# Advanced Techniques for Realistic Real-Time Skin Rendering

Hamza El Rafoulli
Tiago Lobato Gimenes

19 février 2015

# 1 Introduction

Ce document discute l'implémentation d'un algorithme de rendu en temps réel de la peau humaine. D'abord, nous présenterons le modèle adopté par les chercheurs chez NVIDIA, ses avantages, ses limitations, pour ensuite détailler notre propre implémentation.

# 2 Modèle

Le rendu de la peau est difficile à simuler à cause de sont caractère translu-minescent (elle est translucide). Nous avons donc besoin d'un modèle de peau qui tienne en compte de cet aspect, ce que nous allons faire en considérant que la peau est constituée de trois couches distinctes. Couche huileuse/grasse

## 2.1 Couche huileuse

### 2.1.1 Présenté dans l'article

Cette couche supérieure est la seule couche où la réflectance est spéculaire. Au lieu d'utiliser le terme spéculaire de la BRDF de Phong, nous allons utiliser une BRDF physiquement plus plausible qui, selon l'article, donne un aspect plus naturel à la peau et plus fidèle à la réalité. Le modèle choisit est le modèle de Kelemen/Szirmay-Kalos. L'expression analytique du terme spéculaire de ce modèle est donné par la formule générale :

$$f_{r,spec} = P_{\vec{H}}(\vec{H}).F(\lambda, \vec{H}.\vec{L}).G(\vec{N}, \vec{L}, \vec{V})$$

où $P_{\vec{H}}(\vec{H})$ est la densité de probabilité du *halfway vector*, $F(\lambda, \vec{H}.\vec{L})$ est le terme de Fresnel et $G(\vec{N}, \vec{L}, \vec{V})$ est le terme géométrique. Dans le modèle de Kelemen/Szirmay-Kalos, ces termes sont définis comme étant :

$$P_{\vec{H}}(\vec{H}) = \frac{\exp\left(-\frac{\tan\alpha}{m}\right)^2}{m^2.\left(\vec{N}.\vec{H}\right)^4},$$

$$G(\vec{N}, \vec{L}, \vec{V}) = (\vec{L} + \vec{V})^2 = \vec{h}.\vec{h},$$

$$F(\lambda, \vec{H}.\vec{L}) = \left(1 - \vec{V}.\vec{H}\right)^5 + F_0\left(1 - \left(1 - \vec{V}.\vec{H}\right)^5\right)$$

où $\vec{h}$ est le *halfway vector* non normalisé et $F_0$ est la reflectance lorsque l'incidence est normale.

Selon l'article, pour rendre ce calcul plus rapide, il est possible de pré-calculer le terme $P_{\vec{H}}(\vec{H})$ et de le stocker dans une texture. Nous ne partageons (ou ne comprenons) pas cet avis des auteurs puisque, lorsque l'objet bouge dans la scène, les vecteurs de vision de la camera (V) et de lumière (L) changent aussi, ainsi que $P_{\vec{H}}(\vec{H})$, ce que rend nécessaire le calcul à nouveau de la texture, d'où la perte d'intérêt. Vu que l'article date de 2007, cet optimisation était peut être nécessaire à l'époque, mais avec les cartes graphiques modernes, ce pré-calcul n'est plus nécessaire, et notre implémentation n'en fait pas usage. Par contre, la technique présentée par l'article rend possible la variation du coefficient $m$ au sein de la texture, ce qui permet d'améliorer le rendu.

## 2.2 Sous couches

Les couches suivantes permettent de rendre compte du phénomène de diffusion de la lumière. Ces couches ont la propriété de renvoyer la lumière dans toutes les directions (isotropie), ce qui nous permet de ne pas prendre en comte la direction des rayons, mais seulement l'intensité reçue en chaque point. Deux modèles différents sont présentés. Le modèle du dipôle ( Jensen et al 2001) et le modèle du multi pole (Donner et Jensen 2005). L'article montre que le modèle du dipôle n'est pas satisfaisant et que le modèle du multi pole avec au moins trois couches l'est.

Nous avons approximé les courbes de diffusion par une combinaison linéaire de six gaussiennes, ce qui permet d'avoir à la fin une précision suffisante

pour notre champ d'application. Cette projection est définie par la formule :

$$\underset{G_n}{\text{proj}}\, M = \underset{G_{sum} \in G_n}{\arg\min} \left\{ \frac{\sqrt{\int_0^\infty r \left(R(r) - G_{sum}\right)^2 dr}}{\sqrt{\int_0^\infty r \left(R(r)\right)^2 dr}} \right\} \tag{1}$$

Pour vérifier la qualité du « fitting », on peut tracer rR(r) en fonction de r, et vérifier que l'aire sous la courbe est proportionnelle à la reflectance diffuse.

Après avoir calculé les gaussiennes, la technique pour rendre compte de la diffusion est convoler ces gaussiennes avec la texture, ce qui nous permet de calculer la texture d'irradiance, pour enfin combiner linéairement les résultats pour chacune des gaussienne et appliquer la texture final simulant le phénomène de diffusion. D'après les auteurs de l'article, après la projection de la texture sur la géométrie, celle ci subit une distorsion, ce qui rends nécessaire une correction (étirement) de la gaussienne avec laquelle nous allons la convoler. Vu que la courbure sur le visage reste assez faible (d'après une remarque faite par des auteurs) cette correction n'a pas trop d'impact dans notre cas. Nous avons donc décidé de ne pas implémenter cette partie et de nous concentrer sur le traitement de la texture.

Dans l'article, les auteurs proposent de faire ce floutage de texture "en ligne", en utilisant des Frame Buffer Object. Vu que cela rend le code plus compliqué, nous avons décidé de faire cette étape de manière "off line", juste en chargeant la texture afin de calculer la combinaison linéaire finale.

Pour le coloriage final, les auteurs proposent deux approches différentes. La première consiste à normaliser vers le blanc la combinaison linéaire des gaussiennes (Post Scatter Rendering) et la deuxième consiste à colorer les pixels lorsque nous calculons la combinaison linéaire des gaussiennes (Pre Scatter Rendering), ce qui donne une image plus « douce » et aussi plus convaincante. Nous avons implémenté la seconde technique de coloriage.

La suite de l'article présente des techniques pour simuler le transport de lumière dans les parties fines de la peau, ainsi que l'étude de la conservation de l'énergie que nous n'avons pas eu le temps d'implémenter ici.