



Rendu Par Points

Tamy Boubekeur

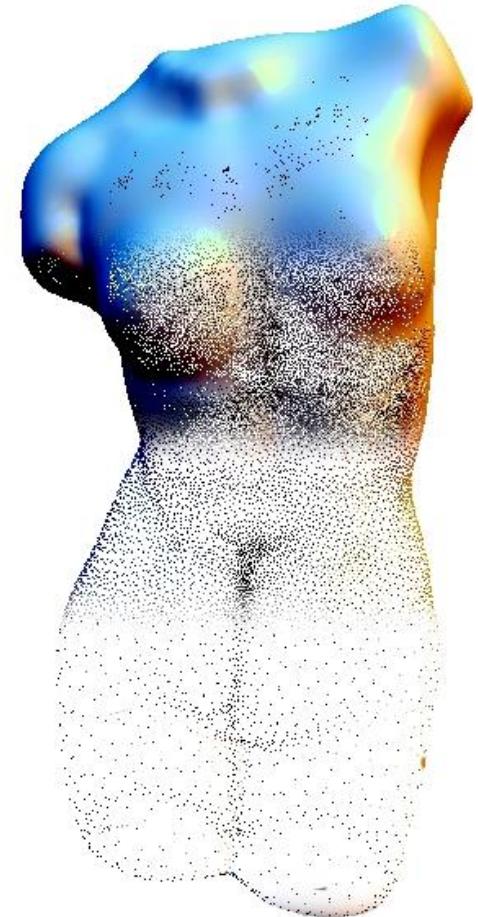
Adobe & Institut Polytechnique de Paris

Sommaire

1. La Synthèse d'images
2. Surfels et splatting
3. Rendu par polygonisation
4. Lancer de rayon
5. Structures de données

Rendu Par Points

- Entrée:
 - nuage de points 3D
 - Attributs par point : position, normales, couleur, coefficients de réflectance, etc
- Problème : la probabilité qu'un rayon de lumière intersecte un point est nulle
- Rendu par points : de multiple solutions à ce problème.



Algorithmes

- Rendu par projection (*rasterization*) directe
 - *Surface Splatting*
- Rendu par polygonisation intermédiaire
 - Patches
 - *Triangle Fans*
 - *Surfel Strips*
- Rendu par lancer de rayon
 - Fondé sur l'opérateur MLS et la *surface de points* qu'il défini

LA SYNTHÈSE D'IMAGES

Synthèse d'Image

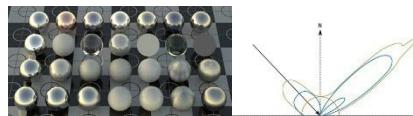
Modèles



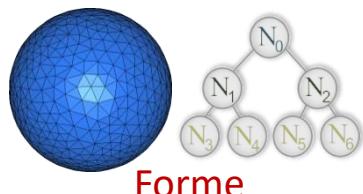
Capteur



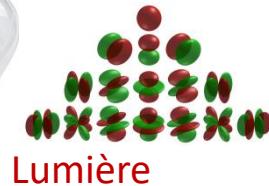
Mouvement



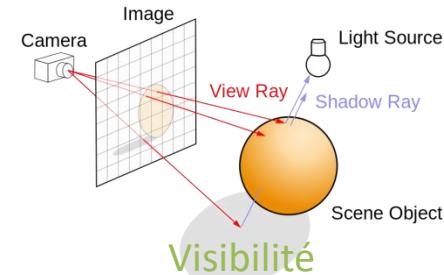
Apparence



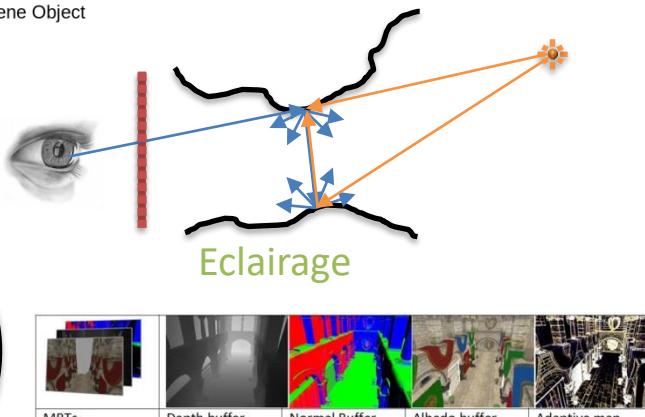
Forme



Lumière



Algorithmes



Post-processing

Input Assembler
Vertex Shader
Control Shader
Tessellator
Evaluation Shader
Geometry Shader
Rasterization
Fragment Shader
Output Manager

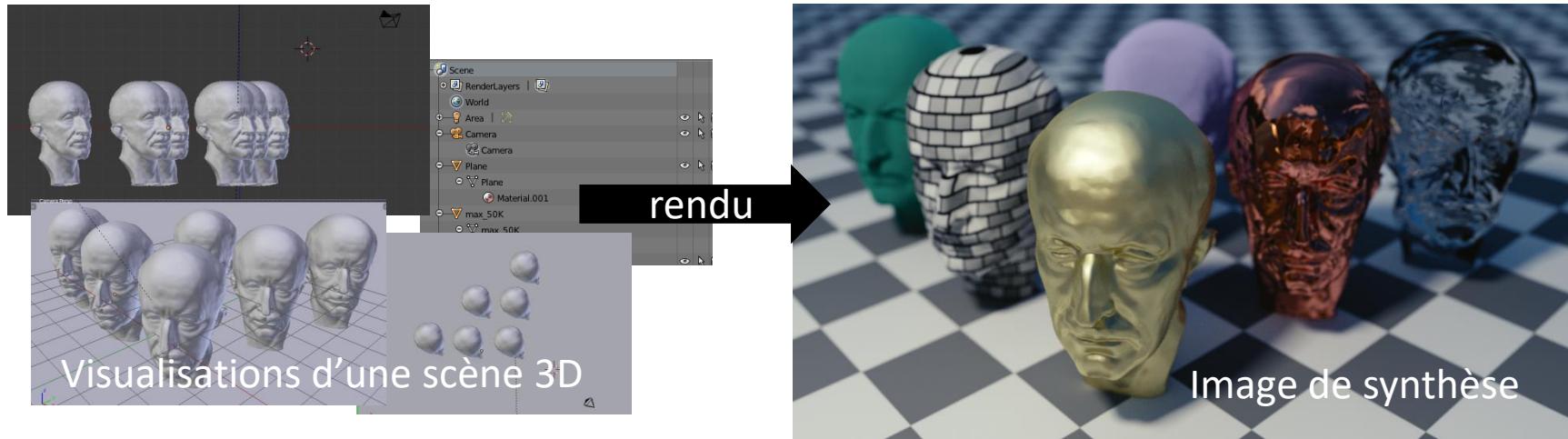
Pipeline GPU

Rendu Temps-Réel



Implémentations

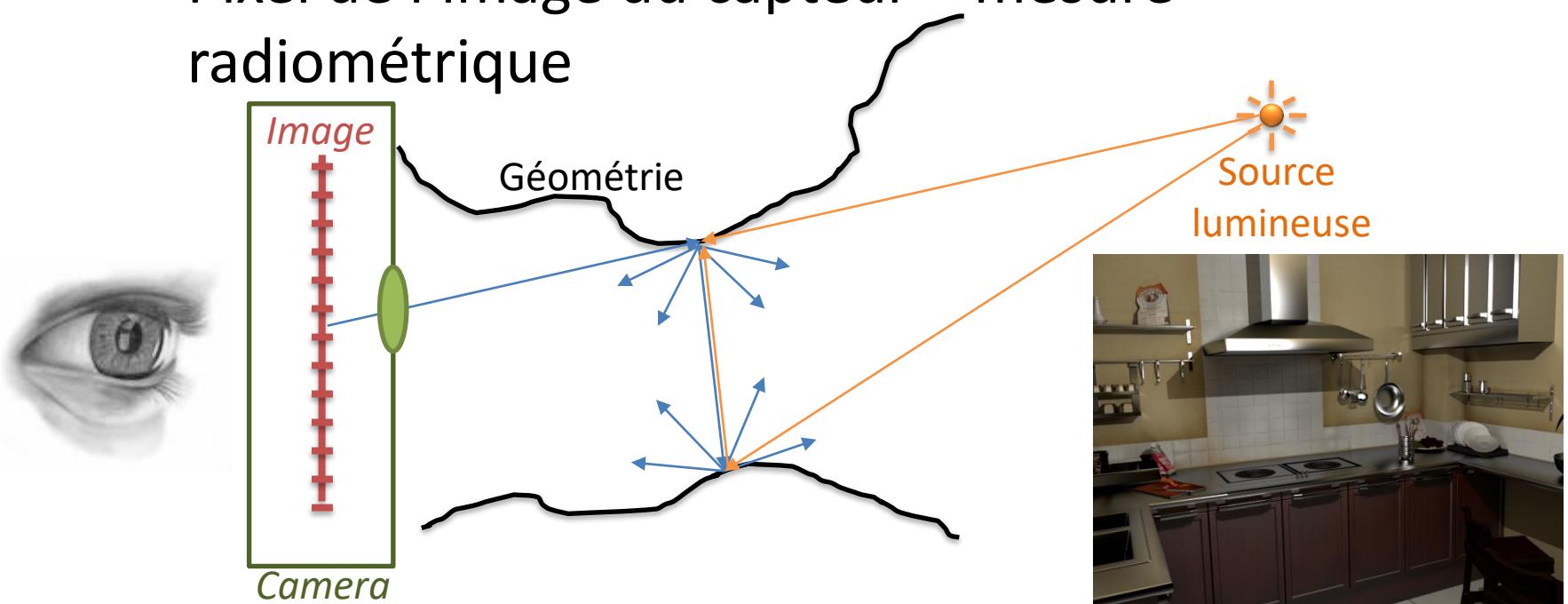
Un processus de simulation



- Rendu = Synthèse d'Image
- Génération d'une image numérique à partir d'une scène 3D
- Réaliste (physiquement plausible) ou expressif

Rendu Fondé sur la Physique

- Simulation du transport de la lumière
 - De la source au capteur
 - Dans une scène virtuelle
 - Pixel de l'image du capteur = mesure radiométrique



Rendu Temps Réel

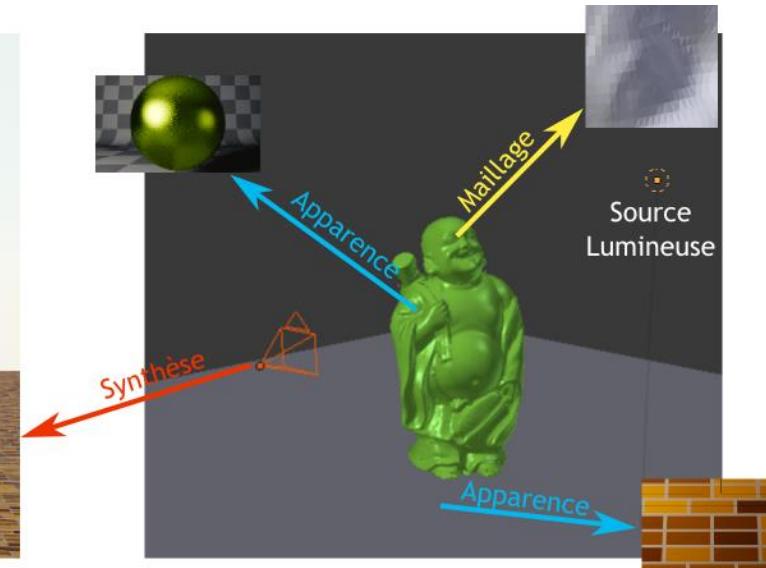
- Pour les systèmes interactifs
- Approximation géométrique et radiométrique de la scène
- Calcul parallèle (GPU)
- APIs OpenGL, DirectX, Vulkan, Metal

