# Rapport des différents algorithmes et techniques

Hugo Rens, Matthias Roves et Jehan-Antoine Vayssade

12 novembre 2017

## Intro

### Cadre

Ce document constitue l'analyse des différents algorithmes, techniques et problématiques liés a l'achèvement du Chef d'Œuvre de Master 2 Informatique Graphique et Analyse d'Image de l'année 2017/2018. Il permet ainsi de démontrer notre compréhension du sujet qui nous est proposé et des outils algorithmiques et mathématiques impliqués. Il permettra aussi à l'équipe de retrouver des informations clés.

### Résumé des "Besoins"

Le sujet, proposé par M.Paulin, porte sur l'intégration dans le Radium Engine – the coolest engine ever made – d'un moteur temps réel orienté réalisme, afin de le rendre encore plus cool. Ce moteur doit intégrer les fonctionnalités suivantes :

— Quelques **BSDFs de type Disney** (basés sur PBRT v3).
— Un éclairage direct, fourni par des **sources linéaires, polygonales et disques**.
— Un éclairage indirect, assuré par des **sources ponctuelles nombreuses**.

**Note :** Les fonctionnalités marquées *(direction privilégiée)* sont les fonctionnalités que nous prévoiront dans notre architecture comme une amélioration future mais n'implémenteront pas dans ce chef-d'œuvre.

| | |
|---|---|
| $x$ | Point d'intersection entre le point et la scène |
| $w_i$ | La direction du rayon lumineux |
| $w_o$ | La direction du point vers la camera |
| $n_x$ | la normale de la surface en $x$ |
| $h_x$ | Le demi vecteur à la surface en $x$ |

TABLE 1 – Liste des symboles.

# 1 Techniques et Algorithmes

## 1.1 BSDFs

En rendu, la BSDF (Bidirectional Scattering Distribution Function) est la fonction déterminant la probabilité qu'un rayon lumineux soit dispersé dans une direction souhaitée. Celle-ci peut se distinguer en deux types de BxDF : La BRDF qui prend en compte les interactions à l'impact du rayon et la BTDF qui simule les rayons traversant le matériaux. Dans le cadre de ce chef-d'oeuvre, nous nous concentrerons surtout sur les interactions à la surface.

La BSDF peut ainsi se modéliser par la somme de deux types de réaction à la surface : une interaction diffuse qui va répartir la lumière dans toutes les directions et une interaction spéculaire qui va concentrer une grande partie de l'énergie dans une seule direction. On peut modéliser cette fonction ainsi :

$$bsdf(x, w_i, w_o) = bsdf_{diff} + bsdf_{spec} \tag{1}$$

### 1.1.1 Modèle de Cook-Torrance

L'interaction spéculaire à la surface étant dépendante de sa géométrie, la mise en place d'un modèle physiquement réaliste va donc passer par le choix d'une BSDF spéculaire représentant ce type de surface. Le modèle de Cook-Torrance [3] nous permet de résoudre ce problème :

$$\rho_{ct}(x, w_i, w_o) = \frac{D * G * F}{4 * (w_o \cdot n_x) * (w_i \cdot n_x)} \tag{2}$$

On voit que ce modèle repose sur trois fonctions : $D$, $G$ et $F$. La fonction $D$ est la fonction de distribution des normales à la surface dans une direction donnée. Par exemple si 50% des normales sont dans la direction $lambda$ alors $D(lambda)$ vaut 0,5. De nombreuses versions de cette fonction existent dans la littérature, mais nous utiliserons celle de Trowbridge-Reitz [11] pour ce chef-d'oeuvre :

$$\chi(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x \leq 0 \end{cases}$$

$$D(h_x, n_x, \alpha_x) = \frac{\alpha^2 \chi(h_x \cdot n_x)}{\pi((h_x \cdot n_x)^2(\alpha_x^2 + \tan^2(\frac{1-(h_x \cdot n_x)^2}{(h_x \cdot n_x)^2}))^2} \tag{3}$$

