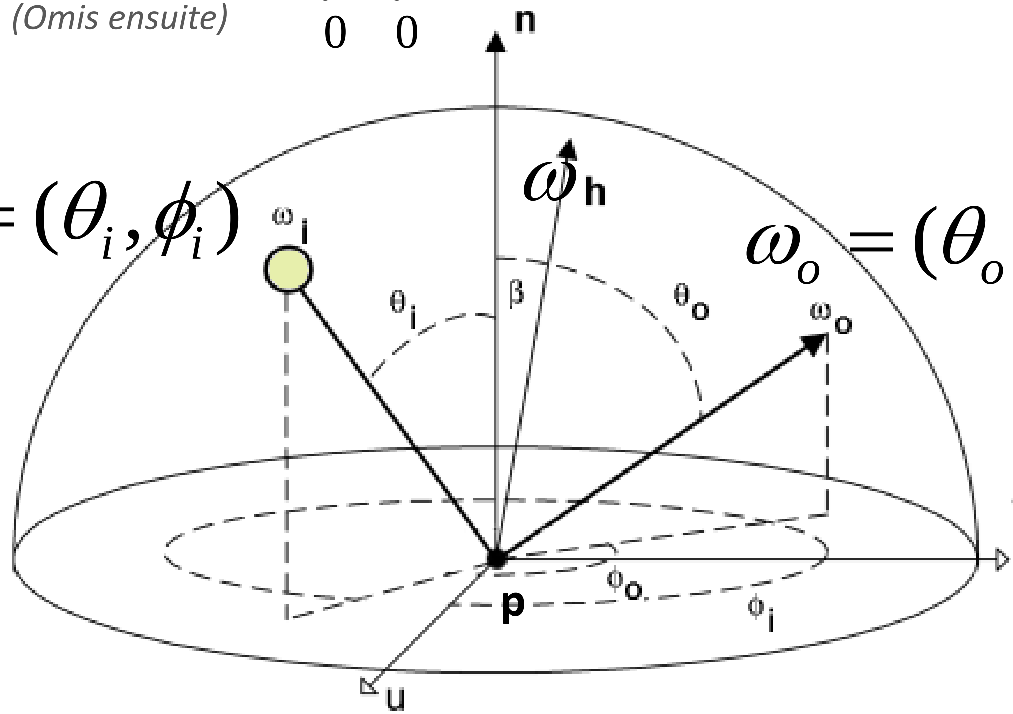
Lumière Reçu

Réflectance

$$L_o(\omega_o) = L_e(\omega_o) + \int_0^{2\pi}\int_0^{\pi/2} L_i(\omega_i)\, f(\omega_i,\omega_o)\cos\theta_i\, d\theta_i\, d\phi_i$$

(Omis ensuite)

Direction incidente de la lumière $\omega_i = (\theta_i, \phi_i)$



$\omega_o = (\theta_o, \phi_o)$ ***Direction d'émission***

$$\omega_h = \frac{\omega_i + \omega_o}{\left\|\omega_i + \omega_o\right\|}$$

Equation du rendu

Simplification pour une **Source ponctuelle** unique

$$L_o(\omega_o) = L_i(\omega_i)\, f(\omega_i, \omega_o)(n \cdot \omega_i)$$

Plusieurs sources ponctuelles

$$L_o(\omega_o) = \sum_i L_i(\omega_i)\, f(\omega_i, \omega_o)(n \cdot \omega_i)$$

Evaluation dans le cas non ponctuel (e.g. sources étendues) par la méthode de *Monte-Carlo*

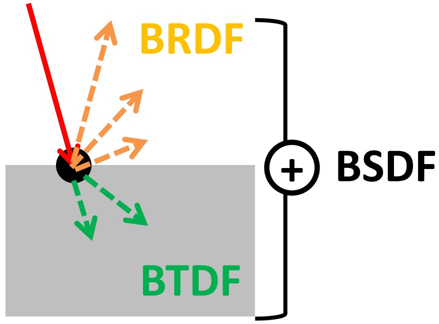
Eclairage direct

Considérer un point de surface indépendamment

- Avec ou sans ombre porté
- Pas d'échange lumineux avec les autres points de la scène
- Modélisation du matériau par une BRDF (*Bidirectional Reflectance Distribution Function*)
  - Plusieurs modèles analytiques existent
  - Paramètres
    - uniforme sur une surface
    - Ou variant et spécifié par une carte sur la surface
      - carte couleur/albedo diffus
      - Carte de brillance
      - etc

