REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique
Laboratoire LESIA

N° d'ordre:.....

Série:.....

Thèse

Présentée en vue de l'obtention du diplôme de
DOCTORAT EN SCIENCE

Spécialité : Informatique

Intitulée

Illumination globale efficace de scènes 3D à base de points: application aux fluides basés particules

par

Ali BEDDIAF

Soutenue le 26/06/2018 devant le jury composé de:

NourEddine DJEDI	Professeur	Université de Biskra	Président
Mohamed Chaouki BABAHENINI	Professeur	Université de Biskra	Rapporteur
Amar BALLA	Professeur	ESI Oued Smar Alger	Examinateur
Mohamed BENMOHAMMED	Professeur	Université de Constantine 2	Examinateur
Azeddine BILAMI	Professeur	Université de Batna 2	Examinateur
Leila DJEROU	MCA	Université de Biskra	Examinatrice

A ma chère ***mère,***

A l'âme de ***mon père,***

A la mémoire de notre collègue et sœur ***Docteur Nesrine Ouannes,***

et à mes vrais amis, je dédie ce modeste travail.

Remerciements

Je remercie tout d'abord **Allah** pour m'avoir donné le courage, la santé et la capacité pour accomplir ce travail.

Ce travail n'aurait pas pu être achevé sans la contribution de plusieurs personnes que je remercie.

Mes chaleureux remerciements vont à mon directeur de thèse, Docteur Mohamed Chaouki BABAHENINI, professeur à l'université de Biskra, pour m'avoir encadré durant la préparation du magister et du présent doctorat qui se sont étalés tous les deux sur 8 ans à peu près de travail fructueux. Ses conseils judicieux, sa grande disponibilité et aussi ses liens scientifiques avec des chercheurs externes, m'ont tous permis de progresser, et faire des mobilités sans lesquelles ce travail ne sera pas terminé.

Je remercie le président du jury Docteur NourEddine DJEDI, professeur à l'université de Biskra qui nous a fait le grand honneur de présider le jury.

Un grand merci au Docteur Amar BALLA, professeur à l'ESI (École nationale Supérieure d'Informatique) pour l'intérêt qu'il a porté à ce travail et pour le temps qu'il a consacré en acceptant d'être examinateur.

Mes remerciements vont aussi au Docteur Mohamed BENMOHAMMED, professeur à l'université de Constantine 2, d'avoir bien voulu examiner ce travail de thèse.

Je remercie également mon collègue Docteur Azeddine BILAMI, professeur à l'université de Batna 2, pour m'avoir fait l'honneur d'évaluer mon travail.

Je remercie aussi Docteur Leila DJEROU, maître de conférences à l'université de Biskra pour l'intérêt qu'elle a bien voulu porter à ce travail en acceptant d'être examinatrice.

Sans oublier de passer mes sincères remerciements à tous ceux qui ont participé de près ou de loin à l'aboutissement de ce travail.

Résumé

Les images rendues en utilisant des algorithmes d'illumination globale sont plus photo-réalistes que les images rendues en utilisant des algorithmes d'illumination locale. Cependant, ils sont également beaucoup plus lents et plus coûteux en termes de temps de calcul. Une approche commune est de calculer l'illumination globale d'une scène et de stocker cette information avec la géométrie (principe de radiance). Ces données stockées peuvent alors être employées pour produire des images de différents points de vue, produisant ainsi différentes images d'une même scène sans devoir refaire un calcul d'éclairage à chacun des changements d'angle.

Les technologies récentes d'acquisition de données en trois dimensions fournissent un grand nombre de points non-structurés en trois dimensions ainsi que le temps de calcul généré par les modèles polygonaux complexe, ont motivé l'apparition du rendu à base de points, ainsi chaque point est considéré indépendamment et possède ses propres caractéristiques.

La thèse s'intéresse aux modèles de rendu réaliste en synthèse d'images, en particulier à la simulation de fluides basés particules. Leur intérêt est de calculer précisément une solution d'illumination qui permet notamment de produire des images réalistes basée sur les interactions complexes entre la lumière et les matériaux dans le fluide.

Après un tour d'horizon des différentes méthodes employées pour le rendu d'images, nous avons en premier lieu présenter notre première contribution consistant en une technique de lancer de rayon adapté aux points reconstruits par des splats. Dans une seconde étape, nous nous sommes intéressés par les méthodes de simulation de fluides basés particules où notre principale contribution consiste à éviter la transformation des particules en une grille, les particules sont alors utilisées comme entrée pour définir directement une surface implicite. L'objectif est de manipuler le fluide comme un milieu participant hétérogène avec des frontières réfractives.

Mots clés : *Représentation à base de points, Illumination globale, niveaux de détail, rendu 3D réaliste, GPU*.

ملخص

الصور المقدمة باستخدام خوارزميات الإضاءة الشاملة هي أكثر واقعية من الصور المقدمة باستخدام خوارزميات الإضاءة المحلية. ومع ذلك ، فهي أيضا أبطأ بكثير وأكثر تكلفة من حيث وقت الحوسبة. الطريقة الشائعة هي حساب الإضاءة الشاملة للمشهد وتخزين هذه المعلومات مع الهندسة (مبدأ الإشعاع). ويمكن بعد ذلك استخدام هذه البيانات المخزنة لإنتاج صور من وجهات نظر مختلفة ، وبالتالي إنتاج صور مختلفة لنفس المشهد دون الحاجة إلى إعادة حساب الإضاءة في كل من التغييرات الزاوية.

توفر تقنيات اكتساب البيانات الحديثة ثلاثية الأبعاد عددًا كبيرًا من النقاط ثلاثية الأبعاد غير المهيكلة بالإضافة إلى وقت الحساب الناتج عن النماذج متعددة الأضلاع المعقدة ، حفز ظهور العرض المستند إلى نقطة ، حيث تعتبركل نقطة مستقلة ولها خصائصها الخاصة.

تركز الأطروحة على نماذج التقديم الواقعي في تركيب الصورة ، ولا سيما محاكاة السوائل القائمة على الجسيمات. تكمن الأهمية في حساب حل الإضاءة الذي يسمح على وجه الخصوص بإنتاج صور واقعية تعتمد على التفاعلات المعقدة بين الضوء والمواد في السائل.

بعد نظرة عامة على الطرق المختلفة المستخدمة لتقديم الصورة ، نقدم أولاً أول مساهمة تتمثل في تقنية تتبع الشعاع تتكيف مع النقاط التي يعاد بناؤها عن طريق الدعامات. في الخطوة الثانية ، نحن مهتمون بأساليب محاكاة السوائل المعتمدة على الجسيمات حيث تتمثل مساهمتنا الرئيسية في تجنب تحويل الجسيمات إلى شبكة ، ثم تُستخدم الجسيمات كمدخل لتعريف السطح الضمني بشكل مباشر. الهدف هو معالجة السائل كوسيط المشاركة غير المتجانسة مع حدود الانكسار.

الكلمات المفتاحية : التمثيل القائم على نقطة، والإضاءة الشاملة ، ومستويات التفاصيل، الرسم ثلاثي الأبعاد الواقعي، وحدة معالجة الرسوميات.

Abstract

Images rendered using global illumination algorithms are more photorealistic than images rendered using local illumination algorithms. However, they are also much slower and more expensive in terms of computing time. A common approach is to compute the global illumination of a scene and to store this information with geometry (radiance principle). This stored data can then be used to produce images from different points of view, thus producing different images of the same scene without having to redo a lighting calculation at each of the angle changes.

Recent three-dimensional data acquisition technologies provide a large number of unstructured three-dimensional points as well as the computation time generated by complex polygonal models, motivated the appearance of point-based rendering, so each point is considered independently and has its own characteristics.

The thesis focuses on realistic rendering models in image synthesis, more specifically, the simulation of particle-based fluids. Their interest is to precisely calculate an illumination solution that allows to produce realistic images based on the complex interactions between light and materials in the fluid.

After an overview of the various methods used for image rendering, we present our first contribution that consists of a ray tracing technique adapted to the points reconstructed by splats. In a second step, we are interested in particle based fluid simulation methods where our main contribution is to avoid the transformation of particles into a grid, the particles are then used as input to directly define an implicit surface. The goal is to manipulate the fluid as a heterogeneous participating medium with refractive boundaries.

Keywords : *Point-based representation, global illumination, levels of detail, realistic 3D rendering, GPU*.

Publications et Communications

Communications nationales

1. A. Beddiaf, M. C. Babahenini, "**Towards accurate splat-based ray tracing of point-based objects**", IGVA'2013, Biskra, Algérie, 2013.
2. A. Beddiaf, M. C. Babahenini, "**Path tracing : quadrature numérique de Gauss**", IGVA'2014, Biskra, Algérie, 2014.

Communications internationales

1. A. Beddiaf, M.C. Babahenini, "**Efficient accelerated ray tracing of point-based objects**", ICMCS'2014, Marrakech, Maroc, 2014.
2. A. Beddiaf, M.C. Babahenini, "**An improved splat-based ray tracing for point-based objects**", ISPS'2015, Alger, Algérie, 2015.

Publications

1. A. Beddiaf, M.C. Babahenini, "**Physically-based rendering of particle-based fluids with light transport effects**", *3D Research*, 2018.

Table des Matières

Liste des Figures

Liste des Tableaux

Introduction générale

En infographie, le terme illumination globale (GI: Global illumination en anglais) désigne la manière dont la lumière est rebondie d'une surface sur une autre (éclairage indirect) au lieu de se limiter uniquement sur l'effet direct de la lumière sur une surface (éclairage direct). La prise en compte de l'éclairage indirect permet de donner lieu à des effets rendant la scène beaucoup plus réaliste et consistante parce qu'en réalité l'apparence d'un objet est affectée par son entourage (**Figure 1**).

Figure 1. Effets de l'éclairage indirect calculé par une méthode d'illumination globale.

Habituellement, dans le domaine des jeux vidéo à titre d'exemple, on se limite à faire un éclairage direct parce que le calcul de l'éclairage indirect nécessite un temps important et ceci ne convient pas aux applications temps réel. Le cout engendré par les méthodes d'illumination globale qui calculent l'éclairage indirecte d'une manière physiquement exacte (lancement et suivi d'une infinité de rayons lumineux réfléchis lorsqu'un rayon percute une surface diffuse par exemple) est élevé en matière de temps. Ceci peut être considéré comme un vrai frein pour plusieurs domaines d'application. Comme remède à ce problème, soit on établit un rapport qualité/prix par la réduction du nombre de rayons calculés et on aura par la suite une image plus ou moins réaliste dans un temps raisonnable. Soit on utilise des méthodes d'illumination globale non exactes mais approchées qui donnent des images ayant bien des effets d'éclairage indirect dans un temps réduit.

Usuellement, les scènes 3D sont constituées à base d'un ensemble de primitives qui sont des polygones, ou souvent des triangles. Récemment d'autres primitives de modélisation 3D sont apparues, permettant d'avoir de différentes représentations, dont la représentation à base de points fait partie.

Contrairement à la représentation classique polygonale, où les concepteurs jouent le rôle le plus important de modélisation des objets 3D (qui est considéré comme une œuvre d'art), la représentation par points est donnée simplement à l'issue d'une phase dite d'acquisition (souvent faite via des appareils sophistiqués d'acquisition comme les scanners 3D). Il est à noter que les points peuvent être aussi donnés par d'autres programmes (comme dans notre travail, les points sont donnés par des simulateurs), voir **Figure 2**.

Figure 2. Logiciel de modélisation polygonale (à gauche), un scanner 3D (à droite).

Motivation

La représentation à base de points a connu de véritables avantages par rapport à la représentation classique à base de polygones (comme la facilité de numérisation des objets du monde réel d'une manière automatique), quoique la dérivation et le développement des techniques permettant le rendu efficace et réaliste de ce type de scènes restent un axe de recherche actif dans lequel plusieurs travaux ont été publiés.

Dans ce travail de thèse, nous nous attaquons au problème de calcul d'images photo-réalistes via les techniques d'illumination dite globales. Ces dernières promettent de donner en sortie des images les plus proches à la réalité parce qu'elles respectent les lois de la physique de transport de lumière dans la nature. Cette problématique a été déjà abordée bien avant le présent travail par de nombreuses recherches, mais nous focalisons plutôt sur un type de scènes bien particulier qui est les scènes à base de nuages de points. Ces nuages de points sont souvent fournis suite à une phase d'acquisition (via des scanners 3D).

Dans notre étude, ces points nous seront fournis plutôt suite à une phase de simulation physique de mouvement des fluides (via des solveurs de fluides). Il s'agit bien d'un ensemble de points tridimensionnels (ayant trois composantes x, y et z) associés à d'autres attributs (comme la couleur, etc.). Le rendu réaliste de ces fluides, utilisant des techniques d'illumination globale, s'avère crucial, car les phénomènes lumineux compliqués qui se produisent (réflexion, réfraction multiple, diffusion multiple, dispersion, absorption) doivent tous être bien pris en compte pour que le fluide soit vu comme étant réel.

Contributions

Dans une première partie, nous avons visé l'étude et l'implémentation d'un lancer de rayon pour les objets à base de points où les points sont reconstruits localement par des splats. Nos principales contributions sont :

- Accélération de l'algorithme du lancer de rayon proposé sur les splats, en minimisant le coût de la phase de calcul d'intersections rayon-splat par l'adaptation de trois techniques classiques d'accélération au cas des splats, à savoir: les volumes englobants, les subdivisions spatiales, et les accélérations géométriques.
- Amélioration de la qualité surfacique donnée par notre lancer de rayon sur les splats par la résolution des conflits présents lors du calcul d'intersections rayon-splat, et qui sont dus au phénomène de chevauchement et de superposition des splats. Ces conflits ont été enlevés via l'introduction de quelques règles basées sur l'étude géométrique des courbures discrètes formées par les splats. Les résultats ont montré une amélioration de la qualité visuelle de l'image surtout sur la silhouette des objets.

Dans la deuxième partie de nos travaux de recherche, nous nous sommes intéressés à l'implémentation de la technique du path tracing dans sa version dédiée aux milieux participants pour les points issus d'un simulateur physique de mouvement de fluides. Ces points représentent les positions des particules de fluides, qui sont associées à d'autres attributs. Dans cette implémentation, la surface n'est pas reconstruite par des splats mais plutôt par des surfaces implicites (c'est ce qui est recommandé dans le cas des objets mous: fluides). Le choix de la technique du path tracing stochastique est motivé par le but visé de faire un rendu réaliste, car le rendu des fluides présente des phénomènes lumineux compliqués qui doivent être pris en compte (tels que: la diffusion multiple, **TIR**: Total Internal Reflection, etc.).

Organisation de la thèse

Notre thèse est divisée en cinq chapitres, comme suit :

- **Le premier chapitre** comporte trois sections où chacune englobe une catégorie de techniques de rendu à base de points. La première catégorie cite des techniques basées sur la reconstruction locale dans l'espace objet. La deuxième met en évidence des techniques basées sur le lancer de rayon. La troisième catégorie inclut des techniques basées sur la reconstruction dans l'espace écran.
- Dans **le deuxième chapitre,** nous présentons notre première contribution qui consiste en un lancer de rayon dédié aux objets à base de points reconstruits localement par des splats, qui est amélioré sur l'axe rapidité de calcul (quantitatif) et sur l'axe qualitatif (qualité de la surface ou précisément la silhouette des objets).
- Dans **le troisième chapitre**, nous détaillons les différentes techniques d'illumination globale (physiques et approchées) permettant le calcul d'images réalistes comportant et l'éclairage direct et indirect.
- **Le quatrième chapitre**, est dédié à la présentation, d'un état de l'art sur les techniques de simulation de mouvement des fluides, sur lequel notre contribution principale va porter.
- Dans **le cinquième chapitre**, nous montrons notre contribution principale, qui consiste au rendu physique des données (points tridimensionnels associés à des attributs additionnels) issues d'une phase de simulation physique de mouvement pour le cas des fluides.
- Finalement, nous terminerons par une conclusion générale dans laquelle, nous présentons une synthèse des contributions apportées, ainsi que par des perspectives possibles pour des travaux futurs.

PARTIE I :

Problématique d'illumination pour les scènes à base de points

Chapitre 1 : Rendu à base de points

1.1. Introduction

Avec l'avènement des scanners 3D, les objets du monde réel peuvent être acquis automatiquement sous la forme d'un modèle décrit par un nuage discret de points où chaque point possède des informations, à savoir, la position, la couleur, la normale ou encore les propriétés du matériau. Afin qu'un algorithme de rendu quelconque puisse être appliqué, les points nécessitent probablement de passer par des étapes d'optimisation pour éliminer le bruit et les redondances souvent présents, et qui sont dus au processus d'acquisition.

Une reconstruction de la surface (en un maillage de polygones à titre d'exemple) est requise, à priori, pour pouvoir appliquer la plupart des algorithmes de rendu. Tandis que d'autres algorithmes n'exigent pas une reconstruction préalable, et gardent le point comme primitive de rendu.

L'utilisation du point comme primitive de rendu date de 1985 (Levoy et Whitted 1985) bien que le rendu de certains d'objets (tels que la fumée, les nuages, le feu ou les arbres) sous forme de points remonte à la fin années soixante dix (Blinn 1982, Csuri, et al. 1979, Reeves 1983, Smith 1984).

Ce chapitre s'articule en trois sections où chacune englobe une catégorie de techniques de rendu à base de points. La première catégorie cite des techniques basées sur la reconstruction locale dans l'espace objet. La deuxième catégorie met en évidence des techniques basées sur le lancer de rayon. La troisième catégorie inclut des techniques basées sur la reconstruction dans l'espace écran. Une conclusion est donnée à la fin en mettant l'accent sur un ensemble de recommandations pouvant être utiles pour tout système de rendu à base de points.

1.2. Techniques basées sur la reconstruction locale dans l'espace objet

1.2.1. Rendu des points discrets

Quand la complexité géométrique des scènes augmente (en matière de nombre de polygones), le rendu devient délicat (élimination des surfaces cachées, le mélange de couleur, etc.). Levoy et Whitted (Levoy et Whitted 1985) baptisent dans ce travail, une nouvelle primitive de rendu qui est le point, qui découple la phase de modélisation de celle de rendu. Selon cette approche, les points, issus d'une phase préalable de conversion (échantillonnage) de la géométrie, représentant la scène, ont exactement trois attributs (position, couleur et opacité) et sont envoyés à un pipeline graphique. Les points sont projetés sur l'écran tout simplement grâce à une multiplication par la matrice de projection perspective. Afin de boucher le vide dû à l'espacement entre les points, autrement dit les trous, présents dans l'image, une opération de

reconstruction peut être appliquée, c'est un filtrage (un noyau gaussien) dont le rayon est inversement proportionnel à la distance entre le point et sa projection (c'est naturel, plus le point s'éloigne plus sa contribution sur l'écran diminue). Le rayon du filtre doit être suffisamment large pour garantir l'absence des trous et aussi le mélange de sa couleur avec celle des points voisins.

Afin de résoudre les conflits dus à l'intersection et au chevauchement de différentes surfaces de la scène durant la visualisation, l'élimination des surfaces cachées est assurée via l'utilisation de l'algorithme du Z-buffer. Deux cas sont possibles, mélange de couleurs ou occultation (visibilité) suivant si il s'agit des points d'une même surface ou de surfaces différentes respectivement, quoique le problème réside dans le fait qu'il est pas évident de dire qu'un tel point fait partie de telle surface. Ce problème est résolu en commençant d'abord par le mélange de couleurs pour des groupes de points qui appartiennent chacun à la même surface, et qui sont stockés ensuite dans des tampons. Après, le test de visibilité est effectué entre les tampons deux à deux du plus loin au plus proche par rapport à la caméra (**Figure 3**).

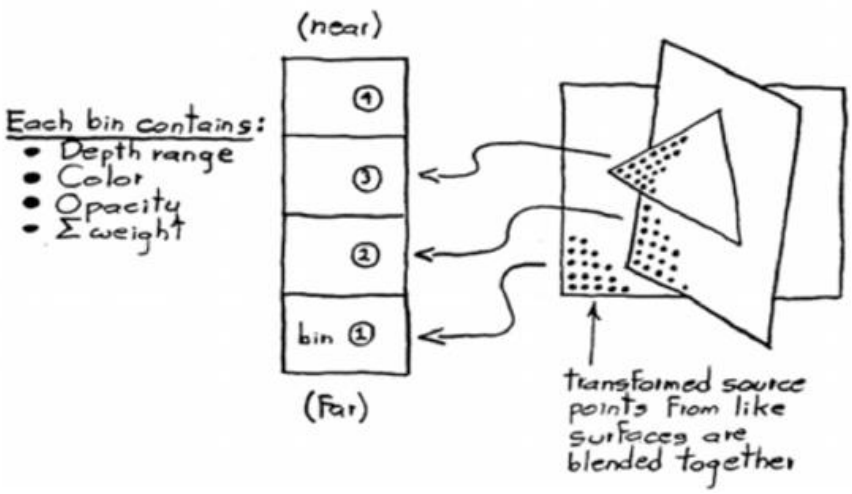


Figure 3. Mélange de couleurs et test de visibilité (Levoy et Whitted 1985).

1.2.2. Rendu basé sur les images de profondeurs : Surfel

Pfister et al. (Pfister, et al. 2000) ont proposé un nouveau paradigme de rendu des scènes complexes avec un taux d'affichage interactif. Premièrement la scène est discrétisée en un ensemble de points sans aucune connectivité. Chaque point, appelé surfel (SURFace ELement) est associé à une texture de couleur pré-filtrée sous la forme d'une ellipse qui correspond à la projection du disque tangent à la surface au point en question, sur l'espace image, suivie d'une application d'un filtre EWA (Elliptical Weighted Average). La superposition des surfels donne à la fin une surface sans présence de trous.

À partir d'un maillage de départ, la technique procèdent à l'échantillonnage de la surface en un ensemble de surfels selon le LDC (Layer Depth Cube), c'est une grille hiérarchique (octree) alignée avec l'espace écran permettant de discrétiser la surface en traçant des rayons (ray casting) depuis les trois directions orthogonales de la boite engloblante de la scène (**Figure 4**), ce qui donne lieu à trois images de profondeur LDIs (Layer Depth Image) orthogonales. Cet échantillonnage a l'avantage de donner une densité optimale des surfels contrairement aux méthodes d'échantillonnage qui travaillent dans l'espace objet (puisque la densité d'échantillonnage correspond à la résolution de l'image de sortie). La résolution de la grille peut être ajustée afin qu'au moins un surfel est projeté sur un pixel de l'écran (pour un rendu efficace). Dans ce cas, le rayon des surfels doit être au minimum $\sqrt[3]{h_0}$ (h_0 est l'espacement entre les pixels du LDI) afin de garantir le chevauchement entre un surfel et ses voisins directs aussi bien dans l'espace objet que dans l'espace image. La résolution de la grille LDC peut être aussi augmentée pour un rendu plutôt de haute qualité (avec un peu plus de temps). L'intérêt de l'échantillonnage LDC est double; une combinaison (réduction) assez facile des trois LDIs en une seule (les surfels des trois LDIs se voient déplacer, autrement dit ré-échantillonner, afin d'être alignés à la grille LDC, comme montre la figure **Figure 5**) ce qui permet de diminuer les points redondants présents à la fois dans plus d'une LDI, ainsi une structure hiérarchique contenant les surfels est déduite automatiquement. L'hiérarchie sert à éliminer les surfaces cachées et celles en dehors le la pyramide de vue (en utilisant des cônes de visibilité (Grossman et Dally 1998), comme montre la figure **Figure 6**) en parcourant l'arbre des blocs du LDC d'une manière hiérarchique, tout en effectuant le rendu par projection des surfels visibles.

Comme suite de ce travail, Zwicker et al. (Zwicker, et al. 2001) ont rajouté à la primitive de rendu (surfel), en plus du filtre de reconstruction à l'espace objet, un autre filtre passe bas dans l'espace écran ce qui donne un filtrage de ré-échantillonnage anisotrope de haute qualité (une nouvelle formulation en espace écran du filtre EWA).

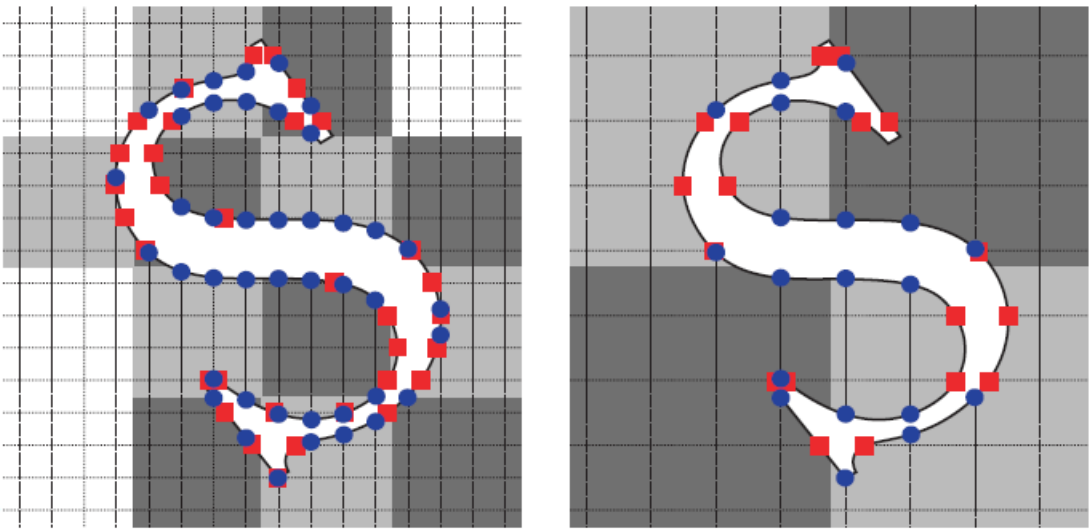


Figure 4. Représentation 2D du LDC à deux niveaux différents (Pfister, et al. 2000).

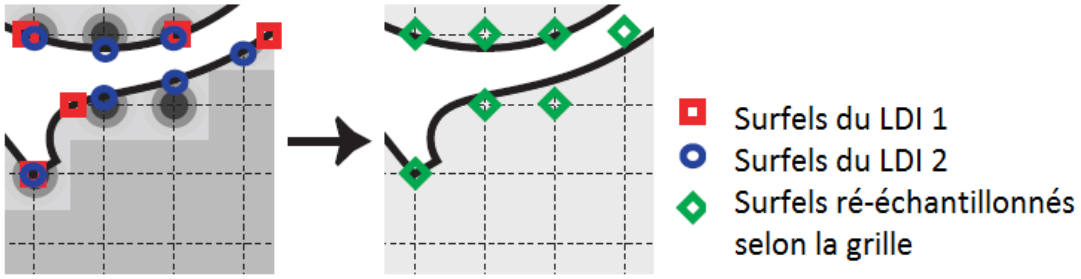


Figure 5. Réduction 3-à-1 (Pfister, et al. 2000).