Modèles de Scène 3D



Image Numérique

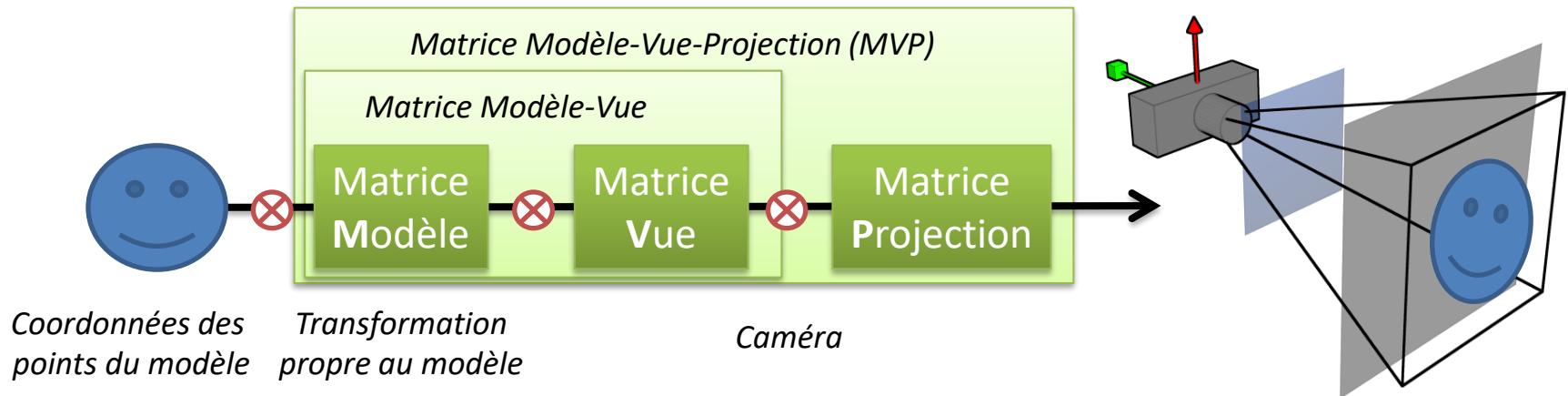
- Une collection de modèles :
 - Capteur (caméra)
 - Géométries
 - Maillages, particules, iso-surfaces, etc
 - Apparence
 - Matériaux, textures
 - Lumières
 - Animation
 - Évolution temporelles des paramètres



Scène 3D

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

Transformation et Projections



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

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

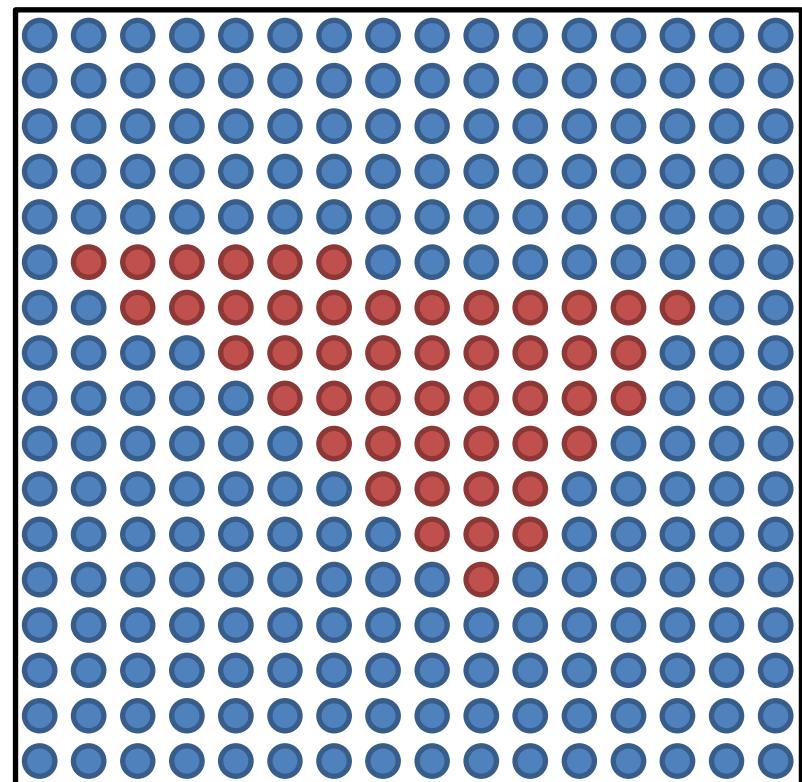
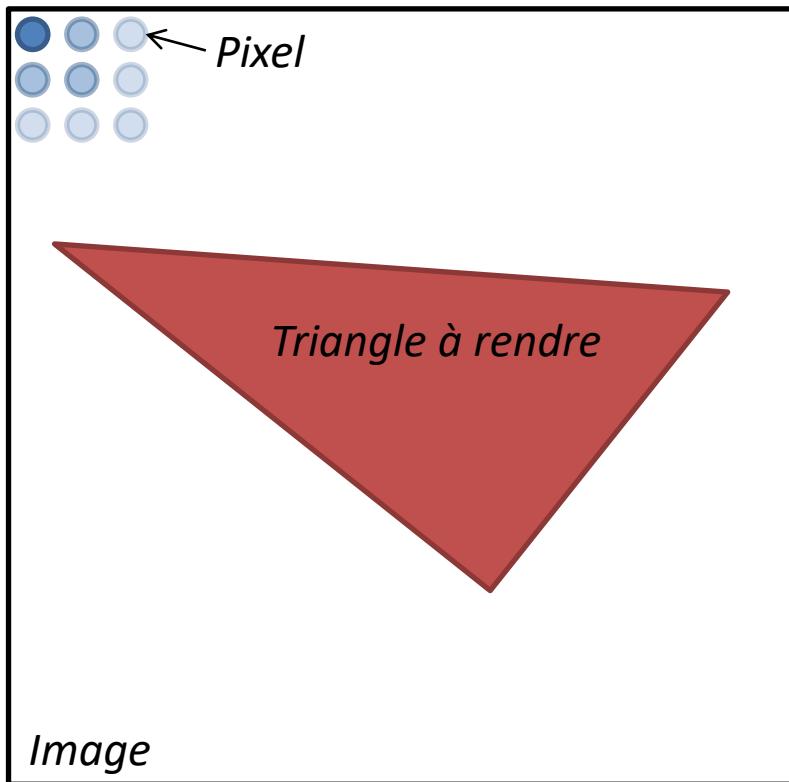
Lancer de rayon

Ray casting/tracing

- Intersection rayon/triangle
- Récursion

Rasterisation

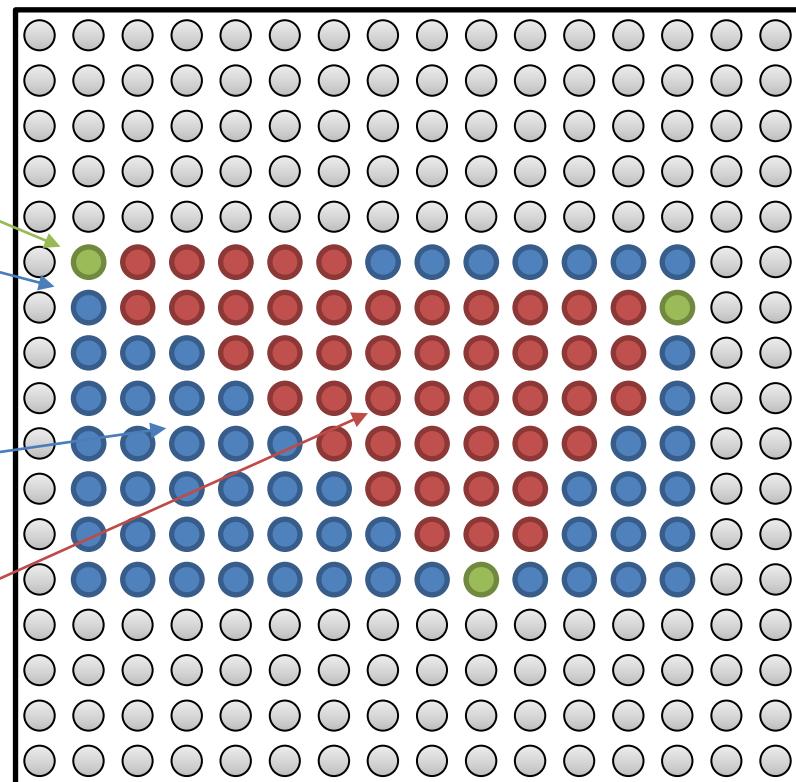
- Discrétisation d'un polygone dans une grille (image)
 - Plusieurs algorithmes alternatifs
 - 1 triangle > n **fragments**
- **Fragment = {x,y}**
 - **Pixel** = fragment visible
 - Plusieurs triangles peuvent couvrir les même pixel
 - Visibilité déterminé plus tard



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 $\{\min X, \min Y, \max X, \max Y\}$ des pixels candidats
3. Calcul des coordonnées barycentriques $\{b_0, b_1, b_2\}$ de ceux-ci en fonction du triangle
4. Classification des pixels effectivement couverts
 - $b_0, b_1, b_2 \geq 0$ et $b_0 + b_1 + b_2 = 1$
5. Emission des fragments correspondant



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 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)

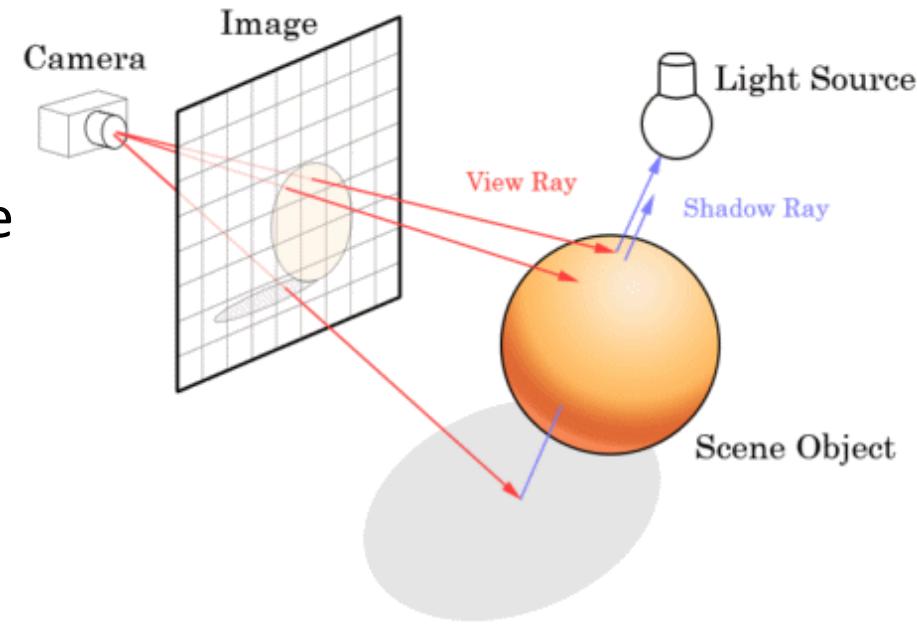
- 😊 Rapide, support GPU, linéaire en temps
- 😢 Plusieurs polygones par pixel / aliasing