La fonction $G$ quant à elle définit l'atténuation lumineuse à la surface dûe à l'auto-occultation des microfacettes. Cette fonction est donc fortement liée à la fonction de distribution des normales à la surface. Nous allons ici aussi utiliser celle de Trowbridge-Reitz [11] :

$$G_p(\omega, h_x, n_x, \alpha_x) = \chi(\frac{\omega \cdot h_x}{\omega \cdot n_x}) \frac{2}{1 + \sqrt{1 + \alpha_x^2 \frac{1-(\omega \cdot h_x)^2}{(\omega \cdot h_x)^2}}} \tag{4}$$

$$G(\omega_i, \omega_o, h_x, n_x, \alpha_x) = G_p(\omega_o, h_x, n_x, \alpha_x) G_p(\omega_i, h_x, n_x, \alpha_x) \tag{5}$$

La dernière fonction $F$ décrit les phénomènes de réfraction-réflexion à la surface selon le coefficient de Fresnel. Ce coefficient s'exprimant comme la moyenne des réflectances des deux plans de polarisation (parallèle et perpendiculaire) du rayon incident :

$$R = \frac{(R_s + R_p)}{2} \tag{6}$$

avec $R_s$ la réflectance selon le plan perpendiculaire et $R_p$ celle selon le plan parallèle.

| | |
|---|---|
| $c_{d,x}$ | La couleur diffuse de la surface en $x$ |
| $\alpha_x$ | La rugosité de la surface en $x$ |
| $\eta_x$ | L'indice de réfraction de la surface en $x$ |

TABLE 2 – Paramètre pour la BSDF Plastic.

| | |
|---|---|
| $\alpha_x$ | La rugosité de la surface en $x$ |
| $\eta_x + ik_x$ | L'indice de réfraction complexe de la surface en $x$ |

TABLE 3 – Paramètre pour la BSDF Métal.

### 1.1.2 BSDF de type plastique

Comme son nom l'indique, cette BSDF tend à modéliser des interactions lumineuses similaires à celle d'un matériau plastique. Pour faire cela nous allons donc user de la fonction de diffusion Lambertienne et de la fonction spéculaire de Cook-Torrance.

$$diff_{Lambert}(x, w_i, w_o) = \frac{c_{d,x}}{\pi} \tag{7}$$

Les matériaux plastiques étant de type Diélectrique, le fresnel utilisé par le modèle de Cook-Torrance sera donc en adéquation avec ce type de matériau [5] :

$$R_s(\eta_x, \theta) = \frac{\cos\theta - \eta_x\sqrt{1 - (\frac{1}{\eta_x}^2(1-\cos^2\theta))}}{\cos\theta + \eta_x\sqrt{1 - (\frac{1}{\eta_x}^2(1-\cos^2\theta))}} \tag{8}$$

$$R_p(\eta_x, \theta) = \frac{\sqrt{1 - (\frac{1}{\eta_x}^2(1-\cos^2\theta))} - \eta_x\cos\theta}{\sqrt{1 - (\frac{1}{\eta_x}^2(1-\cos^2\theta))} + \eta_x\cos\theta} \tag{9}$$

avec $\theta$ l'angle entre le rayon et la micro-normale à la surface. Dans le cas du calcul du terme spéculaire, on utilisera le demi-vecteur $h_x$.

### 1.1.3 BSDF de type métallique

Dans le cas d'une surface métallique, l'interaction lumineuse est complètement spéculaire. Nous allons donc seulement utiliser le modèle de Cook-Torrance pour représenter ce type de surface. Contrairement aux matériaux plastiques, le métal est conducteur. Nous choisissons donc le Fresnel adéquat [10] :

$$a^2 = 0.25(\sqrt{(\eta_x^2 - k_x^2 - \sin^2\theta)^2 + 4\eta_x^2k_x^2} + \eta_x^2 - k_x^2 - \sin^2\theta)$$

$$b^2 = 0.25(\sqrt{(\eta_x^2 - k_x^2 - \sin^2\theta)^2 + 4\eta_x^2k_x^2} - \eta_x^2 + k_x^2 +^2 \sin^2\theta)$$

$$R_s(\eta_x, k_x, \theta) = \frac{a^2 + b^2 - 2a\cos\theta + \cos^2\theta}{a^2 + b^2 + 2a\cos\theta + \cos^2\theta} \tag{10}$$

$$R_p(\eta_x, k_x, \theta) = R_s(\eta_x, k_x, \theta) * \frac{a^2 + b^2 - 2a\sin\theta\tan\theta + \sin^2\theta\tan^2\theta}{a^2 + b^2 + 2a\sin\theta\tan\theta + \sin^2\theta\tan^2\theta} \tag{11}$$

avec $\theta$ l'angle du rayon par rapport à la micro-normale de la surface. Dans le cas du calcul du terme spéculaire, on préférera utiliser le demi-vecteur $h_x$.