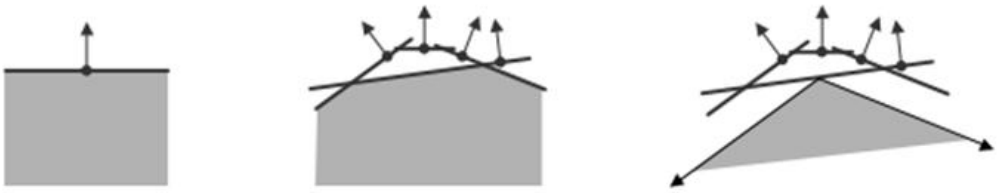


Figure 6. Cône de visibilité.

1.2.3. Rendu progressif accéléré: Qsplat

Rusinkiewicz et Levoy (Rusinkiewicz et Levoy 2000) ont proposé un système de rendu basé points accéléré par une structure de données hiérarchique de sphères englobantes. Cette structure permet d'avoir un rendu efficace de point de vue élimination des surfaces cachées (utilisation des cônes

de normales (Shirmun et Abi-Ezzi 1993)) et celles en dehors de la pyramide de vue, en plus de ça, elle permet d'avoir plusieurs niveaux de détails (LOD). L'hiérarchie peut être générée à partir d'un maillage polygonale, mais aussi des voxels ou des nuages de points, puis stockée dans le disque dans une phase de prétraitement. Le parcours est fait en largeur (**Figure 9**) tout en abandonnant les sous-arbres non visibles. Afin d'avoir un rendu efficace, les sous-arbres de l'hiérarchie doivent être parcourus si l'aire de la sphère correspondante au nœud du sommet projetée sur l'écran dépasse un seuil prédéfini (ceci arrive quand le sommet couvre plus qu'un pixel). Ce seuil peut être ajusté selon le temps de rendu désiré.

La construction de l'hiérarchie (**Figure 7**) suit les étapes suivantes :

1. Construction des sphères
   La taille de chaque sphère (diamètre) doit être la distance maximale entre le point et tous les autres points directement connectés à ce dernier. Ceci garantit l'absence de trous.
2. Division du volume englobant les points sur deux, suivant le plus long axe. Chaque moitié sera englobée par une sphère et fera l'objet d'un nouveau nœud.
3. Application de l'étape 2 sur les points à l'intérieur de chaque sphère jusqu'à avoir un seul point.

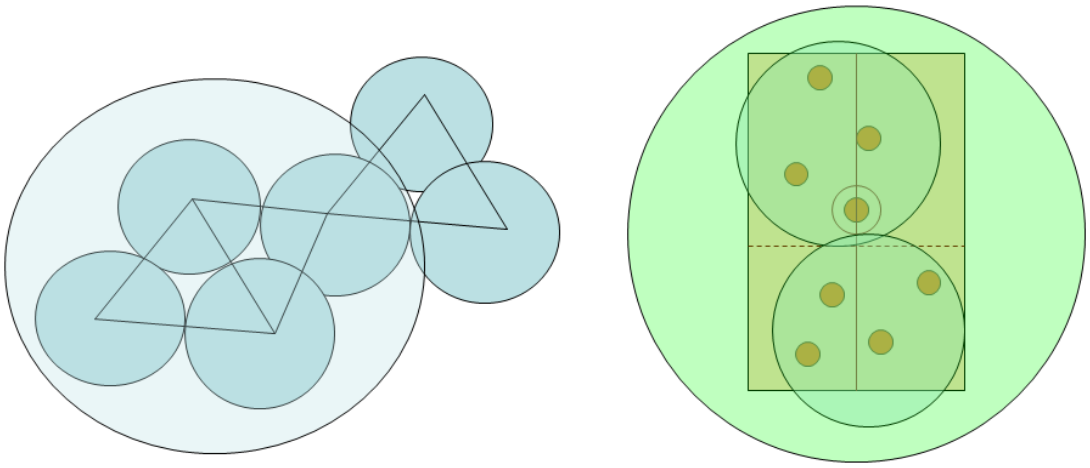


Figure 7. Processus de construction de l'hiérarchie.

À la fin du processus de construction de l'hiérarchie, les nœuds intérieurs possèdent aussi des attributs qui ont comme valeur la moyenne de celles des nœuds fils.

Les attributs sont codés selon une quantification bien précise en un nombre de bits fixe pour chaque attribut (**Figure 8**). Certains attributs (position

et la taille de la sphère) reçoivent des valeurs relatives (offset) à celles des attributs parents, ce qui optimise le nombre de bits requis au codage des valeurs d'attributs. Le nombre de branches est limité supérieurement à quatre afin que la taille finale de l'hiérarchie en disque ne soit pas énorme.

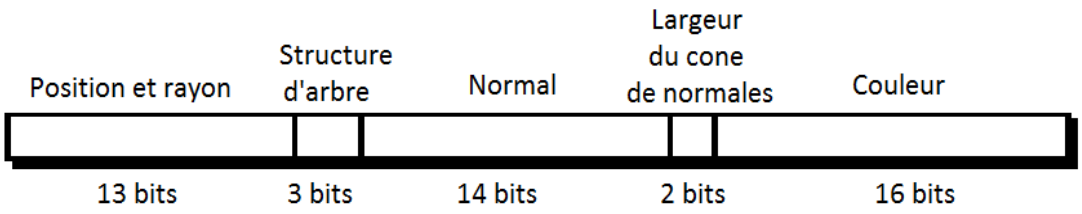


Figure 8. Quantification des attributs d'un nœud de l'hiérarchie.

La primitive de rendu peut être un carré (point), un disque, un disque associé à un filtre gaussien (splat) ou encore un splat elliptique. Ces derniers ne garantissent pas l'absence de trous dans les régions de forte courbure.

Dans le cas où le splat est la primitive de rendu choisie, il s'avère délicat de faire un mélange de couleurs tout en assurant l'élimination des surfaces occultées. Avec OpenGL, Rusinkiewicz et Levoyont ont proposé de faire deux passes, une première avec le Z-buffer activé où la coordonnée Z est mise à jour, dans la deuxième passe, le Z-buffer est désactivé et toutes les surfaces sont rendues dans le framebuffer mais avec une fonction de comparaison avec la profondeur qui existait déjà, ce qui assure un mélange de couleurs juste avec les facettes voisines l'intérieur d'un intervalle z_0. Le résultat est un rendu temps réel de larges scènes.

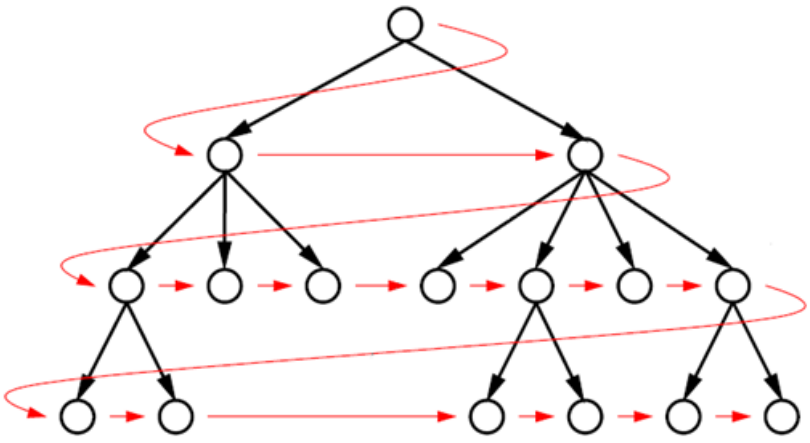


Figure 9. Hiérarchie de sphères englobantes (Rusinkiewicz et Levoy 2000).

1.2.4. Rendu avec ombrage de Phong sur GPU: Phong splatting

Phong splatting a été proposé par Botsch et al. (Botsch, Spernat et Kobbelt 2004) comme extension du surface splatting introduit par Zwicker et al. (Zwicker, et al. 2001) avec plutôt un ombrage par-pixel.

Les points sont rendus par des disques elliptiques générés sur la base d'une analyse en composantes principales (ACP) qui permet de déterminer les courbures principales, afin de faire localement une adéquation d'un disque elliptique à un sous-ensemble de points (les k points voisins) du sens des moindres carrés (Wu et Kobbelt 2004). Ce disque est le splat elliptique qui remplace les points en question.

Dans les systèmes classiques de rendu basé splats, il y a une et une seule normale par splat. Du fait que la normale soit fixe tout au long du splat, l'ombrage résultant est un ombrage plat (ou ombrage de Gouraud si le splat est associé à un filtre de reconstruction gaussien). Botsch et al. ont proposé de faire une interpolation linéaire de la normale d'afin d'avoir un ombrage par-pixel équivalent à celui de Phong dans le cas des maillages classiques.

Ceci peut être fait grâce à un champ de normales (Botsch, Spernat et Kobbelt 2004). Chaque splat est associé à une information additionnelle, autrement dit, le champ de normales. Ce dernier est calculé dans une phase de prétraitement utilisant les courbures principales des sous-ensembles du nuage de points initial (**Figure 10**). Le résultat est la normale centrale, deux directions de courbures principales et deux coefficients pour chaque splat. Le rendu est complètement basé GPU (vertex et fragment shaders).

La génération des splats est faite d'une manière approximative au sens du plan des moindres carrés afin qu'un splat S_j représente le plus fidèlement le sous-ensemble des points $p_i = (u_i, v_i)$. Les normales doivent être aussi approchées d'une manière linéaire au sens des moindres carrés, par la résolution d'un système d'équations formé de l'équation suivante, en prenant en entrée les normales des points en question (n_i).

$$N_j(u_i, v_i) = \bar{n}_j + \alpha_j u_i u + \beta_j v_i v$$

N_j : normale approchée au sens des moindres carrés au point p_i.

$\bar{n}_j$: normale centrale du splat S_j.

(u, v): directions principales.

(u_i, v_i):coordonnées locales du point p_i.

α_j , β_j : coefficients.

où les inconnus $\alpha_j, \beta_j, \bar{n}_j$ doivent être trouvés de façon à ce que $N_j(u_i, v_i)$ approche, au sens des moindres carrés, la normale n_i du point p_i situé sur le splat S_j.

Pour un rendu efficace, un seul point est envoyé au vertex shader par splat, dont la taille (point size) est suffisamment large pour couvrir l'aire du splat elliptique selon une formule bien définie. Dans l'espace écran, uniquement les pixels appartenant à la projection du splat sont pris en compte, les autres sont abandonnés. Ceci peut être fait en repartant de la caméra vers le splat (plan du splat) en passant par le pixel via le lancement d'un rayon (**Figure 11**). L'intersection du rayon avec le splat permet aussi de déterminer la coordonnée Z du point appartenant au splat.

La coupure des morceaux de splats superposés dans les régions de haute courbure (bord, coin, etc.) est possible avec ces splats elliptiques. Les résultats sont efficaces grâce à l'implémentation sur GPU qui a permis un rendu de haute qualité jusqu'à quatre millions points par seconde.

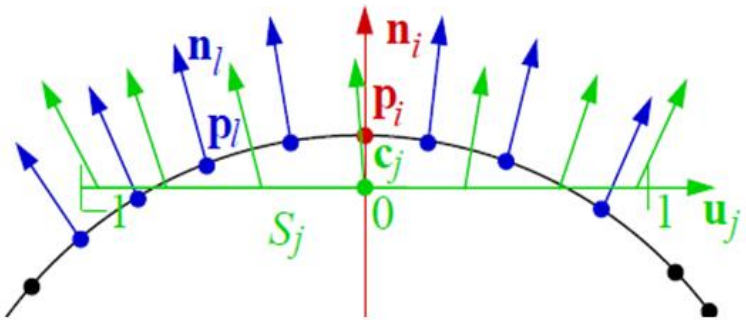


Figure 10. Champ de normales (Botsch, Spernat et Kobbelt 2004).

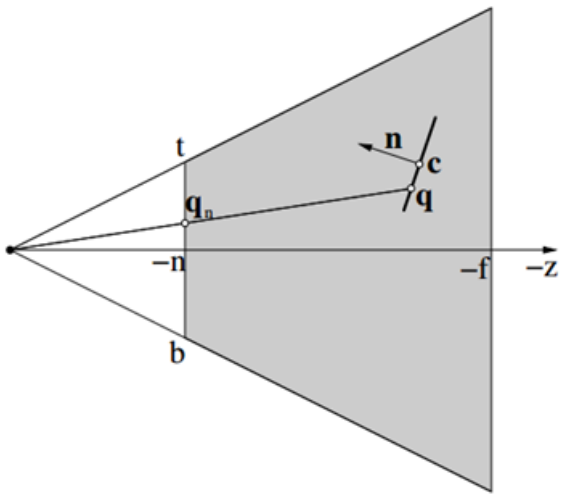


Figure 11. Calcul de l'aire d'un splat elliptique projeté sur l'écran.

1.2.5. Rendu par sur-échantillonnage des surfaces d'approximation MLS

Une surface MLS (Moving Least Square) est une approximation locale de la surface à partir d'un ensemble de points, par des polynômes bi-variés utilisant la régression MLS. Le principe est de partir d'un nuage initial de points duquel un ensemble réduit est sélectionné pour définir la surface MLS. C'est une manière adaptative d'ajuster la densité des points afin que l'approximation finale soit aussi lisse dans les régions de basse courbure que dans les régions de haute courbure.

Dans ce travail (Alexa, et al. 2001), les points sont stockés dans une structure hiérarchique de sphères englobantes Qsplat (Rusinkiewicz et Levoy 2000). Chaque nœud intermédiaire de l'hiérarchie contient une position, un rayon, une normale et probablement une couleur. Tandis que les feuilles contiennent, en plus de ces attributs, l'orientation du plan MLS et les coefficients du polynôme. Les points sont rendus par projection sur l'écran tout en parcourant l'hiérarchie.

Un sur-échantillonnage aura lieu dans l'espace objet si l'aire d'un nœud projeté couvre bien plus qu'un pixel dans l'espace écran (ceci est traduit par l'extension de l'hiérarchie par un nouveau sous-arbre). La création de nouveaux points se base sur le diagramme de Voronoï (**Figure 12**). Au fur et à mesure de nouvelles cellules (sur le plan 2D d'approximation locale des points de départ) sont ajoutées au diagramme où les sommets servent comme des nouveaux points après avoir subi une projection sur la surface MLS.

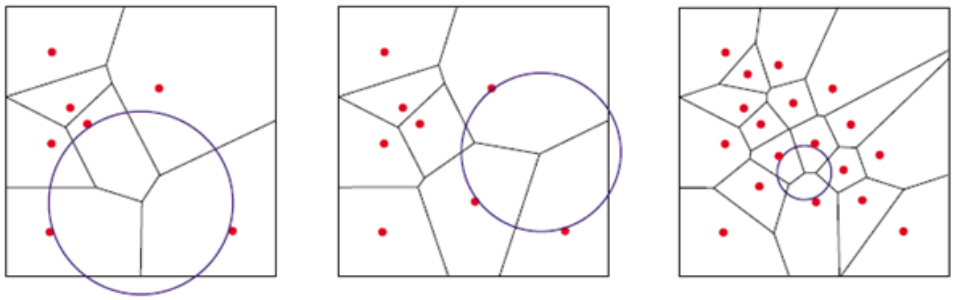


Figure 12. Sur-échantillonnage sur le plan d'approximation locale par le diagramme de Voronoï (Alexa, et al. 2001).

L'avantage de l'hiérarchie est double, elle sert à éliminer les objets en dehors de la pyramide de vue ainsi que les surfaces cachées. En plus de ça, le parcours en profondeur n'est fait qu'en cas de nécessité (le point couvre bien plus qu'un pixel).

La procédure de projection est le processus d'approximation d'un ensemble de points en une surface polynomiale selon une méthode de régression

(MLS). MLS ressemble aux moindres carrés sauf que le plan suit les points proches à un point donné **r** plus que d'autres plus loin.

La procédure de projection MLS se déroule en deux phases (**Figure 13**), la première consiste à trouver le plan qui approche un ensemble de points en s'approchant à un point donné **r** fixé au départ. Ceci est un problème d'optimisation (minimisation) qui peut être résolu par des méthodes numériques itératives (équation (a)). La deuxième phase consiste à calculer les coefficients du polynôme d'approximation de la surface en résolvant un système d'équation (équation (b)). À la fin, grâce à la procédure de projection, la position de n'importe quel point à l'intérieur de la surface est déduite à partir de ses coordonnées locales sur le plan MLS.

$$\sum_{i=1}^{N}(< n, p_i > -D)^2\theta(\|p_i - q\|) \qquad (a)$$

$$\sum_{i=1}^{N}(g(x_i, y_i) - f_i)^2\theta(\|p_i - q\|) \qquad (b)$$

$$D =< n, q >$$

θ : est une fonction monotone décroissante positive.

f_i : est l'hauteur du point p_i depuis le plan MLS.

$g(x_i, y_i)$: est le polynôme de la surface MLS qui est fonction de deux coordonnées locales (x_i, y_i) de n'importe quel point p_i sur le plan MLS.

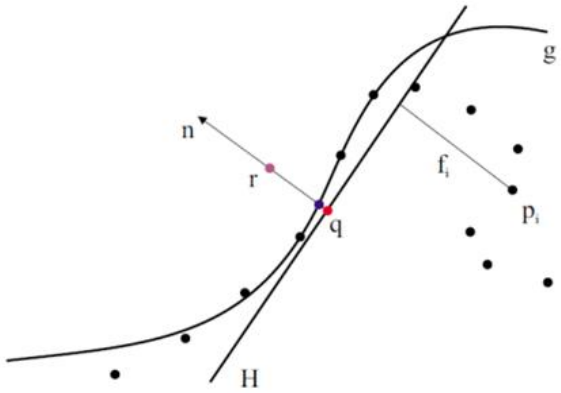


Figure 13. Procédure de projection.

Les opérations mathématiques correspondantes aux méthodes numériques itératives, ainsi qu'au calcul des coefficients du polynôme par la résolution du système d'équation, sont couteuses en matière de temps de calcul. La surface doit être suffisamment échantillonnée afin d'atteindre un bon résultat.

Il est primordial de déterminer les aires des surfaces MLS projetées sur leur plan MLS. Ces dernières doivent être entrelacées pour qu'il y ait pas de trous entre elles. Pour ceci, les surfaces doivent être définies pour des sous-ensembles de points appartenant à une sphère de rayon **h** (**h** est la distance maximale d'espacement entre les points). Ceci va garantir l'existence de chevauchement. En parallèle, il ne faut pas que ces aires soient trop chevauchées entre elles sinon des erreurs vont apparaitre dans certains cas. Pour cela, l'aire d'une surface MLS est définie par le rectangle englobant tous les points d'une surface MLS, projetés sur leur plan MLS.

L'ajout de nouveaux points à la surface polynomiale est nécessaire pour pouvoir remplir la zone correspondante projetée dans l'espace écran. En principe, ces points doivent être projetés par la procédure de projection susmentionnée. Malheureusement la procédure n'est pas rapide suffisamment, donc un échantillonnage des polynômes est envisageable (par évaluation selon une approche de différence en avant) sur une grille de points dont la taille de cellule est **d**. La question posée est dans quelle orientation cette grille doit être mise. L'idéal est qu'elle soit perpendiculaire à la direction de vue sinon l'échantillonnage doit être réajusté.

Cette approche est dépendante du point de vue (view-dependant) et donc un sur-échantillonnage (ou des évaluations des polynômes sur les points) doit être refait à chaque changement des paramètres de la caméra (position, orientation ou changement de la taille de fenêtre de vue). Comme remède, une utilisation d'une pyramide hiérarchique est envisageable, ou encore la taille des points (point size) peut s'adapter selon la configuration de la caméra.

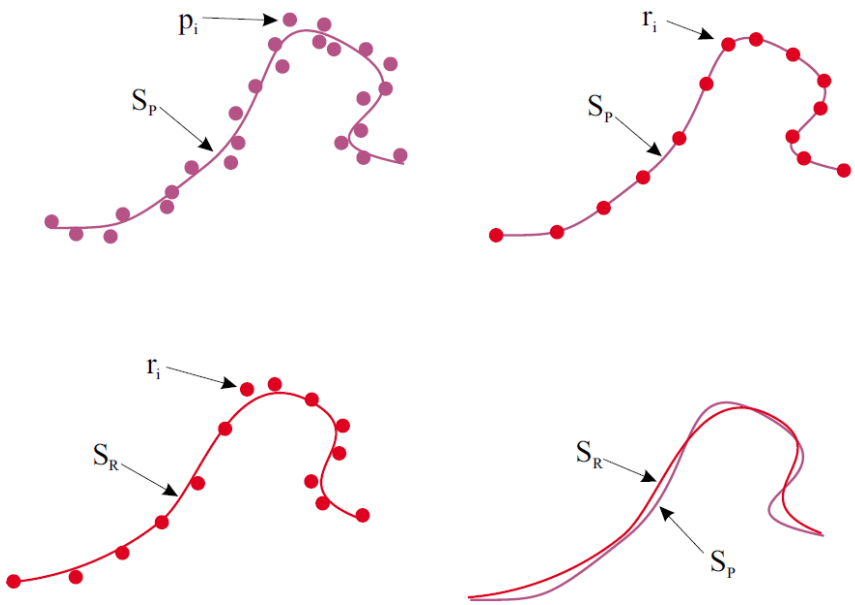


Figure 14. Surface MLS formée à partir du nuage initial et du nuage réduit (Alexa, et al. 2001).

Le rendu peut attendre 5 FPS pour des objets allant de 1kp jusqu'à 900kp. Il faut noter que toute l'approche dépend des points **r**. Ce sont de nouveaux points ajoutés au voisinage du nuage de points de départ, et qui résument toute la surface MLS correspondante aux différents sous-ensembles de points. Ce sont bien l'ensemble réduit des points qui représente une surface MLS proche à celle faite sur la base du nuage initial (**Figure 14**).

1.3. Techniques basées sur le lancer de rayon

1.3.1. Lancer de rayon sur les points

Dans le but d'avoir un rendu réaliste avec des caustiques (telles que la réflexion spéculaire ou encore la brillance), Schaufler et al. (Schaufler et Jensen 2000) ont introduit un modèle basé sur le lancer de rayon qui travaille sur un nuage discret de points. Le principe est de rassembler un sous-ensemble du nuage de points percuté par un cylindre dont l'axe est le rayon (**Figure 15**).

Le calcul d'intersection est fait entre le rayon et les disques centrés aux points, ayant un rayon égale à la plus grande distance d'espacement entre les points du nuage. Ces calculs sont accélérés par une grille hiérarchique où uniquement les nœuds appartenant à la projection du cylindre sur les deux plans 2D (ayant comme axe commun la composante la plus longue de la direction du cylindre) sont pris en compte.

Les attributs finaux (position, normale et la couleur) des points intersectés, sont pondérés selon la distance entre le centre de ces points et l'axe du cylindre. Du fait que la position du point d'intersection entre le rayon et la surface est pondérée suivant le sous-ensemble de points pris à partir d'un point de vue donné, l'image finale devient dépendante du point de vue (view-dependent) (Schaufler et Jensen 2000). Comme ce sous-ensemble de points change d'un point de vue à un autre le même morceau de surface peut avoir des positions différentes selon l'endroit de la caméra.

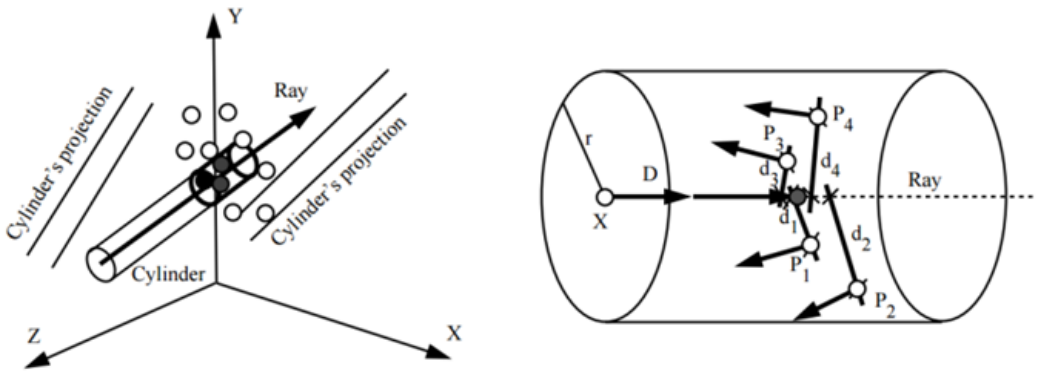


Figure 15. Calcul d'intersection rayon/point (Schaufler et Jensen 2000).

1.3.2. Lancer de rayon cohérent

Wald et al. (Wald, Slusallek, et al. 2001) ont proposé des techniques d'accélération pour le lancer de rayon. L'objectif est de lancer les rayons d'une manière cohérente, c'est à dire, de façon à ce que le code accède à la mémoire cache au maximum, au lieu de la mémoire principale (qui a une latence considérablement plus grande). L'idée est de lancer tout un paquet de rayons (quatre dans l'article) ayant la même direction et origine (rayons voisins), et qui opèrent (parcours et calcul d'intersection) simultanément utilisant des instructions parallèles du genre SIMD (extension SSE disponible sur les machines Intel à partir du Pentium 3), sur la même partie de données (la même surface percutée par un paquet de rayons cohérents) afin de réduire la bande passante de la mémoire et favoriser l'utilisation du cache.

1.3.3. Lancer de cônes multi-résolution

Contrairement au lancer de rayon conventionnel, le lancer de cônes multi-résolution (Wand et Straßer 2003) fournit des images sans bruits et artefacts, sauf qu'il est quatre fois plus lent. Ce qu'apporte le lancer de cônes multi-résolution par rapport le lancer de cônes tel qu'il était présenté initialement par Amanatides (Amanatides 1984), c'est la capacité de traiter des scènes très complexes (tels que les nuages de points) en un temps proportionnellement réduit.

1.3.4. Lancer de rayon sur les surfaces implicites

Adamson et al. (Adamson et Alexa 2003) ont proposé de changer le calcul d'intersection rayon/surface de la formulation paramétrique vers une autre implicite. Donc le problème de détermination du point d'intersection devient un problème de détermination de la racine d'une fonction.

Étant donné que le modèle de surface d'approximation MLS initialement proposé par Alexa et al. (Alexa, et al. 2001) peut être considéré comme un modèle de surface implicite, Adamson et al. étend ces surfaces implicites par des sphères autour de chaque point, afin de permettre un calcul d'intersection rayon/surface à deux étapes implicitement.

