

Rapport de projet
Synthèse réaliste d'image
par *raytracing* et *photon mapping*

François BEAUNE
beaune@aist.enst.fr

12 janvier 2004

Table des matières

| | | |
|----------|--|----------|
| 1 | Notes préliminaires | 2 |
| 2 | Introduction au <i>photon mapping</i> | 2 |
| 3 | Détails d'implémentation | 3 |
| 3.1 | Structure d'un photon | 3 |
| 3.2 | Construction de la <i>photon map</i> | 3 |
| 3.2.1 | Émission des photons | 3 |
| 3.2.2 | Propagation des photons à travers la scène | 4 |
| 3.2.3 | Interaction photon–surface diffuse | 4 |
| 3.2.4 | Interaction photon–surface spéculaire | 5 |
| 3.2.5 | Stockage des photons | 5 |
| 3.3 | Rendu final de la scène | 5 |
| 3.3.1 | <i>Raytracing de la scène</i> | 5 |
| 3.3.2 | Estimation de l'illumination indirecte | 6 |
| 4 | Résultats | 7 |
| 5 | Conclusions | 7 |
| | Références | 8 |

1 Notes préliminaires

Ce document n'a pas pour vocation de présenter tous les aspects du *photon mapping*, ni de donner tous les détails permettant de réaliser une implémentation complète. Il se contente de compléter la présentation intitulée « Présentation de *toxic*—Synthèse réaliste d'image par *raytracing* et *photon mapping* » et de mentionner quelques détails d'implémentation méritant l'attention.

Tous les pointeurs du type JENSEN : page X, section X.X.X font référence à l'ouvrage de Henrik Wann JENSEN intitulé *Realistic Image Synthesis Using Photon Mapping*.

Les quelques temps de calcul mentionnés ici et là sont donnés à titre purement indicatif et ne doivent pas faire l'objet de comparaison avec des temps de calcul obtenus par d'autres outils de rendu, le code utilisé étant un code de recherche et non un code de production. Tous les rendus ont été effectués sur un PC équipé d'un processeur INTEL Pentium IV à 2.8 Ghz et de 512 Mo de mémoire vive (FSB à 800 Mhz), le tout fonctionnant sous MICROSOFT Windows XP Professionnel.

2 Introduction au *photon mapping*

La technique du *photon mapping* a été développée par Henrik Wann JENSEN au cours de ses études de *PhD* dans les années 1993-1994. Les premiers articles mentionnant cette technique ont été publiés dans les années 1995-1996.

Le *photon mapping* se décompose en deux étapes distinctes et successives. La première étape consiste à émettre des photons depuis chaque source lumineuse de la scène. Chaque photon est propagé à travers la scène jusqu'à disparition (le photon est sorti de la scène) ou absorption par une surface. À chaque fois qu'un photon interagit avec une surface diffuse, il est stocké dans une structure particulière appelée *photon map*. Éventuellement, un nouveau photon est alors réémis.

La seconde étape consiste à rendre la scène de façon classique à l'aide d'un *raytracer* récursif (réflexions et réfractions spéculaires) et distribué (ombres douces, profondeur de champs). À chaque intersection entre un rayon primaire (directement issu de l'observateur) ou un rayon secondaire (issu d'une ou de plusieurs réflexions ou réfractions spéculaires) et une surface diffuse de la scène, une illumination est calculée. L'illumination d'un point de la scène se divise en deux contributions : l'illumination directe (la lumière provenant directement des sources lumineuses) et l'illumination indirecte (la lumière reçue après au moins une réflexion diffuse). L'illumination indirecte est calculée en estimant la densité de photons dans la *photon map* au point considéré.

3 Détails d'implémentation

3.1 Structure d'un photon

Un photon est une particule élémentaire de lumière. Il est émis depuis une source lumineuse. Après parcours à travers la scène, il est stocké dans une structure particulière appelée *photon map*. La représentation des photons et celle de la *photon map* sont complètement découplées de la géométrie de la scène. C'est ce découplage qui donne toute sa puissance au *photon mapping*.