### 1.1.4 BSDF de type Matte

Afin de représenter une surface complètement matte, le modèle ne doit posséder aucun terme spéculaire. On va donc annuler ce terme dans l'équation de BSDF et mettre un terme diffus tel que décrit par Oren-Nayar [9] :

$$s = (w_i \cdot w_o) - (n_x \cdot w_i) * (n_x \cdot w_o)$$

$$t = \begin{cases} 1 & si \quad s \leq 0 \\ max(n_x \cdot w_i, n_x \cdot w_o) & si \quad s > 0 \end{cases}$$

$$A = \frac{1}{\pi + (\frac{\pi}{2} - \frac{2}{3}) * \alpha_x} \quad B = \frac{\sigma'}{\pi + (\frac{\pi}{2} - \frac{2}{3}) * \alpha_x}$$

| | |
|---|---|
| $c_{d,x}$ | La couleur diffuse de la surface en $x$ |
| $\alpha_x$ | La rugosité de la surface en $x$ |

TABLE 4 – Paramètre pour la BSDF Matte.

| | |
|---|---|
| $c_{d,x}$ | La couleur diffuse de la surface en $x$ |
| $c_{s,x}$ | La couleur spéculaire de la surface en $x$ |
| $\alpha_x$ | La rugosité de la surface en $x$ |

TABLE 5 – Paramètre pour la BSDF Substrate.

$$diff_{Matte}(x, w_i, w_o) = c_{d,x}(n_x \cdot w_i) * (A + B * \frac{s}{t}) \tag{12}$$

#### 1.1.5 BSDF de type Substrate

Pour ce type de BSDF nous allons nous servir d'un type de spéculaire de Cook-Torrance modifié. Afin de mieux représenter le facteur géométrique de ce type de surface, le terme $G$ de l'équation à été développé directement dans la formule de Cook-Torrance, ce qui nous donne :

$$diff_{Substrate}(x, w_i, w_o) = \frac{28 * c_{d,x}}{23 * \pi} * (1 - ks_x) * (1 - 0.5 * |w_i \cdot n_x|)^5 * (1 - 0.5 * |w_o \cdot n_x|)^5 \tag{13}$$

$$spec_{Substrate}(x, w_i, w_o) = \frac{D(\alpha_x, w_h) * F(c_{s,x}, w_i \cdot w_h)}{4 * (w_i \cdot w_h) * \max(|w_i \cdot n_x|, |w_o \cdot n_x|)} \tag{14}$$

#### 1.1.6 Principled BSDF (Modèle de Disney)

Le Principled Shader de Disney [1] fut créé sur l'envie de modéliser un grand nombre de matériaux physiquement réalistes à l'aide d'une seule fonction fortement paramétrisée. Pour faire cela, ils se sont donc basé sur les nombreux types de BSDF déjà disponibles et ont simplement composé ces dernières avec des paramètres définissant l'influence de chaque part sur le rendu final.

La fonction ainsi obtenue prend la forme suivante :

$$diff_{disney}(x, w_i, w_o) = (1 - \alpha_{met})((1 - \alpha_{ss})diffuse_{disney} + \alpha_{ss} * SS_{disney} + \alpha_{sheen} * sheen_{disney}) \tag{15}$$

$$spec_{disney}(x, w_i, w_o) = \frac{D_{disney} G F_{shlick}}{4 \cos(w_i \cdot n_x) \cos(w_o \cdot n_x)} + \alpha_{cc} * CC_{disney} \tag{16}$$

On voit ainsi l'apparition de différentes fonctions qui vont venir définir les différentes parties du principled BSDF.