L'inconvénient de l'application du lancer de rayon sur ces surfaces implicites est toujours les calculs surchargés dus aux approximations surfaciques MLS à partir d'un nuage de points (rendu en plusieurs heures).

1.3.5. Lancer de rayon interactif sur des surfaces implicites

Wald et al. (Wald et Seidel 2005) ont proposé un lancer de rayon interactif qui atteint 24 millions points par seconde grâce à des techniques d'accélération (structure KD-tree avec un parcours optimal via les heuristiques SAH) appliquées sur le même modèle de lancer de rayon sur des surfaces implicites proposé par Adamson et al. (Adamson et Alexa 2003).

Dans un premier temps, Wald et al. avaient fait des expérimentations utilisant le lancer de rayon sur des disques (splat) avec des techniques d'accélération de calcul d'intersection et de parcours des structures de donnée (KD-tree) avec possibilité de coupure des disques superposés, ainsi que l'interpolation des normales (en se basant sur les disques voisins). Les résultats obtenus sont de qualité acceptable mais très couteux en terme de temps de calcul.

Ce qu'ont proposé Wald et al. est de mélanger les splats pour en former plutôt un modèle de surfaces d'approximation MLS d'Alex et al. (Alexa, et al. 2001). Les surfaces MLS ont l'avantage d'enlever les artefacts (présents dans la plupart des techniques de splatting dans les régions de silhouette ou de forte courbure) et de donner une surface lisse et continue, à condition que la surface soit suffisamment échantillonnée.

Un plan d'approximation (défini par un point $\bar{\boldsymbol{p}}$ et une normale $\bar{\boldsymbol{n}}$) est déterminé par régression, à partir des points voisions à une position prédéfinie **x** où chaque point $\boldsymbol{p_i}$ a un champs d'influence sphérique $\boldsymbol{\omega_i}$ (nul au delà du rayon $\boldsymbol{r_i}$) qui est défini par une fonction de diminution $\boldsymbol{W}$ inversement proportionnelle à la distance entre le point et la position **x**.

$$\bar{p}(x) = \frac{\sum \omega_i(x) p_i}{\sum \omega_i(x)}$$

$$\bar{n}(x) = \frac{\sum \omega_i(x) n_i}{\sum \omega_i(x)}$$

$$\omega_i(x) = W\left(\frac{\|x - p_i\|}{r_i}\right)$$

$$W(r) = \begin{cases} 1 - r & ;\ r < 1 \\ 0 & ;\ r \geq 1 \end{cases}$$

La surface implicite est définie à partir de ce plan, par la racine de la fonction suivante :

$$f(x) = \left(x - \bar{p}(x)\right)\bar{n}(x)$$

Sachons que le nuage est organisé selon une structure KD-tree, l'algorithme de calcul d'intersection (**Figure 16**) déroule selon les étapes suivantes:

1. Le rayon est échantillonné en une séquence de positions séparées par un intervalle fixe (**Figure 17**).
2. La fonction $\boldsymbol{f}(\boldsymbol{x})$ est évaluée à chaque position au fur et à mesure, jusqu'à ce qu'une nouvelle valeur diffère en signe de l'ancienne, ce qui signifie qu'une intersection a eu lieu.
3. La position de l'intersection est interpolée linéairement selon les deux valeurs, nouvelle et ancienne de $\boldsymbol{f}(\boldsymbol{x})$.

```
bool INTERSECT(Ray R, Splats S[])
float oldF = 0; oldT = t_near;
for (i=0..k-1)
   t = Interpolate(i/(k-1),t_near,t_far);
   x = origin + t * direction;
   W = 0; N = 0; P = 0;
   for (each splat S)
      w = S.w(x);
      if (w == 0) continue;
      W += w;   N += w * S.n;   P += w * S.p;
   end
   if (W==0) continue; // not contd in any S
   F = (W*x - P) * N;
   if (F*oldF < 0 /* different signs ! */)
      t_hit = Interpolate(oldF/(oldF-F),
                          oldT,t));
      n_hit = ... /* same loop as above */
      return HIT;
   end
   oldF = F; oldT = t;
end
return NO_HIT;
```

Figure 16. Algorithme de calcul d'intersections (Wald et Seidel 2005).

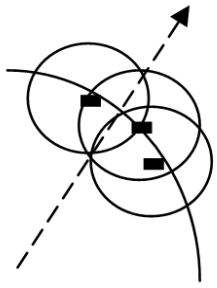


Figure 17. Rayon traversant progressivement une surface implicite MLS.

Plus le rayon d'influence des points est grand, plus y aura de splats à considérer, ceci dit, plus d'itérations. Pour une meilleure efficacité, il faut que le rayon des splats soit minimal, tout en assurant une couverture totale de la surface (absence de trous).

L'hiérarchie KD-tree utilisée doit être bien faite pour assurer un parcours optimal via des techniques comme les heuristiques (SAH).

1.3.6. Lancer de rayon sur des splats

Linsen et al. (Linsen, Müller et Rosenthal 2007) ont proposé un lancer de rayon basé plutôt splats, pour les nuages de points où ils réutilisent le champ de normale (Botsch, Spernat et Kobbelt 2004) dans une phase de prétraitement. Les attributs des splats superposés sont pondérés de la même façon que celle de Schaufler et al. (**Figure 18**). Les calculs d'intersection sont accélérés par une structure de données qui est la grille hiérarchique (octree).

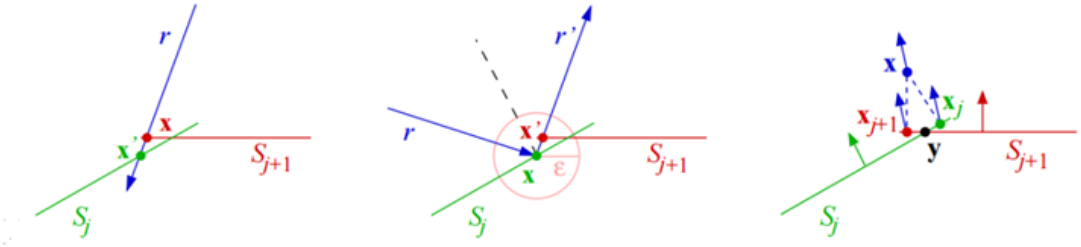


Figure 18. Calcul d'intersections rayon/splat.

1.3.7. Lancer de rayon basé GPU interactif

Les auteurs (Goradia, et al. 2010) ont proposé un lancer de rayon basé splats combiné a une technique d'accélération qui est la grille hiérarchique (octree) avec implication des GPU permettant au rayon de parcourir la scène d'une manière rapide, et avoir à la fin un rendu interactif avec des effets de réflexion, de retransmission et d'ombre.

Le principe des rayons cohérents, c'est à dire, les rayons qui empruntent des chemins similaires a été exploité précédemment (Wald, Slusallek, et al. 2001) pour accélérer le lancer de rayon sur des architecture SIMD. Ce même principe peut être étendu sur les architecture GPU grâce à CUDA.

Le parcours de la grille hiérarchique du nœud racine (premièrement percuté par le rayon) jusqu'au nœud feuille se fait d'une manière descendante, après linéarisation de la grille via la formule SFC (Space Filling Curves).

1.3.8. Lancer de rayon efficace des scènes dynamiques sur GPU utilisant CUDA

Un travail récent de qualité a été publié (Zhou, et al. 2008) permettant la construction temps réel de KD-tree sur GPU (via le modèle de programmation CUDA de NVIDIA) et ceci pour toutes les phases de construction. Ce qu'ont apporté les auteurs, est que les nœuds du KD-tree sont construits au même ordre du parcours en profondeur.

Le fait de pouvoir construire les KD-tree en temps réel, rend la technique applicable pour les traitements des scènes dynamiques.

Les domaines potentiels d'application sont, entres autres, le lancer de rayon, le lancer de photons, le path tracing et aussi les scènes dynamiques à base de points. Pour ce dernier domaine, la construction du KD-tree sur GPU permet indirectement de faire l'estimation de la normale et la densité locale du nuage de points par la technique des **k** voisins les plus proches, ainsi que la mise à jour des champs de déformation pour les outils de déformation de formes libres (FFD: Free Form Deformation).

1.3.9. Illumination globale interactive sur GPU

Les auteurs ont proposé un Framework (Wang, et al. 2009) dans lequel, une utilisation des ISM: Imperfect Shadow Maps (Ritschel, et al. 2008) qui calculent l'éclairage indirect pour les surfaces diffuses, est combinée à un lancer de photons basé GPU via un KD-tree (Zhou, et al. 2008), pour calculer des effets de réflexions et de caustiques des surfaces spéculaires.

1.4. Techniques basées sur la reconstruction dans l'espace écran

1.4.1. Rendu par interpolation pull-push

Le rendu dans l'espace écran (ou l'espace image) utilisant l'interpolation pull-push a été utilisé vivement dans la littérature. (Gortler, et al. 1996) ont utilisé l'interpolation par un algorithme de la pyramide pull-push. Grossman and Dally l'ont adapté en 1998 pour faire la reconstruction des nuages de points non denses dans l'espace image afin d'éliminer les trous entre les points projetés (Grossman et Dally 1998). En 2000, Pfister et al. ont utilisé cette pyramide pour combler les trous entre les splats et pas les pixels (Pfister, et al. 2000).

Ce que les auteurs apportent dans ce travail (Marroquim, Kraus et Cavalcanti 2007), c'est surtout un rendu efficace grâce à une implémentation GPU qui permet d'atteindre un taux d'affichage qui touche 50 millions points par seconde.

Les points sont rendus par projection sur écran tout en stockant leurs attributs (normal, profondeur, rayon, etc.) dans des tampons. Chaque pixel prend les attributs d'un seul point, d'après les auteurs, ceci rapporte du temps (contrairement au splatting qui sur-dessine les splats superposés). Il faut mettre l'accent sur le fait que le rendu est fait en une seule passe (contrairement au splatting qui est fait en deux, une pour la visibilité et l'autre pour les attributs).

Dans la phase pull, la moyenne des attributs d'un bloc de quatre pixels valides (contenant une projection d'un point) du niveau fin, est calculée et stockée dans le pixel correspondant du niveau grossier. Il est exclu du calcul de la moyenne, tout pixel du bloc, éloignant du premier pixel (au sens du plus petite coordonnée Z) d'une distance supérieure au rayon de ce premier pixel là (test d'occultation).

Donc le niveau grossier ne prend pas en considération les points occultés tandis qu'ils sont toujours présents dans le niveau fin et ils seront recalculés dans la phase push. Les attributs des pixels du niveau grossier sont calculés par la moyenne des attributs du niveau fin sauf pour l'intervalle de profondeur qui est plutôt l'union des intervalles des pixels du niveau fin.

La reconstruction est fait utilisant un filtrage elliptique qui correspond à la projection orthogonale du cercle centré au point ayant un rayon **R** (calculé au préalable localement sur le nuage) sur l'écran. L'axe majeur correspond au vecteur perpendiculaire à la normale projetée, sa longueur est deux fois le rayon du point. L'axe mineur est parallèle à la normale projetée, et sa longueur est la longueur de l'axe majeur multipliée par la coordonnée **Z** de la normale.

À la phase push, la reconstruction elliptique, en plus d'agir comme un filtre d'interpolation, elle sert aussi comme un test de visibilité. Les attributs des pixels invalides ou occultés (en dehors de l'intervalle de profondeur du niveau grossier) du niveau fin sont recalculés selon une pondération des attributs des pixels du niveau grossier, valides et dont leur rayon couvre bien le pixel en question du niveau fin (**Figure 19**).

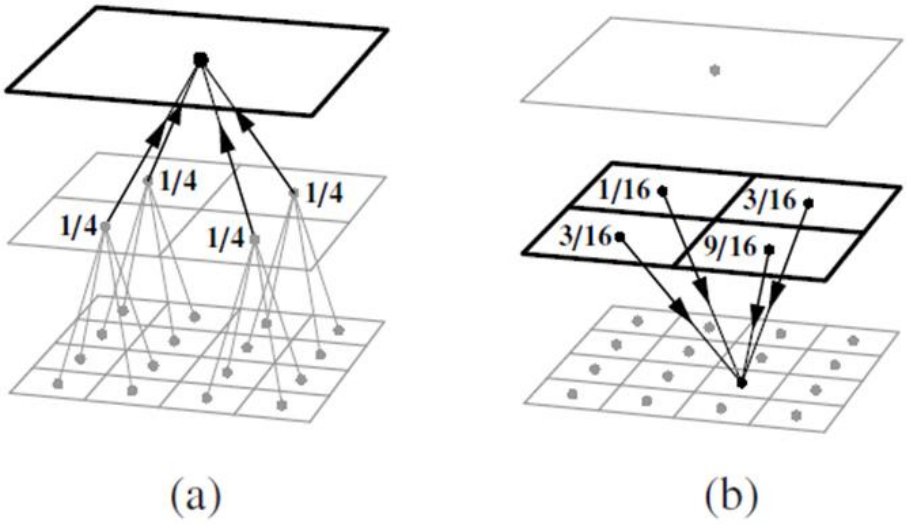


Figure 19. Deux phases d'interpolation: (a) Pull. (b) Push (Marroquim, Kraus et Cavalcanti 2007).

1.4.2. Rendu interactif via l'opérateur HPR

L'opérateur HPR: Hidden Point Removal (Katz, Tal et Basri 2007) permet de déterminer les points visibles d'un nuage de points initial à partir d'un point de vue donné. Il fonctionne en deux étapes, la première est la transformation ou l'inversion sphérique des points (**Figure 20**), puis le calcul de l'enveloppe convexe est fait dans la deuxième étape comme il est lent et son implémentation n'est pas convenable sur les GPUs actuels.

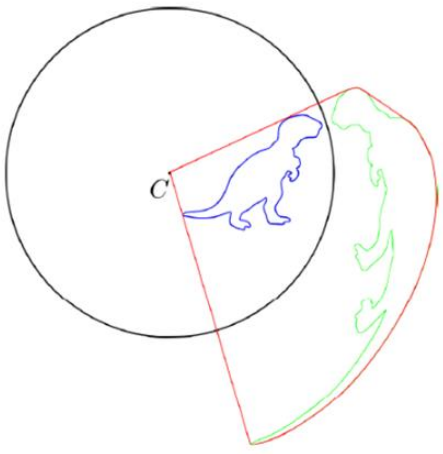


Figure 20. Opérateur HPR (Katz, Tal et Basri 2007).

Certains anciens travaux de calcul approximatif de l'enveloppe convexe, commencent d'abord par réduire le nuage initial en un autre moins dense (où ces points choisis ont une forte probabilité d'appartenir à l'enveloppe convexe). D'autres méthodes se basent sur des heuristiques.

Dans ce travail, Esperanca et al. (Esperanca, Oliveira et others 2012) ont proposé un algorithme de calcul approximatif de l'enveloppe convexe pouvant être implémenté facilement sur GPU ce qui permet un rendu interactif de larges nuages de points.

L'opérateur HPR inverse sphériquement les points selon l'équation suivante:

$$\hat{p}_i = f(p_i) = p_i + 2(R - \|p_i\|)\frac{p_i}{\|p_i\|}$$

Esperanca et al. ont étendu en troisième dimension, une approche intuitive proposée par Kavan et al. (Kavan, Kolingerova et Zara 2006) qui travaille en 2D, pour le calcul approximatif de l'enveloppe convexe. Cette approche permet de dire si un point appartient ou pas à l'enveloppe convexe d'un ensemble de points, suivant les étapes suivantes:

1. Sélectionner un point origine $\boldsymbol{q}$ à l'intérieur de l'enveloppe convexe.
2. Diviser la sphère qui englobe les points en $\boldsymbol{k}$ secteurs $\frac{\boldsymbol{2\pi}}{\boldsymbol{k}}$.
3. Sélectionner des points candidats : déterminer le point $\boldsymbol{p_i}$ le plus loin de l'origine $\boldsymbol{q}$ pour chaque secteur $\boldsymbol{i}$ dans la direction du bissecteur $\boldsymbol{d_i}$ (**Figure 21**).
4. Propagation des points candidats : comparer chaque point candidat $\boldsymbol{p_i}$ du secteur $\boldsymbol{i}$ avec tous les autres points candidats $\boldsymbol{p_j}$ et le mettre à jour si $\boldsymbol{p_j}$ est plus loin que $\boldsymbol{p_i}$ dans la direction $\boldsymbol{d_i}$ (ou juste les secteurs immédiatement voisins pour plus de rapidité de calcul).

À la différence de ce que Kavan et al. cherchent à faire (création de l'enveloppe convexe), l'auteur veut juste trouver les points du nuage appartenant à l'enveloppe convexe.

L'algorithme est accéléré en utilisant une structure de données contenant les points répartis sur les différents secteurs où le parcours est fait en parallèle afin de sélectionner et propager les points candidats.

Afin de faire le rendu, une phase de reconstruction surfacique partielle par triangulation est effectuée dans l'espace objet, à partir des points candidats appartenant à l'enveloppe convexe.

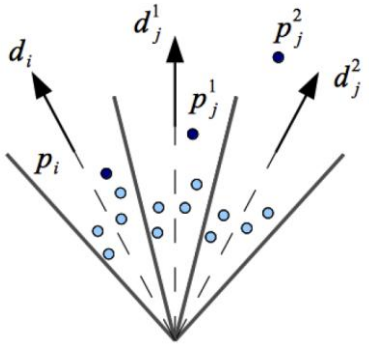


Figure 21. Calcul des points candidats de chaque secteur.

En 2014, ce travail a été repris mais avec un rendu dans l'espace écran par la technique d'interpolation pull-push (Machado e Silva, et al. 2014) en prenant comme entrée les buffers contenant les points candidats.

1.4.3. Rendu par depth peeling

Le depth peeling est une technique de partitionnement des scènes 3D en des couches triées (**Figure 22**). Ces couches sont extraites l'une après l'autre d'une manière itérative en partant de la caméra. La première partie visible de la scène est projetée et stockée dans la première couche, mais aussi supprimée de la scène, avant de calculer la prochaine couche.

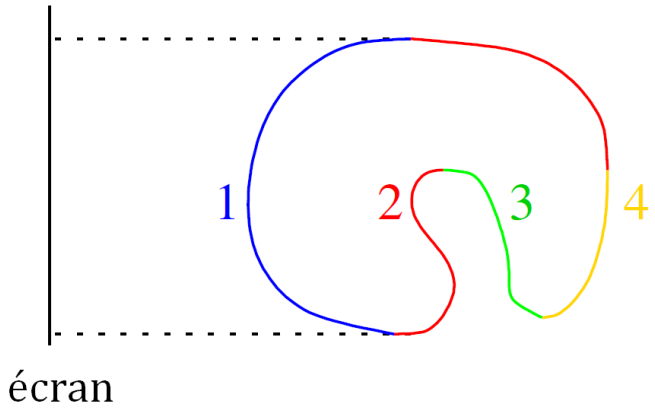


Figure 22. Depth peeling.

Dans ce travail (Dobrev, Rosenthal et Linsen 2010), les auteurs ont exploité cette technique pour proposer une méthode de rendu dans l'espace image, par un algorithme multi-passe (le nombre des passes est le nombre des couches souhaitées). Les passes sont exécutées avec un test de profondeur

activé. Une fonction de supérieur strictement est utilisée dans le test de profondeur afin que la couche courante soit comparée avec celle précédente.

Les trous présents entre les points projetés d'une couche **i** permettent la pénétration d'autres points appartenant à une couche **i+1** (**Figure 23**).

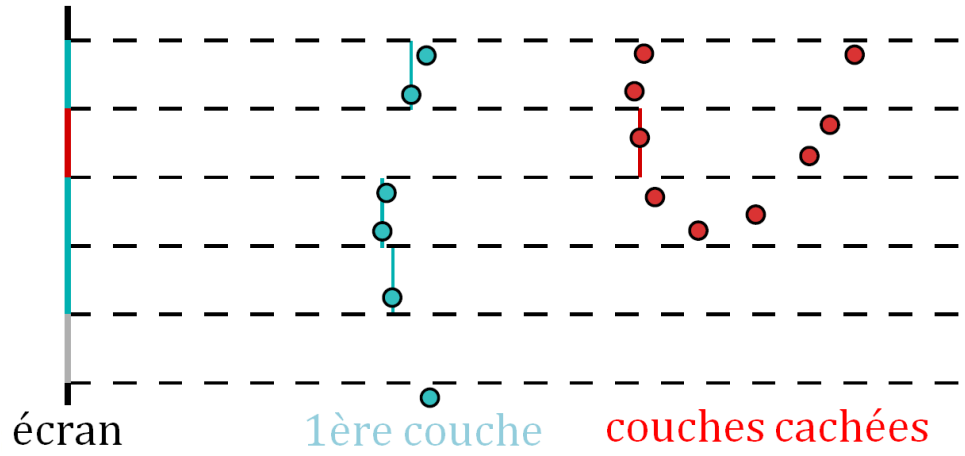


Figure 23. Les trous.

Pour remédier à ce problème, une opération de remplissage approximatif de trous dans l'espace écran doit être faite en se basant sur l'information de profondeur, en quatre étapes comme suit:

1. Remplissage des trous représentant l'arrière plan (qui sont présents que dans l'espace image mais pas dans l'espace objet).
2. Remplissage des trous représentant les pixels occultés.
3. Filtrage de l'image pour une qualité meilleure de l'image de la couche.
4. Anti-aliassage est appliqué à la silhouette de l'image finale comportant toutes les couches.

Afin d'identifier les trous d'arrière plan, un masque 3x3 est utilisé (**Figure 24**). Si l'une des configurations ci-dessous est présente donc il s'agit d'un trou d'arrière plan qui est présent aussi dans l'espace objet et donc qu'il ne faut pas le remplir, sinon c'est un trou dû à une faible densité des points et donc il faut le remplir (sa profondeur et sa couleur) par la valeur (de la profondeur et la couleur) du pixel ayant une profondeur minimale au tour du pixel candidat. Le processus est itératif pour les trous de taille plus grande.

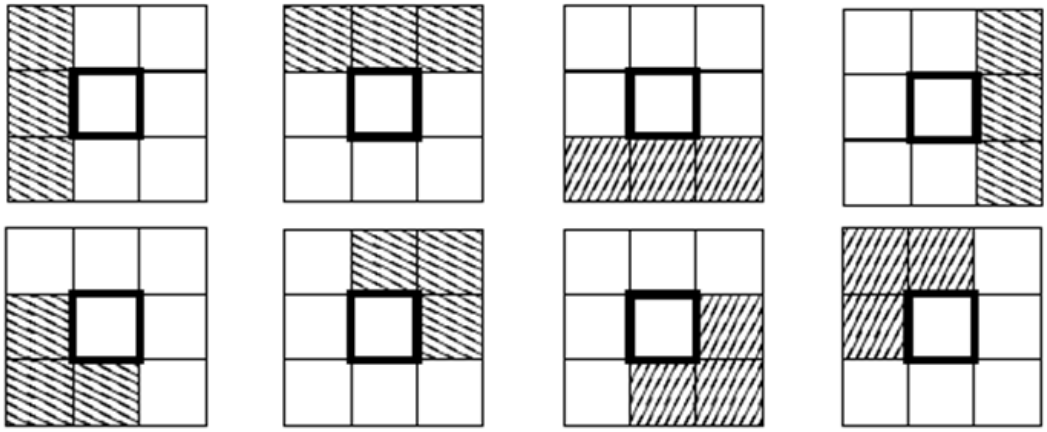


Figure 24. Masque 3x3 pour la détection de trous sur l'image contenant les points projetés (Dobrev, Rosenthal et Linsen 2010).

Afin d'identifier les trous d'occultation, un paramètre doit être calculé empiriquement selon le nuage de points en entrée, représentant la distance minimale entre les points de deux couches consécutives $\boldsymbol{D_{min}}$. Le même masque est appliqué et un remplissage doit avoir lieu si la majorité des pixels voisins ont une profondeur inférieure à la profondeur du pixel candidat $\boldsymbol{D}$ plus $\boldsymbol{D_{min}}$.

Le filtrage d'image de chacune des couches est souhaitable pour une meilleure qualité, pour cela, un filtre gaussien de 3x3 est proposé sauf qu'il n'est pas applicable sur la silhouette.

La silhouette peut être détectée par une vérification des pixels voisins au cas où leur profondeur change brusquement de plus de $\boldsymbol{D_{min}}$. Une fois la silouhette est détectée, un lissage est appliqué afin de réduire l'aliassage.

Il se trouve à la fin que la combinaison de toutes les couches avec différents degrés d'opacité, donne un rendu des surfaces transparentes.

1.5. Conclusion

À la fin de ce chapitre, il nous paraît à travers les travaux que nous avons pu parcourir que le rendu des nuages de points via des approximations locales communément nommées splats est assez efficace et ceci grâce à des techniques qui s'avèrent inévitables pour tout système de rendu à base de points. Nous citons entre autres:

- L'utilisation des structures de données hiérarchiques pour le parcours et le calcul de visibilité (octree, Qsplat, LDC, etc.).
- L'utilisation des filtres pour combler les trous et permettre un mélange de couleur, ce qui donne l'impression d'une surface continue (filtre gaussien pour chaque splat).

- La réduction du nombre de points en un nombre minimal avec élimination des points redondants et les bruits (techniques de ré-échantillonnage et de simplification).
- L'estimation de l'aire du splat rastérisé à l'écran afin d'ajuster le compromis qualité/efficacité.
- L'utilisation des GPU pour accélérer des parties du calcul (vertex/fragment shader).

Tandis que certaines techniques comme les surfaces d'approximation MLS comportent des calculs numériques surchargés et leur efficacité reste discutable malgré qu'elles constituent une reconstruction continue la plus exacte de la surface à partir d'un ensemble de points discrets, donnant lieu à des résultats meilleurs visuellement.

Le rendu dans l'espace écran est une vraie tendance efficace alternative des méthodes classiques de splatting. Malgré le manque de l'appui physique, certains effets d'illumination globale restent toujours reproductibles approximativement, permettant d'avoir des images plus ou moins réalistes à moindre coût.

Le lancer de rayon se trouve ressuscité, malgré qu'il était longuement considéré comme la technique la plus chère en temps de calcul, et qui est loin d'être utilisé pour des applications qui demandent un temps de réponse interactif, tout cela grâce à la parallèlisation du calcul sur GPU, ainsi que l'utilisation des structures de données accélérées (KD-tree).