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
```



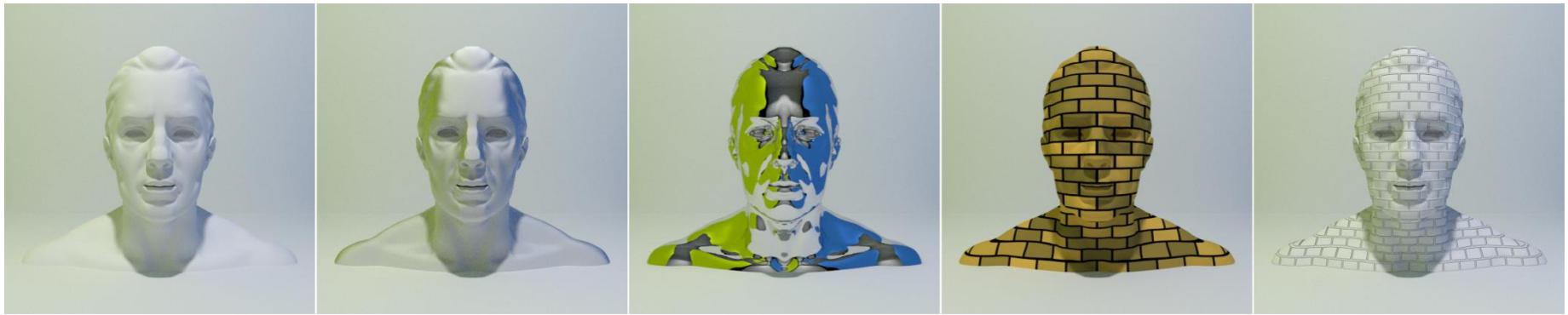
Accélération:

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

😊 Simple, facilement généralisable

😢 Couteux

Apparence



Surface Diffuse

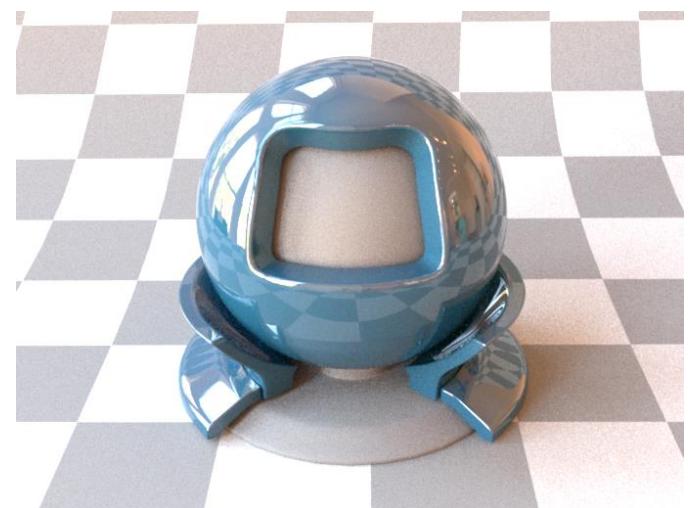
Surface Glossy

Surface Spéculaire

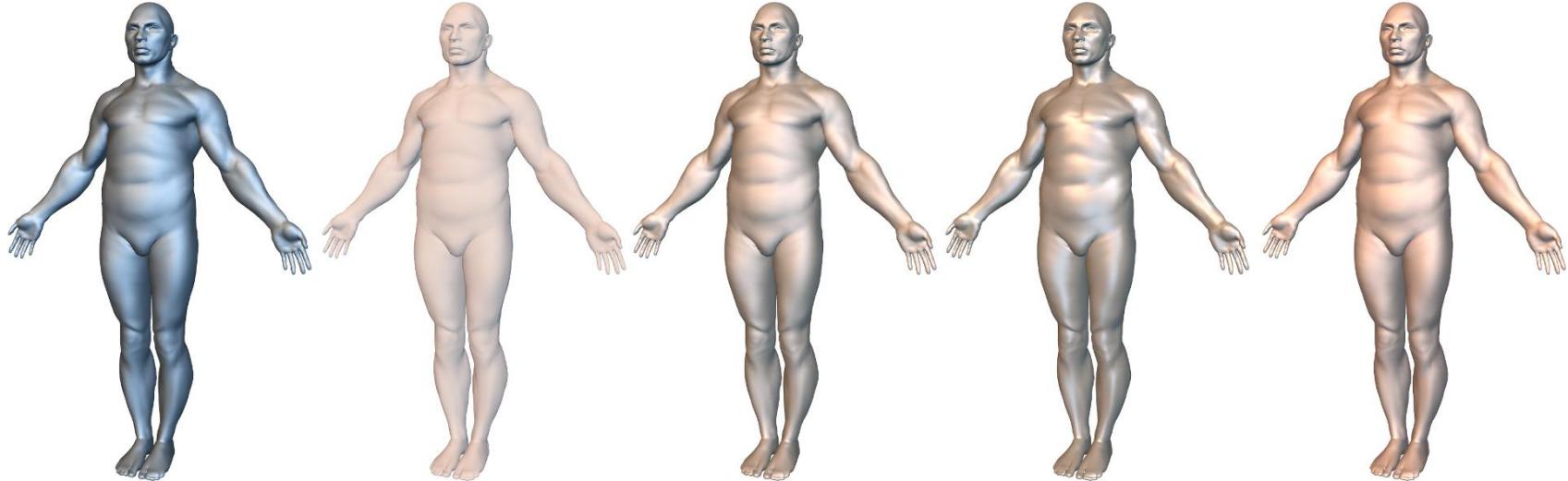
Carte Couleur

Carte Relief

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



Apparence

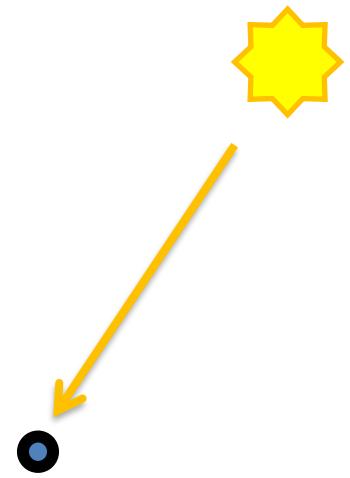


- Couleur en un point p :
 - depuis un point de vue donné
 - pour une scène donnée
- Fonction de:
 - L'**éclairage** (illumination) en p
 - La **réflectance** du matériau en p

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 ($\text{W} / \text{m}^2\text{sr}$)



Champ de Lumière

- Ou *Light Field*
- $L(p, \omega), p \in \mathbb{R}^3, \omega \in S^2$
- **Radiance** (*éclairement*) au point p , dans la direction ω
- 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



Equation du Rendu

*Intensité
renvoyée*

$$L_o(\omega_o)$$

*Emission
intrinsèque*

$$L_e(\omega_o) + \int_0^{2\pi} \int_0^{\pi/2}$$

(Omis ensuite)

*Lumière
Reçu*

$$L_i(\omega_i) f(\omega_i, \omega_o) \cos \theta_i d\theta_i d\phi_i$$

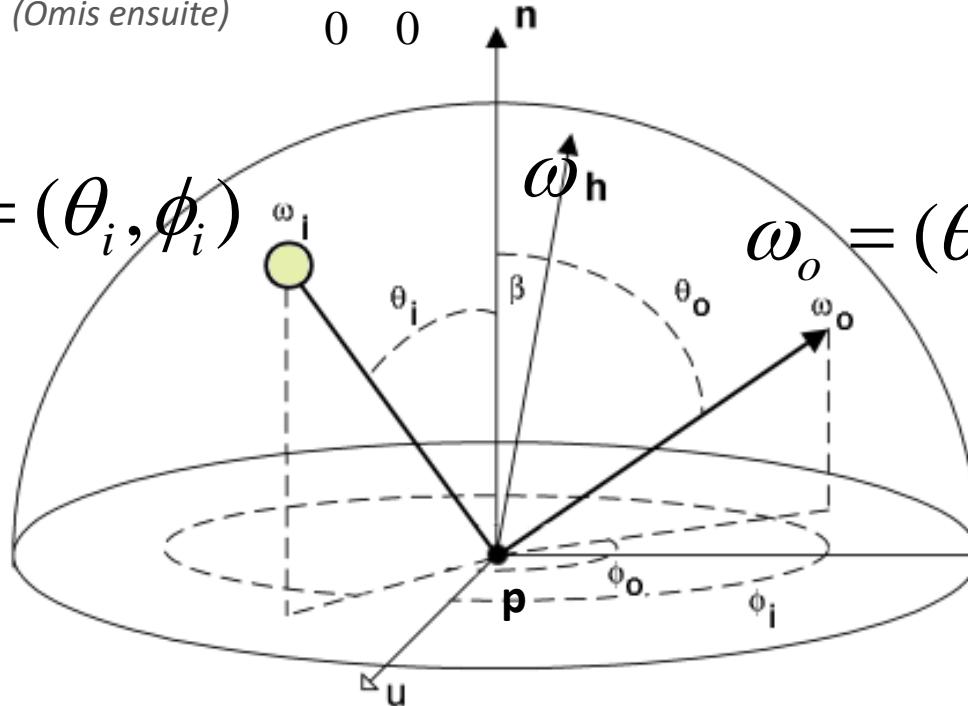
Réflectance

*Direction
incidente de
la lumière*

$$\omega_i = (\theta_i, \phi_i)$$

*Direction
d'émission*

$$\omega_o = (\theta_o, \phi_o)$$



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

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*

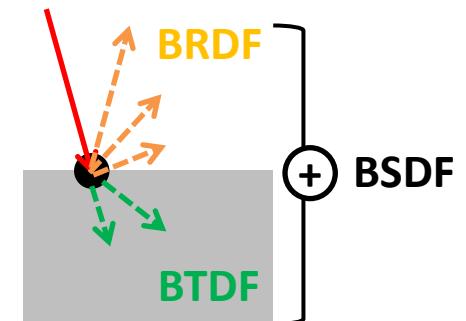
Eclairage Direct

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 (**diffusion/dispersion**)
 - Pour l'instant, on oublie la composante transmissive (**BTDF**)



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

BRDF

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

$$f: S^2 \times S^2 \rightarrow \mathbb{R}^+$$

$$\omega_i \times \omega_o \rightarrow r$$

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

Lumière incidente *Direction d'émission* *HalfVector*

Propriétés Désirables

- **Réciprocité**

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

- **Conservation**

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

- **Positivité**

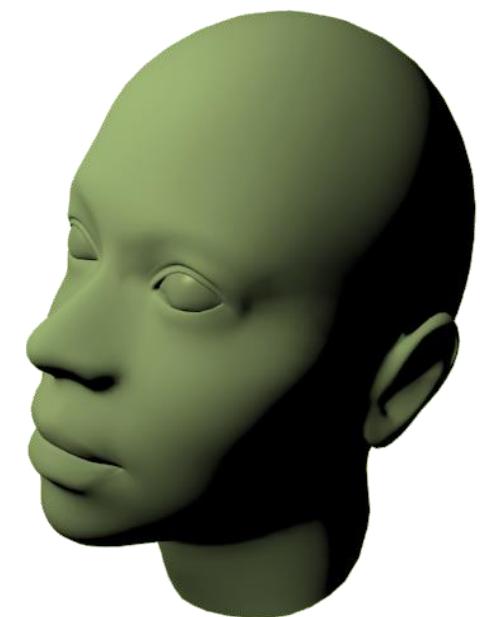
$$f(\omega_i, \omega_o) \geq 0, \forall \omega_i, \omega_o$$

BRDF diffuse

- Modèles de Lambert

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

- 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.

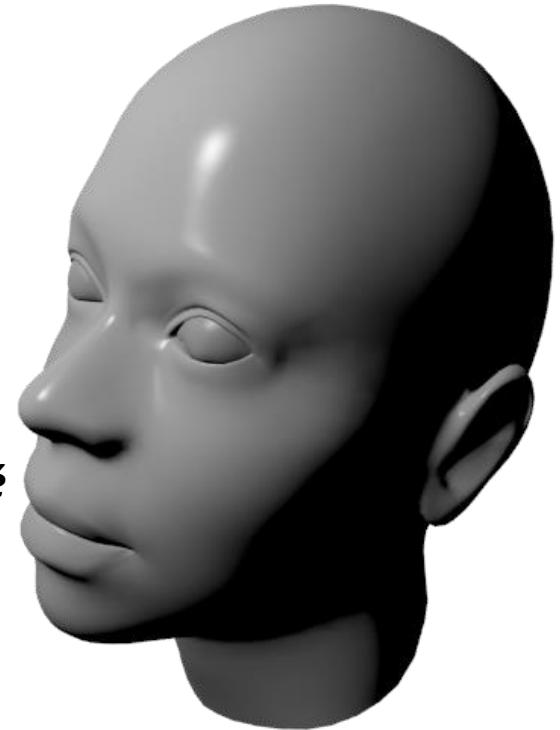


BRDF de Phong

- Terme spéculaire

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

Brillance
↓
 $r = 2n(\omega_i \cdot n) - \omega_i$
↑
Coefficient de spécularité



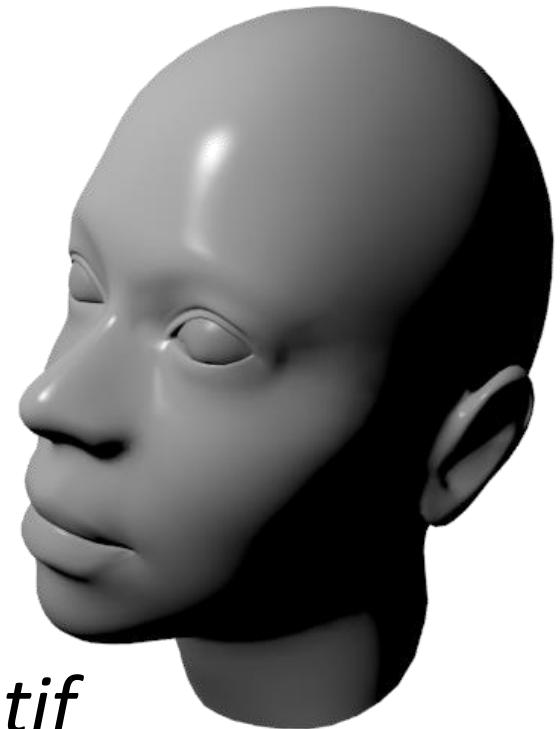
- Indice de **brillance** (*shininess*)
- Terme diffus : Lambert
- *Modèle empirique non conservatif*

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

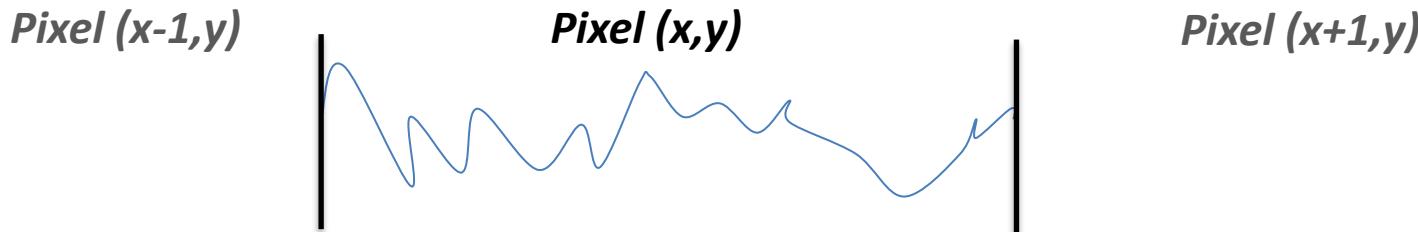
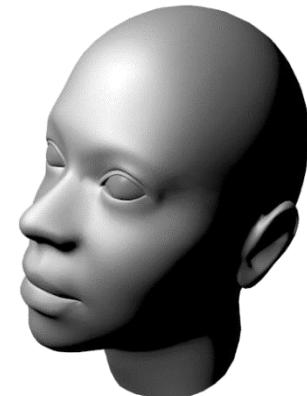


Modèles à Micro-Facettes

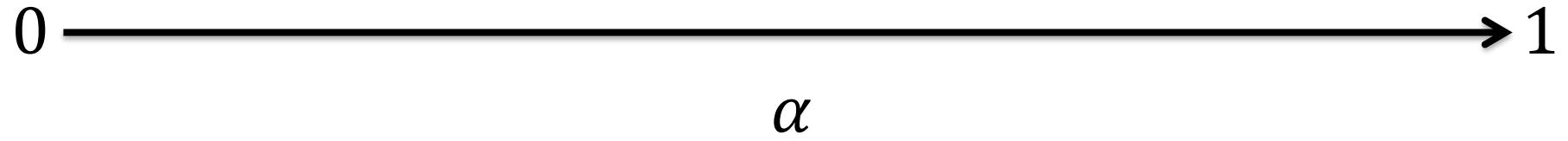
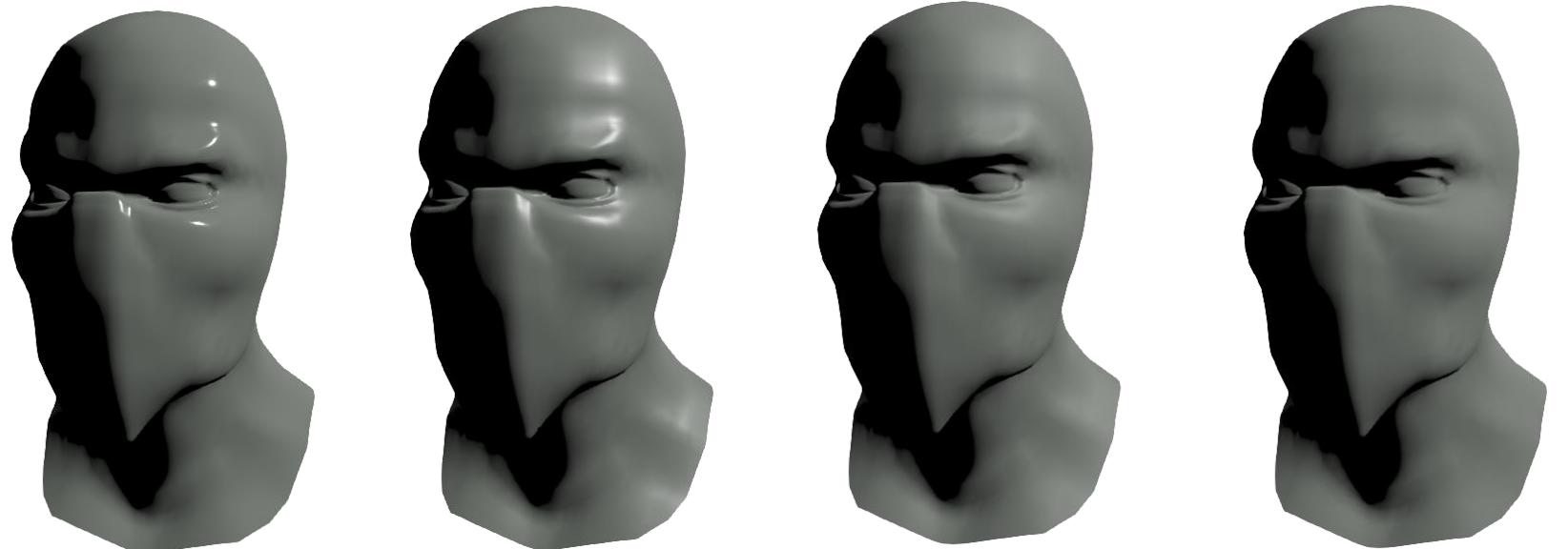
$$f^s(\omega_i, \omega_o) = \frac{D(\omega_h)F(\omega_i, \omega_h)G(\omega_i, \omega_o, \omega_h)}{4(n \cdot \omega_i)(n \cdot \omega_o)}$$

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

- Modèle statistique de la micro-géométrie
- $\alpha \in [0,1]$ la rugosité du matériau
 - En pratique souvent élevée au carré
- 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 utilisée dans l'étude des ondes électromagnétiques
- Déterministe
- Suppositions :
 - 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)

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

Standard industriel (Disney, Unreal Engine, etc)