Réflectance

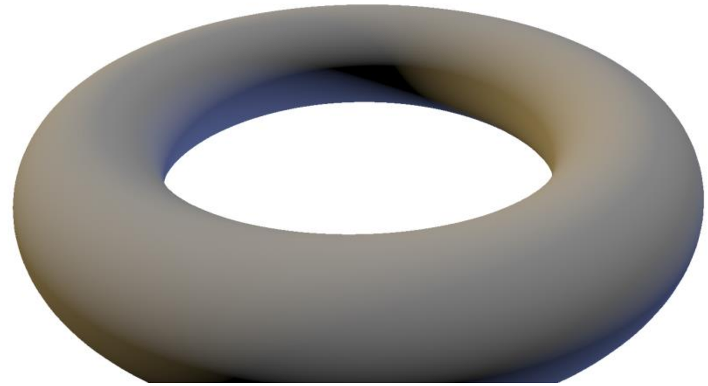
- Définit en un point par la fonction de distribution de réflectance bidirectionnelle ou **BRDF**
  - **BRDF :** composante **réflective** de la **BSDF (dispersion)**
    - Pour l'instant, on oublie la composante transmissive (**BTDF**)



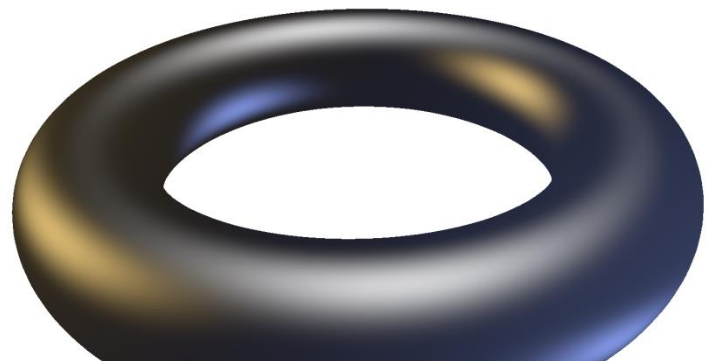
- Défini la réflexion de l'énergie reçu (radiance)
- Composantes classiques :
  - **Diffuse** : distribution de l'énergie dans toutes les directions
  - **Spéculaire/Fresnel** : réflexion directionnelle
    - Miroir : réflexion spéculaire parfaite

Compositions

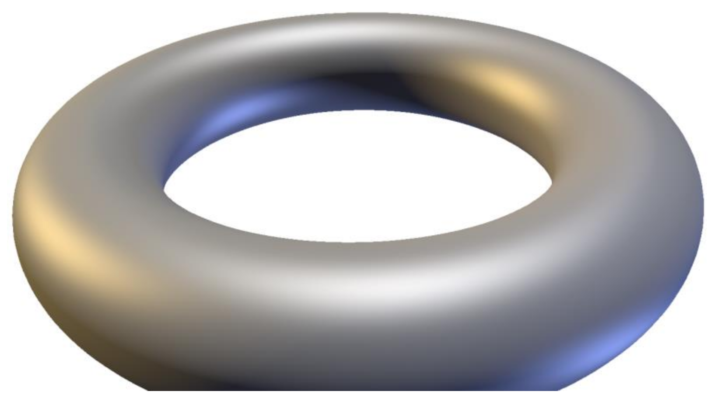
Terme diffus



Terme spéculaire



Diffus + Spéculaire



BRDF

- Définit la **micro-structure** d'un matériau dans le cadre de l'optique géométrique.
- Cas classique : une fonction à 4 dimensions

$$f : S^2 \times S^2 \rightarrow [0,1]$$
$$\omega_i \times \omega_o \rightarrow r$$

$$\omega_i = (\theta_i, \phi_i) \quad \omega_o = (\theta_o, \phi_o) \quad \omega_h = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$$

Lumière incidente ***Direction d'émission*** ***HalfVector***

- Valeur en inférieur ou égale à 1

$$L_o(\omega_o) = L_i(\omega_i) f(\omega_i, \omega_o)(n \cdot \omega_i)$$

BRDF Physiquement Plausibles

- Respecte la **réciprocité d'Helmotz**

$$f(\omega_i, \omega_o) = f(\omega_o, \omega_i)$$

- **Conservative**

$$\int_\Omega f(\omega_i, \omega_o) \cos\theta_o d\omega_o \leq 1$$

- **Positivité**

$$f(\omega_i, \omega_o) \geq 0, \forall\, \omega_i, \omega_o$$

Propriétés

- Isotropie/Anisotropie
- Nombre réduit de paramètres
- Séparabilité « diffus/spéculaire »

$$f(\omega_i, \omega_o) = f^d(\omega_i, \omega_o) + f^s(\omega_i, \omega_o)$$

- Evaluable par la méthode de Monte-Carlo

Note sur la couleur

- Pour l’instant, on considère le cas où la BRDF s’applique aux trois canaux couleur de la même manière
- Albedo diffuse : couleur de base du matériau
- Couleur spéculaire :
  - Modèles physiques : dépend du coefficient de Fresnel et de la conductivité du matériau
  - Modèles empirique : spécifiée indépendamment