**Terme diffus :** Pour représenter la diffusion de ses matériaux, la BSDF utilise un modèle Lambertien auquel elle multiplie un Fresnel tel que présenté dans le cours SIGGRAPH de Burley en 2015 [2] :

$$diffuse_{disney} = \frac{c_x}{\pi}(1 - 0.5(1 - (w_o \cdot n_x)^5))(1 - 0.5(w_i \cdot n_x)^5) \tag{17}$$

**Approximation du SubSurface Scattering :** Afin d'obtenir un résultat toujours plus réaliste, le terme diffus est mélangé avec une approximation d'un rendu modélisant la dispersion sous la surface de l'objet. Pour ce faire, l'équipe Disney s'est inspiré du modèle présenté par Hanrahan-Kruger [4] :

$$Fss = lerp((w_o \cdot n_x)^5, 1, (w_i \cdot h_x)^2 \alpha_x) lerp((w_i \cdot n_x)^5, 1, (w_i \cdot h_x)^2 \alpha_x)$$

$$SS_{disney} = 1.25(Fss * (\frac{1}{|w_o \cdot n_x| + |w_i \cdot n_x|} - 0.5) + 0.5) \tag{18}$$

**Modélisation du Sheen :** Le Sheen (ou brillance en français) sert à représenter l'effet de brillance qui peut être vu é angle rasant sur certains tissus. Cet ajout est donc modélisé comme une BSDF proche d'un Fresnel :

$$sheen_{disney} = lerp(\alpha_{sheenTint}, 1, kd_x)(1 - \cos(w_i \cdot h_x))^5 \tag{19}$$

| | |
|---|---|
| $c_x$ | La couleur de la surface en $x$ |
| $\alpha_{specTint}$ | paramètre contrôlant la teinte du spéculaire en $x$ |
| $\alpha_{met}$ | paramètre contrôlant l'aspect métallique de la surface en $x$ |
| $\alpha_{rou}$ | La rugosité de la surface en $x$ |
| $\alpha_{anisotropic}$ | paramètre contrôlant le taux d'anisotropie en $x$ |
| $\alpha_{ss}$ | paramètre contrôlant le taux de SS en $x$ |
| $\alpha_{sheen}$ | paramètre contrôlant le taux de Sheen en $x$ |
| $\alpha_{sheenTint}$ | paramètre contrôlant la couleur de l'éclat spéculaire en $x$ |
| $\alpha_{cc}$ | paramètre contrôlant le taux de ClearCoat en $x$ |
| $\alpha_{ccGlow}$ | paramètre contrôlant la brillance du ClearCoat en $x$ |
| $\eta_x$ | L'indice de réfraction de la surface en $x$ |

TABLE 6 – Paramètre pour la BSDF Principled.

**Distribution des Normales :** Le modèle de BSDF de Disney peut aussi représenter des modèles anisotropiques. De ce fait un recalcul de D a été nécessaire afin de représenter ces surfaces [1] :

$$aspect = \sqrt{1 - 0.9 * \alpha_{anisotropic}}$$

$$\alpha_{rou,x} = \alpha_{rou}^2/aspect \quad \alpha_{rou,y} = \alpha_{rou}^2 \cdot aspect$$

$$D_{disney} = \frac{1}{\pi}\frac{1}{\alpha_{rou,x}\alpha_{rou,y}}\frac{1}{((h_x \cdot \vec{x})^2/\alpha_{rou,x}^2 + (h_x \cdot \vec{y})^2/\alpha_{rou,y}^2 + (h_x \cdot n_x)^2)^2} \tag{20}$$

**Fresnel :** Selon l'équipe Disney, l'approximation de Schlick est une méthode suffisamment précise pour obtenir les coefficients de Fresnel [1] :

$$F_o = (1 - \alpha_{met})\frac{\sqrt{\eta - 1}}{\sqrt{\eta + 1}}((1 - \alpha_{specTint}) + \alpha_{specTint}c_x) + \alpha_{met}c_x$$

$$F_{shlick}(F_o, \theta) = F_o + (1 - F_o)(1 - \cos\theta)^5 \tag{21}$$

Avec $Fo$ la réflexion spéculaire à incidence normale et $\theta$ l'angle entre le rayon et la micro-normale ($h_x$ dans le cas présent). Mais l'on remarquera que le modèle peut présenter des erreurs dans le cas de matériaux métalliques.

