Advanced Techniques for Realistic Real-Time Skin Rendering

Hamza El Rafoulli
Tiago Lobato Gimenes

18 février 2015

1 Introduction

Ce document montre et discute les problèmes trouvés lors de l'implémentation de l'algorithme de rendu en temps réel de la peau humaine. D'abord, il sera discuté l'implémentation modèle standard de NVIDIA et ses problèmes et/ou limitations pour ensuite détailler notre implémentation.

2 Modèle

Ce que fait la peau tellement difficile à rendre c'est sa propriété de que la lumière incidente est absorbé en partie et est émise dans un point différent. Pour faire un modèle cohérent il nous faut un modèle qui tienne conte de ces plusieurs couches et que les traite de façon particulières.

2.1 Couche Huileux

2.1.1 Présenté dans l'article

Cette couche est la seule couche où la réflectance est spéculaire. Au lieu d'utiliser le terme spéculaire de la BRDF de Phong, il vaux mieux d'utiliser celui d'une BRDF physique, vu que, selon les auteurs de l'article, elle augmente quelques lignes de code et donne un résultat expressive. Le modèle choisit est le modèle de Kelemen/Szirmay-Kalos. Le terme spéculaire dans ce type de modèle est généralement donné par :

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

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 à l'incidence normale.

Les auteurs de l'article disent que, pour rendre le calcul plus vite, nous pouvons faire une pré-calcul du terme $P_{\vec{H}}(\vec{H})$ et le estoquer dans une texture. Nous ne partageons pas les mêmes avis que les auteurs puisque, lorsque l'objet bouge dans la scène, les vecteurs de vision de la camera et de lumière vont changer, ainsi que $P_{\vec{H}}(\vec{H})$, ce que fait nécessaire un re-calcul de la texture, en prenant du temps de calcul et en faisant le code plus compliqué. Vu que l'article date de 2007, cet optimisation pouvait être nécessaire, mais avec les cartes modernes ce gain de performance ne vaut pas le cout d'un code plus compliqué, donc l'implémentation ici présent faire explicitement le calcul de la probabilité. Par contre, la technique présenté par l'article rend possible de varier m entre les régions de la texture, ce que rend le rendu plus naturel.

2.2 Sous couches

Les autres sous couches sont des couches avec une réflexion diffuse. Pour la peau, ces couches on la propriété de renvoyer la lumière dans tout les directions également, ce que nous permet de ne pas considérer la direction de la lumière mais seulement l'intensité reçue en chaque point.

Deux modèles différents ont été utilisé. Le modèle du dipole de Jensen et al 2001 et le modèle du multipole de Donner et Jensen 2005. L'article montre que le modèle du dipole n'est pas satisfaisante et que le modèle du multipole avec trois couches est un modèle satisfaisante.

L'article essaies de faire une simplification en faisant la projection du modèle du multipole sur un espace des gaussiennes $G_n = \left\{\frac{1}{2\pi\nu}e^{\frac{r^2}{2\nu}}\right\}$ tel qu'elles

aient un impulse unitaire. Cette projection est définie comme :

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

Pour vérifier la qualité du fitting il faut faire un plot de $rR(r)$ vs. r puisque l'aire sous la courbe est proportionnelle à la reflectance diffuse.

Après avoir calculé les gaussiennes, l'astuce pour rendre la diffusion c'est de faire une floutage avec ces gaussiennes sur la texture, ce que nous rend la texture d'irradiance, pour enfin pouvoir les combiner linéairement et appliquer la texture finale à la face.

Les auteurs aussi disent que, d'après un effet de projection, la convolution avec les gaussiennes dans l'espace texture ne sera pas bien convaincante dans l'espace 3D de l'objet, ce que rends nécessaire une correction d'étirement de la gaussienne avec laquelle nous allons faire convoler les textures. D'après le remarque des auteurs que dit que cette correction n'a pas trop d'impact dans le cas étudié vu que le visage n'a pas une courbure très grande, nous avons décidé de ne pas implémenter cette partie et se concentrer sur des parties plus centrales, comme la combinaison linéaire des gaussiennes.

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

Pour la coloriage finale les auteurs proposent deux approches différentes, une qui consiste en normaliser vers le blanc la combinaison linéaire des gaussiennes (Post Scatter Rendering) et une autre que consiste à colorer les pixels lorsque nous faisons la somme des gaussiennes (Pre Scatter Rendering), ce que rend l'image plus flou et aussi plus convaincante. Nous avons implémenté la dernière technique de coloriage.

La suite de l'article présente des techniques pour simuler le transport de lumière dans les parties fines de la peau, aussi comme des techniques pour conserver l'énergie qui ne sont pas implémenté ici.