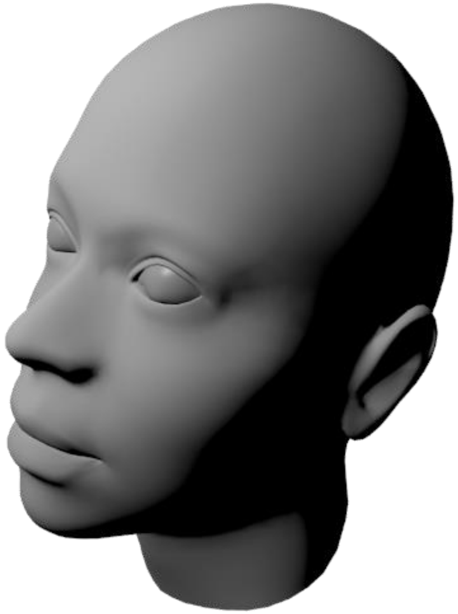
BRDF diffuse

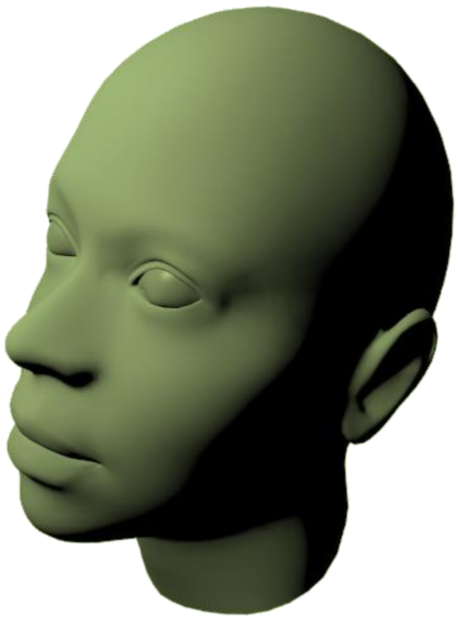
- Modèles de Lambert

$$f^{d}(\omega_i,\omega_o)=\frac{k_d}{\pi}$$

Coefficient Diffus

- Standard
- Indépendant du point de vue
- Réutilisé dans la plupart des autres modèles, qui se concentrent sur les réflexions spéculaires dépendante du point de vue.





BRDF de Phong

- Terme spéculaire

Brillance

$$f^s(\omega_i, \omega_o) = k_s (r \cdot \omega_o)^s$$

$$r = 2n(\omega_i \cdot n) - \omega_i$$

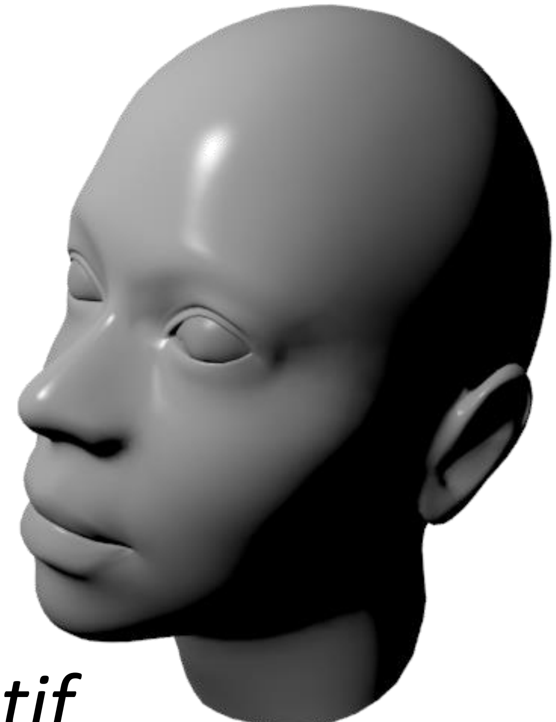
Coefficient de spécularité

- Indice de **brillance** (*shininess*)
- Terme diffus : Lambert
- *Modèle empirique non conservatif*

BRDF de Blinn-Phong

- Modèle de Phong modifié

$$f^s(\omega_i, \omega_o) = k_s (n \cdot \omega_h)^s$$



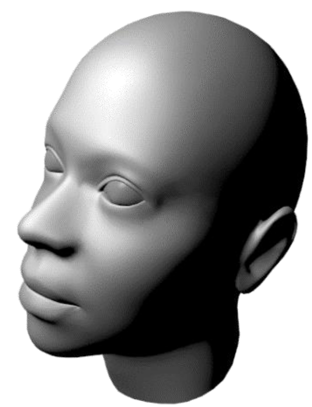
- Simple, efficace
- *Modèle empirique non conservatif*
- Normalisé en 2008