Terme de Fresnel (F)

- Fraction réfléchie de la lumière incidente pour une surface plate.
- Distingue les matériaux conducteurs et diélectriques
 - Diélectriques/isolant (non-métaux) : absorbe immédiatement la lumière réfractée
 - Conducteurs (métaux) : renvoie une partie de l'énergie réfractée à l'extérieur de l'objet > coloration du reflet

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 la *reflectance de Fresnel* :

- couleur spéculaire caractéristique du matériau
- dépendante du matériau
- observée dans la direction du vecteur normal
- corrélée à l'*indice de réfraction* n : $F_0 = \left(\frac{n-1}{n+1}\right)^2$
 - lorsque le matériau est observé à l'air libre.

Exemples de Réflectance Spéculaires

Diélectrique	F0 (linéaire)	F0 (sRGB)
Eau	0.02, 0.02, 0.02	0.15, 0.15, 0.15
Plastique/Verre (faible)	0.03, 0.03, 0.03	0.21, 0.21, 0.21
Plastique (élevée)	0.05, 0.05, 0.05	0.24, 0.24, 0.24
Verre (élevée)	0.08, 0.08, 0.08	0.31, 0.31, 0.31
Diamant	0.17, 0.17, 0.17	0.45, 0.45, 0.45

Conducteur	F0 (linéaire)	F0 (sRGB)
Or	1.00, 0.71, 0.29	1.00, 0.86, 0.57
Argent	0.95, 0.93, 0.88	0.98, 0.97, 0.95
Cuivre	0.95, 0.64, 0.54	0.98, 0.82, 0.76
Fer	0.56, 0.57, 0.58	0.77, 0.78, 0.78
Aluminium	0.91, 0.92, 0.92	0.96, 0.96, 0.97

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 d'ombrage
inter-facettes

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

Ombrage *Masquage*

Terme Géométrique GGX

Terme géométrique classiquement 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 :

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

« 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*

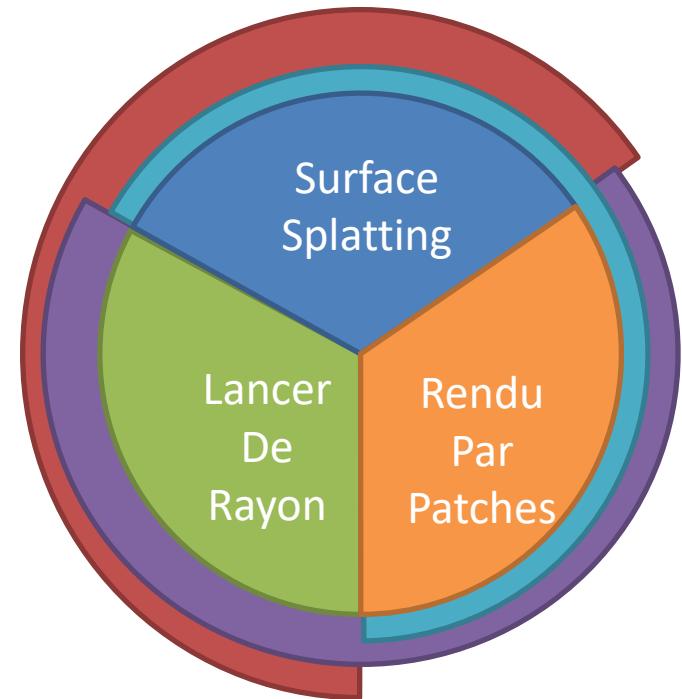
SYNTHÈSE D'IMAGE POUR NUAGES DE POINTS

Problème

- La synthèse d'image est avant tout employée sur des maillages polygonaux car:
 - Il est facile de projeter sur un plan (l'image) ou d'intersecter un triangle avec un rayon
 - Les processeurs graphiques sont conçus pour traiter ce type de données
 - Les outils de conception 3D peuvent tous, d'une manière ou d'une autre, générer des maillages surfaciques de leurs modèles

Rendu par Points

- Le rendu de nuage de points peut se résumer à 2 problématiques :
 - *Pas assez de points, que faire ?*
 - Remplir les pixels de l'images en projetant les points 3D vers la caméra ne suffit pas à boucher les trous
 - **Solution** : reconstruction et/ou super-échantillonnage
 - *Trop de points, que faire ?*
 - Trop de points saturent le programme/matériel de rendu et créent des artefacts (aliasing)
 - **Solution** : sous-échantillonnage, *Level-of-details (LoD)*, structures de partitionnement hiérarchique



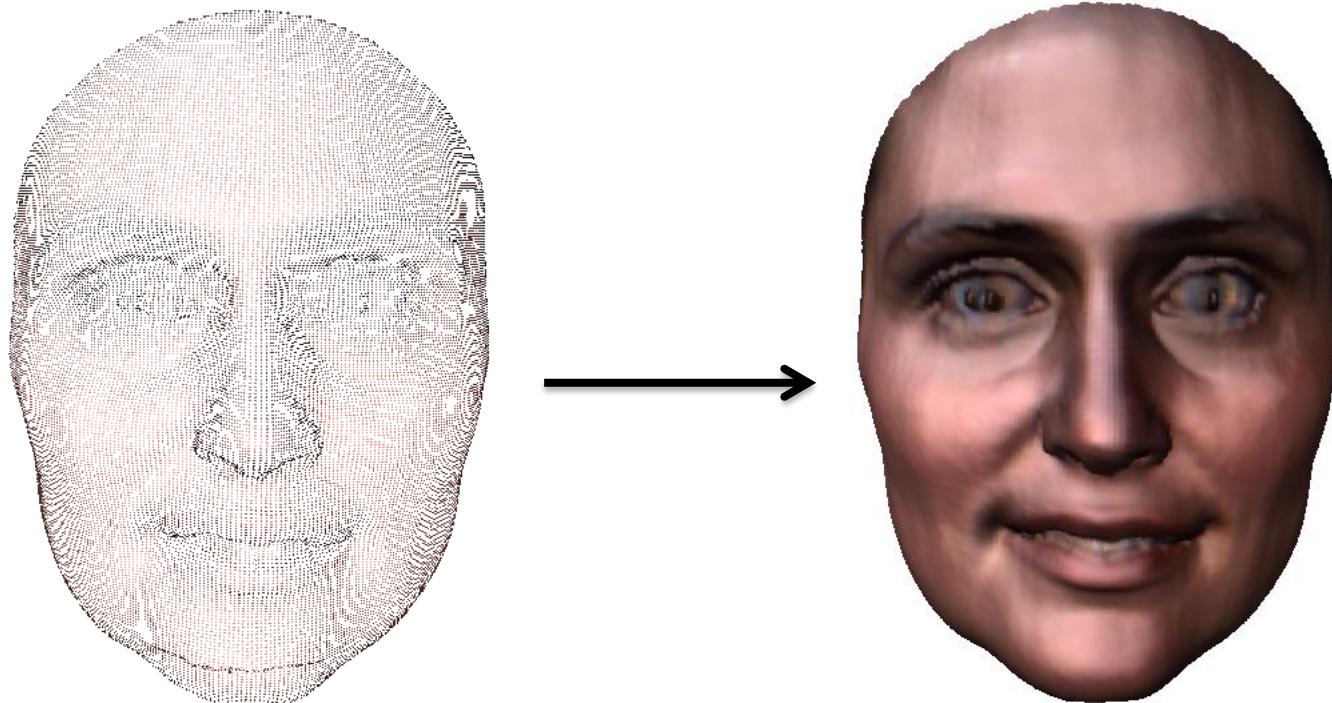
Wish list

- Fidélité
 - Reproduire les zones lisses ou fortement courbées de la **surface sous-jacente**, de manière robuste de tous les points de vue
- Résistance au bruit
 - Fréquent dans les nuages de points
- Rapidité
 - Support GPU, applications temps-réel
- Localité
 - Les nuages de points peuvent être massifs
- Solution sans maillage global
 - *Sinon, il n'y a aucune raison de ne pas faire du rendu polygonal !*

Rendu par point simple

- Rendu par rasterization (Z-Buffer)
 - Eclairage par point = éclairage par pixel
 - 1 point = 1 pixel
 - Méthode simple et rapide, support matériel
 - Primitive `GL_POINTS` et `glDrawArrays` en OpenGL
 - Pic de performance du rendu GPU
 - Problèmes
 - Sous-échantillonnage (trous)
 - Sur-échantillonnage (crénelage)
- Besoin d'une forme de reconstruction
- Dans l'espace image
 - Point > **surfel** (élément de surface)
 - Echantillonnage adaptatif, filtrage et reconstruction



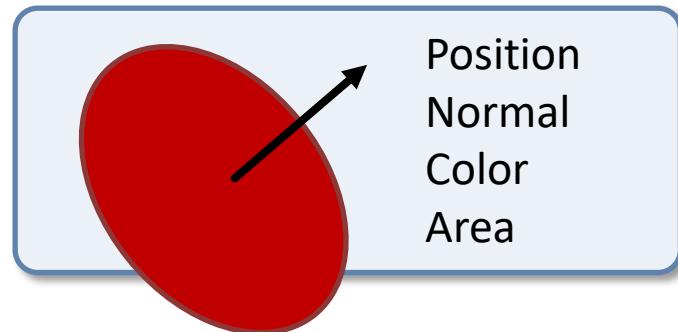


SURFELS ET SPLATTING

Du point au surfel

- Surfel : *surface element* [Grossman 98]