Chapitre 2 :

Contribution1 : lancer de rayon basé splats pour les objets à base de points

2.1. Introduction

Comme nous travaillons sur des scènes à base de points, deux contributions ont été réalisées sur le sujet du lancer de rayon. Il s'agit d'un lancer de rayon adapté aux points reconstruits par des splats. Ce que apporte notre implémentation par rapport à ce qui existe dans la littérature, est situé sur l'axe quantitatif et l'axe qualitatif, comme détaillé dans les deux sections suivantes.

2.2. Première contribution: Efficient accelerated splat-based raytracing

Vu que dans n'importe quel algorithme de lancer de rayon, la phase de calcul d'intersections rayon-surface est la partie la plus coûteuse en matière de temps de calcul. Pour cette raison, notre lancer de rayon sur les splats a été accéléré par des optimisations géométriques, ainsi qu'en adaptant les techniques accélératrices classiques pour le cas des points (ou splats), à savoir les volumes englobants ainsi que les subdivisions spatiales.

Dans cette première contribution (Beddiaf et Babahenini 2014), on a fait la sélection d'un **volume englobant** (parmi plusieurs) qui est la sphère (**Figure 25**). Ce choix est justifié par le fait qu'on travaille avec des disques (splats) orientés, et donc le seul volume englobant qui colle le mieux à un disque sera une sphère. En plus ce ça, il est connu que les intersections rayon-sphère sont parmi les intersections rayon-surface les plus rapides et simples à calculer.

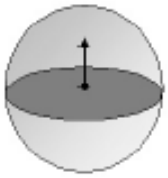


Figure 25. Une sphère englobant un splat.

Ainsi, les intersections sont calculées à deux niveaux:

1. **Intersection rayon-sphère**
   Quand ce test échoue , on passe pas au calcul d'intersection rayon-splat.
2. **Intersection rayon-splat**
   Quand le rayon incident est bien intersecté par la sphère, dans ce cas on vérifie si le rayon percute aussi le splat dedans (**Figure 26**).

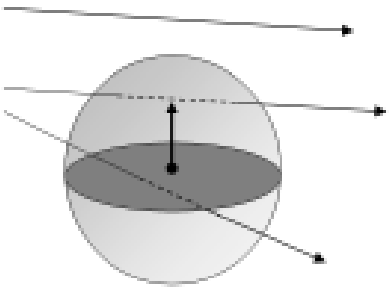


Figure 26. Un rayon peut percuter la sphère, le splat, les deux, ou rien.

Formellement parlant, si le rayon est donné sous la forme paramétrique suivante:

$$R(t) = \left(o, \vec{d}\right) = o + t\,\vec{d}$$

o est l'origine du rayon et $\vec{\boldsymbol{d}}$ est sa direction.

Ainsi que la sphère est donnée sous la forme de l'équation implicite suivante:

$$(S_x - c_x)^2 + (S_y - c_y)^2 + (S_z - c_z)^2 = r^2$$

Tel que ***c*** est le centre de la sphère, ***S*** sont les points résidents sur la surface de la sphère, et ***r*** est le rayon de la sphère.

Donc, la première intersection du rayon avec la sphère est donnée par:

$$T' = \min(\max(t_0, 0), \max(t_1, 0))$$

où

$$t_0 = \frac{-B - \sqrt{B^2 - 4AC}}{2A}$$

$$t_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

Sachons que A, B et C sont des coefficients constants:

$$A = d_x^2 + d_y^2 + d_z^2$$

$$B = 2(d_x(o_x - c_x) + d_y\left(o_y - c_y\right) + d_z(o_z - c_z))$$

$$C = (o_x - c_x)^2 + \left(o_y - c_y\right)^2 + (o_z - c_z)^2 - r^2$$

Pour les subdivisions spatiales, on a choisi la grille uniforme pour deux raisons. Premièrement, parce que les sphères englobant les splats ont une taille (diamètre) fixe, et donc en fixant la résolution de la grille uniforme sur la base de ce diamètre (longueur de cellule=diamètre splat), le stockage des splats dans la grille sera plus simple (un splat va appartenir à seulement les 8 cellules environnant le centre du splat) qu'avec d'autres subdivisions spatiales telles que

KD-tree ou octree (**Figure 27**). Deuxièmement, les grilles uniformes sont connues par leur algorithme de parcours très simple et rapide, et ceci est l'objectif visé.

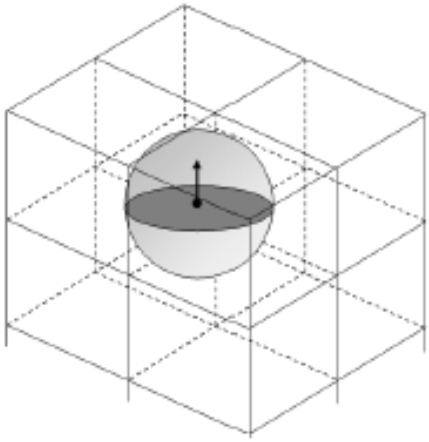


Figure 27. Huit cellules remplies par un seul splat.

Nous avons appliqué notre lancer de rayon basé splats sur trois objets basés points comme suit:

- Une sphère miroir (complètement spéculaire) ayant exactement 2562 points
- Un bunny (complètement diffus avec une couleur grise) ayant exactement 35945 points
- Un rabbit (complètement diffus avec une couleur grise) ayant exactement 70658 points

Les images résultantes sont d'une bonne qualité visuelle comme montre **Figure 28**. De point de vue quantitatif, les trois images confirment un bon gain en matière de temps de calcul (**Figure 29**) lors de l'application des différentes techniques d'accélération. L'ingrédient clé ayant l'impact le plus important sur le temps de calcul était le volume englobant (sphère) par rapport aux accélérations géométriques et la subdivision spatiale (grille uniforme).

Commençons par la sphère, nous avons observé que le lancer de rayon accéléré est 33 fois plus rapide (accélération du temps de calcul) que la version non accélérée, et ceci grâce au volume englobant à 82%. Pour le bunny, nous avons observé que le lancer de rayon accéléré est 600 fois plus rapide que la version non accélérée, et ceci grâce au volume englobant à 91%. Pour le rabbit, nous avons observé que le lancer de rayon accéléré est 656 fois plus rapide que la version non accélérée, et ceci grâce au volume englobant à 80%.

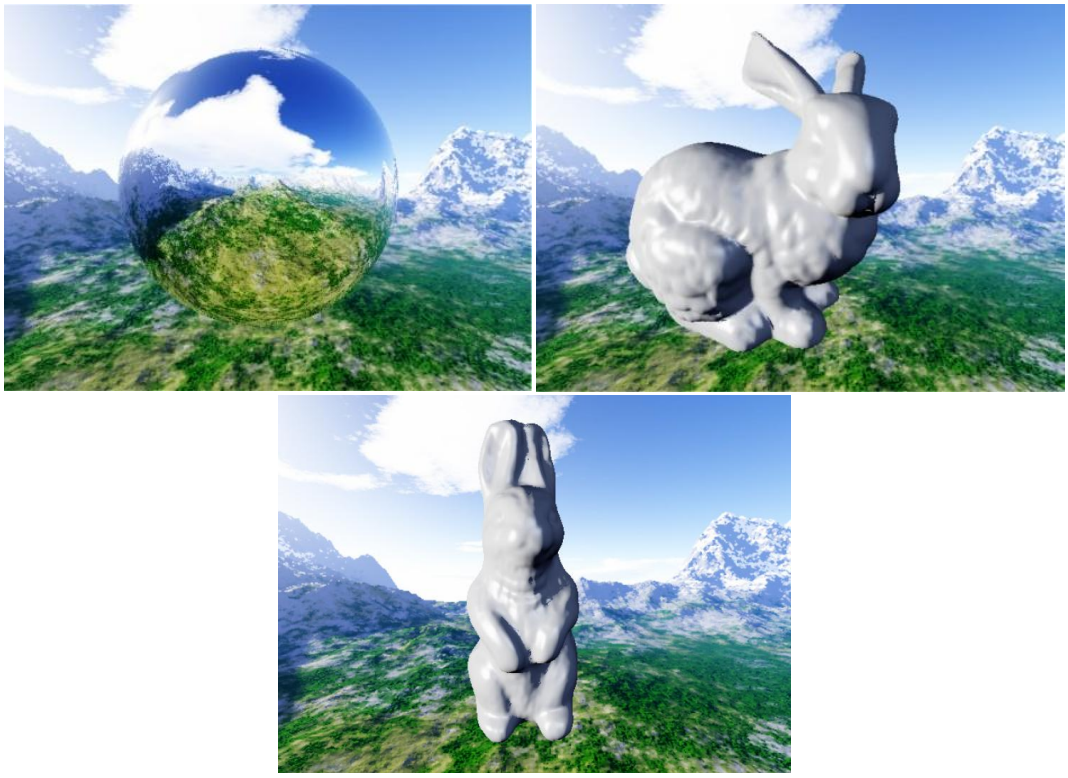


Figure 28. Une sphère miroir, un bunny et un rabbit rendus avec notre lancer de rayon accéléré.

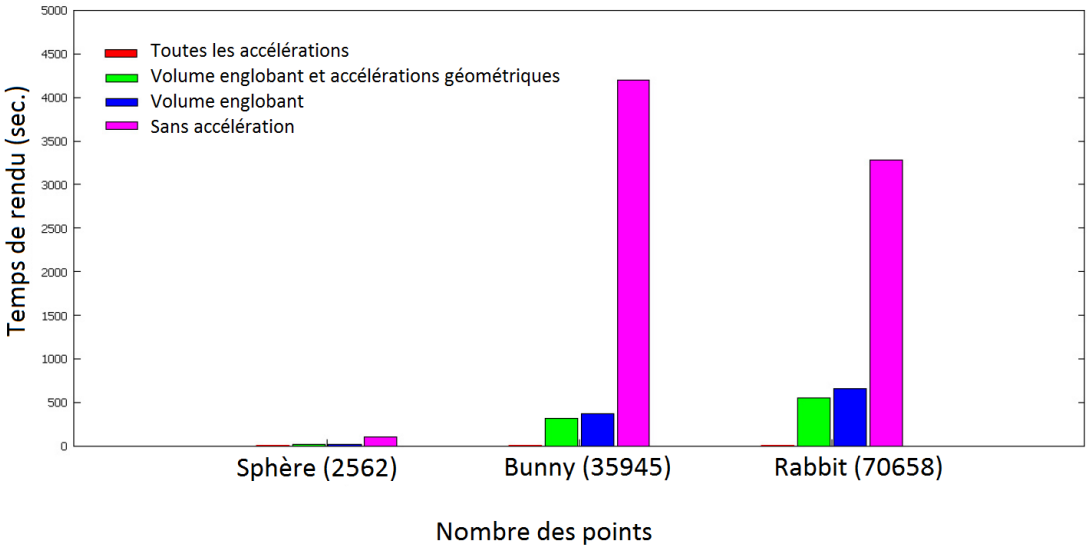


Figure 29. Temps de rendu des trois scènes selon l'activation de trois accélérations.

2.3. Deuxième contribution: An improved splat-based raytracing

Cette contribution a un apport sur le volet qualitatif et plus précisément la silhouette des objets à base de splats (Beddiaf et Babahenini 2015). On s'est

attaqué à un problème bien pointu qui est le phénomène de superposition et de chevauchement inter-splats qui se produit souvent sur les frontières d'objets. Ce chevauchement engendre un problème d'identification d'une surface continue et lisse lors de l'utilisation du lancer de rayon.

Pour cela, on a proposé une liste de règles pour ôter le conflit de détermination du point d'intersection d'un rayon traversant des splats superposés et chevauchés. On devrait même estimer (interpoler) une normale à ce point d'intersection de façon à ce que l'ombrage appliqué sera d'une bonne qualité (idéalement proche à l'ombrage de Phong).

Les conflits ont été résolus sur la base de l'étude géométrique des courbures discrètes formées par les splats (**Figure 30**). Si l'angle formé par les deux splats est inférieur à **180°** donc la courbure est concave, sinon elle est convexe. L'angle est calculé comme suit:

$$\omega = \omega_1 + \omega_2 = \arccos\left(\frac{\overrightarrow{c_j c_{j+1}} . \overrightarrow{N_J}}{\left\|\overrightarrow{c_j c_{j+1}}\right\| . \left\|\overrightarrow{N_J}\right\|}\right) + \arccos\left(\frac{\overrightarrow{c_{j+1} c_j} . \overrightarrow{N_{J+1}}}{\left\|\overrightarrow{c_{j+1} c_j}\right\| . \left\|\overrightarrow{N_{J+1}}\right\|}\right)$$

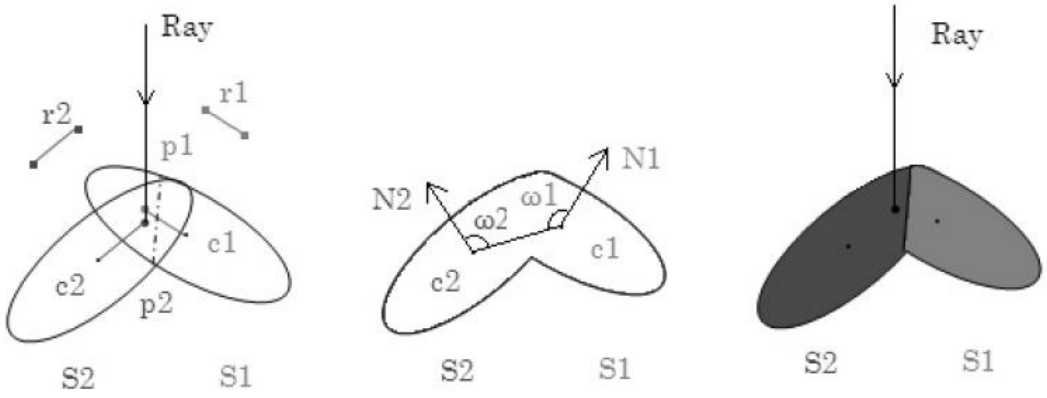


Figure 30. Résolution des conflits des splats chevauchés sur la base de l'étude de courbures.

Dans le cas où la courbure est concave, on prend le premier point d'intersection avec le premier splat au lieu du deuxième, et inversement. Tandis que pour l'estimation de la normale à ce point, elle sera une pondération des deux normales des splats concernés, qui dépend de la distance qui sépare les points d'intersections des centres des splats en question (**Figure 31**).

$$Normal_x = \sum_{i=0}^{n} \frac{Normal_i \, . \, d_i^{'}}{r}$$

$$d_i^{'} = (r - d_i)$$

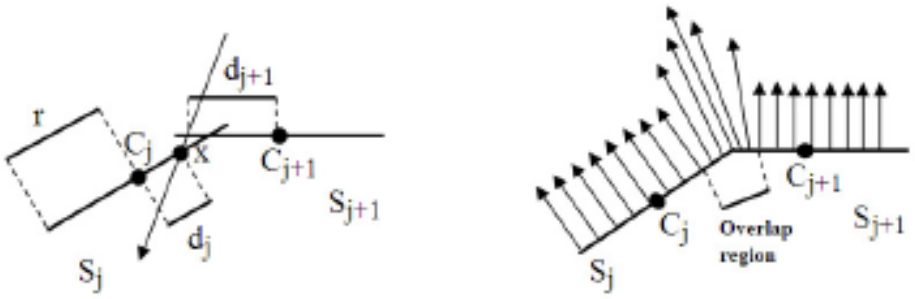


Figure 31. Interpolation polynomiale de la normale.

La résolution des conflits présents lors du calcul d'intersections d'un rayon avec une surface ayant une simple courbure concave ou convexe a bien réussi. Cependant pour des objets arbitraires ayant à la fois des courbures fortes et faibles, un calcul de visibilité plus compliqué doit être trouvé. Pour cette raison, nous avons proposé un algorithme à deux phases qui opère comme suit:

1. **Phase des intersections loin**
   A. **Cas d'un ou de multiples intersections avec des faces avant/arrière**
      On prend les toutes premières intersections à l'intérieur d'un petit intervalle, qui se présentent avant la rencontre une intersection d'une face arrière (**Figure 32**).
   B. **Cas d'un ou de multiples intersections avec des faces avant**
      Dans ce cas, les intersections avec les faces avant ne seront pas prises en compte parce que il n'y a pas d'intersection avec une face arrière qui vient après. C'est à dire, il s'agit d'une intersection avec une frontière de la silhouette. Il est à noter que l'objet doit être fermé et sans trous pour que cet algorithme réussisse (**Figure 32**).
2. **Phase des intersections proches**
   A l'issue de la phase 1, on a une intersection (type loin), donc on se base sur l'étude de courbures géométriques discrètes (convexité et concavité) des splats superposés à l'intervalle $\boldsymbol{\varepsilon}$, afin de fixer la position et la normale finale comme montré précédemment (**Figure 32**).

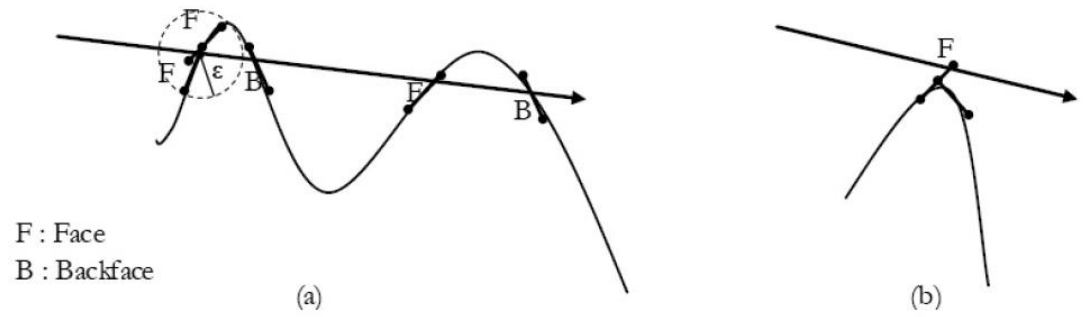
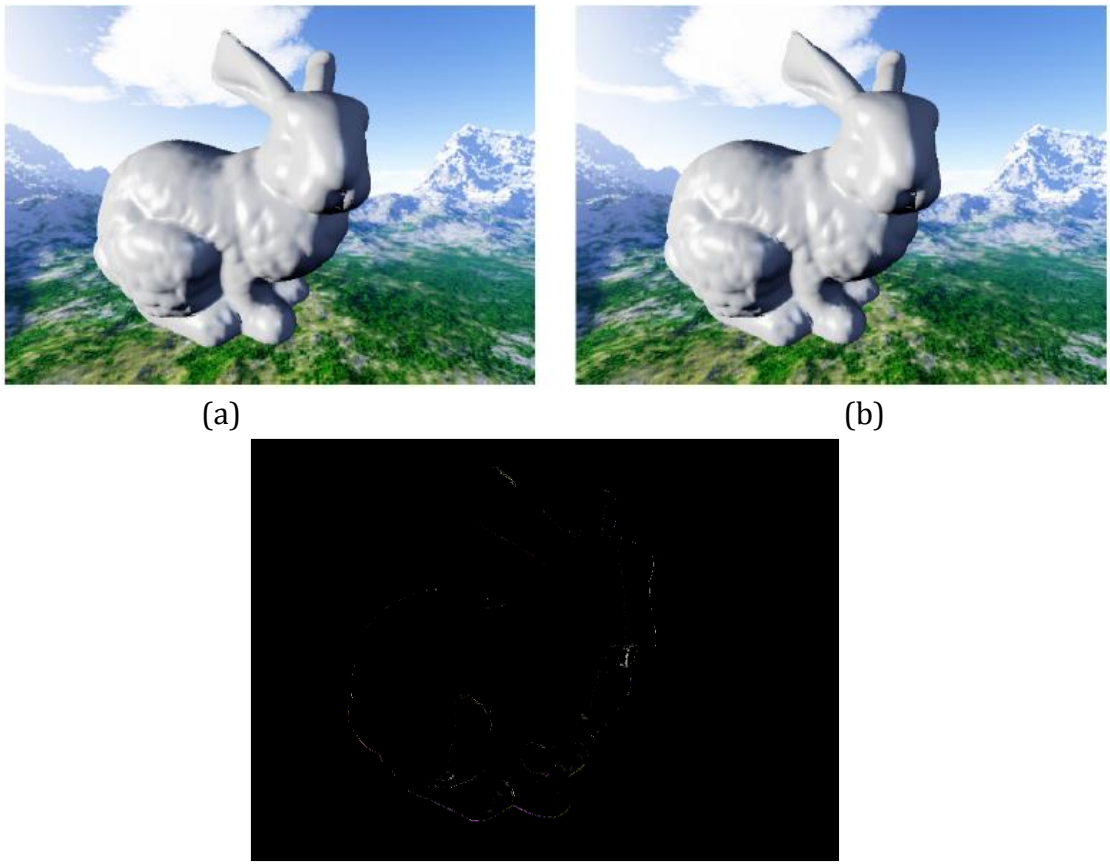


Figure 32. (a)Multiples intersections avec des faces avant/arrière. (b) intersection unique avec une face avant.

.

Les résultats obtenus après l'application de l'algorithme à deux phases, sont montrés dans **Figure 33**. Notons que l'image de différence est calculée en parcourant chaque pixel de l'image et en faisant la différence sur chaque canal RGB.



(a) (b)

(c)

Figure 33. Bunny rendu (a) avec des splats chevauchés. (b) avec résolution de conflits. (c) Image de différence.

2.4. Conclusion

Après avoir implémenté un noyau de lancer de rayon sur les points utilisant la technique de reconstruction locale par splats, l'avoir augmenté par rapport à ce qui est dans l'état de l'art sur deux points importants; rapidité de calcul, et qualité de la surface finale reconstruite, nous pouvons aller vers une technique d'illumination globale basée sur le lancer de rayon afin d'atteindre notre objectif de départ visé dans ce travail de recherche.

Pour cette raison, dans le prochain chapitre, nous détaillons les techniques existantes d'illumination globale et nous focalisons surtout sur celles basées sur le lancer de rayon afin d'étendre notre lancer de rayon vers l'une des techniques d'illumination globale.

Chapitre 3 :

Techniques d'illumination globale

3.1. Introduction

La production des images de synthèse photo-réalistes est assurée utilisant des techniques d'éclairage appelées l'illumination globale. Cette dernière peut être décomposée en deux parties indépendantes : directe et indirecte.

L'éclairage direct qui concerne l'éclairage des surfaces en utilisant uniquement les sources lumineuses de la scène. Son calcul est relativement simple, tandis que l'éclairage indirect est dû aux multiples réflexions et réfractions de la lumière par les objets de la scène. Le calcul précis de cet éclairage est très coûteux en temps de calcul et en espace mémoire. Les techniques d'illumination globale sont classées en deux catégories: celles dites exactes (ou physiques), et d'autres dites approchées (non physiques).

3.2. Problème de résolution de l'équation de transport de lumière

L'équation de transport de lumière ou autrement dit l'équation de rendu telle qu'elle est présentée par Kajiya (Kajiya 1986) est formulée comme suit:

$$L_s(k_o) = \int_{k_i} \rho(k_i, k_o) L_f(k_i) \cos\theta_i d\sigma_i$$

Cette équation définit l'éclairage indirect par la luminance en un point '**s**' dans la scène comme étant l'intégrale de la luminance incidente (L_f) provenant de toutes les directions multipliée par une fonction BRDF (ρ) et le cosinus de l'angle d'incidence (θ_i).

Cette équation peut être résolue par une méthode d'éléments finis en remplaçant l'intégrale par une sommation (méthode de radiosité). Elle peut être résolue aussi par des méthodes stochastiques comme la méthode de Monte Carlo. A part ces deux méthodes (physiques) qui calculent l'illumination globale en résolvant l'équation de rendu, Il existe d'autres méthodes non physiques où le point de départ n'est pas l'équation de rendu, mais elles focalisent sur le calcul d'éclairage d'une manière approchée et rapide.

3.3. Techniques physiques d'illumination globale

3.3.1. Radiosité

C'est une technique de calcul d'éclairage global en se basant sur la physique du transfert radiatif de la lumière entre les surfaces élémentaires d'une scène tridimensionnelle. Elle est basée sur des considérations d'équilibre énergétique, ainsi les inter-réflexions entre surfaces diffuses sont bien prises en compte. La radiosité a été introduite à l'université de Cornell (Cohen et Greenberg 1985). Malgré qu'elle est très coûteuse en matière de calcul, elle a

cependant l'avantage d'être indépendante du point de vue (position de l'observateur), et donc le calcul illumination peut se faire en amont (comme phase de prétraitement), ensuite une navigation dans des environnements virtuels (avec probablement un casque de réalité virtuelle) peut avoir lieu en changeant la position de l'observateur (sans refaire les calculs d'éclairage).

L'expression des équations de radiosité est faite sur la base de l'hypothèse que les phénomènes d'émission et de réflexion sont dus à des réflecteurs parfaits, donc la direction des rayons est perdue après toute réflexion. On commence par définir les termes suivants:

- **Radiosité (B):** c'est la quantité de base qu'on cherche à calculer pour chaque surface; c'est une énergie par unité de surface et de temps.
- **Réflectivité R:** c'est la fraction de lumière réfléchie sur une surface (elle est normalisée entre 0 et 1). On note que l'absorption vaut **(1-R)**.
- **Facteur de forme (F):** c'est la fraction de la lumière quittant une surface et arrivant à une autre (elle est normalisée entre 0 et 1).
- **Emission (E):** c'est l'énergie émise par la surface (comme pour le cas d'une source lumineuse). Elle est exprimée par unité de surface et de temps.

Pour calculer la radiosité d'un élément de surface dA_i (voir **Figure 34**), l'intensité de cet élément va dépendre de toute la lumière qu'il émet directement plus la lumière qui est réfléchie. Cette lumière réfléchie dépend de la lumière quittant toute autre surface dans l'environnement. Une fraction de la lumière quittant toute autre surface peut arriver à l'élément de surface en question et peut être réfléchie à son tour dans l'environnement. La fraction dépend du facteur de forme entre les surfaces et la réflectivité de l'élément de surface.

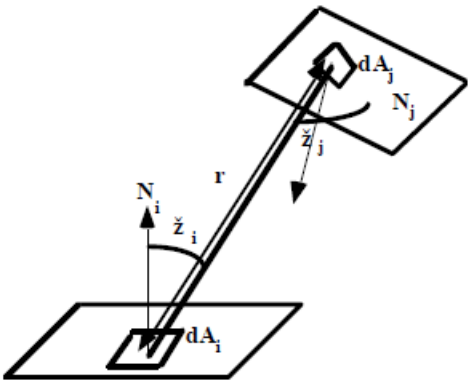


Figure 34. Interprétation géométrique.