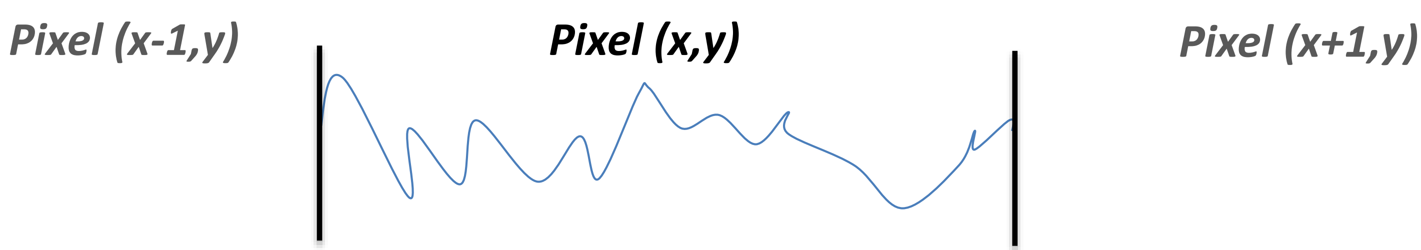
Modèles à Micro-Facettes

Distribution de microfacettes ***Terme de Fresnel*** ***Terme Géométrique***

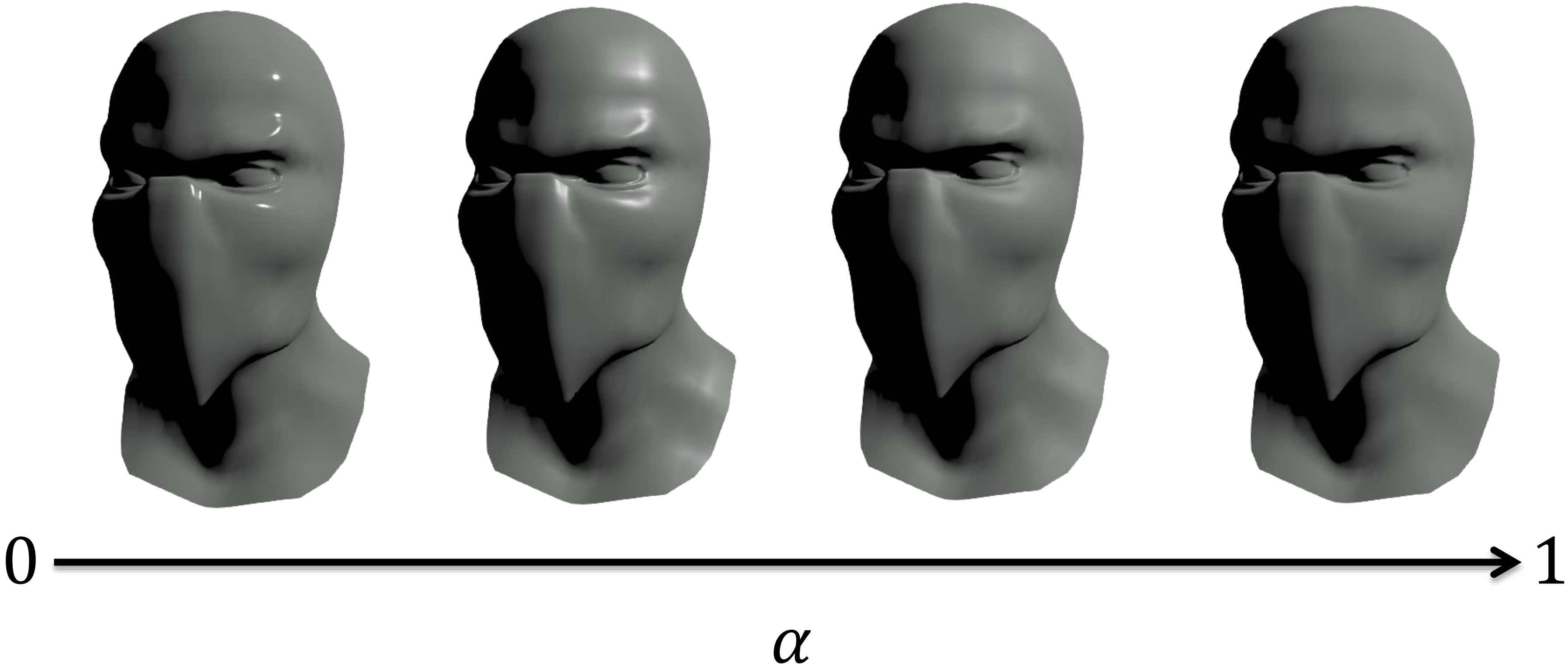
$$f^s(\omega_i, \omega_o) = \frac{D(\omega_h)F(\omega_i, \omega_h)G(\omega_i, \omega_o, \omega_h)}{4(n \cdot \omega_i)(n \cdot \omega_o)}$$

- Modèle statistique de la micro-géométrie
- $\alpha \in [0,1]$ la rugosité du matériau
  - En pratique souvent élevée au carrée
- Caractérisation géométrique





Rugosité



Distribution de micro-facettes

- D : Modèles la **distribution de normales** de la surface à l'échelle microscopique
- Exemple :
  - Distribution de Beckmann (BRDF Cook-Torrance, 1981-1982)
  - Distribution GGX/Trowbridge-Reitz
    - Et variantes
- Idéalement : F et G doivent être dérivés de D

Distribution de Beckmann

$$D(\omega_h) = \frac{1}{\pi\alpha^2(n \cdot \omega_h)^4} e^{\frac{(n\cdot\omega_h)^2 - 1}{\alpha^2(n\cdot\omega_h)^2}}$$

- Employée par la **BRDF de Cook-Torrance**
- Distribution des ondes électromagnétiques
- Déterministe
- Supposition :
  - toutes les facettes ont la même aire
  - Toute facette a une facette symétrique par rapport à la normale

Distribution GGX

Introduit par Trowbridge et Reitz (1975), généralisé par Burley (2012)

$$D(\omega_i, \omega_o) = \frac{{\alpha_p}^2}{\pi\left(1 + \left({\alpha_p}^2 - 1\right) \cdot (n \cdot \omega_h)^2\right)^2}$$

Standard industriel (Disney, Unreal Engine, etc)

Terme de Fresnel

Fraction réfléchi de la lumière incidente pour une surface plate.