Un photon est une petite structure comportant au moins deux informations essentielles :

- sa position dans la scène : c'est un point (x, y, z) de l'espace ;
- la puissance qu'il transporte : c'est une fraction de la puissance totale de la source lumineuse depuis laquelle il est émis.

Dans *toxic*, la position (x, y, z) du photon est codée par trois flottants sur 32 bits, soit 12 octets au total pour la position. La puissance transportée est codée en utilisant l'exposant commun de WARD (*WARD's shared-exponent RGB format*) [9] : quatre entiers (r, g, b, e) de 8 bits chacun suffisent, soit 4 octets au total pour la puissance. D'autres informations sont également stockées, comme la direction d'incidence (la direction suivant laquelle le photon est arrivé sur la surface à laquelle il appartient). Finalement, il est possible de représenter un photon de façon compacte par une structure de 20 octets. Voir JENSEN : page 69, section 6.2 pour plus de détails sur la représentation des photons.

3.2 Construction de la *photon map*

3.2.1 Émission des photons

Un certain nombre de photons est émis depuis chaque source lumineuse de la scène. Le nombre de photons émis depuis une source lumineuse donnée est un facteur de qualité (plus le nombre de photons émis est grand, plus l'illumination est finement calculée) puisque la puissance totale de la source est répartie entre tous les photons. Il peut être intéressant d'envoyer d'autant plus de photons que la source lumineuse est puissante, ceci afin d'obtenir, à l'issue de la phase d'émission, un nuage de photons d'intensités approximativement égales (ce qui contribue notablement à la qualité de l'illumination indirecte qui en résulte). On peut également envoyer davantage de photons depuis les sources lumineuses qui contribuent le plus à l'illumination indirecte, mais la difficulté réside alors dans le choix de ses sources lumineuses (aucune de ces deux techniques n'a pour le moment été implémenté dans *toxic*).

Dans la suite, par souci de clarté, toutes les sources lumineuses sont considérées comme étant parfaitement diffuses (lambertiennes). *toxic* est plus souple puisqu'il autorise l'utilisation de profils d'émission arbitraires.

Pour plus de détails sur l'émission des photons (*photon tracing*), se référer à JENSEN : page 55, section 5.

Émission depuis une source ponctuelle Si elle n'est pas physiquement plausible, c'est la source lumineuse la plus simple : les photons partent tous du même point (la position de la source) et sont émis aléatoirement et uniformément dans toutes les directions.

Émission depuis une source étendue C'est la source de lumière la plus répandue, puisque c'est la seule source physiquement plausible. L'émission de chaque photon se compose de deux étapes : un point est d'abord choisi aléatoirement et uniformément sur la surface de la source, puis une direction est choisie aléatoirement et uniformément dans l'hémisphère soutenue par la surface au point considéré.

3.2.2 Propagation des photons à travers la scène

En l'absence de milieu non-homogène (*participating media*, voir JENSEN : page 113, section 10 à ce sujet), le milieu ambiant est considéré comme étant le vide. Les photons voyagent librement dans ce milieu, en ligne droite, sans subir d'altération.

3.2.3 Interaction photon–surface diffuse

Pour simplifier, on ne considère ici que des surfaces parfaitement diffuses. *toxic* n'impose aucune contrainte sur les *BRDF* : toute *BRDF* ou combinaison de *BRDF* peut être utilisée pour caractériser les propriétés de réflexion d'une surface. Le détail des algorithmes permettant l'utilisation de *BRDF* quelconques est hors de propos de ce document, qui se contente de donner un aperçu de la méthode dans les cas simples.

Lorsqu'un photon arrive sur une surface diffuse, il est stocké dans la *photon map* à la position de l'impact avec la surface¹.

Si l'on suppose que la surface (lambertienne) sur laquelle le photon est arrivé a une réflectance ρ (comprise entre 0 et 1), on peut utiliser la technique de la roulette russe [8] [1] pour décider si le photon est absorbé ou réfléchi. Cette technique donne toujours le bon résultat, quelque soit la définition choisie pour p . Elle permet de concentrer les ressources de calcul sur les photons importants et de supprimer les photons peu importants.