Formellement parlant, la radiosité est exprimée comme suit:

$$B_{dA_i} dA_i = E_{dA_i} + \rho_{dA_i} \int B_{dA_j} F_{dA_j - dA_i} dA_j$$

où:

$\boldsymbol{B_{dA_i}}$ est la radiosité de l'élément de surface $\boldsymbol{dA_i}$

$\boldsymbol{E_{dA_i}}$ est l'émission de l'élément de surface $\boldsymbol{dA_i}$

$\boldsymbol{\rho_{dA_i}}$ est la réflectivité de l'élément de surface $\boldsymbol{dA_i}$

$\boldsymbol{F_{dA_j - dA_i}}$ est le facteur de forme de $\boldsymbol{dA_j}$ à $\boldsymbol{dA_i}$; c'est la fraction d'énergie quittant $\boldsymbol{dA_j}$ pour arriver en $\boldsymbol{dA_i}$.

Pour résoudre cette équation, il faut discrétiser l'environnement en le subdivisant en des régions discrètes pour lesquelles on assume que la radiosité et l'émission sont constantes dans la région. On aura donc la nouvelle équation:

$$B_{A_i} A_i = E_{A_i} + \rho_{A_i} \sum_j B_{A_j} F_{A_j - A_i} A_j$$

En considérant que la fraction d'énergie émise par un élément et reçue par un autre est identique à la fraction d'énergie émise dans l'autre sens, on peut exprimer les facteurs de forme entre les surfaces **i** et **j** simplement comme le rapport des surfaces:

$$F_{A_i - A_j} = F_{A_j - A_i} * \frac{A_j}{A_i}$$

On en déduit l'équation fondamentale pour calculer la radiosité:

$$B_{A_i} = E_{A_i} + \rho_{A_i} \sum_j B_{A_j} F_{A_i - A_j}$$

Si l'environnement est divisé en **N** régions, il en résultera un système linéaire de **N** équations que l'on peut mettre sous forme matricielle et le résoudre avec la méthode de Gauss-Seidel.

3.3.2. Path tracing

Le path tracing est une variante du lancer de rayon, pour le calcul de l'illumination globale comprenant à la fois l'éclairage direct et direct, pour produire des images photo-réalistes. Nous commençons d'abord par l'étude du lancer de rayon

3.3.2.1. Lancer de rayon

Le lancer de rayon (en anglais **raytracing**), quant à lui, est une ancienne technique basée sur la simulation numérique d'optique géométrique. Intuitivement, on peut considérer une méthode dans laquelle les rayons

lumineux sont lancés ou tracés de la source de lumière, suivant leur chemin jusqu'à l'observateur.

Cette approche est délicate et même impossible car seuls quelques rayons arrivent à l'observateur. Pour cette raison, il est préférable de renverser la direction de propagation des rayons, en traçant les rayons à partir de l'observateur, comme montré sur **Figure 35**.

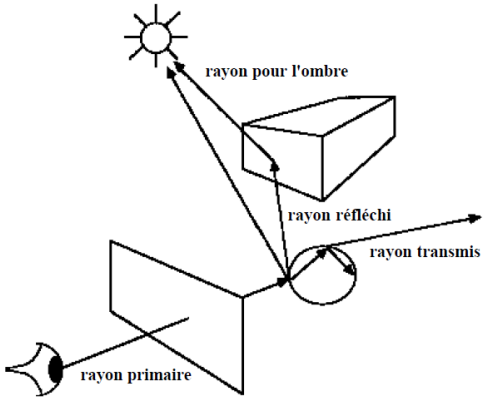


Figure 35. Principe du lancer de rayon.

Un algorithme de lancer de rayon revient à lancer des rayons à partir de l'observateur pour chaque pixel, puis calculer les intersections rayon-objet de la scène, ensuite, obtenir les informations photométriques afin de trouver les composantes de couleur du pixel en question.

A chaque point d'intersection rayon-surface, de nouveaux rayons sont engendrés. Un rayon pour l'ombre est ainsi lancé à partir du point d'intersection vers la (ou les) source(s) de lumière pour déterminer si un éclairage direct a eu lieu. Si c'est le cas, une intensité du rayon est calculée en se basant sur les propriétés de réflectance (diffuse et spéculaire) de la surface, ainsi l'intensité de la source de lumière. Des rayons réfléchis et transmis sont probablement engendrés. Pour ces derniers, il faut sans doute appliquer récursivement le même processus afin de déterminer les intersections de ces rayons avec d'autres surfaces. De cette manière, un arbre d'intersection est construit pour chaque pixel (**Figure 36**). Ce processus récursif s'arrête dans l'un des cas suivants:

- soit le rayon quitte la scène,
- soit le rayon percute une surface ni spéculaire ni transparente,
- soit la contribution du rayon devient négligeable
- ou encore une profondeur maximale de la récursivité est atteinte.

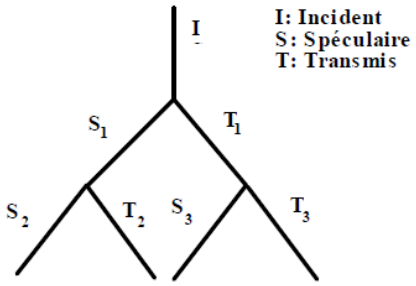


Figure 36. Arbre d'intersections.

Lorsque l'arbre est créé, il est parcouru en appliquant une équation à chaque nœud afin de calculer l'intensité. Selon ce processus récursif, l'intensité pour le nœud courant est obtenue quand tous les sous-nœuds sont évalués. L'équation de calcul d'intensité à appliquer à chaque nœud, est celle de Whitted (Whitted 1980) qui est basée sur les lois de la réflexion et de la réfraction (loi de Snell-Descartes). Selon le modèle de Whitted, l'illumination est donnée par l'équation:

$$I = I_a + I_d + I_s + I_t$$

où la lumière ambiante et la lumière diffuse sont calculées selon la formule de Phong. La lumière spéculaire est obtenue par l'équation:

$$I_s = k_s S$$

où **S** est la lumière spéculaire incidente (de direction **R**) et $\boldsymbol{k_s}$ est une constante. On a en plus une lumière transmise donnée par l'équation:

$$I_t = k_t T$$

où **T** est l'intensité du rayon transmis (de direction **P**) et $\boldsymbol{k_t}$ est une constante.

Les directions des rayons sont obtenues à partir des équations de la réflexion et de la réfraction (Snell-Descartes) où on considère l'angle d'incidence **i**, l'angle de réfraction **r** et l'indice de réfraction **n** du premier matériau relativement au second. On a:

1. Le rayon incident, la normale au point incident et le rayon réfracté sont dans un même plan.
2. Le rayon incident et le rayon réfléchi forment un même angle avec la normale, c'est à dire **r = i**.

On détermine ainsi la direction **R** du rayon réfléchi:

$$V' = \frac{V}{|V.N|}$$
$$R = V' + 2N$$

3. Le sinus de l'angle d'incidence est en rapport avec le sinus de l'angle de réfraction, comme suit:

$$\sin(i) = n\sin(r)$$

On détermine ainsi la direction **P** du rayon réfracté:

$$P = \frac{N + V'}{\sqrt{n^2|V'|^2 - |V' + N|^2}} - N$$

Figure 37.nous montre la situation d'un point de vue géométrique.

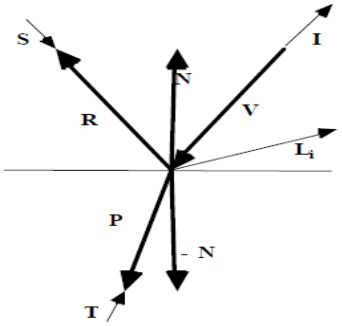


Figure 37. La réfraction.

L'algorithme du lancer de rayon est très puissant et permet de traiter pratiquement tous les aspects du réalisme. Malheureusement, les calculs d'intersections sont très coûteux en temps, car ils sont étroitement liés au nombre de rayons lancés, et seules des machines puissantes peuvent se permettre de tels calculs dans un temps raisonnable. Il faut donc trouver des méthodes d'optimisation. Pour cela, il existe plusieurs stratégies envisageables; augmenter la vitesse de traitement de ces intersections, trouver des moyens pour tester plus rapidement s'il y a possibilité d'intersection, diminuer le nombre de rayons lancés, ou encore utiliser des rayons plus complexes que des droites (faisceaux ou cônes, par exemple).

3.3.2.2. Path tracing

On désigne communément par le terme path tracing, la méthode de path tracing stochastique de Monte Carlo. On appelle ainsi toute méthode de calcul qui se base sur les variables aléatoires. Généralement c'est une estimation d'un calcul (une intégrale) via la moyenne des échantillons qui ont chacun une probabilité. C'est une estimation et pas une approximation (l'erreur est inconnue).

On se sert du calcul d'intégrale via la méthode de Monte Carlo pour la résolution de l'équation de rendu telle qu'elle est présentée par Kajiya'86, qui se résume en la somme d'une composante d'éclairage direct (propre à un point de la surface) avec une composante indirecte (qui est en elle-même la somme de toutes les luminances provenant des autres surfaces voisines).

L'équation de rendu selon Kajiya (Kajiya 1986) est la suivante (voir **Figure 38**):

$$L_s(k_0) = \int_{all\ k_i} \rho(k_i, k_0) L_f(k_i) cos\theta_i d\sigma_i$$

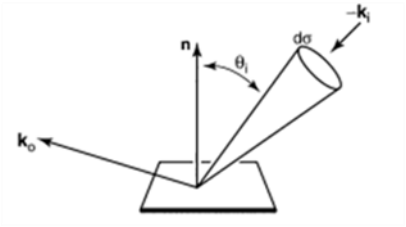


Figure 38. Angle solide.

Les k_i sont les angles solides. Un angle solide est défini comme suit : $\Omega = \mathrm{A}/\mathrm{r}^2$

D'après cette équation, le calcul d'intégrale sur un domaine des angles solides, revient à un calcul d'intégrale sur un domaine de surface (la surface de l'hémisphère comme montre **Figure 39**).

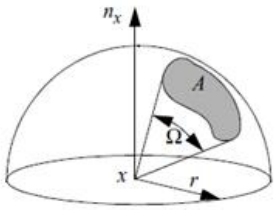


Figure 39. Hémisphère.

L'intégrale précédente est approximée ainsi :

$$\frac{1}{N}\sum_{i=1}^{N} \frac{L(r(x, \Psi_i) \rightarrow -\Psi_i) f_r(x, \Psi_i \leftrightarrow \Theta) \cos(n_x, \Psi_i)}{p(\Psi_i)}$$

Où $p(\Psi_i)$ est la fonction de densité de probabilité (PDF) de la variable aléatoire.

On peut choisir une entre plusieurs PDFs : uniforme, cosinusoïdale, etc.

- ***Echantillonnage uniforme***

$$\mathrm{p}(\Psi_i) = \frac{1}{2\pi}$$

$$\frac{2\pi}{N}\sum_{i=1}^{N} L(r(x,\Psi_i) \rightarrow -\Psi_i) f_r(x,\Psi_i \leftrightarrow \Theta)\cos(n_x,\Psi_i)$$

- ***Echantillonnage cosinusoïdale***

$$p(\Psi_i) = \cos(n,\Psi_i)/\pi$$

$$\frac{\pi}{N}\sum_{i=1}^{N} L(r(x,\Psi_i) \rightarrow -\Psi_i) f_r(x,\Psi_i \leftrightarrow \Theta)$$

L'idéal est de choisir la loi de probabilité la plus adaptée à la surface à intégrer, autrement dit, l'échantillonnage cosinusoïdale est le plus adapté à la surface de l'hémisphère ; mathématiquement parlant, l'équation de rendu devient simplifiée car des termes de l'équation seront éliminés.

- ***Contribution***: Nous avons implémenté le path tracing stochastique (méthode de Monte Carlo) avec un échantillonnage cosinusoïdale (et précisément l'échantillonnage d'importance) sur une scènes 3D complètement diffuse, comportant un cornell-box et un objet dedans (bunny) à base de points (plus spécifiquement des splats) avec 9 échantillons et 9 rebonds, comme montre **Figure 40**.

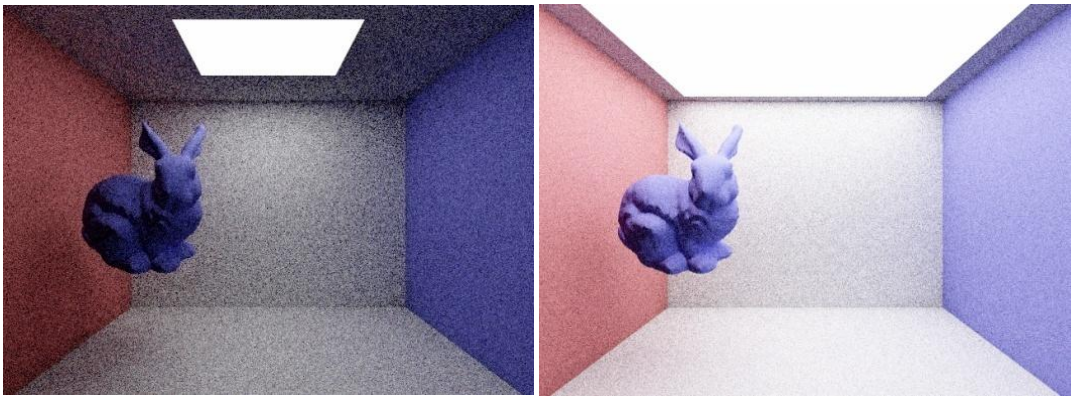


Figure 40. Rendu par path tracing d'une scène à base de points.

On voit que la qualité des images est acceptable, cependant les images produites par les méthodes stochastiques restent toujours bruitées, et nécessitent un temps énorme pour converger (en augmentant le nombre de rayons envoyés par pixel). Nous avons même d'intégrer l'équation de rendu utilisant des techniques numériques (telles que la quadrature de Gauss) afin d'éviter les bruits dus à l'échantillonnage stochastique, mais cette idée ne nous

a pas donné de bons résultats (*idée communiquée dans une conférence nationale*).

3.4. Techniques approchées d'illumination globale

3.4.1. Cache de luminance

La technique du cache de luminance (Ward, Rubinstein et Clear 1988), est une technique efficace pour l'accélération du calcul de l'illumination indirecte pour les scènes diffuses. Utilisant un échantillonnage de Monte Carlo pour le calcul de la luminance incidente (irradiance en anglais) à un point nécessite des centaines d'opérations de lancer de rayon.

Chaque opération, peut engendrer à son tour d'autres rayons dans la scène, c'est pour cette raison que le calcul devient extrêmement lent. La technique de cache de luminance exploite le fait que la luminance aux surfaces diffuses varie en général d'une manière lisse.

Ainsi, la luminance préalablement calculée, est stockée dans une structure de données, et en cas de besoin cette valeur stockée sera interpolée pour approcher la luminance des surfaces voisines.

La luminance est suffisante pour représenter l'éclairage indirect diffus, parce que la fonction BRDF est indépendante du point de vue (view-independent):

$$\boldsymbol{E(p) = \int_{\Omega} L_i(p, \omega_i) cos\theta_i d\omega_i}.$$

La luminance émise dans une direction aléatoire $\boldsymbol{\omega_0}$ peut être calculée en fonction de la luminance incidente:

$$\boldsymbol{L_0(p, \omega_i) = E(p)\rho(p)/\pi}$$

où $\boldsymbol{\rho(p)}$ est la réflectance diffuse (albedo) au point **p**.

Chaque valeur de luminance est calculée utilisant un couteux échantillonnage de l'hémisphère, et stockée dans un cache pour une utilisation future. Afin d'éviter le calcul de luminance à chaque point de la scène, un schéma de cache est utilisé comme suit (voir **Figure 41**):

- **Si** une ou plusieurs valeurs de luminance sont stockées dans le voisinage **Alors**
  Utiliser cette valeur.
- **Sinon**
  Calculer et stocker la luminance à ce point.

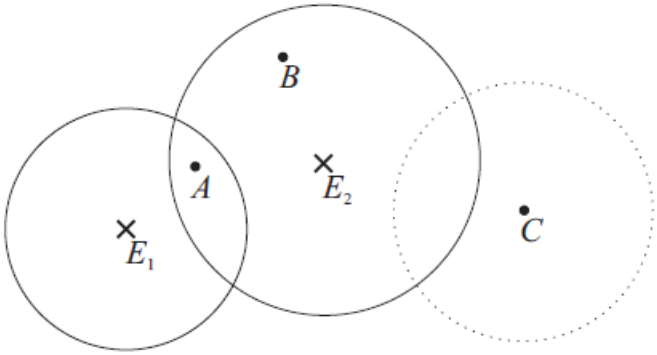


Figure 41. Luminances pré-calculées et stockées aux points E_1 et E_2. (Ward, Rubinstein et Clear 1988).

La luminance au point **p** est interpolée à partir des valeurs stockées dans le cache comme suit:

$$E_\vartheta(p) = \frac{\sum_{i\in\vartheta}(E_i + (n_i * n) \,.\, \nabla_r E_i + (p - p_i) \,.\, \nabla_t E_i) w_i(p)}{\sum_{i\in\vartheta} w_i(p)}$$

où $\mathbf{p_i}$ est la position du i[ème] valeur de luminance du cache ($\boldsymbol{E_i}$), et $\boldsymbol{n_i}$ est la normale de la surface au point $\mathbf{p_i}$. $\boldsymbol{\nabla_r E_i}$ et $\boldsymbol{\nabla_t E_i}$ sont les gradients de la luminance, qui sont utilisés pour une interpolation lisse. Ils sont calculés et stockés dans le cache aussi.

La pondération de la **i**[ème] valeur de luminance est faite selon la formule suivante:

$$w_i(p) = \left(\frac{\|p - p_i\|}{R_i} + \sqrt{1 - n \cdot n_i}\right)^{-1}$$

où $\boldsymbol{R_i}$ est la moyenne harmonique des distances des objets visibles depuis le point $\mathbf{p_i}$. $\boldsymbol{\vartheta}$ est l'ensemble des valeurs de luminance qui peuvent être utilisées pour l'interpolation au point **p**, défini comme suit:

$$\vartheta = \{i;\ \ w_i(p) > \frac{1}{a}\ \}$$

où **a** est l'erreur d'approximation désirée définie par l'utilisateur. En d'autres mots, la valeur **i** peut être utilisée seulement aux alentours de son endroit où la pondération $\boldsymbol{w_i(p)}$ est supérieure strictement au seuil $\mathbf{1/a}$. Plus la valeur de **a** est grande, plus la tolérance d'interpolation est grande et moins l'image finale est exacte. Si l'ensemble $\boldsymbol{\vartheta}$ est vide à un point donné, une nouvelle valeur de luminance est calculée et stockée dans le cache.

La pondération $\boldsymbol{w_i(p)}$ est égale à l'inverse de la borne supérieure du gradient de la luminance. En conséquence, la densité de la luminance stockée

dans le cache est petite dans les régions de scène ouvertes et plates où la luminance ne change pas rapidement. Par contre, cette densité est grande est importante dans les surfaces courbées où des changements brusques de la luminance sont probables.

3.4.2. Photo mapping

La technique du photon mapping (Jensen 1996) est un algorithme à deux passes, tout comme le path tracing bidirectionnel, qui trace des chemins d'illumination depuis des sources de lumière vers l'œil de l'observateur et vice versa. Cependant, cette approche stocke et réutilise des valeurs d'illumination pré-calculées pour des raisons d'efficacité. Dans la première passe, des photons sont tracés depuis des sources de lumière dans la scène. Ces photons, qui portent un flux d'informations, sont stockés dans une structure de données appelée carte de photons (photon map). Dans la deuxième passe, une image est rendue utilisant les informations de la carte de photons (**Figure 42**).

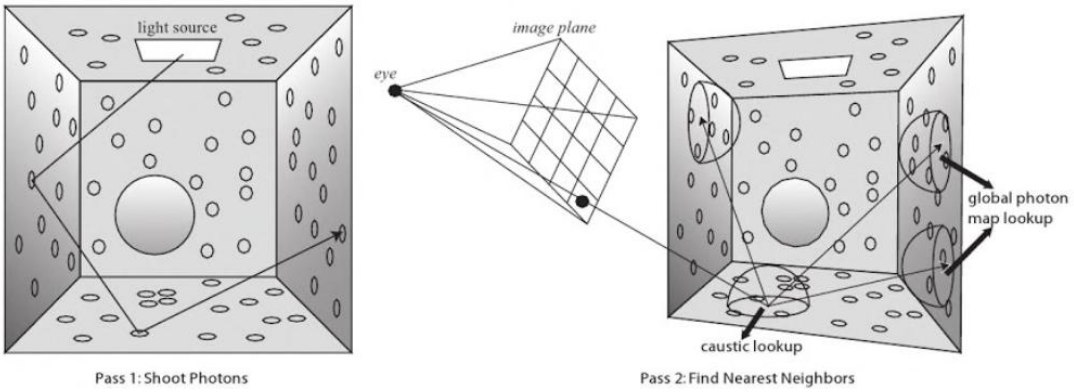


Figure 42.Deux passes de l'algorithme de photon mapping: (a) distribution des photons. (b) rendu par estimation de luminance.

3.4.2.1. Première passe: tracé de photons

Le tracé de photons est le processus d'émission de photons discrets à partir des sources de lumière à travers la scène. L'objectif principal de cette passe est de remplir les cartes de photons utilisées dans la passe de rendu pour calculer la luminance réfléchie sur les surfaces, et la luminance diffuse dans les milieux participants.

3.4.2.1.1. Emission de photons

Le cycle de vie d'un photon commence depuis la source de lumière. Pour chaque source de lumière dans la scène, on crée un ensemble de photons et on divise la puissance globale de la source lumineuse entre eux. Les lumières plus brillantes émettent plus de photons que des lumières plus faibles. La détermination du nombre de photons à créer à chaque lumière dépend en grande partie du fait que des estimations de luminance peuvent être faites pendant la passe de rendu. Pour de bonnes estimations de luminance, la densité

locale des photons sur les surfaces doit fournir une bonne statistique de l'illumination.

Jensen (Jensen 1996) déclare que tout type de source de lumière peut être utilisé, et décrit plusieurs modèles d'émission. Pour les sources lumineuses ponctuelles, on doit émettre des photons uniformément dans toutes les directions. Pour les sources lumineuses surfaciques, on choisit une position aléatoire sur la surface, puis une direction aléatoire dans l'hémisphère au-dessus de cette position.

La stratégie d'émission peut être modifiée en fonction de la scène. Par exemple, pour les scènes dispersées, on doit concentrer notre émission de photons sur la géométrie, sinon la plupart des photons seront perdus. La solution de Jensen est d'utiliser des cartes de projection. Les cartes de projection optimisent l'émission de photons en dirigeant les photons vers des objets importants. Les cartes de projection sont généralement implémentées en tant que bitmaps qui sont déformées sur une forme de délimitation pour la source de lumière où chaque bit détermine si la géométrie d'importance est dans cette direction.

Les cartes de projection sont également très importantes pour des effets tels que les caustiques. Les caustiques sont générés à partir de la lumière focalisée provenant des surfaces spéculaires et nécessitent une densité plus élevée de photons pour une estimation précise de la luminance. Avec les cartes de projection, on peut concentrer plus de photons vers des surfaces spéculaires.

3.4.2.2.2. Diffusion de photons

Les photons émis par les sources lumineuses sont dispersés à travers une scène et sont finalement absorbés ou perdus. Lorsqu'un photon frappe une surface, on peut décider quelle quantité d'énergie est absorbée, réfléchie et réfractée en fonction des propriétés de la surface.

Pour tenir en compte la distribution de l'énergie du photon à la surface, on décompose en petits photons si nécessaire. Cependant, il est facile de voir que la génération de nouveaux photons à chaque interaction avec la surface, sera très coûteuse en termes de calcul et de stockage. Au lieu de faire cela, Jensen préconise l'utilisation d'une technique de Monte Carlo standard appelée la roulette russe. Nous utilisons la roulette russe pour décider de façon probabiliste si les photons sont réfléchis, réfractés ou absorbés. En utilisant cette technique, nous réduisons les coûts de calcul et de stockage tout en obtenant le résultat correct.

Il est important de noter que la puissance du photon réfléchi n'est pas modifiée. L'exactitude du résultat global convergera avec plus d'échantillons. La probabilité est donnée par la réflectivité d'une surface. Si la réflectivité pour une

surface est de 0.5, alors 50% des photons seront réfléchis à pleine puissance tandis que les 50% restants seront absorbés. Pour montrer pourquoi la roulette russe réduit les coûts de calcul et de stockage, considérons 1000 photons sur une surface avec une réflectivité de 0.5. On peut refléter 1000 photons à la moitié de la puissance ou on peut refléter 500 à pleine puissance en utilisant la roulette russe.

3.4.2.2.3. Stockage de photons

Pour une scène donnée, on peut tirer des millions de photons des sources lumineuses. Il est souhaitable que notre carte de photons soit compacte pour réduire les coûts de stockage. on veut également qu'elle supporte des recherches spatiales tridimensionnelles rapides car on doit interroger la carte de photons des millions de fois pendant la phase de rendu.

La structure de données que Jensen recommande d'utiliser pour la carte des photons est un KD-tree. C'est l'une des rares structures de données idéales pour traiter des distributions non uniformes de photons. La complexité du cas le plus défavorable pour localiser des photons dans un KD-tree est de complexité **O**(**n**) où, comme s'il était équilibré, il est **O** (**log n**). Après que tous les photons soient stockés dans la carte, on doit s'assurer que l'arbre est équilibré.

Pour chaque photon, on stocke sa position, sa puissance et sa direction d'incidence. Jensen propose la structure suivante.

```
struct photon {
float x, y, z ;        // position (3 flottants sur 32 bits)
char p [4] ;           // puissance (RGB) sur 4 caractères
char phi, theta ;      // direction d'incidence
short flag ;           // drapeau utilisé pour KD-tree
}
```

Jensen prescrit le format RGB de Ward pour coder la puissance en 4 octets. **phi** et **thêta** sont des coordonnées sphériques qui correspondent à 65536 directions possibles.

Cependant, pour les implémentations les plus sérieuses, la structure sera aussi compressée que possible pour permettre des scènes extrêmement volumineuses et complexes. Pour les moteurs de rendu novices, la structure peut rester largement non compressée pour faciliter l'utilisation.

3.4.2.2. Deuxième passe: rendu

Le photon mapping est utilisé pour calculer les effets d'éclairage indirect et des caustiques. Trop de photons seraient nécessaires dans la carte des photons pour traiter avec précision les surfaces spéculaires/brillantes.