Distingue les matériaux et conducteurs et diélectriques

Approximation de Schlick [1993]

$$F(\omega_i, \omega_h) = F_0 + (1 - F_0)(1 - max(0, \omega_i \cdot \omega_h))^5$$

Avec $F_0 \in \mathbb{R}$ l’indice de réfraction de Fresnel, dépendant du matériau (. Exemples :

- Plastique (diélectrique) : 0.02 à 0.05
- Aluminium (conducteur) : [0.91, 0.92, 0.92], « reflet coloré », variance significative selon la longueur d’onde

Terme Géométrique

- Modélise les effets d ‘ombrage et de masquage des micro-facettes entre elles
- Dépend de la rugosité et de la distribution D

Terme Géométrique Cook-Torrance

Distingue les effets de masquage et de l'ombrage inter-facettes

$$G = \min[1, \frac{2(n\cdot\omega_h)(n\cdot\omega_i)}{(\omega_o\cdot\omega_h)}, \frac{2(n\cdot\omega_h)(n\cdot\omega_o)}{(\omega_o\cdot\omega_h)}]$$

Ombrage ***Masquage***

Terme Géométrique GGX

Terme géométrique associé à la distribution de micro-facettes GGX

$$G^{Smith}(\omega_i, \omega_o) = G_1^{Smith}(\omega_i) G_1^{Smith}(\omega_o)$$

$$G_1^{Smith}(\omega) = \frac{2(n \cdot \omega)}{n \cdot \omega + \sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot \omega)^2}}$$

Approximation de Schlick :

$G_1^{Schlick}(\omega) = \frac{(n \cdot \omega)}{(n \cdot \omega)(1-k)+k}$ avec k $= \alpha \sqrt{\frac{2}{\pi}}$

Remarque

- Dans le cas purement physique : F et G doivent être dérivés de D
- Formelle, dans le cas des modèles à microfacettes,
  - $f(\omega_i, \omega_o) := \cancel{f^d(\omega_i, \omega_o) +} f^s(\omega_i, \omega_o)$
  - Mais : on garde en pratique le terme diffus car
    - il permet de compenser empiriquement beaucoup de phénomène non pris en compte dans par les microfacettes
    - Par convention, de nombreuses technologies de rendu le maintiennent

Matériaux métalliques

- Conducteurs
- Couleur réfléchie modulée par le matériau
  - En pratique si matériau métallique : réflexion avec la couleur de base
  - Sinon : la lumière incidente conserve sa teinte

« Un » modèle pour le rendu basé - physique

- Matériau « PBR » (*Physically Based Rendering*)
  - Albedo (couleur diffuse moyenne)
  - Rugosité (entre 0 et 1)
  - « Metallique »
    - valeur binaire (conducteur ou pas)
- Voir le modèle plus complet proposé
  - *Physically-based Shading*, *SIGGRAPH*

Couches de BRDF

- Nécessaire en pratique pour modéliser de nombreux matériaux
- Reproduit l'empillement de matériaux semi-transparents.

Couche à dominante diffuse

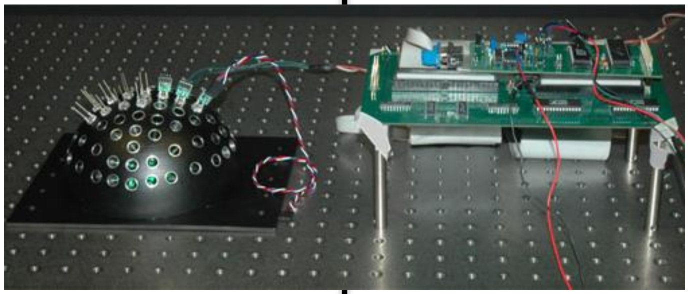
Couche à dominante spéculaire

Empillement

BRDF Basée-Donnée

- Echantillonnée à partir d'un véritable matériel
- Pas de forme analytique simple
- Représentée par un nombre réduit de coefficients une fois projetée une base de fonctions spécifique :
  - harmoniques sphériques
  - ondelettes