La probabilité p que le photon soit réfléchi est posée égale à ρ . On procède alors comme suit : on tire un nombre aléatoire ξ entre 0 et 1. Si ξ est inférieur à p , le photon est réfléchi avec la même puissance que le photon

¹Il n'est pas très clair, à la lecture de JENSEN et autres articles, si tous les photons arrivant sur une surface diffuse doivent être stockés dans la *photon map*, ou bien si seuls les photons ayant subi au moins une réflexion (diffuse ou spéculaire) doivent être stockés.

incident, sinon il est absorbé et on émet un nouveau photon depuis la source lumineuse.

3.2.4 Interaction photon–surface spéculaire

De même que dans le cas des surfaces diffuses, on ne considère ici que des surfaces parfaitement spéculaires.

Un photon arrivant sur une surface spéculaire voit sa puissance multipliée par la réflectance de la surface spéculaire avant d’être réfléchi suivant la normale à la surface. Aucun photon n’est stocké dans la *photon map*.

3.2.5 Stockage des photons

Dans la phase de lancé et de tracé des photons, les photons arrivant sur une surface diffuse sont stockés les uns à la suite des autres dans un tableau linéaire de dimension 1.

Durant le rendu de l’image, la *photon map* est une structure statique qui permet d’évaluer l’illumination globale en tout point du modèle. L’évaluation de l’illumination est basée sur la recherche des M plus proches photons autour du point considéré dans la scène. Il est donc essentiel que ce type de requête soit la plus rapide possible, mais également que la structure d’organisation des photons soit la moins gourmande possible en mémoire, puisqu’on projette d’utiliser jusqu’à plusieurs millions de photons pour une seule image. Il est clair que la structure de tableau linéaire à une dimension, si elle possède l’empreinte mémoire minimale, n’est pas adaptée à la recherche des plus proches photons d’un point donné.

Une structure plus adaptée est le *kd-tree*. Un *kd-tree* est un arbre binaire multidimensionnel (dans notre cas, de dimension 3), où chaque noeud contient un photon et partitionne une des trois dimensions de l’espace en deux sous-espaces. Il a été montré [2] que, dans un *kd-tree* contenant N photons, la recherche des M plus proches photons autour d’un point donné de l’espace a une complexité de l’ordre de $O(M + \log N)$. En outre, la structure de *kd-tree* peut être représentée de façon extrêmement compacte [7], sans aucun *overhead* par rapport à un tableau.

Se référer à JENSEN : page 67, section 6.1 et JENSEN : page 70, section 6.3 pour plus de détails concernant la construction du *kd-tree*, et à JENSEN : page 72, section 6.4 pour l’algorithme de recherche des plus proches photons autour d’un point donné de l’espace.

3.3 Rendu final de la scène

3.3.1 Raytracing de la scène

Le rendu d’une scène en *photon mapping* est effectué par un *raytracer* classique. Dans *toxic*, le *raytracer* est récursif [10] — rendu des surfaces

spéculaires : réflexion, réfraction — et stochastique [3] — ombres douces, profondeur de champ, flou de mouvement, réflexions floues, translucence, etc.

Le coeur d'un *raytracer* réside dans la routine d'intersection entre un rayon et les objets de la scène. Il n'est pas rare que le *raytracer* lance de plusieurs millions à plusieurs *milliards* de rayons à travers une scène, notamment pour l'évaluation de l'illumination (directe ou indirecte). Il est donc primordial de rendre le code d'intersection rayon-scène le plus efficace possible afin d'obtenir des temps de calcul raisonnables.

toxic organise les objets dans une structure d'*octree*, si cela en vaut la peine (c'est-à-dire si le nombre d'objets est suffisant pour que l'*overhead* dû à l'*octree* soit négligeable). De même, il peut organiser les triangles d'un *mesh* dans un *octree* propre au *mesh* (si le nombre de triangles est suffisant). L'algorithme utilisé pour traverser les *octrees* est un des plus efficaces actuellement. Il est décrit dans [6].

Le code d'intersection rayon-triangle utilisé dans *toxic* est tiré de [5]. C'est également un des codes d'intersection rayon-triangle les plus rapides.