Au cours de la passe de rendu, nous utilisons une **approximation** de l'équation de rendu pour calculer la luminance réfléchie aux endroits de la surface. L'équation de rendu est décrite comme suit:

$$L_r(x,\vec{\omega}) = \int_{\Omega} f_r\left(x \quad ,\overrightarrow{\omega'},\vec{\omega}\right) L_i(x,\overrightarrow{\omega'}) \cos\theta_i \, d\overrightarrow{\omega'}$$

où:

- **x** est une position de la surface.
- $\overrightarrow{\boldsymbol{\omega'}}$ est la direction de la luminance incidente
- $\overrightarrow{\boldsymbol{\omega}}$ est la direction émise depuis la surface.
- $\boldsymbol{f_r}$ est la BRDF.
- $\boldsymbol{L_i}$ est la luminance incidente.
- $\boldsymbol{\theta_i}$ est l'angle d'incidence.
- $\boldsymbol{d\overrightarrow{\omega'}}$ est l'angle solide.
- **Ω** est l'hémisphère.

L'approximation de l'équation de rendu donne:

$$L_r(x,\vec{\omega}) = \sum_{p=1}^{n} f_r(x,\overrightarrow{\omega_p},\vec{\omega}) \frac{\Delta\phi_p(x,\overrightarrow{\omega_p}}{\pi r^2}$$

Afin d'obtenir une bonne estimation de la luminance, on a besoin d'un nombre suffisant de photons dans notre estimation de densité. Bien sûr, cela peut directement se rapporter au nombre de photons émis par les sources lumineuses. Plus de photons sont utilisés, plus l'estimation est précise. On doit également faire attention à la façon dont on collecte les photons.

L'idée principale derrière la collecte de photons est qu'on espère avoir une idée de ce qu'est l'illumination à **x** en examinant les **N** photons les plus proches. Si on inclut des photons depuis autres surfaces, avec des normales de surface radicalement différentes, donc on peut dégrader la précision de l'estimation.

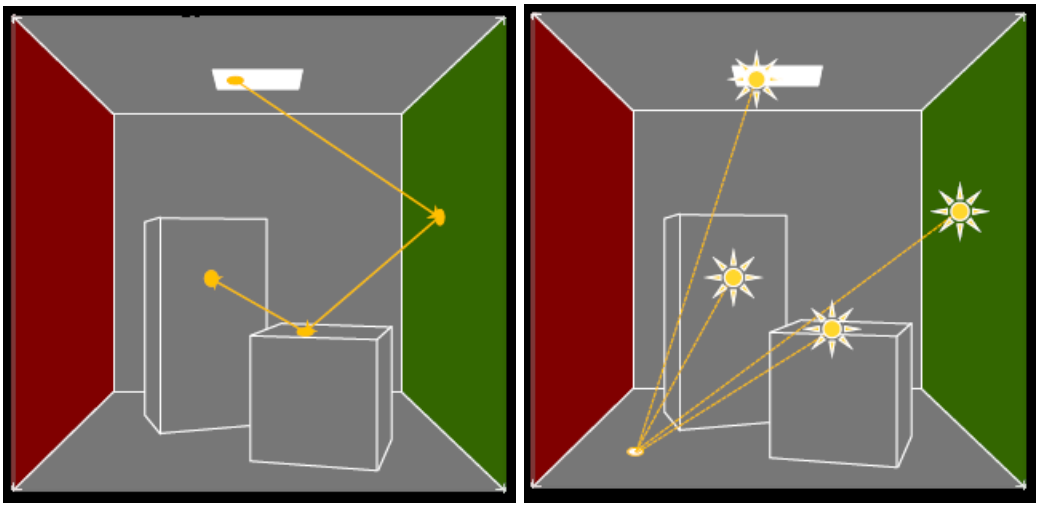
3.4.3. VPL: Virtual Point Light

VPL, autrement appelé la radiosité instantanée (Keller 1997), est une technique hybride liée au path tracing bidirectionnel. Son principe est de

remplacer l'illumination diffuse indirecte dans une scène par une illumination diffuse directe de plusieurs sources ponctuelles de lumière. Les sources ponctuelles sont placées dans des endroits où des trajectoires de photons sont supposées passées à travers. Cette approche peut être vue comme une méthode d'estimation de la densité, utilisant le noyau de l'intégrale d'équation de radiosité. L'avantage principal de la radiosité instantanée se situe dans l'échantillonnage corrélé positivement pour tous les pixels, parce que les mêmes chemins de lumières sont utilisés pour tous les pixels. Ainsi, une image calculée avec la radiosité instantanée, semble être lisse et ne présente pas d'artefacts de type bruit (habituellement vus avec le path tracing bidirectionnel).

Comme montre **Figure 43**, la radiosité instantanée passe par deux étapes:

1. Des photons sont tracés depuis la source de lumière dans la scène, puis les sommets des chemins créés par ce tracé de photons seront considérés comme des VPLs: Virtual Point Light (**Figure 43a**).
2. La scène est rendue plusieurs fois pour chaque VPL (**Figure 43b**).



(a) **(b)**

Figure 43. Deux étapes de la radiosité instantanée. (a) le placement des VPLs. (b) le rendu.

3.4.4. RSM: Reflective Shadow Maps

C'est un algorithme (Dachsbacher et Stamminger 2005) qui étend la technique du shadowmap utilisée pour le calcul d'ombre, pour faire un rendu interactif comprenant de l'illumination indirecte. Contrairement ce que les pixels du shadowmap signifient, chaque pixel du RSM est considéré comme une source de lumière indirecte. Ainsi, la composante indirecte du calcul d'illumination est interpolée dans l'espace écran à partir de ces sources indirectes via des calculs au niveau du processeur graphique programmable (**Figure 44**).

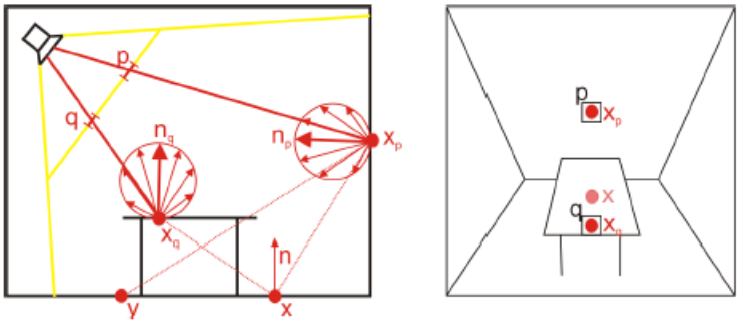


Figure 44. Calcul de l'illumination indirecte via la carte d'ombre réflective (Dachsbacher et Stamminger 2005).

3.4.5. AO: Ambient Occlusion

En infographie, l'occlusion ambiante est une technique d'ombrage et de rendu utilisée pour calculer l'exposition de chaque point d'une scène à l'éclairage ambiant. L'occlusion ambiante peut être vue comme une valeur d'accessibilité calculée pour chaque point de surface. Le résultat est un effet d'ombrage diffus et non-directionnel qui ne projette pas d'ombres claires mais qui assombrit les zones fermées et abritées et peut affecter le ton général de l'image rendue (voir **Figure 45**).

Contrairement aux méthodes locales telles que l'ombrage de Phong, l'occlusion ambiante est une méthode globale (Bunnell 2005), ce qui signifie que l'éclairage à chaque point dépend de l'autre géométrie de la scène. Cependant, il s'agit d'une approximation très grossière de l'illumination globale complète. L'aspect obtenu par l'occlusion ambiante seule est similaire à la façon dont un objet peut apparaître un jour couvert.

Figure 45. Occlusion ambiante d'une voiture.

3.4.6. ISM: Imperfect Shadow Maps

C'est une méthode de calcul interactif de l'illumination indirecte dans des scènes entièrement dynamiques basées sur des tests de visibilité (Ritschel, et al. 2008). Pendant que la nature élevée de fréquences de l'éclairage direct nécessite une visibilité précise, l'éclairage indirect consiste principalement à des gradations lisses, qui ont tendance à masquer les erreurs dues à des erreurs de visibilités. Cette technique exploite cela, en calculant par approximation la visibilité pour faire une illumination indirecte via des carte d'ombres imparfaites de basses résolutions, à partir d'une représentation brute de la scène basée points (**Figure 46**).

Ces cartes sont utilisées conjointement avec un algorithme d'illumination globale basé sur des VPLs permettant l'illumination indirecte des scènes dynamiques en temps réel.

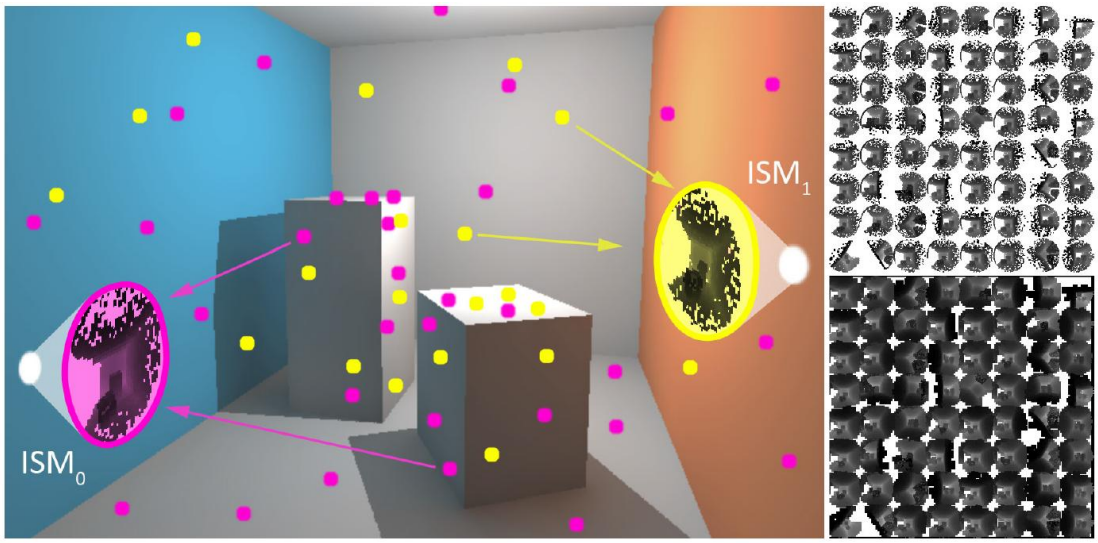


Figure 46. Carte d'ombre créée en rendant la scène depuis plusieurs sources ponctuelles (Ritschel, et al. 2008).

Comme mentionné précédemment, l'éclairage indirect est basé sur l'idée de la radiosité instantanée, où les VPLs sont distribués de manière stochastique le long des chemins de lumière qui agissent ensuite comme expéditeurs de lumière indirecte. En sommant la contribution de tous les VPLs, tout en tenant compte de l'effet d'ombre, cela donne des résultats d'éclairage indirect. Les cartes d'ombre imparfaites sont utilisées pour déterminer la visibilité pour chaque VPL, car elles sont très efficaces; des centaines d'ISM peuvent être rendues en une seul passe et leur coût de rendu est dépendant uniquement du nombre total de polygones. Afin de maintenir un haut taux d'affichage, les VPLs sont créés à partir de la source de lumière, complètement sur le GPU. La position 3D de chaque VPL est déterminée en rendant un cubemap depuis un point de vue centré à la source lumineuse ponctuelle, sur laquelle on effectue un échantillonnage d'importance. Chaque VPL crée une carte ISM parabolique,

orientée selon la normale de la surface capturant l'ensemble de l'hémisphère en une seule vue. Seul un sous-ensemble de tous les points est utilisé pour chaque ISM et toutes les ISM sont stockées dans une grande texture. Pour réduire davantage le coût de rendu, un G-Buffer est utilisé pour rassembler efficacement des VPL lors du rendu de l'image finale.

3.4.7. PBGI: Point-Based Global Illumination

Per H. Christensen a publié une approche approximative alternative aux méthodes de simulation physique (path tracing) moins exacte mais plus efficace (Christensen 2008). Elle convient surtout au domaine de la production des films. Cette méthode procède en trois étapes. Dans la première étape, un nuage de points représentant les surfaces éclairées est directement généré. Dans la deuxième étape, les points (surfels) sont organisés dans un octree, et l'illumination de chaque nœud de l'octree est représenté comme un seul point (surfel) ou en utilisant des harmoniques sphériques. Le rendu est fait dans la troisième étape, avec un calcul d'éclairage global à chaque point en utilisant la rastérisation sur un cube.

Les avantages de cette méthode d'illumination globale sont: le calcul rapide, des résultats non bruités, l'illumination globale est aussi rapide que l'occlusion ambiante, et finalement l'éclairage de l'environnement ne prendre pas du temps supplémentaire.

L'inconvénient principale réside dans le fait que l'approche est multi-passe, et que les résultats sont pas garantis d'être aussi précis que le lancer de rayon. L'objectif n'est pas l'exactitude, mais juste des résultats visuellement acceptables, cohérents et sans bruit (Cependant, la méthode converge vers le résultat correct car les surfels deviennent plus petits et plus denses et la résolution de pixellisation est augmentée). La méthode ne convient pas pour la réflexion de miroir, la réfraction aigüe ou l'ombres dures. Le lancer de rayon est simplement mieux à cet effet.

3.5. Conclusion

A la fin de ce chapitre, nous pouvons conclure que les techniques d'illumination globale, généralement, se basent toutes (à part la radiosité) sur l'algorithme de lancer de rayon. Etant donné que la catégorie des techniques approchées vise bien la production des images photo-réalistes, mais surtout un rendu efficace afin de satisfaire les contraintes temporelles des applications diverses (qui exigent souvent un rendu interactif et/ou temps réel). Cependant, pour un rendu correct et exact physiquement parlant, le path tracing est

considéré comme l'unique solution pouvant donner lieu à des images les plus réalistes possible. Pour cette raison nous avons opté pour ce choix afin de proposer, dans le chapitre 5, une solution de rendu pour un cas d'étude, qui est les données (des points tridimensionnels associés à d'autres informations) issues des simulateurs physiques de fluides, mais avant, dans le chapitre suivant (4), nous introduisons d'abord le domaine de simulation et de rendu de fluides.

Partie II:

Fluides à base de particules

Chapitre 4:
Simulation et rendu des fluides

4.1. Introduction

En infographie, le mouvement des fluides est un problème important lors de la simulation de phénomènes quotidiens, par exemple: la pluie, la boue, l'eau qui coule, la fumée de cigarette, la vapeur, la mousse, les vagues océaniques, etc. Dans le domaine de l'animation graphique, deux grandes classes de simulation physique ont été proposées pour étudier la dynamique des fluides: **(1)** les approches Eulériennes basées grilles et **(2)** les approches Lagrangiennes basées particules.

4.2. Approches Eulériennes de simulation de fluides

Dans cette première catégorie d'approches (Fedkiw, Stam et Jensen 2001, Blasi, Saec et Schlick 1993, Losasso, Gibou et Fedkiw 2004), une grille eulérienne est utilisée par le solveur pour stocker dans chaque cellule les propriétés du fluide (densité, vitesse, pression, etc.). Ces méthodes sont largement utilisées, ainsi qu'elles se basent sur les équations de **Navier-Stokes** étendues pour la simulation de nombreux effets intéressants, notamment la tension de surface pour l'animation de l'eau et du lait, la viscoélasticité et l'élastoplasticité du mucus, le pudding et l'argile, et la flottabilité thermique pour l'animation du gaz chaud et turbulent. La majorité des implémentations eulériennes et leurs extensions fonctionnent hors ligne, c'est-à-dire qu'elles ne contribuent pas à une solution interactive.

Les équations de **Navier-Stokes** sont comme suit:

$$\rho\left(\frac{\partial}{\partial t} + u \cdot \nabla\right)u = -\nabla p + \mu\nabla \cdot (\nabla u) + f$$

$$\nabla \cdot u = 0$$

où $\boldsymbol{u}$ est la vélocité, $\boldsymbol{\rho}$ est la densité, et $\boldsymbol{p}$ est la pression, $\boldsymbol{\mu}$ est la viscosité du fluide et $\boldsymbol{f}$ est l'ensemble des forces extérieures qui agissent sur le fluide.

Ces équations formulent le mouvement d'un fluide eulérien. Le fluide est cependant composé de cellules de fluide, alignées dans une grille régulière, chacune contient un certain nombre de molécules de fluide, ou particules. **Figure 47** illustre une disposition de base d'un fluide à base de grille, qui a été réduite à deux dimensions pour des raisons de clarté. En raison de la résolution grossière de la grille, toute la surface du fluide est contenue dans la même rangée, ce qui est considéré comme un problème commun pour les détails et la visualisation.

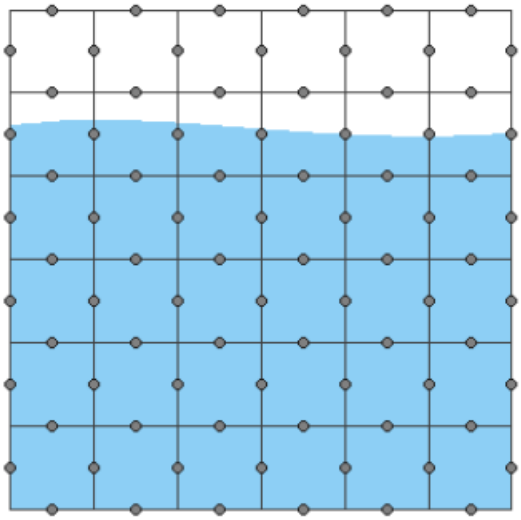


Figure 47. Structure d'un fluide basé grille (vue 2D).

Les équations de **Navier-Stokes** défient une solution analytique complète, mais peuvent être résolues numériquement en utilisant plusieurs étapes pour chaque composante des équations. L'une des étapes les plus importantes est la décomposition de **Helmholtz-Hodge**, qui est une technique mathématique. Les différentes étapes sont résolues en utilisant des intégrateurs explicites, implicites et semi-implicites. La grille fournit une solution pour estimer les dérivés en utilisant une méthode de différence finie (**FDM**).

4.3. Approches Lagrangiennes de simulation de fluides

Dans cette deuxième catégorie (Monaghan 1992, Müller, Charypar et Gross 2003), le mouvement du fluide est déterminé en résolvant les forces d'interaction de l'ensemble de particules. Le résultat de cette opération est un ensemble de points 3D et d'autres variables de simulation (**Figure 48**).

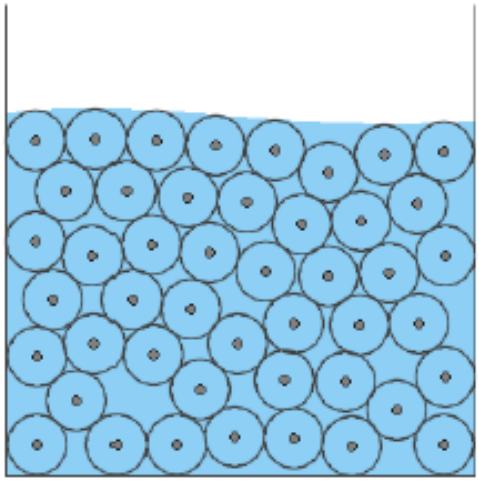


Figure 48. Structure d'un fluide basé particules (vue 2D).

Dans cette classe, on trouve la méthode Smoothed Particle Hydrodynamics (SPH) qui a été initialement proposée dans le but d'étudier et de simuler les problèmes astrophysiques (Gingold et Monaghan 1977). SPH est une méthode d'interpolation utilisée pour approximer les valeurs et les dérivées des grandeurs de champ continues en utilisant des points d'échantillonnage discrets. Les points d'échantillonnage sont identifiés comme des particules lissées qui portent des entités concrètes, par exemple la masse, la position, la vitesse, etc., mais les particules peuvent également porter des quantités de champ physique estimées selon le problème étudié, par exemple, la masse volumique, la température, la pression, etc. Les quantités SPH sont macroscopiques et obtenues en moyenne pondérée à partir des particules adjacentes.

Comparée à d'autres méthodes bien connues pour l'approximation numérique de dérivés, par exemple la méthode des différences finies, qui nécessite que les particules soient alignées sur une grille régulière, SPH peut approcher les dérivées de champs continus en utilisant une différenciation analytique sur des particules situées de manière totalement arbitraire. Chaque particule est considérée comme occupant d'une fraction de l'espace du problème, et pour obtenir des moyennes pondérées plus précises, les particules de l'échantillon doivent être denses.

SPH est fondamentalement une méthode d'interpolation. L'interpolation est basée sur la théorie d'interpolation d'intégrale utilisant des noyaux approchant une fonction. L'intégrale de toute fonction de quantité $\boldsymbol{A}(\boldsymbol{r})$ (qui peut être: la densité, la pression , la vélocité, etc.) est définie sur tout l'espace $\boldsymbol{\Omega}$ par:

$$A_I(r) = \int_{\Omega} A(r^{'})W(r - r^{'}, h)dr'$$

où **r** est un point aléatoire dans **Ω**, **W** est un noyau de lissage de largeur **h.** L'approximation numérique de l'intégrale par sommation donne:

$$A_S(r) = \sum_j A_j \, V_j W(r - r_j, h)$$

où $\boldsymbol{j}$ parcourt l'ensemble des particules, $\boldsymbol{V_j}$ est le volume attribué implicitement a une particules **j**, $\boldsymbol{r_j}$ est une position, et $\boldsymbol{A_j}$ est la valeur d'une quantité quelconque $\boldsymbol{A}$ à la position $\boldsymbol{r_j}$.

Plus tard (chronologiquement), la technique SPH a été utilisée dans le domaine graphique pour simuler le comportement du feu et des gaz (Stam et Fiume 1995).

Ensuite, SPH a été utilisé pour animer des corps déformables (Desbrun et Gascuel 1996). Dans le domaine de la simulation des fluides, Müller et al. (Particle-based fluid simulation for interactive applications 2003) ont été les premiers qui ont proposé l'utilisation de SPH.

4.4. Rendu de fluides

En fait, Müller et al. ont proposé même une technique pour rendre les particules SPH. Dans leur approche, en utilisant la technique du champ de couleur, les particules qui appartiennent à la surface sont identifiées avec leurs normales. Ensuite, les particules de surface sont soit rendues directement avec une technique de splatting (Zwicker, et al. 2001), soit après la phase de triangulation en utilisant l'algorithme de Marching Cubes (Lorensen et Cline 1987). Les images résultantes de cette approche n'étaient pas réalistes suffisamment (Müller, Charypar et Gross 2003).

Les points, comme primitive de rendu, ont attiré de nombreux efforts de recherche et ont montré leur efficacité. Néanmoins, les particules ne peuvent pas être considérées comme des points car elles n'ont pas de normales. En conséquence, les techniques classiques basées sur des points ne parviennent pas à rendre directement les nuages de particules. Pour résoudre ce problème, une extension de la technique de splatting a été proposée pour rendre l'isosurface des particules (Co, Hamann et Joy 2003). Cependant, même cette approche semble peu pratique car on ne sait pas comment le champ de densité est calculé.

La plupart des données de simulation basées sur des particules sont rendues en utilisant l'information de densité qui est organisée en une grille 3D. Chaque cellule de grille stocke une pondération de la densité de particules voisines. L'isosurface peut être extraite de la grille de densité par polygonalisation en utilisant l'algorithme de Marching Cubes (Lorensen et Cline 1987). Cet algorithme est considéré comme une solution très gourmande en bande passante mémoire. L'isosurface peut également être restituée à partir de la grille de densité (généralement après la phase de lissage / raffinement) en utilisant la méthode level set (Enright, et al. 2002). Cette dernière est coûteuse en termes de calcul.

La grille de densité peut également être utilisée pour rendre l'isosurface par des approches de type lancer de rayon sur GPU (Goswami, et al. 2010, Hadwiger, et al. 2005, Imai, Kanamori et Mitani 2016). Pour le rendu de volume, le modèle basé sur la grille peut être rendu par ray casting volumique. Dans un tel procédé, le rayon traversant la grille est échantillonné et la couleur des échantillons est mélangée de l'avant à l'arrière (Drebin, Carpenter et Hanrahan 1988).

D'autres méthodes ne transforment pas les particules en une grille, et les manipulent comme des fonctions de base (noyaux isotropes / anisotropes). Plus spécifiquement, en utilisant le ray marching, où une fonction scalaire (distance) est calculée à différentes positions en additionnant la contribution des différents noyaux de particules proches. De cette façon, une surface implicite est trouvée. Le premier travail dans cette catégorie, qui utilise un noyau isotrope, a été présenté par Zhu et Bridson (Zhu et Bridson 2005). Un second travail attribue un

noyau anisotrope à chaque particule après l'analyse en composantes principales (PCA) du nuage de particules (Yu et Turk 2013), et ceci afin de gérer les caractéristiques aigues du fluide. Comme troisième exemple, dans (Laan, Green et Sainz 2009), les données de simulation basées sur les particules ont été utilisées pour rendre directement une surface lisse sur l'espace de l'écran en utilisant le flux de courbure.

Le rendu via des techniques physiquement réalistes a l'avantage de produire des images réalistes qui contiennent des effets d'illumination globale (c'est-à-dire que l'éclairage direct et indirect est pris en compte pendant le calcul de l'image). En ce qui concerne les milieux participants, l'équation de transfert radiatif (RTE) est résolue soit par une version volumique des techniques de path tracing (Kulla et Fajardo 2012, Lafortune et Willems 1996) soit par le photon mapping (Jarosz, et al. 2011, Jensen et Christensen 1998). Plusieurs articles récents ont abordé le problème de rendu des milieux participants avec des frontières réfractives (Holzschuch 2015, Wang, Gascuel et Nolzschuch 2016) par lequel nous sommes intéressés dans ce travail.

4.5. Conclusion

Dans le présent travail, nous sommes intéressés par la deuxième catégorie de méthodes de simulation de fluides basées particules (précisément les fluides SPH qui sont très utilisés de nos jours et qui sont aussi considérés comme une tendance), et nous évitons la transformation de particules en grille, ainsi que nous utilisons les particules comme entrée pour définir directement une surface implicite (en utilisant une approche basée sur le lancer de rayon, comme expliqué précédemment). L'objectif est de manipuler le fluide comme un milieu participant hétérogène avec des frontières réfractives. Car, en effet, la sortie de la phase de simulation physique n'est pas seulement la position des particules mais aussi leurs propriétés physiques/photométriques (comme la densité qui comprend l'absorption et la diffusion, la couleur, la fonction de phase, etc.).

Chapitre 5:

Contribution 2 : l'illumination globale des données de simulation de fluides à base de particules

5.1. Introduction

Rappelons que l'objectif de notre thèse était de développer une approche permettant le calcul de l'éclairage direct et indirect pour le cas particulier des scènes qui sont représentées par un ensemble discret de points issu d'une phase dite d'acquisition.