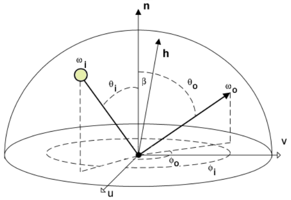
Echantillon de matériau réel



Données 4D

Projection base de fonctions

Ensemble de coefficients



Rendu

TEXTURES

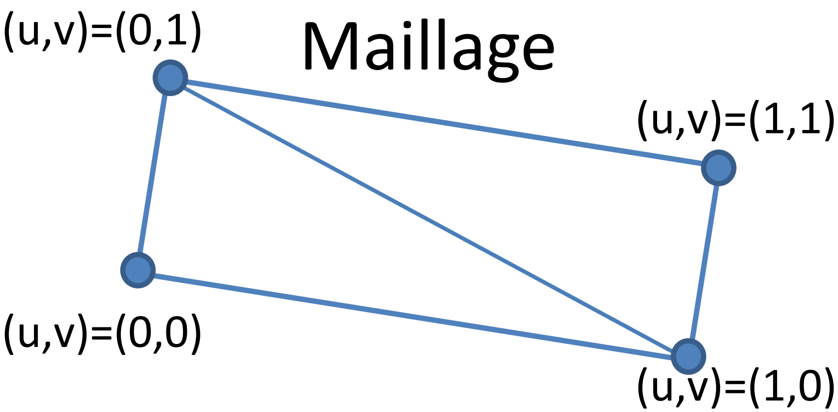
Définition

- Une carte de valeurs (*map*)
  - Albedo, définit une distribution de points colorés sur les surfaces (*color map*) ; autre paramètre de la BRDF aussi modulables via une carte (rugosité, brillance)
  - Normal, définit une distribution de vecteur normaux sur une surfaces (*normal map*)
  - Déplacement, définit une distribution de vecteurs de déplacements sur une surface (*displacement map*)
- Textures 2D (carte scalaire ou vectorielle) plaquée sur une surface 3D via sa *paramétrisation UV*

Plaquage de texture

Texture couleur (image 2D RGB)

Maillage



Maillage texturé

Coordonnées paramétriques

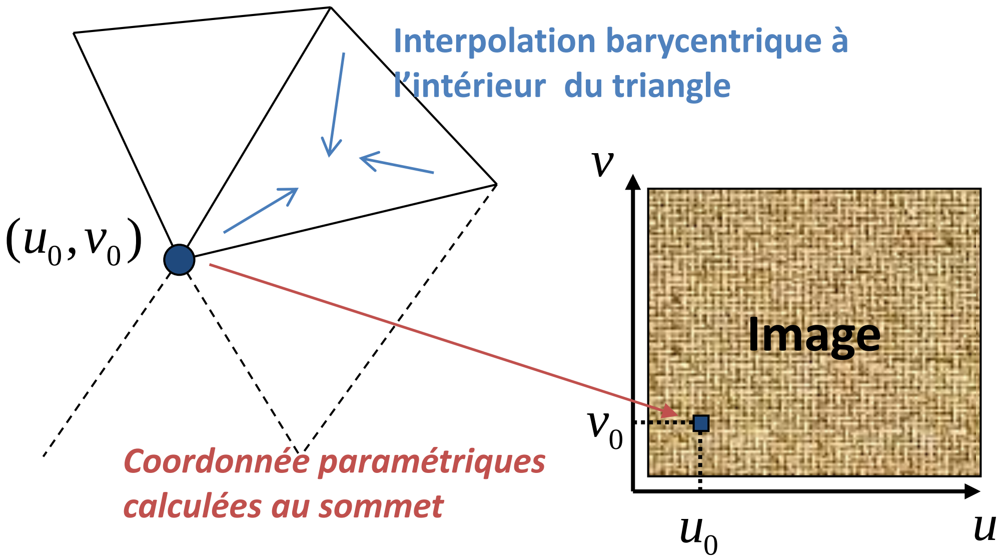
a.k.a « Coordonnées de texture »,
a.k.a « Coordonnnées uv »

$$(u,v) \in \Re^2$$ par convention: $$(u,v) \in [0,1]^2$$

Définition d'une propriété de surface à partir d'un fonction bivariée:

$$f:$$
$$\Re^2 \to \Re^n$$
$$u,v \to c$$

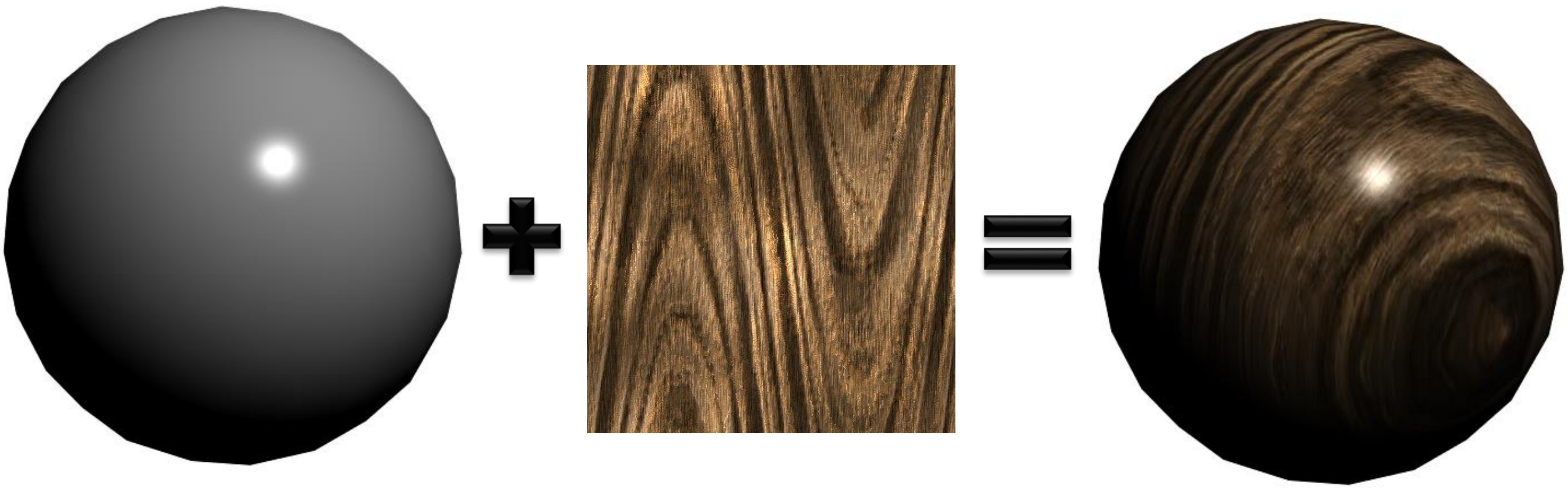
Exemple : valeur d'un pixel dans une images (« texture mapping »)



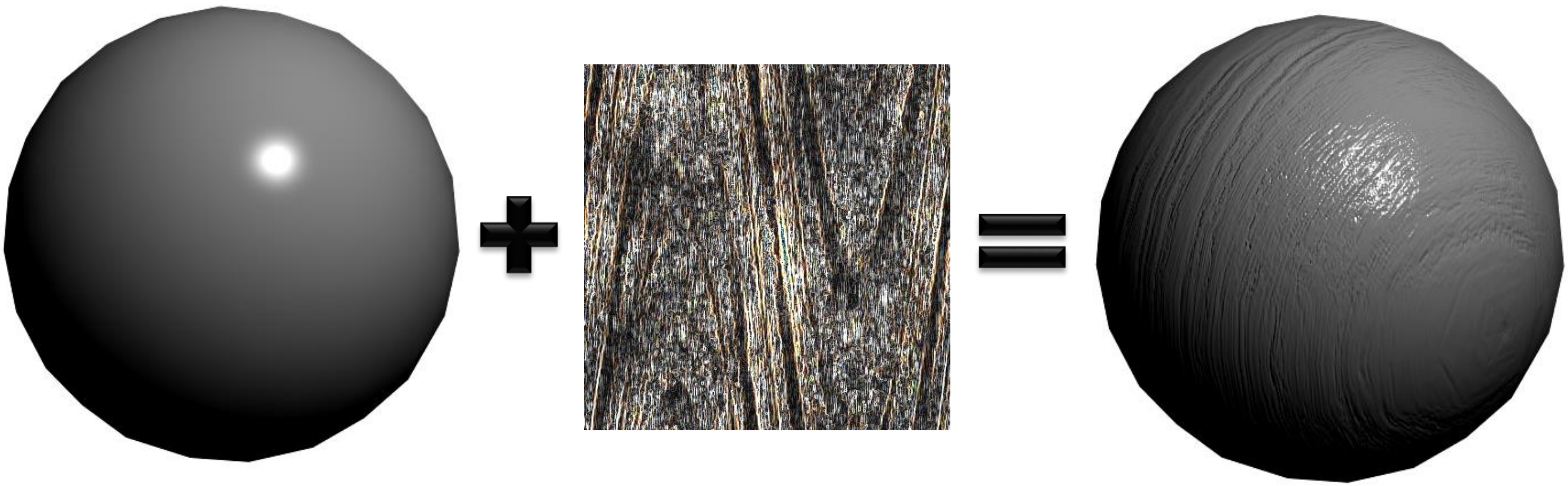
*Définition de coordonnées paramétriques continues sur l'ensemble des sommets d'un maillage : **Paramétrisation***