3.3.2 Estimation de l'illumination indirecte

Intuitivement, l'estimation de l'illumination indirecte en un point donné de la scène peut se résumer à déterminer, en ce point, la densité de photons dans la *photon map* : c'est une visualisation « directe » de la *photon map*. Il suffit pour cela de rechercher les M plus proches photons autour du point auquel on souhaite connaître l'illumination indirecte, d'accumuler leur puissance puis de la diviser par la surface du disque englobant ces photons (la surface est supposée localement plane autour du point considéré ; pour que cela se vérifie en pratique, il faut s'assurer que la *photon map* possède suffisamment de photons).

Malheureusement, même avec un nombre très élevé de photons, l'illumination obtenue par cette méthode se révèle de faible qualité : on note un bruit basse fréquence dans l'illumination, les coins et les arêtes des objets sont anormalement sombres et de nombreux artefacts apparaissent ici et là.

La solution à ce problème est malheureusement très couteuse en temps de calcul. Elle consiste à ajouter un niveau d'indirection dans la consultation de la *photon map*. L'illumination n'est plus estimée directement au point considéré. À la place, on évalue l'illumination reçue en ce point à travers l'hémisphère que la surface soutient en ce point. En pratique, on lance des rayons depuis le point à éclairer et en direction d'un certain nombre de points uniformément choisis sur la surface de l'hémisphère. La densité de photons est estimée aux points d'intersection entre ces rayons et la scène, et l'illumination du point initialement considéré est alors la moyenne des illuminations de tous ces nouveaux points. Ce procédé porte communément le nom de *final gathering*.

4 Résultats

Se reporter à la fin de la présentation *PowerPoint* pour les images rendues avec *toxic*.

5 Conclusions

Le *photon mapping* est une technique qui résoud intégralement l'équation de rendu [4] tout en combinant simplicité, robustesse et performances. Le principal apport de cette technique est le total découplage entre la *photon map* et la géométrie de la scène. Ce découplage permet le rendu en illumination globale de surfaces procédurales (surfaces fractales par exemple) sans qu'une représentation polygonale n'ait besoin d'être générée préalablement (ce qui d'ailleurs n'est pas toujours possible). En outre, le *photon mapping* autorise l'utilisation de *BRDF* arbitraires (et notamment les combinaisons de *BRDF*).

Pour résumer, le *photon mapping* permet de résoudre efficacement l'équation de rendu dans son intégralité (prise en compte de tous les chemins lumineux) avec la généralité, la robustesse et la simplicité du *raytracing* de Monte Carlo.

Références

- [1] James Arvo and David B. Kirk. “Particle Transport and Image Synthesis.” *Computer Graphics (Proc. SIGGRAPH '90)* 24(4): 63–66 (August 1990).
- [2] Jon L. Bentley. “Multidimensional binary search trees in database applications.” *IEEE Trans. on Soft. Eng.* 5(4): 333–340 (July 1979).
- [3] Robert L. Cook, Thomas Porter, and Loren Carpenter. “Distributed Ray Tracing.” *Computer Graphics (Proc. SIGGRAPH '84)* 18(3): 137–145 (July 1984).
- [4] James T. Kajiya. “The rendering equation.” *Computer Graphics (Proc. SIGGRAPH '86)* 20(4): 143–150 (August 1986).
- [5] Tomas Möller and Ben Trumbore. “Fast, minimum storage ray-triangle intersection.” *Journal of graphics tools*, 2(1): 21–28, 1997.
- [6] J. Revelles, C. Urena, and M. Lastra. “An Efficient Parametric Algorithm for Octree Traversal.” In *The 8th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Media*, 2000.
- [7] Robert Sedgewick. *Algorithms in C++*. Reading, MA: Addison-Wesley, 1992.
- [8] Jerome Spanier and Ely Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Reading, MA: Addison-Wesley, 1969.
- [9] Greg Ward. “Real pixels.” In *Graphics Gems II*, edited by James Arvo, pp. 80–83, San Diego: Academic Press, 1991.
- [10] Turner Whitted. “An improved illumination model for shaded display.” *Communications of the ACM* 23(6): 343–349 (June 1980).