Les techniques existantes d'illumination globale ont été conçues à la base pour des scènes représentées par un ensemble de facettes. Ces techniques doivent être étendues pour traiter plutôt des nuages de points d'une manière efficace tout en produisant des images réalistes.

Notre étude sur les techniques de rendu à base de points nous a permis de définir une nouvelle orientation du travail de recherche, à savoir, réaliser des calculs de simulation d'éclairage pour la visualisation de fluides en mouvement.

La simulation physique de la dynamique des fluides utilisant le modèle par particules SPH, a été initiée par Müller et al. en 2003. Actuellement, les outils de simulation basés SPH permettent de donner la position des particules en des différents intervalles de temps, mais aussi d'autre propriétés physiques, telles que: le volume, la vélocité, la masse, la densité, etc.

Afin de visualiser des fluides à base de particules, les recherches bibliographiques ont montré que la plupart des méthodes de rendu utilisées dans le contexte de la visualisation de fluides reposent sur une reconstruction d'un maillage (dans l'espace objet) ou directement sur l'espace écran. Nous pensons que l'utilisation directe des points peut être exploitée par le lancer de rayon pour mener vers des calculs de simulation d'éclairage non biaisés, par path tracing. En admettant que le lancer de rayon soit toujours considéré comme une technique lente, l'utilisation des structures de données accélératrices ainsi que l'exploitation des cartes graphiques modernes offrent la possibilité de produire des résultats de manière efficace.

Nous avons étudié en particulier le contexte des milieux participants, souvent représentés par un ensemble de points organisés ou non, ayant un ensemble d'attributs dont les valeurs sont déterminées par des simulateurs physiques (comme les simulateurs de fluides à titre d'exemple).

Notre objectif est d'exploiter directement les données issues de la phase de simulation physique des fluides (à base de particules SPH), pour en développer une technique de rendu pour les milieux participants, directement adaptée aux simulateurs.

Afin de mettre en œuvre un système informatique de simulation d'éclairage, nous avons choisi de coder notre solution dans l'un des moteurs de rendu physique existants. Nous avons choisi le logiciel MitsubaRenderer, très utilisé dans le monde académique. Outre l'avantage de réduire le temps de développement, cela permet surtout de pouvoir comparer de manière

indiscutable la crédibilité des comparaisons entre notre solution et les méthodes existantes, non seulement en terme de qualité visuelle, mais aussi en terme de temps de calcul. De plus, nous sommes intéressés pour diffuser notre méthode sous la forme d'un module (plug-in) dédié à ce moteur de rendu.

MitsubaRenderer propose un code libre de droits et orienté objet. Il comporte une multitude de méthodes de calcul et de simulation d'éclairage (VPL , photon mapping, path tracing, etc.), des structures d'accélération (KD-tree basé sur l'heuristique SAH), les milieux participants (homogènes ou hétérogènes) ainsi que le code est parallélisé sur CPU via des processus légers (threads).

A cet effet, nous proposons une solution qui adapte le path tracing volumique à l'une des méthodes de simulation à base de particules bien connues, à savoir l'hydrodynamique des particules lissées (SPH). Ce choix s'explique par le fait que le path tracing produit des résultats réalistes (qui incluent des effets d'illumination globale) en raison de sa nature physique. Dans notre proposition, nous considérons le fluide comme un milieu participant avec des frontières réfractives, et ceci dans le but de traiter à la fois l'aspect externe (surface) et interne (volume).

5.2. Bases théoriques du transport de lumière dans les milieux participants

L'équation de rendu (RE) a été généralisée à l'équation de transfert radiatif (RTE) pour simuler le transport de lumière sur les milieux participants (Jensen et Christensen 1998, Lafortune et Willems 1996) par le path tracing et photon mapping. La RTE définit la luminance provenant d'un point **x** dans la direction $\vec{\omega}$ comme suit:

$$\frac{\delta L(x,\overrightarrow{\omega})}{\delta x} = \alpha(x)L_e(x,\vec{\omega}) + \sigma(x)\int_{\Omega} f\left(x,\overrightarrow{\omega'},\vec{\omega}\right)L(x,\overrightarrow{\omega'})d\omega' - \alpha(x)L(x,\vec{\omega}) - \sigma(x)L(x,\vec{\omega}) \quad (1)$$

- $\boldsymbol{L_e}$est la luminance émise.
- $\boldsymbol{\alpha}$ *et* $\boldsymbol{\sigma}$ sont respectivement les coefficients d'absorption et de diffusion. Lorsque ces derniers sont constants, le milieu est dit homogène, sinon il est hétérogène.
- $\boldsymbol{f}$ est la fonction de phase du milieu qui est définie sur $\boldsymbol{\Omega = 4\pi}$. Elle est similaire à la fonction de distribution de réflectance bidirectionnelle (BRDF) sur les surfaces. $\boldsymbol{f}$ est souvent symétrique et ne dépend que de l'angle de phase $\boldsymbol{\theta}$ (entre les directions

d'incidence et de réflexion). Le milieu est isotrope ou anisotrope lorsque la fonction de phase est indépendante ou dépendante de l'angle de phase respectivement.

Nous notons que l'équation (1) possède quatre termes de luminance: l'émission, la diffusion, la dispersion et l'absorption. Habituellement, les termes d'absorption et de diffusion sont combinés en un terme appelé extinction:

$$k(x) = \alpha(x) + \sigma(x)$$

$$k(x)L(x, \vec{\omega})$$

où $\boldsymbol{k}(\boldsymbol{x})$ est le coefficient d'extinction (aussi appelé densité). L'intégration de la RTE le long d'un segment $[\boldsymbol{x_0}, \boldsymbol{x}]$ du rayon traversant un milieu donne:

$$\begin{aligned} L(x, \vec{\omega}) = & \int_{x_0}^{x} \tau(x', x)\alpha(x')L_e(x', \vec{\omega})\, dx' \\ & + \int_{x_0}^{x} \tau(x', x)\sigma(x') \int_{\Omega} f\left(x', \overrightarrow{\omega'}, \vec{\omega}\right) L(x', \overrightarrow{\omega'}) d\omega' dx' \\ & + \tau(x_0, x)L(x_0, \vec{\omega}) \end{aligned} \quad (2)$$

Le terme principal dans l'équation (2) est une double intégrale sur deux domaines:

- Segment du rayon à l'intérieur du médium.
- Sphère, où une fonction de phase est définie à chaque point du segment.

$\boldsymbol{\tau}(\boldsymbol{x'}, \boldsymbol{x})$ est la transmittance le long du segment $[\boldsymbol{x'}, \boldsymbol{x}]$ qui diminue avec la densité intégrée comme suit:

$$\tau(x', x) = e^{-\int_{x'}^{x} k(\xi) d\xi} \quad (3)$$

Comme représenté sur **Figure 49**, la RTE peut être résolue avec une approche stochastique non biaisée comme le path tracing de Monte Carlo (path tracing volumique). En plus du calcul des intersections surfaces/rayons et de la génération des rayons secondaires (déjà pris en compte dans le path tracing sur des scènes surfaciques), un ray marching à l'intérieur du milieu est appliqué avec la génération des rayons de diffusion.

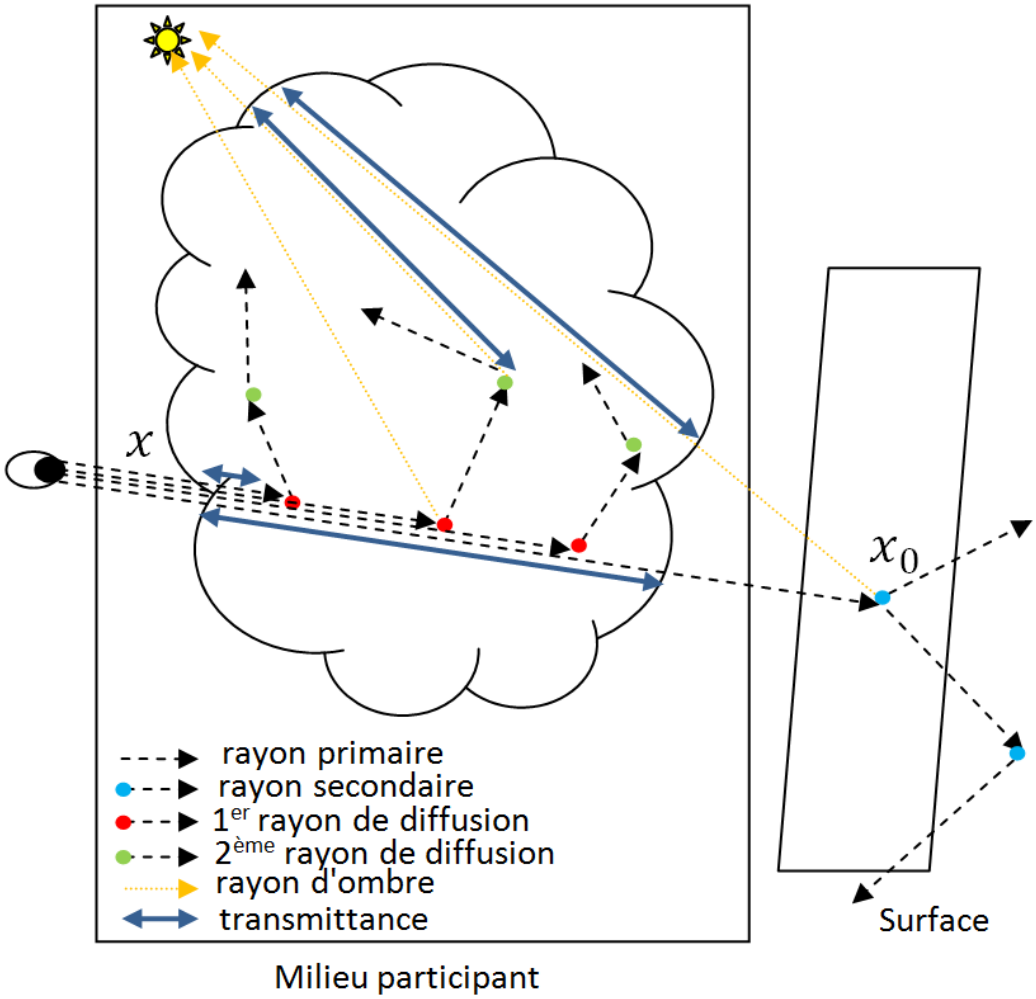
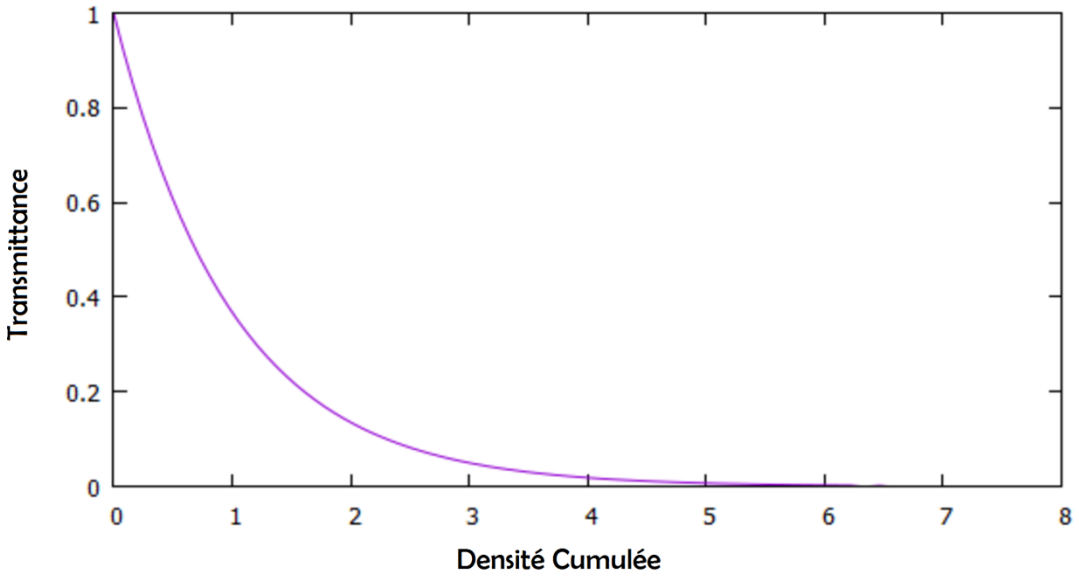
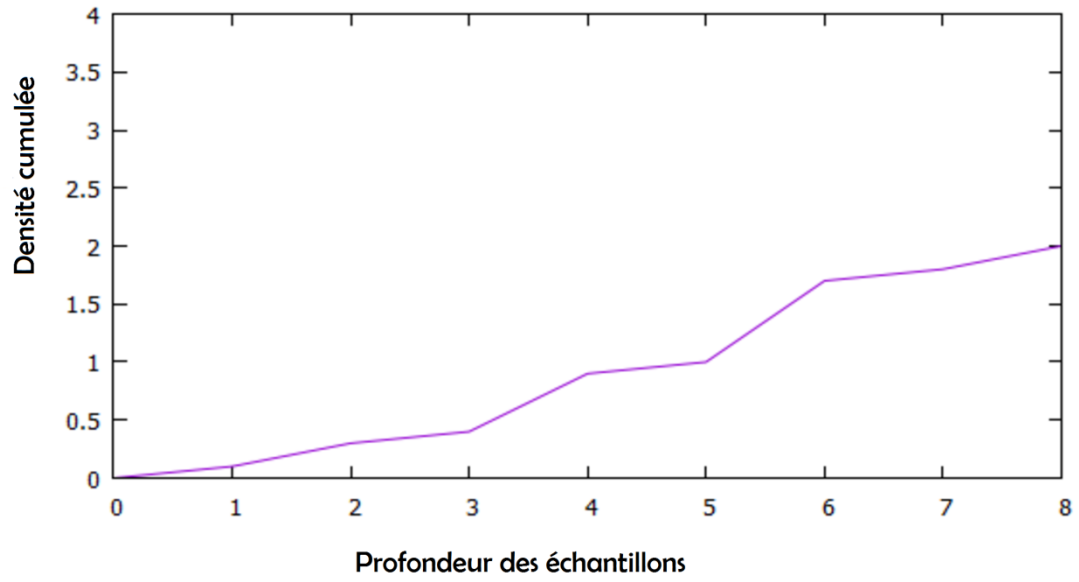


Figure 49. Path tracing d'un milieu participant.

Ce ray marching est guidé par un échantillonnage aléatoire le long du segment de rayon en fonction de la transmittance (densité intégrée) habituellement avec un échantillonnage d'importance. Ce dernier est meilleur que l'échantillonnage uniforme puisque la fonction de transmittance est une fonction exponentielle de la densité intégrée (**Figure 50a**). La densité intégrée est une fonction croissante en fonction de la profondeur des points (échantillons) sur le rayon (**Figure 50b**). Par conséquent, pour toute solution de la RTE utilisant le path tracing, le calcul de la transmittance (densité intégrée), c'est-à-dire l'intégration de l'équation (3), est la première question à laquelle il faut répondre (Jensen 2001).



(a)



(b)

Figure 50. Exemple de: (a) fonction de transmittance. (b) densité intégrée.

En raison de leur complexité, leur matériau translucide et leur forme floue, de nombreux objets dans la nature (comme le brouillard, la fumée, la peau, etc.) ne peuvent pas être représentés (et donc rendus) avec des modèles polygonaux classiques.

La sortie du simulateur physique est un ensemble de points épars ayant des propriétés physiques et photométriques telles que la position, la densité, la couleur, la masse, le volume, la vitesse et la fonction de phase. Les propriétés les

plus importantes pour le rendu sont la position et la densité (qui englobe l'absorption et la diffusion).

L'une des techniques les plus courantes (Fedkiw, Stam et Jensen 2001, Losasso, Gibou et Fedkiw 2004) utilisée pour calculer l'intégrale de densité le long du rayon traversant le milieu participant, consiste premièrement à transformer la densité des particules en une grille tridimensionnelle. Deuxièmement, le rayon est échantillonné en un ensemble de points. Troisièmement, la densité à chaque point est prise en utilisant une interpolation tri-linéaire de la cellule environnante. Enfin, l'intégration est effectuée en utilisant des approches numériques comme l'intégration de Simpson.

5.3. Solution proposée: illumination globale pour les fluides SPH

Dans cette section, nous décrivons notre solution pour le rendu des fluides SPH. En termes plus spécifiques, notre approche adapte le path tracing volumique pour les modèles à base de particules et son fonctionnement est divisé en deux étapes: (1) calcul d'intégrale de densité par une approche basée particules, et (2) définition de la surface du fluide.

5.3.1. Calcul d'intégrale de densité par une approche basée particules

Comme notre approche considère des milieux participants hétérogènes (c'est-à-dire, des particules ayant des densités différentes), un algorithme efficace (**Algorithme 1**) a été proposé pour calculer l'intégrale de densité sur le rayon passant par des particules chevauchées (**Figure 51**).

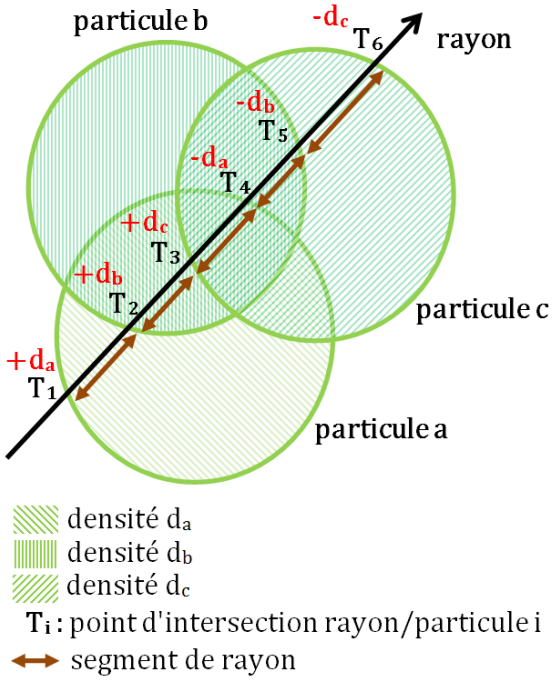


Figure 51. Rayon traversant un fluide basé particules.

Algorithm 1: Density integral calculation.

```
Data: Particle Set P, Ray ray, Desired Density DD
Result: Position T
/*** Data setting ***/
struct Intersection{
float tpos ;
float density ;
int particleID;
Intersection(float t, float d, int id)
{
tpos ⟵ t;
density ⟵ d;
particleID ⟵ id;
}
}
vector<Intersection> IT;
int i,counter; float t1,t2;
bool FOUND;
float TotalDensity, newDensity, difference;
float step, offset, TotalStep, D, scale;
/*** Initialization ***/
scale ⟵ 100.0;
TotalStep ⟵ 0.0;
TotalDensity ⟵ 0.0;
D ⟵ 0.0;
counter ⟵ 0;
/*** Intersections computation ***/
foreach element i of particle set P do
    if rayIntersectParticle(ray,P[i],t1, t2) then
        IT.push(new Intersection(t1,P[i].density,i) ;
        IT.push(new Intersection(t2,-1*P[i].density,i)
        ;
QuickSort(IT);
/*** Integration of piecewise density function till
 getting the desired density ***/
FOUND ⟵ false;
for j ← 0 to IT.size()-1 do
    newDensity ⟵ 0;
    step ⟵ IT[j + 1].tpos − IT[j].tpos;
    D ⟵ D + IT[j].density;
    if IT[j].density > 0 then
        counter ⟵ counter + 1;
    else
        counter ⟵ counter − 1;
    if counter ≠ 0 then
        /* averaging the density in overlap regions */
        newDensity ⟵ D/counter ;
    TotalDensity ⟵
     TotalDensity + newDensity ∗ step ∗ scale;
    if TotalDensity > DD then
        difference ⟵ TotalDensity − DD ;
        offset ⟵ difference/(newDensity ∗ scale) ;
        TotalStep ⟵ TotalStep + offset ;
        T ⟵ TotalStep ;
        FOUND ⟵ true;
        break ;
    TotalStep ⟵ TotalStep + step ;
/*** Result ***/
if FOUND then
    Print("Desired Density found at", T) ;
else
    Print("Desired Density not found" ) ;
```

Algorithme 1. Calcul d'intégrale de densité.

Une fois que la densité désirée **DD** est donnée aléatoirement en entrée de l'algorithme, cette dernière produit en sortie la position **T** sur le rayon traversant le milieu. En d'autres mots, l'algorithme fonctionne comme suit. Tout d'abord, les intersections sphères/rayons sont déterminées et stockées avec leurs densités respectives dans un tableau de structure. Ce tableau représente en quelque sorte une fonction de densité par morceaux. Deuxièmement, l'intégrale de la fonction de densité est calculée sur les segments de rayon jusqu'à ce que la densité désirée soit atteinte.

Notez qu'en raison du fait que les particules chevauchées appartiennent au même fluide SPH, la densité des segments dans les régions de chevauchement est moyennée. Ce choix peut être changé si plusieurs fluides SPH sont considérés.

Afin d'accélérer la première phase de l'algorithme (calcul des intersections rayon-sphère), les particules ont été organisées en deux subdivisions spatiales différentes; un arbre SAH KD-tree (Wald et Havran 2006) et une grille uniforme. Selon les résultats expérimentaux obtenus, la grille

uniforme surpasse clairement l'arbre KD. En fait, cela est évident car les particules SPH sont généralement uniformément réparties dans l'espace et le parcours d'une grille uniforme nécessite moins de temps par rapport à une hiérarchie KD-tree.

Contrairement aux approches basées sur les grilles, notre modèle de calcul de l'intégrale de densité est plus précis car il est basé sur les particules d'origine (pas celles transformées). Cette caractéristique intéressante donne plus de flexibilité au modèle proposé dans les étapes de rendu suivantes.

5.3.1.1. Support pour de multiples fluides SPH

Dans cette section, nous étendons notre modèle de calcul d'intégrale de densité afin de gérer plusieurs fluides SPH ayant des propriétés différentes (densité, couleur, fonction de phase, etc.).

Comme mentionné précédemment, nous avons choisi de faire la moyenne de la densité dans les régions de chevauchement. Ceci s'explique par le fait que les particules appartiennent au même fluide. Ce qui n'est pas le cas si un milieu avec plusieurs fluides SPH est considéré (voir **Figure 52**).

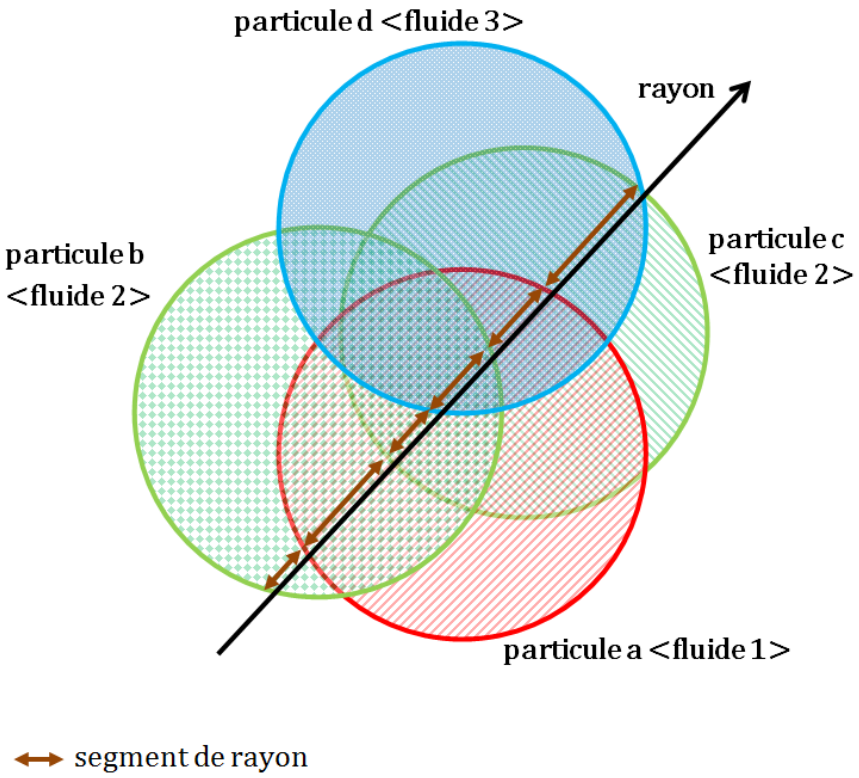


Figure 52. Multiples fluides SPH.

Le problème d'ambiguïté du calcul d'intégrale de densité dans les régions de chevauchement de particules peut être résolu par les trois étapes suivantes:

1) Sur chaque segment, les particules en question doivent être classées en fonction du fluide auquel elles appartiennent.
2) La densité de chaque classe est moyennée.
3) Une classe peut être sélectionnée selon l'une des deux stratégies suivantes:
   a) la plus basse densité.
   b) la plus haute densité.

5.3.2. Définition de la surface du fluide pour les milieux participants

Les fluides sont des objets translucides dont la surface doit être définie avant toute opération de rendu. En fait, de nombreux événements se produisent sur la surface du fluide (réflexion et réfraction), en plus des événements qui se produisent à l'intérieur du milieu comme l'absorption, l'émission, la dispersion et la diffusion.

Les approches basées particules utilisent une technique de champ de couleur pour identifier les particules de surface avec leurs normales. Ensuite, la visualisation est réalisée par splatting (Müller, Charypar et Gross 2003).

Les approches basées grilles utilisent la grille de densité pour définir une fonction de distance afin de reconstruire une surface implicite par ray marching. L'algorithme de Marching Cubes (Lorensen et Cline 1987) est aussi utilisé avec des données de simulation basée grille. Il donne une soupe de polygone, mais avec un prix coûteux en mémoire, surtout quand un maillage très fin est ciblé. Le modèle polygonal pourrait être rendu par une approche projective (rastérisation) ou peut-être rendu par lancer de rayon sur CPU ou GPU.

Dans notre travail, nous proposons une approche basée sur le lancer de rayon pour calculer la densité et rendre la surface avec une technique non biaisée (path tracing).

5.3.2.1. Surfaces de convolution

Les surfaces de convolution sont une approche de modélisation pour un type spécifique de surfaces (appelées objets mous) qui ne peuvent pas être facilement représentées avec un maillage.