Exemples d'algorithme de paramétrisation: LSCM, Floater

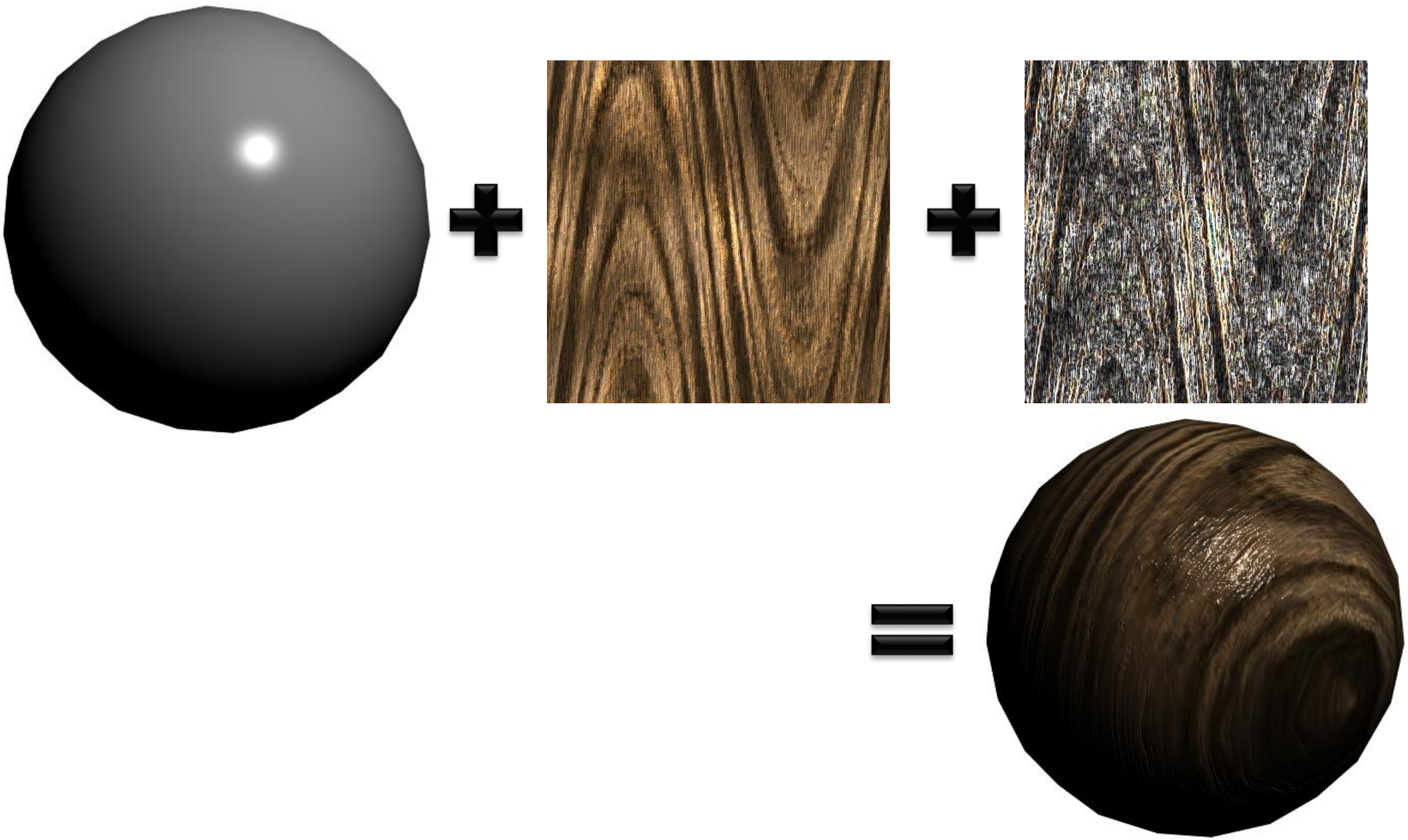
Texture couleur



Texture normale

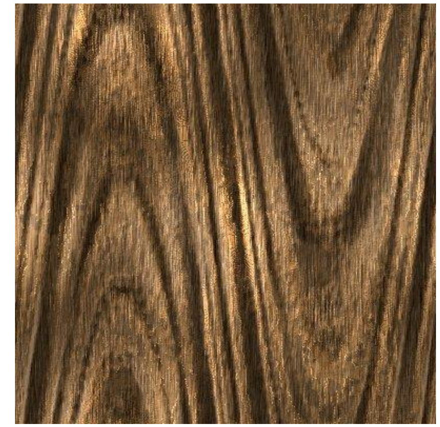


Couleur et normale



Type de textures

Texture 512x512 RGB



- 1D, 2D, 3D, etc
  - scalaire, vectorielle,
  - couleur, alpha, paramètre de BRDF
- 2 types principaux
  - Carte
  - Procédural

Carte de Texture

- « Image » prédéfinis, artificielle ou capturée (photo)
- Coût mémoire parfois élevé
  - Compression/décompression à la volée
- Nécessite un filtrage pour éviter les effets d'aliasing, de moiré, etc
- Evaluation efficace :
  - Accès mémoire + filtrages
- Pré-calculs possibles
  - Traitement d'image (rehaussement, débruitage, etc)
  - Calcul d'une pyramide (mip-map)
  - Quantification et compression

Texture Procédurale

- Une fonction évaluable en tout point d'un domaine
- Léger en mémoire : code la fonction + paramètres
- Couteux à l'évaluation
- A priori, une infinité de fonctions possible, mais quelques propriétés sont souhaitables
  - Déterministe
  - Variation naturelle (pseudo-aléatoire)
  - Faible nombre de paramètres intuitifs
  - Evaluations dé-corrélées les unes des autres
    - Calcul parallèle / GPU
- Exemples : **bruit de Perlin**, bruit en ondelette, bruit de Gabor

Bruit de Perlin

- Introduit par Ken Perlin en 1985
- Bruit de gradient
- Donne une apparence pseudo-aléatoire aux surfaces
- Multi-échelle : sommes de bruit de Perlin à plusieurs échelles