 - Position, normales, couleur et **rayon**

- Calcul du rayon par point :
 - Distance au k-ième plus proche voisin
 - Distribution de points non connue
 - *Kd-tree* à construire
 - Rayon fixe ou variable
 - Distribution connue ou ré-échantillonnage [Cook 86] [Wei 08]



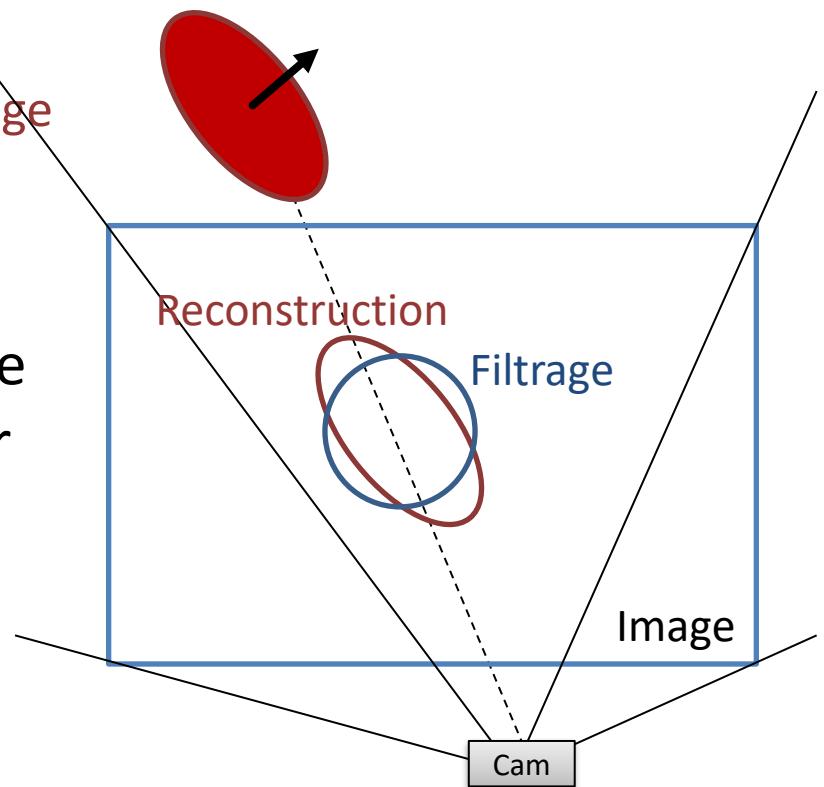
Splatting

- Reconstruire la forme dans l'espace de l'image [Pfister 2000]
- Par interpolation à partir d'un nuage de surfels projetés dans le plan de l'image
- Comment interpoler ?



Interpolation EWA

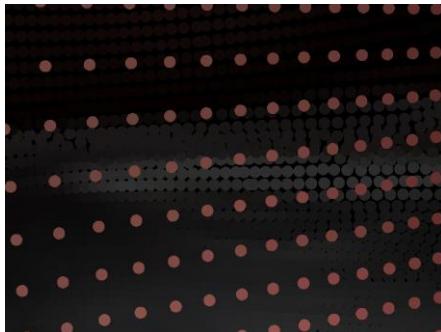
- *Elliptique Weighted Average*
 - Reconstruction : ellipse en espace image
 - Filtrage passe-bas : gaussienne
- Remplit l'image autour du centre de projection du surfel avec sa couleur [Zwicker 2002]
- Antialiasing
- En pratique approximé [Botsch 2005][Guennebaud 2006]



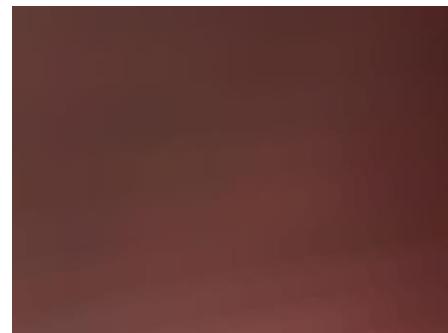
Exemple



Rendu par point simple



Rendu EWA



Algorithme de rendu par EWA splatting

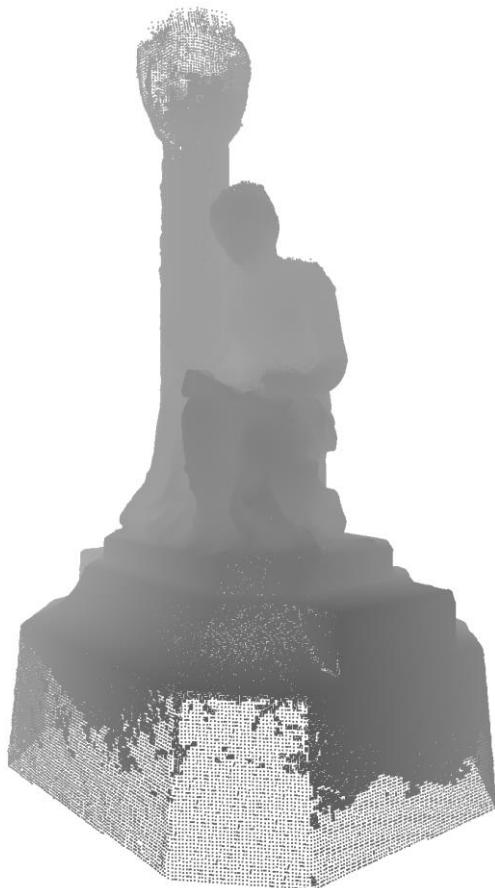
- Prépass: *ombrage de Gouraud* des surfels
 - équation du rendu, BRDF, calcul au centre de chaque surfel
- **Pass 1:**
 - EWA splatting de l'information de profondeur
 - Z-Buffer rempli et « décalé »
- **Pass 2:**
 - EWA splatting couleur + *alpha* (mélange) avec sauvegarde de la somme des pondérations dans *tampon d'accumulation*
 - Image couleur formée, mais dynamique incorrecte
- **Pass 3:**
 - Normalisation de la l'image couleur obtenue par le *tampon d'accumulation*
 - Affichage

Problème : filtrage passe-bas sur les valeur de réponse de la BRDF > perte de fréquence des réflexions spéculaires.

Eclairage par pixel

- Ombrage par pixel (« Phong Shading») : *projection* des paramètres d'apparence plutôt que des valeurs de réponse couleur [Botsch 2005]
- Algorithme
 - Phase 1: Construction d'un **tampon d'attributs (G-Buffer)** par *EWA splatting*
 - Canaux « position/profondeur », normales, coefficient diffus, coefficient spéculaire, etc
 - Phase 2 : **Ombrage différé** du tampon
 - Calcul de la réponse de la BRDF en chaque pixel (*Phong Shading*) à partir des valeurs du tampon d'attribut
 - Affichage

Passe 1: G-Buffer



Profondeur



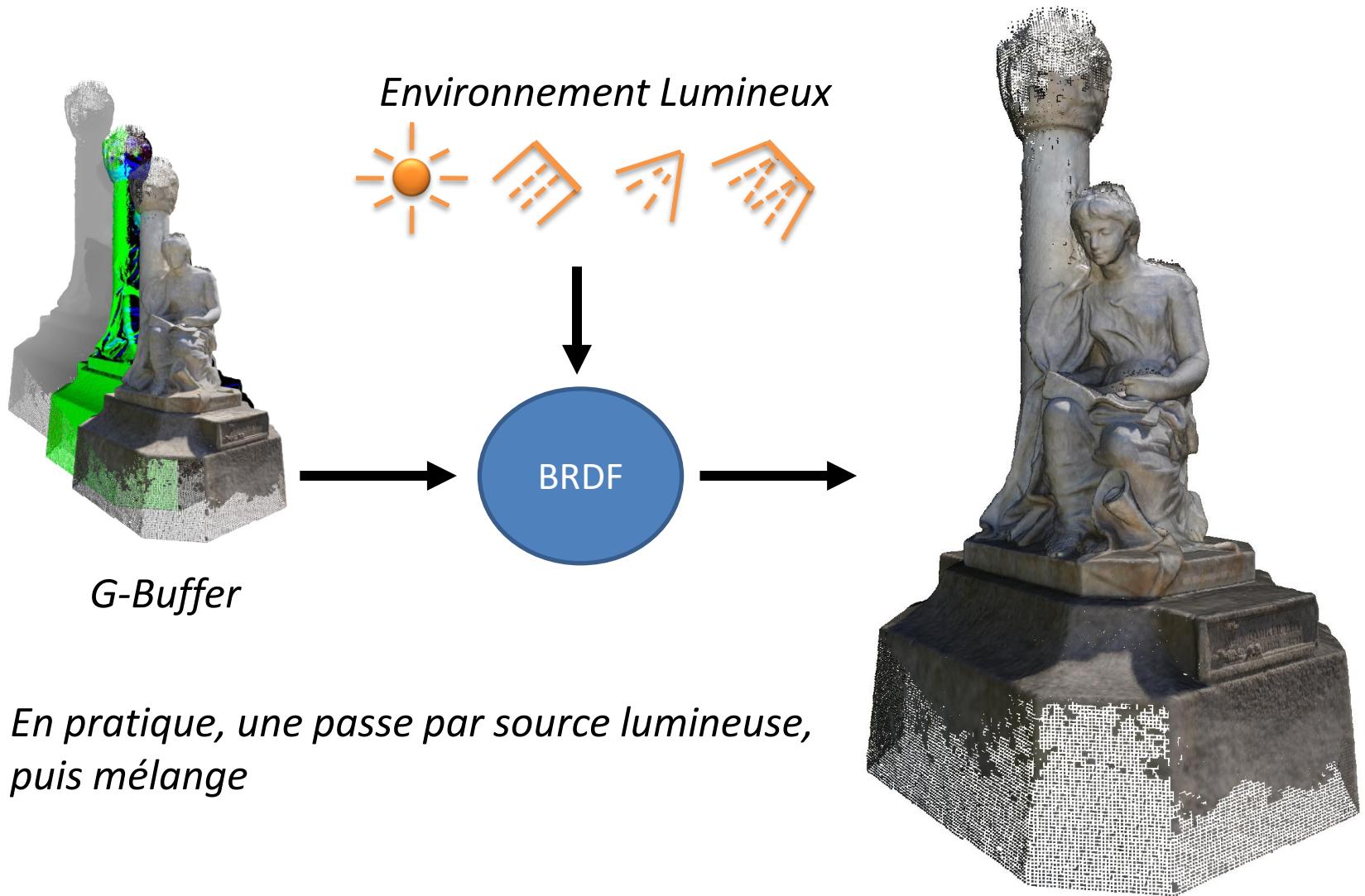
Vecteur normal



Albedo

...

Passe 2 : Ombrage différé



EWA G-Buffer + Ombrage Différé



*BRDF à
microfacettes*



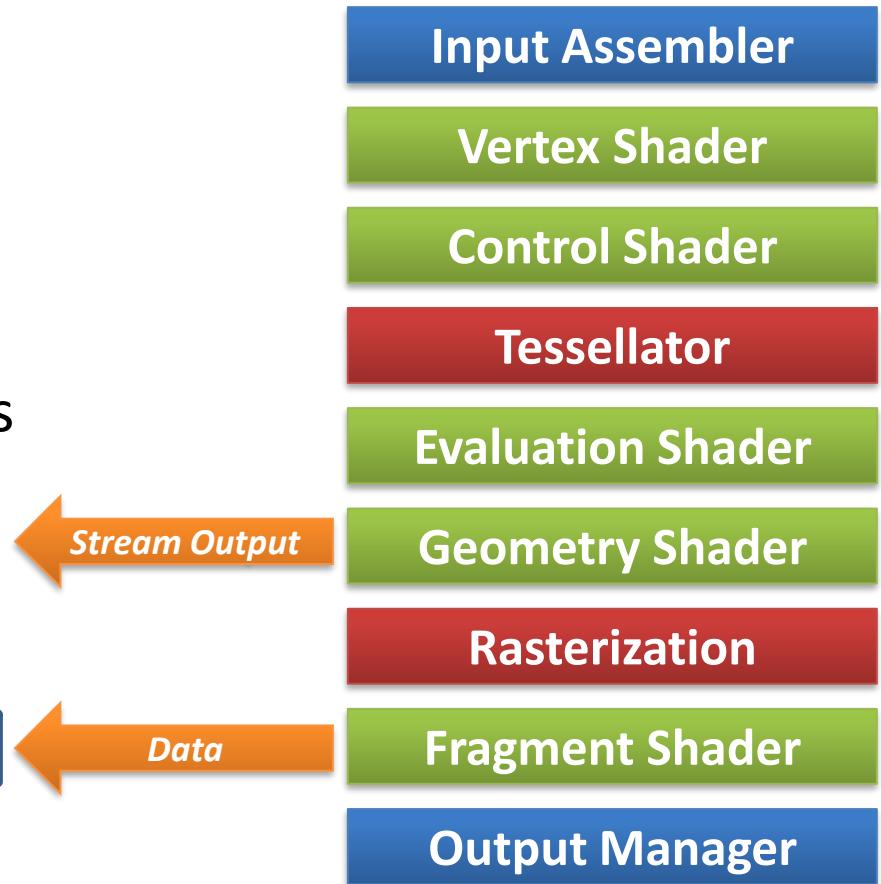
Occultation ambiante

Collection d'effets en espace écran

- Occultation ambiante
- Réflexions
- Flou de mouvement
- Flou de Profondeur

Pipeline Graphique Moderne

- Direct3D 11+ / OpenGL 4+
- Vue API
- Collaboration GPGPU possible (CUDA/OpenCL)
- Implémentation sur processeurs de flux génériques
- Compute Shaders : calcul non graphique de support par texture interposée



Programmable

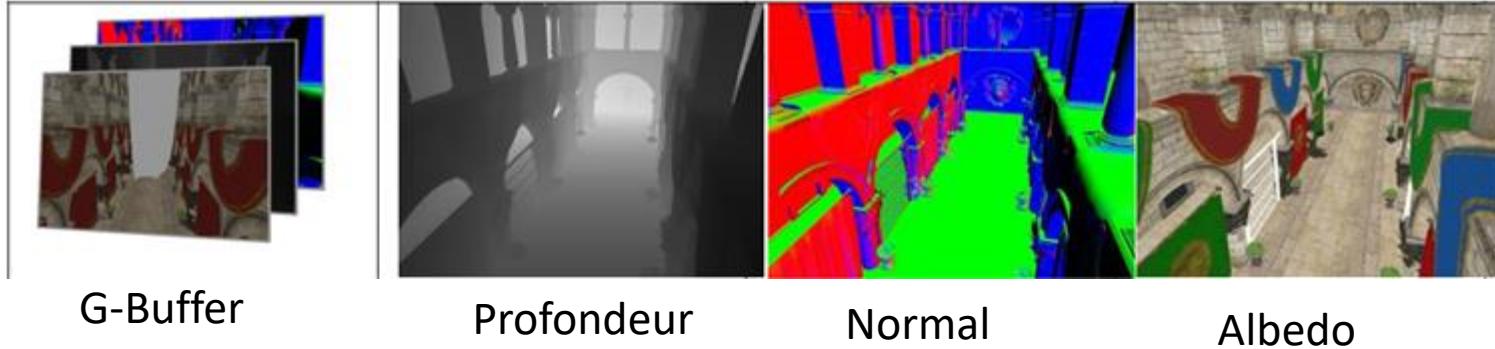


Configurable



Fixe

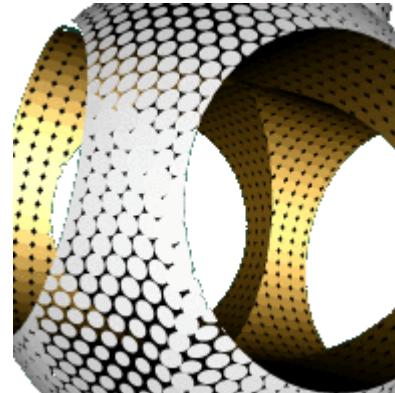
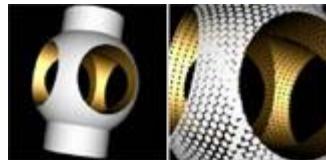
Implémentation GPU



- *Surfel = Point-Sprite*
- Pipeline à concevoir en Ombrage Différé (*Deffered Shading*)
- Stockage des tampons : *Framebuffer (FBO)*
- Remplissage des tampons :
 - *Multiple Render Target (MRT)*
 - depuis un *fragment shader*
 - programme GPU en charge de la coloration des pixels
 - Passe d'éclairage indépendant de la complexité du nuage de points

Extensions

- Arêtes vives [Adams 2003]



- Effets spéciaux [Heinzle 2006]
 - Flou de mouvement
 - Profondeur de champ
- Rendu hybride points/polygones



Vers le rendu différentiel

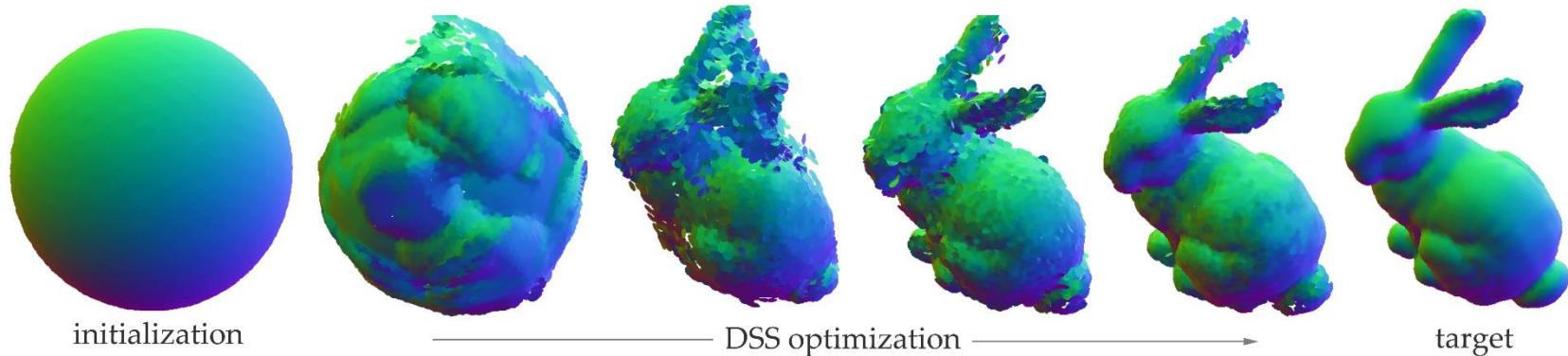


Image tirée de [Yifan et al., SIG Asia 2019]

- Surface splatting différentiel
- Control de la géométrie d'une nuage de point par l'image souhaité
- Dans la mouvance de la *programmation différentielle*