**Addition d'un effet ClearCoat :** L'effet ClearCoat définit par l'équipe Disney permet de modéliser une couche de vernis par dessus la surface de l'objet. Cet effet est obtenu en ajoutant un lobe spéculaire a la BSDF avec une rugosité fixée a 0.25 et un IOR de 1.5 :

$$CC_{disney} = 0.25G(w_i, w_o, h_x, n_x, 0.25)F_{shlick}(0.04, w_i \cdot h_x)D_{GTR1}(h_x, n_x, lerp(\alpha_{ccgloss}, 0.1, 0.001) \tag{22}$$

Contrairement aux autres BSDFs, nous choisissons une distribution GTR 1.0 car celle-ci permet des traînées proche de la réalité dans le cas d'une couche translucide :

$$D_{GTR1}(h_x, n_x, \alpha) = \frac{\alpha^2 - 1}{\pi \log(\alpha^2)(1 + (\alpha^2 - 1)(h_x \cdot n_x)^2} \tag{23}$$

## 1.2 Sources ponctuelles, très nombreuses (Many Lights)

### 1.2.1 Problème

L'éclairage – direct ou indirect – par des sources nombreuses est toujours problématiques en temps réel, car il induit une surcharge de la bande passante et (souvent) du calcul. Plusieurs techniques existent pour réduire les calculs, dont, notamment, le calcul des lumières uniquement en les zones qu'elles éclairent.

Ceci peut être effectué en espace image (tiled rendering) ou en espace scène (clustered rendering). Le tiled rendering fait cependant un clustering plus approximatif des sources, et des cas particuliers existent où la technique devient non-bénéfique (dans des configurations où la profondeur varie trop sur une même tuile pour permettre de sélectionner efficacement les lumières affectantes).

Bien qu'il soit discuté ici des sources ponctuelles nombreuses, il est également possible de clusteriser les sources polygonales de l'éclairage direct. Ceci sera précisé dans la section suivante.

### 1.2.2 Choix de la technique

Nous nous intéresserons donc au **clustered shading** ; la technique présentée dans [7] présente en effet quelques avantages sur le tiled rendering. La segmentation est, de par sa nature 3D, mieux adaptée, et permet d'éviter les variations de performance liées aux changements de vue. Les auteurs précisent que les pires cas sont également mieux gérés. La technique permet également de prendre d'autres paramètres de la scène en compte lors pour l'éligibilité des lumières, et en particulier les normales. Le nombre de lumières supporté annoncé sur papier est estimé à 1,000,000 de sources (ponctuelles).

La gestion des ombres n'est, en revanche, pas plus prise en charge qu'avec du tiled rendering, dans le cadre many-lights.

### 1.2.3 Boucle de rendu

```
0   Z Prepass (Zmin ET Zmax)
1   Clustering (compute shader)
2   // Éclairage direct
3   for each object:
4           for each fragment:
5           Discard if Zfail // Attention au depth figthing [1]
6           for each light in polygons:
7                   Accumulate direct lights
8   // Éclairage indirect
9   for each object:
10          for each fragment:
11          Discard if Zfail // Attention au depth figthing [1]
12          for each light in the fragment many-lights cluster:
13                  Accumulate indirect lights
```

Les deux types d'éclairage sont séparés afin de pouvoir les gérer plus indépendamment. En particulier, on sera en mesure de ne calculer l'éclairage indirect (coûteux) que lorsque la vue ne bouge pas, de manière asynchrone. La structure du cluster peut être similaire pour les deux types d'éclairages, et ainsi passer la liste de clusters adéquat pour chacune des deux étapes.

### 1.2.4 Z Prepass

**Réduction de l'overdraw** L'algorithme de Forward Shading utilisé souffre d'un gros défaut : le calcul d'éclairage est effectué pour chaque fragment, ce qui est sous-optimal sachant que pour chaque pixel, un seul d'entre eux sera affiché. Pour pallier à ce problème, on calcule dans un premier temps le Z-buffer, puis, dans une seconde passe, éliminons les fragments que nous savons à l'avance inutiles.

Calcul du zbuffer
- Disable colormask
- Enable depth test
- Depth func lessEq
- Enable depth mask

Utilisation pour minimiser l'overdraw
- Enable colormask
- Enable depth test
- Depth func lessEq
- Disable depth mask

### 1.2.5 Clustering

**Critère de clustering** L'un des critères les plus directs est la position. Elle sera considérée dans l'espace *Camera*. Le plus naturel serait de subdiviser le frustum uniformément, mais on obtiendrait des clusters très inégaux. Pour pallier à ce problème, on considère le logarithme de ces positions, ce qui offre des clusters répartis de manière plus égale.

Il est également possible d'utiliser d'autres critères pour les clusters. [7] propose un clusternig tenant compte de la normale d'un fragment, ce qui permet de potentiellement réduire encore le nombre de lumières affectant un même fragment.