 - Dont le *deep learning* est un cas particulier



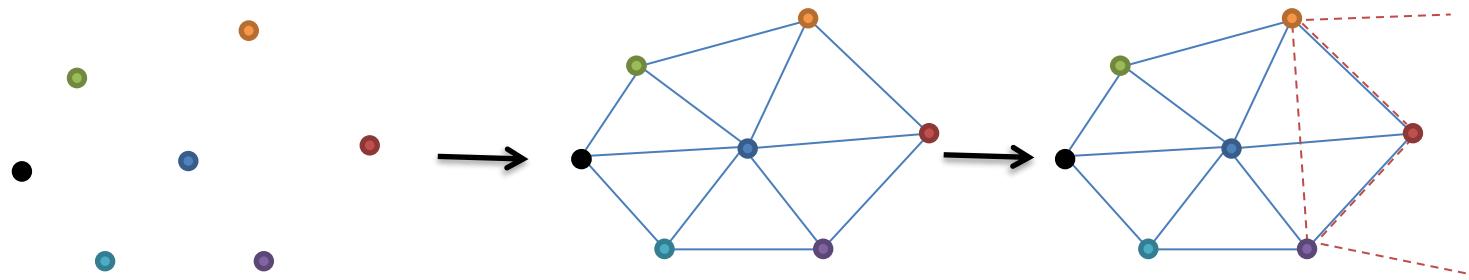
RENDU PAR POLYGONISATION

Idée

- Générer très rapidement des polygones par interpolation des points du nuages afin de remplir l'image sans trou
- Forme de reconstruction de surface partielle/locale
- Peut être combiner avec du rendu par point pure pour les objets distants
- Exploite le pipeline GPU à son maximum lors de la *rasterization*

Triangles Fans

- Générer des éventails de triangles autour des points, en reliant les k plus proches voisins



- Rapide, mais beaucoup d'artefacts:
 - Consistance de la triangulation de proche en proche
 - Des trous subsistes

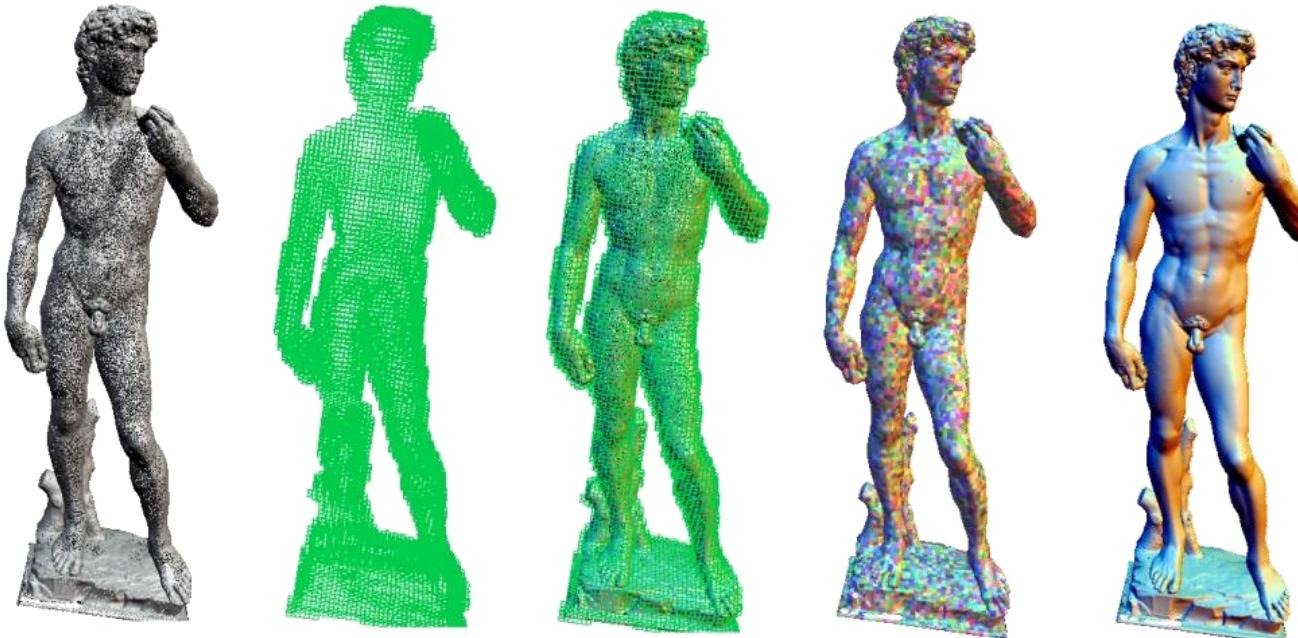
Surfels Strips

Effectuer une reconstruction locale, en dimension inférieures, et stocker les morceaux de surface obtenus dans une structure hiérarchique.

Algorithme

1. Partitionnement spatial du nuage
2. Triangulation locale
3. Compression

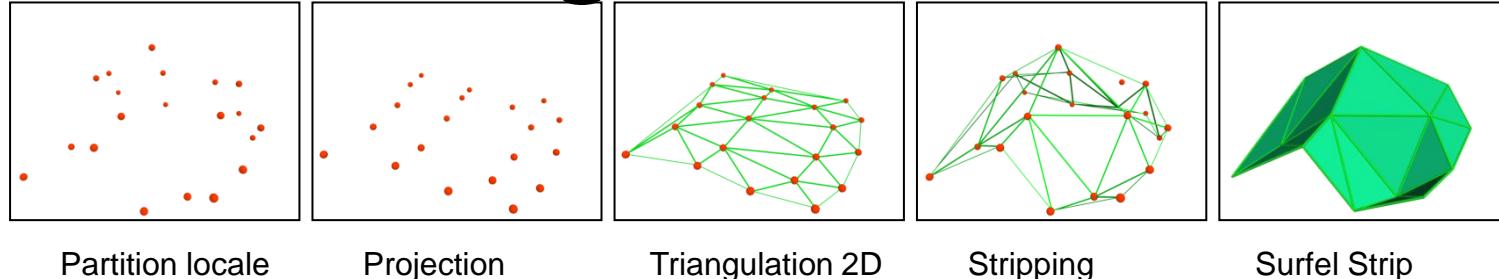
Partitionnement



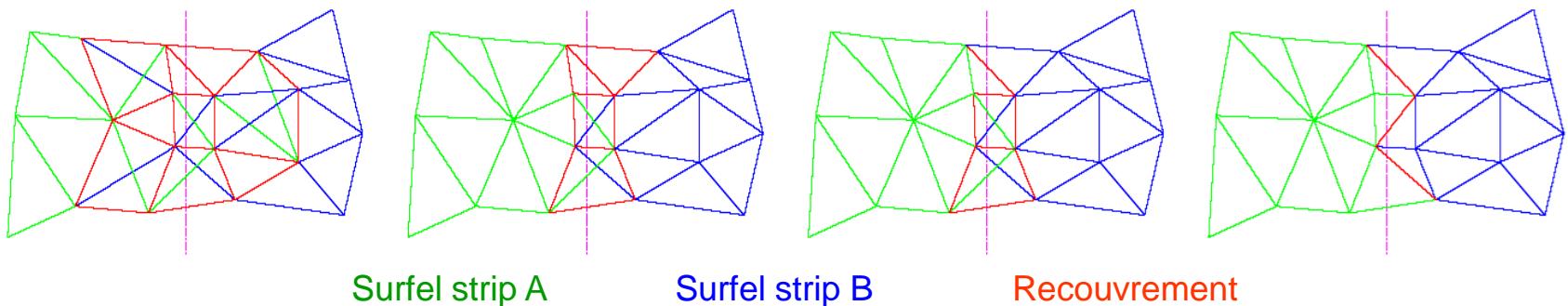
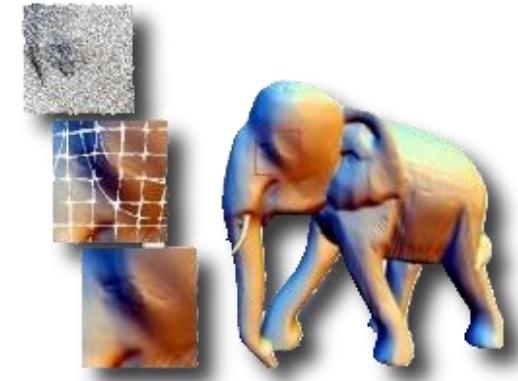
A l'aide d'un octree

- Découpe récursive de l'espace en cubes
- Critère d'arrêt : ensemble de points contenu dans un nœud localement $2.5D >$ estimation via la variation du champ de normales

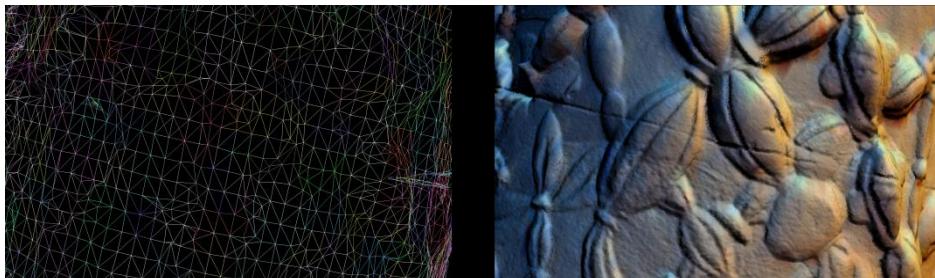
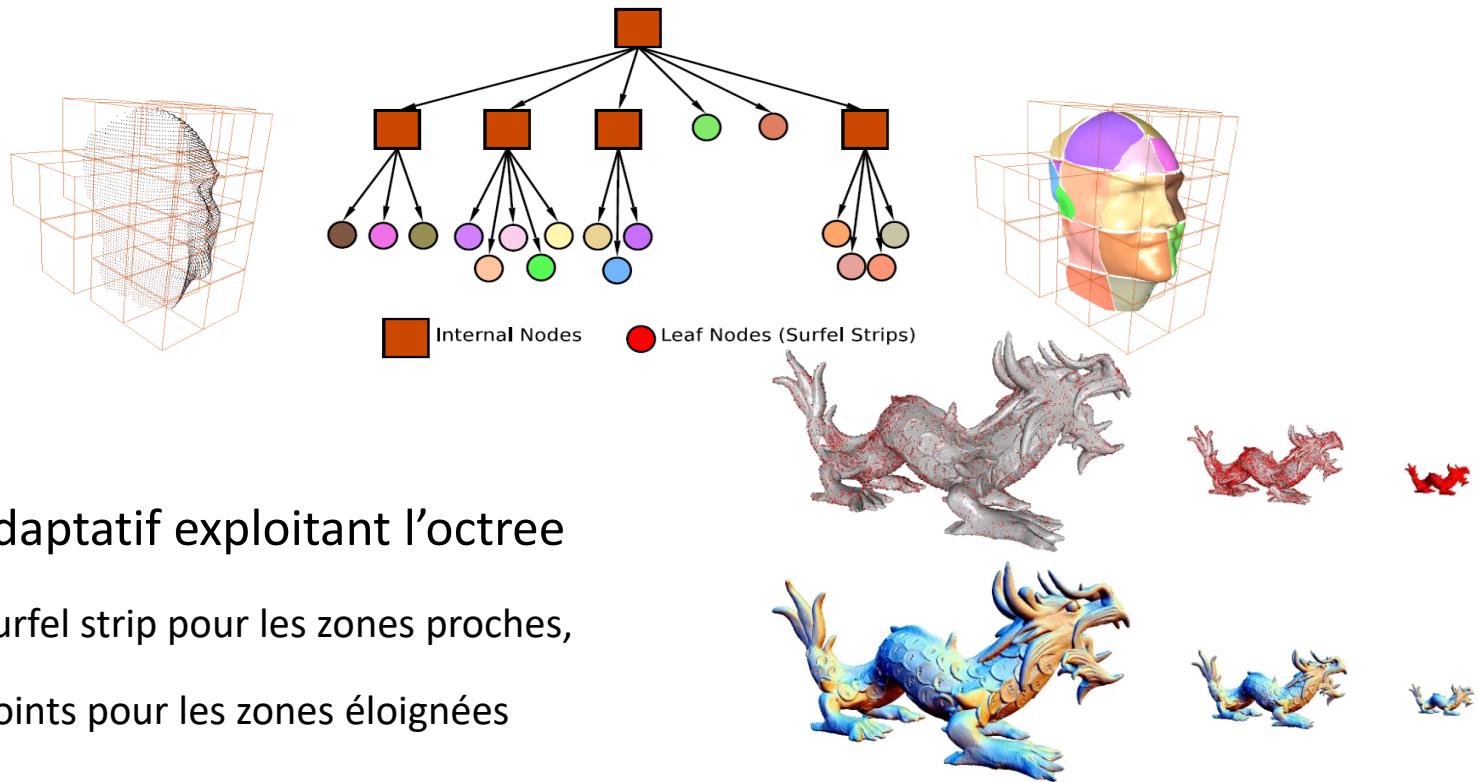
Triangulation Locale



1. Inflation: création de recouvrements entre cellules voisines
2. Projection sur un plan moyen en chaque cellule
3. Triangulation de Delaunay 2D dans chaque plan
4. Decimation des triangles redondants
5. *Stripping*



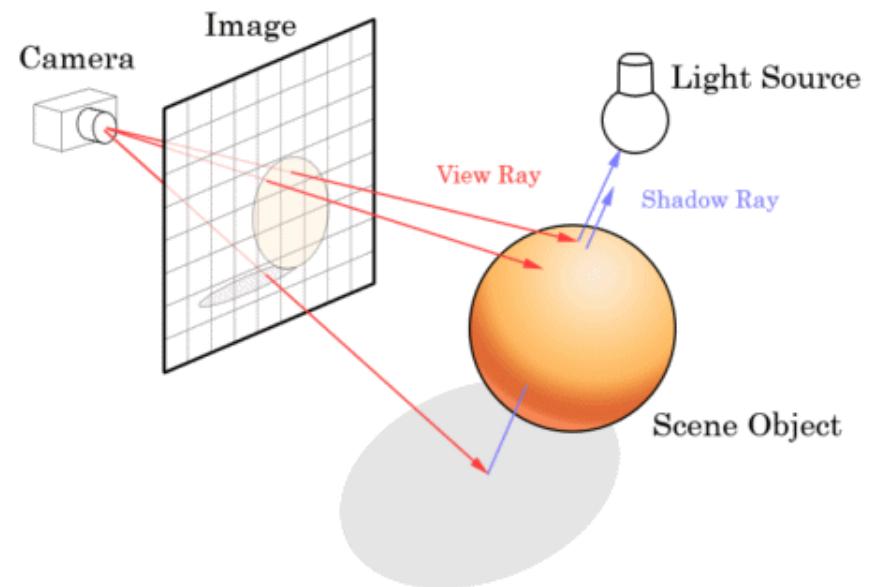
Structure Finale



LANCER DE RAYON

Intersector un nuage de points ?

- Via la surface de points MLS sous-jacente
- A l'exception du test d'intersection :
 - Algorithme identique au lancer de rayon pour maillages



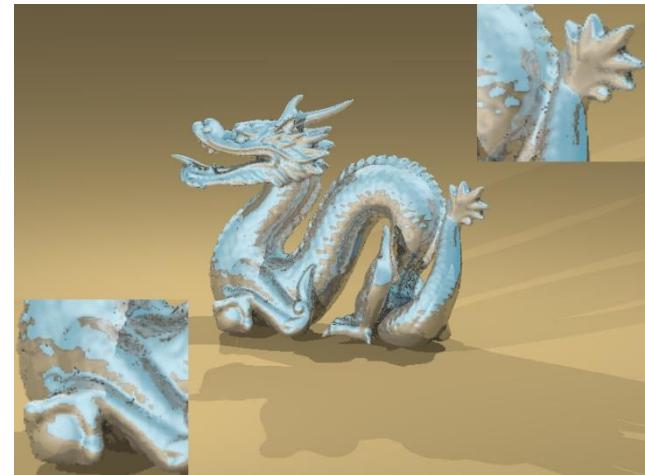
Intersection de Surfaces de Points

- Construction d'un arbre de partitionnement spatial stockant jusqu'à k points par feuilles
 - En général, un kd-tree
- *Ray marching* : on se déplace le long de chaque rayon en testant la **forme implicite** du modèle de surface de points choisi
- Lorsqu'on détecte un changement de signe dans la valeur de fonction implicite associée à la surface de points, on raffine le pas d'itération depuis le point précédent sur le rayon
- Accélération :
 - on équipe chaque point d'un rayon indiquant une boule au-delà de laquelle la surface est garantie absente
 - Le ray marching n'est effectué que dans l'espace de l'union des boules générées par le nuage, testé par ray tracing

Résultats



Ombres



Réflexions

- **Avantages** : nombreux effets calculables trivialement par lancer de rayon, maintenant disponible directement sur les nuages de points
- **Inconvénient** : beaucoup plus lent que les approches par rasterization (splatting, polygonisation).
 - Mais support GPU émergent (NVIDIA RTX, Vulkan RT)

STRUCTURES DE DONNÉES POUR LE RENDU PAR POINTS

Structures Hiérarchique

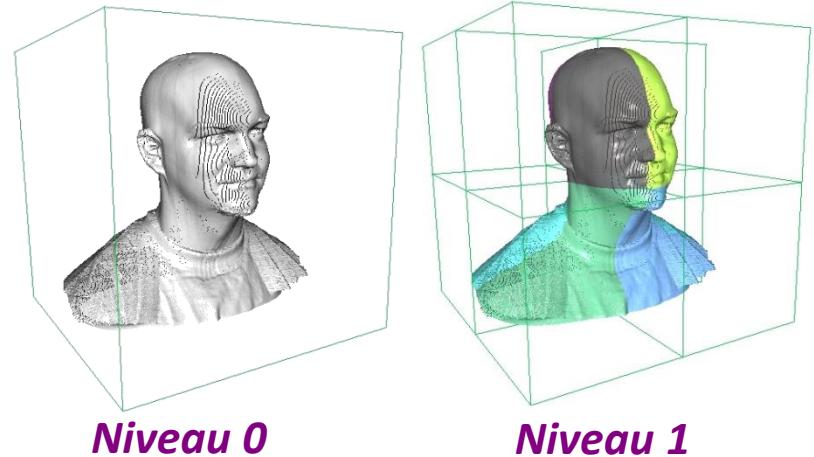
- Principales structures de subdivision spatiale:
 - **kD-Tree** [Bentley 1975] : organisation orthogonale d'un ensemble d'échantillons
 - **BSP-Tree** [Fuchs et al. 1980] : Subdivision binaire et récursive de l'espace par des hyperplans
 - **Quadtree/Octree** [Jackins & Tanimoto 1980] : dimension de l'espace dans la structure (subdivision 1-pour-4 en 2D, 1-pour-8 en 3D)
- Principales structures de subdivision de primitives :
 - **BVH** : Hiérarchie de Volumes Englobant
- **De très nombreuses utilisations et combinaisons dans la littérature.**

Octree

- Généralisation 3D du Quadtree
- Hiérarchie de Cube Englobant

Implémentation récursive très simple

```
TYPE OctreeNode {  
    OctreeNode children[8];  
    Data data; // Bounding Cube + misc. data (e.g. 3D points)  
}  
  
OctreeNode buildOctree (Data data) {  
    OctreeNode node;  
    if (stopCriteria (data))  
        init (node, data); // fill children with NULL and affect data  
    else  
        Data * childData[8];  
        dataSpatialSplit (data, dataChild); // 8-split of the bounding  
                                         // cube and partitioning of data  
        for (int i = 0; i < 8; i++)  
            node.children[i] = buildOctree (childData[i]);  
            node.data = NULL;  
  
    return node;  
}
```

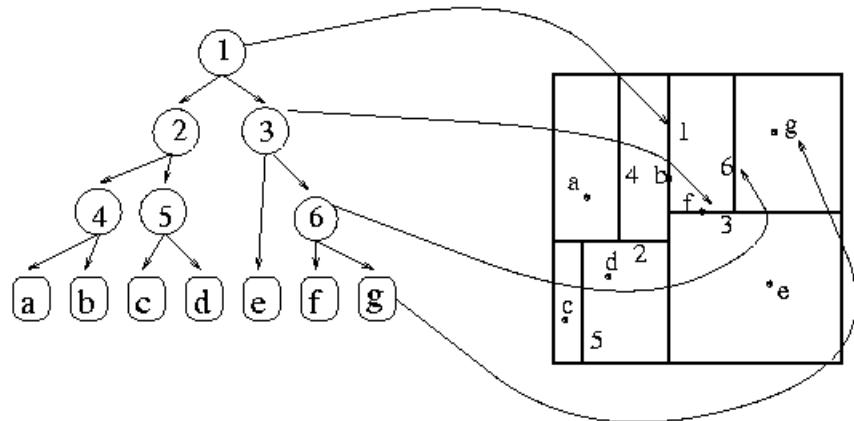


- Peut également être implémenter
- dans une table
 - sans pointeur, arbre quasi-parfait)
 - avec une table de hashage
 - code de Morton

kD-Tree

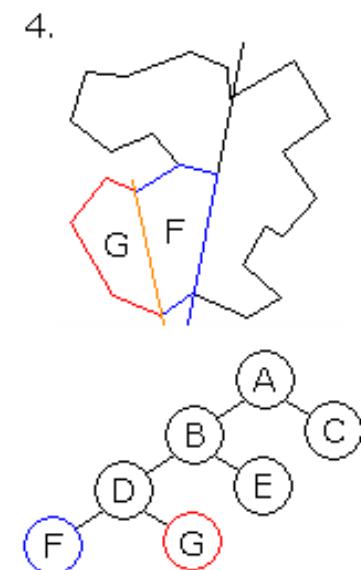
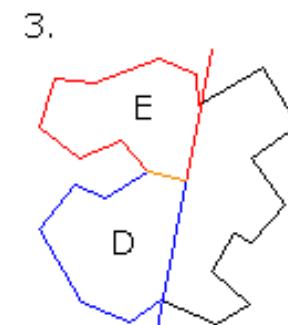
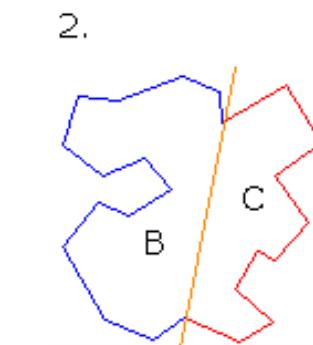
- Structure de partitionnement *orthogonale* d'échantillons
- Arbre binaire. A chaque niveau:
 - Calculer la boîte englobante de P
 - Diviser P le long du plus grand axe (X, Y ou Z)
- Algorithme de construction:

```
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;  
}
```



BSP Tree

- Arbre binaire de partitionnement spatial
- Chaque nœud définit un hyperplan séparant son sous-espace associé
 - Exemple : plan choisi selon un analyse en composantes principales sur P



Comparaison

Adaptivité à une profondeur donnée



Simplicité d'implémentation/Temps construction



Bon Compromis
*Heuristique efficaces pour
(quasi) garantir un temps
d'accès en Log (N) (test
d'intersection)*
Choix habituel du raytracing

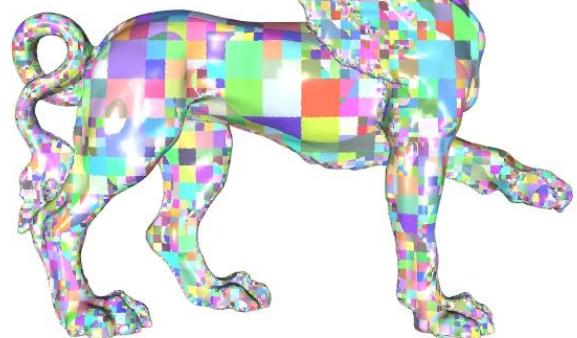
Comparaison



Grille



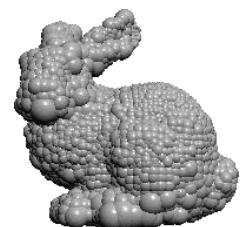
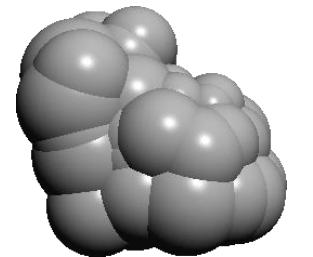
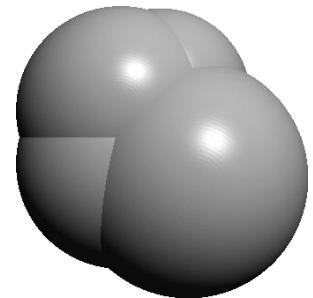
BSP



Octree

Hiérarchie de Volumes Englobants (BVH)

- BVH vs Octree/kd-Tree/BSP-Tree:
 - Noeud != somme de ses enfants
 - Intersection entre volumes englobants (BV)
 - Valence arbitraires des nœuds de l'arbre
 - Forme arbitraire des volumes de partitions
 - En général convexes pour des tests d'intersections plus simples
 - Construction :
 1. Trouver le plus petit BV de P
 2. Subdiviser BV en n sous-BV
 3. Classifier P dans les n sous-BV et recommencer
 - BV usuels :
 - Bounding Sphere (BS)
 - Axis-Aligned Bounding Box (AABB),
 - Oriented Bounding Box (OBB)



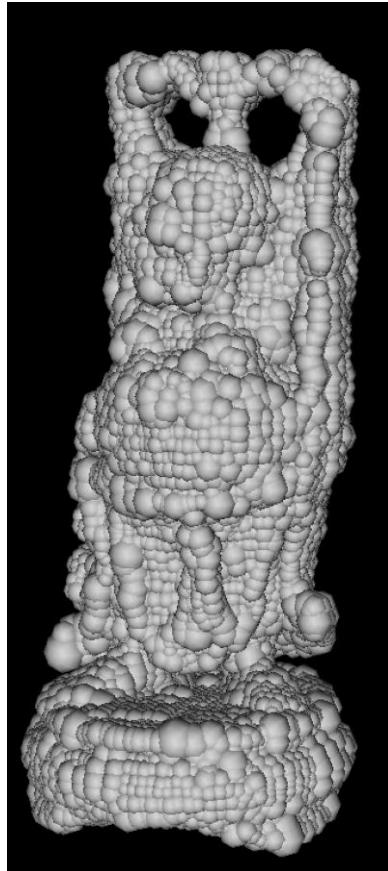
BSH

QSplat

- Exemple d'algoirthme de rendu par point exploitant une structure hiérarchique [Rusinkiewicz 2000]
 - BSH
 - Rendu par point simple (disques)
 - quantification hiérarchique du nuage : 32 à 48 bits/points, incluant position, normale, couleur et rayon
 - Rendu adaptatif : arbre **chargé** et raffiné progressivement en fonction du point de vue
 - Des milliards de points en temps réel sur une machine standard
 - Streaming depuis l'espace de stockage (disque dur, réseau)

Exemple

Visualisation d'un niveau de la structure hiérarchique (une sphère par nœud)



Rendu par point après stabilisation du raffinement hiérarchique

CONCLUSION

Résumé

- Rendu par points : synthétiser des images à partir de modèles 3D sans connectivité
- Réduit « la distance » entre la capture et la visualisation
- Permet de traiter de grandes masses de données 3D
- ... mais peut être utile ailleurs:
 - *Éclairage global pour les effets spéciaux [Pixar 2008]*



Techniques

- **Splatting** : pure rendu par points, reconstruction de la forme en espace image
 - **Polygonisation** : solution hybride, permet de ré-exploiter les algorithmes existants de rendu polygonal
 - **Lancer de rayon** : rendu haute qualité non temps-réel uniquement
-
- Peuvent toutes tirer partie des structures hiérarchique de partitionnement spatial e.g., QSplat
 - Une alternative au couple *reconstruction de surface/rendu polygonal*

Références

Cours :

- Point-Based Graphics, Alexa & Gross, ACM SIGGRAPH 2004 Course Notes

Livres :

- Point-based Graphics, Gross & Pfister, Morgan Kaufman Eds, 2007
- Rendu par points, Schlick, Reuter & Boubekeur, *Informatique Graphique et Rendu*, Hermès Publishing, 2007
- **Fundamentals of Computer Graphics**, by Peter Shirley, Steve Marschner, et alia, A.K. Peters, July 2009.
- **Realistic Ray Tracing**, Shirley & Morley
- **Real-Time Renderting**, Akenine-Möller, Haines, Hoffman
- **Physically Based Rendering**, Pharr, Jakob & Humphreys