Le cas limite suivant doit être évité : il peut tout à fait arriver qu'un fragment soit affecté par toutes les lumières, si elles sont toutes concentrées en un point. Ce cas sera évité en instaurant une limite au nombre de sources affectant simultanément un point. Nous pensons à une limite située entre 128 et 1024 sources. Ce cas devrait arriver très peu en pratique, malgré tout, mais nous permet d'optimiser notre calcul.

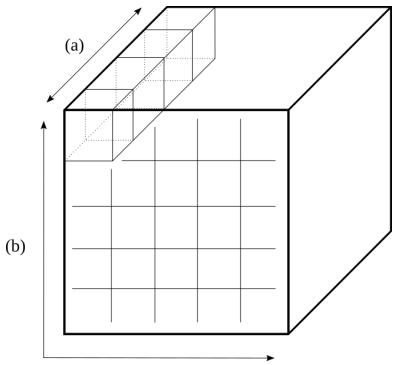


FIGURE 1 – Stockage des sources de lumières. En *(a)* les données pour une lumière, sur les composantes RGBA des textures. En *(b)*, chaque lumière, organisée ou non.

**Stockage et structure de données** Pour chaque cluster il faut stocker, d'une manière ou d'une autre, les sources qu'il contient. Cette question est difficile à résoudre, notamment car elle demande de faire un compromis entre la latence induite par des indirections dans le cas d'une représentation indexée et la difficulté que constitue la recopie et le transfert des données sur GPU dans le cas d'une duplication.

Après discussion, nous sommes arrivés à la conclusion que le système de cache du GPU amortirait probablement très bien la latence liée au déréférencement alors que la recopie des sources poserait de gros problèmes de bande passante pour un gain par rapport au déréférencement minime.

Une représentation possible consiste à créer un index des lumières par cluster et à les regrouper dans une liste, de manière semblable à ce qui est fait en tiled shading dans [8] p.62. La solution que nous prévoyons d'implémenter se base sur les tableaux de textures 2D, comme montré sur la figure 1.

**Organisation des lumières *(direction privilégiée)*** Nous prévoyons de pouvoir améliorer la localité des positions de chaque lumière (en $(x, y)$) en parcourant ces lumières selon une courbe de Hilbert ; deux lumières proches dans l'espace le seraient alors souvent dans le plan.

## 1.3 Sources polygonales, linéaires

### 1.3.1 Choix de la technique

L'éclairage direct de la scène doit être fourni par un ensemble réduit – de l'ordre de la dizaine/centaine – de scènes polygonales, linéaires, et disques. Les techniques envisageables doivent permettre de contourner l'infaisabilité d'un échantillonnage comme fait en rendu hors-ligne.

La technique présentée dans [6] sera utilisée. L'idée est de pouvoir approximer la zone éclairée d'un point par une source de forme polygonale à l'aide d'une fonction simple et aux bonnes propriétés (de norme, de distribution, ...).

### 1.3.2 Principe

Cette technique repose sur le choix d'une distribution sphérique de base, présentée en (24), et sur la transformation de ses vecteurs de direction par une matrice $3 \times 3$, qui produit une distribution alternative avec des propriétés similaires, pour simuler le comportement d'une distribution GGX par rapport à une source.

$$D_o(\omega_o = (x, y, z)) = \frac{1}{\pi} max(0, z) \tag{24}$$

### 1.3.3 Détails

Les parties clés de l'algorithme consistent à trouver la matrice de transformation (fitting) et à intégrer la distribution sur la projection du polygone.

**Fitting** La transformation de la distribution originale dépend de 2 paramètres : l'angle incident $\theta$ (entre la normale du polygone et la normale d'un point de la sphère) et la rugosité $\alpha$ de la BRDF, et forme une matrice

$M$ de la forme (25) qui contient 4 paramètres variables.

$$M = \begin{bmatrix} a & 0 & b \\ 0 & c & 0 \\ d & 0 & 1 \end{bmatrix} \tag{25}$$

Comme on le verra dans la partie Intégration, on a besoin de $M^{-1}$ et non pas de $M$, ce qui fait 4 paramètres plus un cinquième paramètre pour la norme de la BRDF. Ces matrices seront précalculées dans des textures de $64 \times 64$ pour tous les $\theta$ dans un sens et tous les $\alpha$ dans l'autre, et sont ensuite retrouvées par interpolation.

**Intégration** L'intégration est faite pour $D_o$ sur la projection du polygone, et non pas l'inverse, ce qui permet de ne stocker que les 5 paramètres cités précédemment et de généraliser l'intégration. On peut faire ceci car [6] montre que l'intégrale sur la projection du polygone sur la sphère de la distribution sphérique transformée est égale à l'intégrale sur la projection du polygone transformé de la distribution originale (24), pour laquelle on dispose d'une expression analytique, de la forme :

$$\int_{P_o} D_o(\omega_o) d\omega_o \tag{26}$$

où les $\omega_o$ sont les directions associées aux projections des points du polygone transformé $P_o = M^{-1}P$. Cette intégration peut se faire avec le Théorème de Stokes. En considérant la radiance émise $L$ constante sur le polygone, on obtient

$$I = L \int_{P_o} D_o(\omega_o) d\omega_o \tag{27}$$

avec :

$$\int_{P_o} D_o(\omega_o) d\omega_o = E(P_o) = E(p_1, \ldots, p_n) = \frac{1}{2\pi} \sum_{i=1}^{n} acos(\langle p_i, p_j \rangle) \left\langle \frac{p_i \times p_j}{||p_i \times p_j||}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\rangle \tag{28}$$

où :
- $j = (i+1)\%n$ pour décrire localement une arrête du polygone
- $\langle a, b \rangle$ est le produit scalaire

**Lumières texturées *(direction privilégiée)*** Les équations précédentes ne sont plus valables lorsque le polygone émet de manière non-uniforme, et la formule de $I$ doit être réécrite comme montrée dans [6], en section 5. Les textures doivent de plus êtres filtrées de manière adéquate, car il faut connaître la paramétrisation $(u, v)$ et le LOD.

**Clustering *(direction privilégiée)*** Ces sources de lumières, moins nombreuses, peuvent être toutes évaluées pour chaque fragment. Mais on pourrait vouloir les mettre en clusters, de la même manière qu'on le fait pour les sources indirectes. Il faudra définir un volume d'influence simplifié pour ces polygones.

# Conclusion

Le moteur sera de type Forward Clustered, avec une étape de Z-prepass.

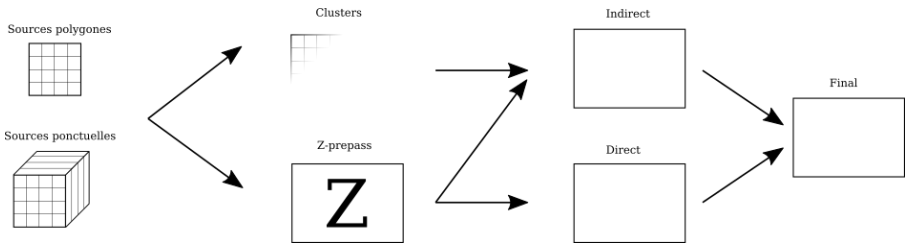


FIGURE 2 – Schéma **simplifié** de l'algorithme de rendu

# Références


[1] Brent Burley. Physically based shading at disney. *SIGGRAPH shading course*, 2012.

[2] Brent Burley. Extending the disney brdf to a bsdf with integrated subsurface scattering. *SIGGRAPH shading course*, 2015.

[3] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1) :7–24, January 1982.

[4] Pat Hanrahan and Wolfgang Krueger. Reflection from layered surfaces due to subsurface scattering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 165–174, New York, NY, USA, 1993. ACM.

[5] Eugene Hecht. *Optics 2nd edition*. Addison-Wesley Publishing Company, 1987.

[6] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *ACM Trans. Graph.*, 35(4) :41 :1–41 :8, July 2016.

[7] Ola Olsson, Markus Billeter, and Ulf Assarsson. Clustered deferred and forward shading. In *HPG '12 : Proceedings of the Conference on High Performance Graphics 2012*, pages 87–96, Paris, France, 2012.

[8] Ola Olsson, Emil Persson, and Markus Billeter. Real-time many-light management and shadows with clustered shading. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, pages 12 :1–12 :398, Los Angeles, California, 2015. ACM.

[9] Michael Oren and Shree K. Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 239–246, New York, NY, USA, 1994. ACM.

[10] Peter S. Shirley. *Physically Based Lighting Calculations for Computer Graphics*. PhD thesis, Champaign, IL, USA, 1991. UMI Order NO. GAX91-24487.

[11] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 195–206, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.