A partir d'un ensemble de points, nous définissons un noyau $\boldsymbol{f(x)}$ avec un rayon **R1** assigné à chaque point. Ces noyaux peuvent se chevaucher. Pour trouver l'interface de la surface, on calcule une fonction de distance **D(x)** qui somme les différentes contributions des points voisins. Lorsque cette fonction atteint un seuil **T**, la surface est considérée comme atteinte (**Figure 53**).

$$D(x) - T = \sum f(x) - T = 0 \qquad (4)$$

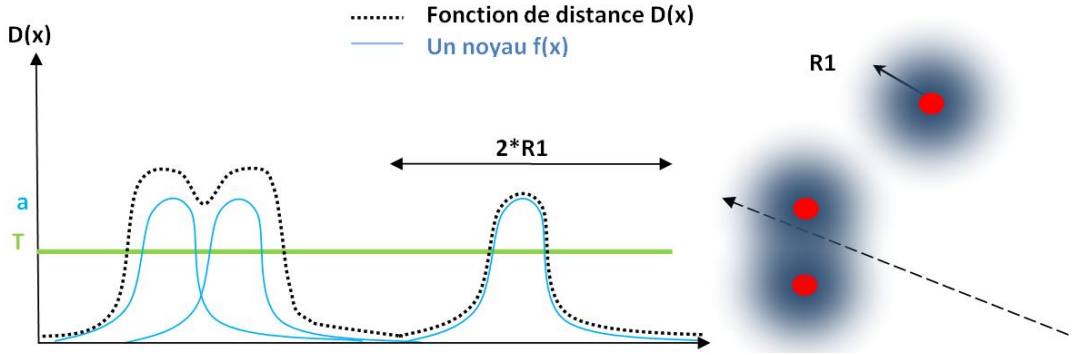


Figure 53. Fonction de distance d'un rayon traversant des particules.

L'équation (4) peut être résolue par Marching Cubes. Cependant, nous nous intéressons plutôt à une approche du genre ray marching dans laquelle la racine peut être trouvée par une approche itérative numérique telle que la sécante ou la technique de bissection.

Nous notons que plus le nombre d'itérations est grand, plus le résultat est exact, mais avec un temps de calcul supplémentaire.

Le noyau des surfaces de convolution doit être une fonction symétrique décroissante continue. Comme premier exemple, considérons le noyau de Jim Blinn défini par $f(r) = ae^{-br^2}$.

L'inconvénient de ce modèle de Jim Blinn réside dans le fait que le noyau n'est pas borné (tous les points contribuent dans **D(x)**) et le terme exponentiel nécessite un coût de calcul supplémentaire.

Comme deuxième exemple, considérons le noyau de Zhu-Bridson (Zhu et Bridson 2005) qui évite la fonction exponentielle:

$$f(r) = \max(0, \left(1 - \left({}^{r}/_{b}\right)^2\right)^3)$$

L'un des exemples classiques de surfaces de convolution est le modèle de metaballs (Blinn 1982) connu par le noyau suivant:

$$f(r) = \begin{cases} a\left(1 - \dfrac{3r^2}{b^2}\right) & 0 \le r \le \dfrac{b}{3} \\ \dfrac{3a}{2}(1 - \dfrac{r}{b})^2 & b/3 \le r \le b \\ 0 & r \ge b \end{cases}$$

Pour la raison d'avoir les avantages d'un noyau borné et le faible coût de calcul, le modèle metaballs a été choisi, dans notre travail, afin de proposer une solution de rendu pour les fluides SPH.

Notez que la valeur du seuil **T** contrôle le volume de la surface résultante (**Figure 54**):

- plus **T** est petit, plus le résultat sera grossier et volumineux.
- plus **T** est grand, plus le résultat sera fin (mais avec perte d'éléments de surface).

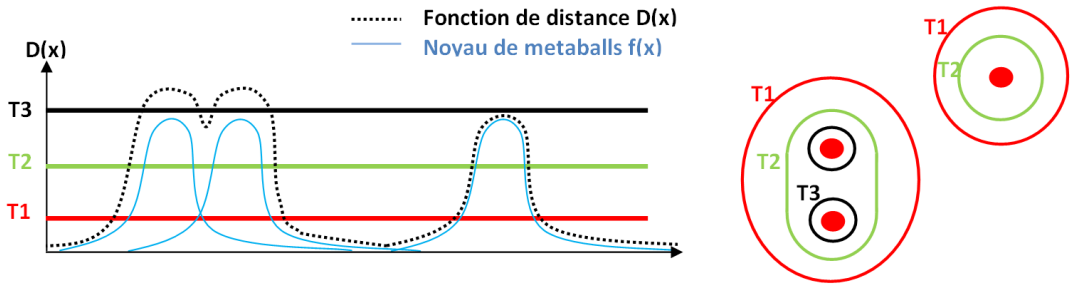


Figure 54. Surface résultante de la fonction de distance avec trois différent seuils.

5.3.2.2. Solution metaballs pour les particules SPH

Selon les simulations physiques conduites des fluides SPH, chaque particule a un rayon constant **R0**. En supposant que les particules sont largement espacées (pas de recouvrement inter-particulaire), un seuil **T0** peut être défini pour que les particules puissent être visualisées comme des sphères de rayon **R0** (voir Equation (5)):

$$D(x) - T0 = \sum f(x) - T0 = f(R0) - T0 = 0 \Rightarrow T0 = f(R0) \qquad (5)$$

La normale peut être trouvée par le gradient de la fonction de distance **D(x)** (en utilisant les dérivées partielles). Comme nous travaillons avec une approche de lancer de rayon, nous estimons la normale comme suit:

$$Normal(x) = \frac{\sum f\left(\left(\overrightarrow{\|x - P_i.center\|}\right) * \left(\overrightarrow{x - P_i.center} \Big/ \overrightarrow{\|x - P_i.center\|}\right)\right)}{\sum f\left(\overrightarrow{\|x - P_i.center\|}\right)}$$

En général, les particules se chevauchent, par conséquent, l'utilisation du seuil précédent donne une surface légèrement plus grossière et plus lisse que les sphères (et ceci, seulement dans les régions de chevauchement). Par conséquent, la surface finale du fluide apparaîtra trop bombée et ne ressemblera pas à un liquide (**Figure 55**).

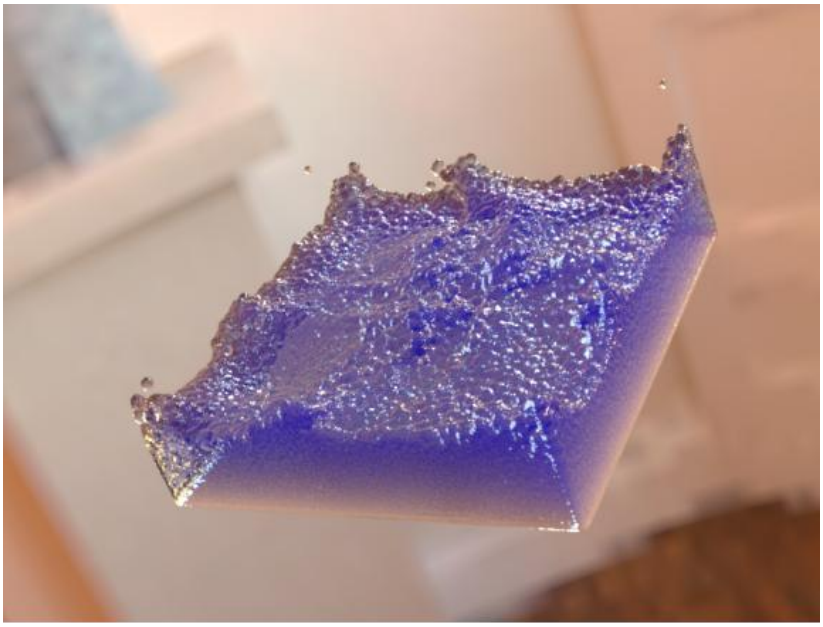


Figure 55. Surface résultante légèrement lisse sur les régions de chevauchement utilisant un rayon d'influence R1.

Afin de rendre la surface du fluide plus lisse et moins bombée, il faut tenir en compte plus de particules provenant des voisinages d'ordre plus élevé au cours du calcul de la normale. Cette idée est inspirée de l'estimation des courbures discrètes sur des maillages triangulaires (Gatzke et Grimm 2006) (**Figure 56**).

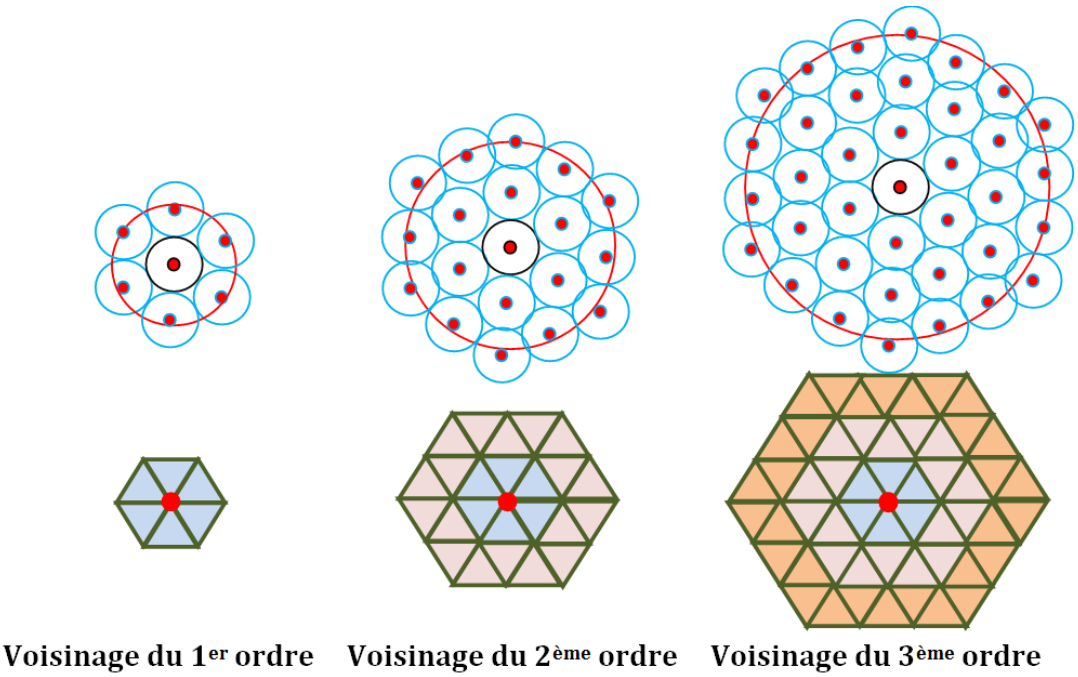


Figure 56. Notion du voisinage topologique (dans un maillage triangulé) et spatial (dans un nuage de particules).

Néanmoins, comme les particules sont discrètes et n'ont pas de topologie, la notion de voisinage ne peut pas être définie. Ce problème peut être résolu en considérant le voisinage spatial et non pas le topologique. Plus précisément, le rayon d'influence des particules doit être augmenté de **R1** à **2*n*R0**, où **n** est l'ordre du voisinage (**Figure 57**). En effet, afin d'éviter d'avoir une surface volumineuse, le seuil doit également être augmenté selon la formule empirique suivante: $\boldsymbol{T = (2 * n - 1) * a}$

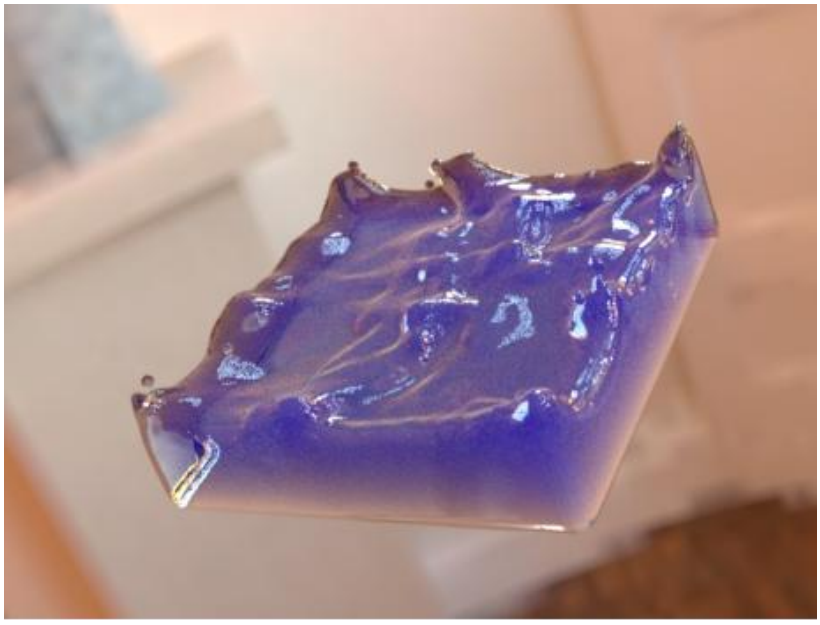


Figure 57. Surface résultante quand R1=2*3*R0 (voisinage du 3ème ordre) est plus lisse mais avec perte des éléments de surface.

De cette façon, l'isosurface devient plus lisse mais avec une perte d'éléments de surface (c'est-à-dire, la perte des particules qui n'ont pas de voisins, ou spécifiquement des particules dont l'amplitude du noyau est inférieure au seuil $\boldsymbol{T}$). Les particules perdues peuvent être récupérées en utilisant un deuxième seuil $\boldsymbol{T'}$ qui correspond au voisinage du premier ordre (**Figure 58**).

$$\boldsymbol{T' = (2 * 1 - 1) * a = a}$$

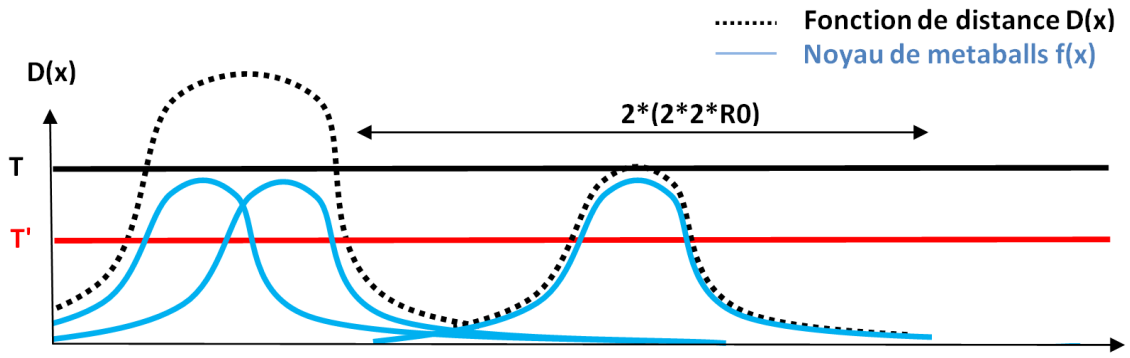


Figure 58. Fonction de distance des metaballs avec un rayon changé à R1=2*2*R0.

5.3.3. Framework du path tracer

Comme mentionné précédemment, les particules sont définies par deux rayons, le rayon réel donné par le simulateur **R0** , et le rayon d'influence **R1 = 2 * n * R0** .

Les particules sont organisées en deux grilles uniformes avec deux résolutions. La grille la plus grossière a une longueur de cellule de **2 * R1**, tandis que la plus fine a une longueur de cellule de **2 * R0**.

Le Framework comporte deux phases:

- **Première phase - identification de surface:** dans cette phase, un lancer de rayon est appliqué sur les particules de la grille la plus grossière. Premièrement, les points d'intersection sphères/rayons sont stockés dans un tableau unidimensionnel. Ensuite, la surface est déterminée en échantillonnant le tableau en utilisant la méthode itérative de la sécante. Une fois l'intersection rayon-surface (**x**) est trouvée, la normale est calculée comme mentionné ci-dessus. A partir de là, des rayons secondaires peuvent être générés (réflexion et réfraction).
- **Deuxième phase - calcul d'intégrale de densité:** dans le but de générer des rayons de diffusion à l'intérieur du milieu, le lancer de rayon est appliqué sur les particules de la grille la plus fine. Tout d'abord, les points d'intersection sont stockés dans un tableau unidimensionnel. Ensuite, **Algorithme 1** est appliqué pour calculer l'intégrale de densité qui correspond à une position **T** sur le rayon. A partir de ce point, de nouveaux rayons de diffusion sont générés dans des directions aléatoires en fonction de la fonction de phase.

Ces deux phases doivent être répétées à chaque fois que le rayon rebondit dans le milieu. Idéalement, le rayon réfracte deux fois; à la surface faisant face à l'avant et faisant face à l'arrière. Mais en général, lorsque le milieu est un liquide (de l'eau par exemple), son indice de réfraction est supérieur à celui de l'air. Il en résulte que le rayon à la rencontre de la face arrière pourrait ne pas quitter le milieu si l'angle d'incidence, noté ($\boldsymbol{\theta_{in}}$) est supérieur à un angle critique ($\boldsymbol{\theta_{critique}}$):

$$\theta_{in} > \theta_{critique}$$

Cette situation est appelée la Réflexion interne totale (TIR: Total Internal Reflection), et selon la loi de Snell, le rayon devrait plutôt se refléter dans le milieu avec le même angle: $\theta_{in} = \theta_{out}$ (voir **Figure 59**).

Dans un tel cas, le rayon subit plusieurs réfractions avant de quitter le milieu.

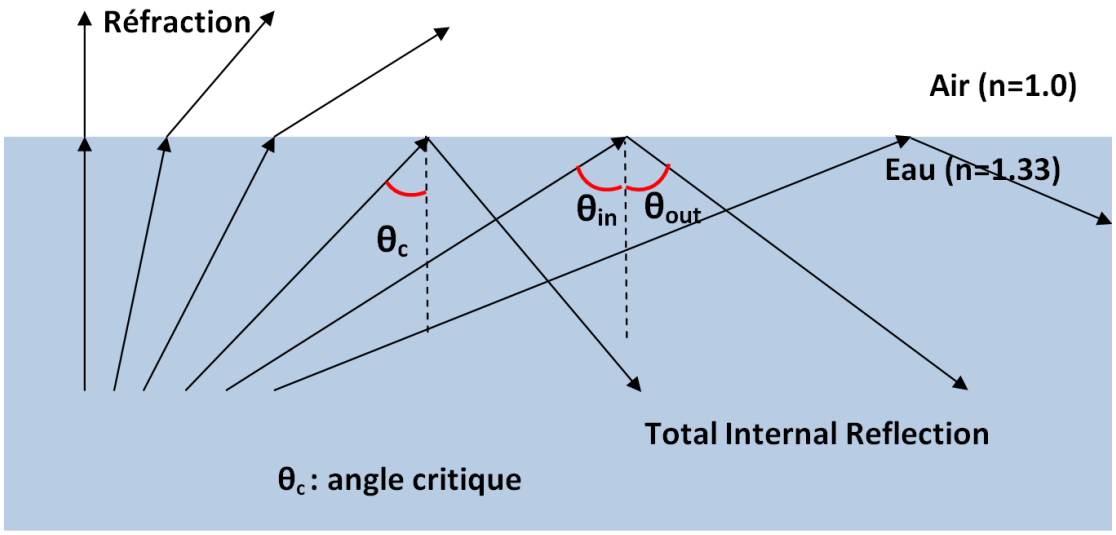


Figure 59. Réfraction de rayons depuis un milieu ayant un indice réfraction élevé vers un autre ayant un indice diminué.

5.3.4. Path tracing étendu pour l'eau de mer

En utilisant notre path tracer SPH, l'eau de mer peut être rendu en spécifiant comment les rayons doivent agir lorsqu'une surface formée par différents types de particules est atteinte (**Figure 60**).

Les particules peuvent être discrètes ou continues, et elles peuvent avoir un matériau BSDF (Bidirectional Scattering Distribution Function) ou un matériau BRDF. Les particules continues contribuent ensemble à définir une surface implicite, tandis que les particules discrètes sont rendues individuellement comme des sphères de rayon **R0**. En fait, des particules discrètes peuvent être rendues même si elles sont situées à l'intérieur du milieu (pas seulement sur la surface). Au cours de la seconde phase du calcul de l'intégrale de densité, lorsqu'une particule discrète est rencontrée, la phase courante est interrompue et des rayons secondaires sont générés.

Dans ce travail, la particule SPH générale est étendue à quatre types de particules, à savoir: l'eau, la mousse, la bulle et le sable. Contrairement aux méthodes précédentes (Takahashi, et al. 2003, Thürey, et al. 2007, Laan, Green et Sainz 2009) les bulles et la mousse ne sont pas rendues comme des effets mais sont physiquement rendues en définissant comment elles doivent interagir avec la lumière.

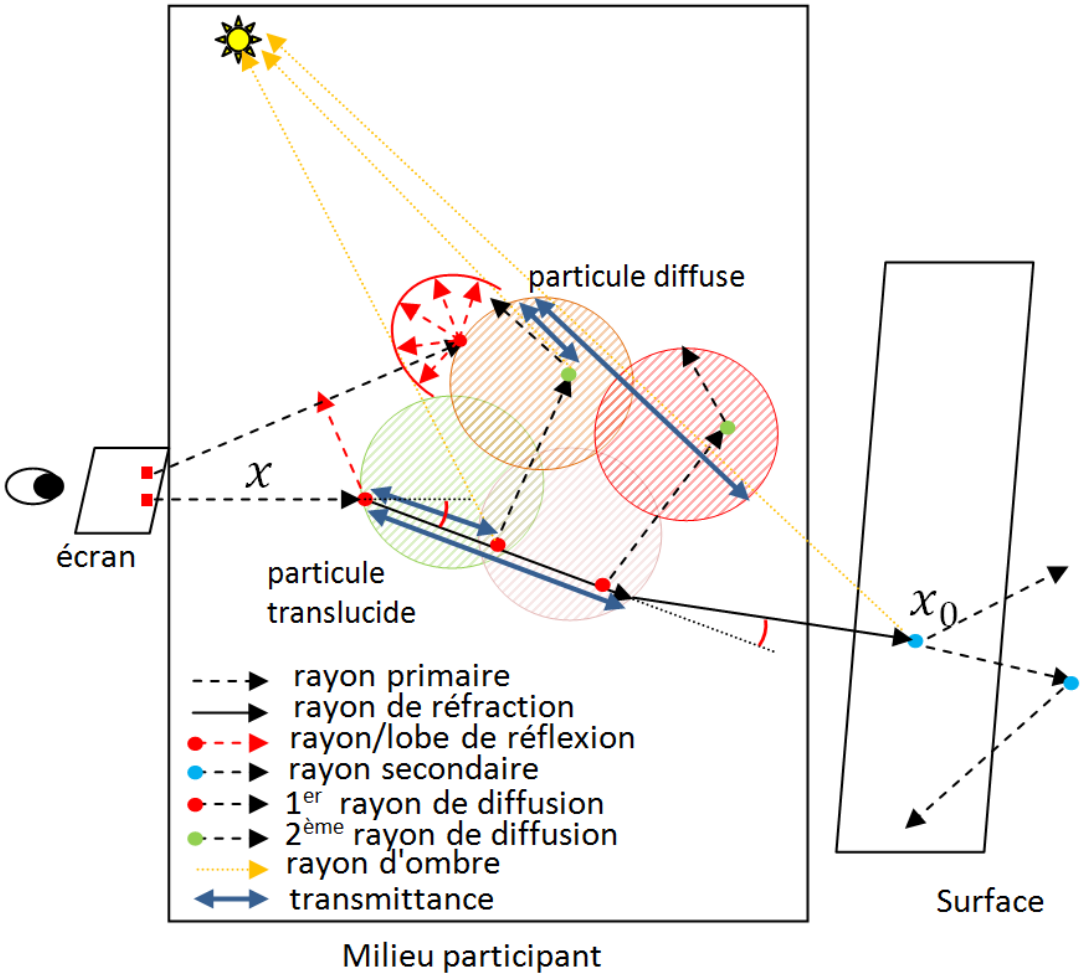


Figure 60. Path tracing d'un milieu participant ayant des frontières réfractives (fluide SPH).

5.3.4.1. Eau SPH

La particule SPH est étendue pour la gestion des particules de type eau en ajoutant les propriétés suivantes:

- Type de surface: continu.
- Matériel: BSDF; La fonction delta de Dirac avec IOR de Fresnel est égale à 1.33 / 1.0.
- Albedo: n'importe quelle couleur.
- Densité: inchangée (ou 0 pour l'eau transparente).

5.3.4.2. Mousse SPH

La particule SPH est étendue pour la manipulation des particules de type mousse en ajoutant les propriétés suivantes:

- Type de surface: aucun.
- Matériel: aucun.

- Albedo: blanc.
- Densité: inchangé.

La mousse est rendue comme volume car elle n'a pas de frontières de réfraction et seul le terme de diffusion est calculé.

5.3.4.3. Bulle SPH

La particule SPH est étendue pour gérer les particules de type bulle en ajoutant les propriétés suivantes:

- Type de surface: discret.
- Matériel: BSDF; La fonction Dirac delta avec Fresnel IOR est égale à 1.0 / 1.33.
- Albedo: pas de couleur.
- Densité: 0.

5.3.4.4. Sable SPH

La particule SPH est étendue pour la gestion des particules de type sable en ajoutant les propriétés suivantes:

- Type de surface: discret.
- Matériel: BRDF; diffusion Lambertienne.
- Albedo: marron.
- Densité: infini.

5.4. Résultats expérimentaux

L'expérimentation est faite sur des données de simulation d'un barrage d'eau rompu (donné par les auteurs de (Brousset, et al. 2016)). Notre solution (path tracing volumique à base de particules) est implémentée sur le MitsubaRenderer (Jakob 2010) et chaque image a une résolution de 768x574 pixels et calculée avec 128 échantillons et 8 rebonds sur une machine **Intel** (**CPU**: **i7** de 2.60 **GHz**, **RAM**: 8 **GB**).

Les données (barrage d'eau rompu) à un intervalle de temps spécifique (qui contient exactement 110375 particules) sont rendues par le path tracing avec notre modèle de metaballs en faisant varier l'ordre de voisinage de un à trois **Figure 62a**, **Figure 62b**, et **Figure 62c**. On notera que la densité de particules a été fixée à zéro afin de ne rendre que l'isosurface (c'est-à-dire, ne considérer que les événements de réflexion / réfraction et omettre le phénomène de diffusion). Il est clair que la surface devient plus lisse lorsque l'ordre de voisinage augmente. D'autre part, Houdini (qui est un logiciel de SideEffects) a été utilisé pour reconstruire une surface maillée à partir du modèle de particules. Ensuite, la surface obtenue a été rendue par le lancer de rayon via le

MitsubaRenderer avec le même nombre d'échantillons et de rebonds (voir **Figure 62d**). L'image résultante est considérée comme l'image de référence avec laquelle nous comparons les trois images précédentes en utilisant la métrique de l'erreur quadratique moyenne (MSE), comme indiqué sur **Figure 65a**. Nous observons que l'erreur diminue quand l'ordre de voisinage augmente. Au-delà du voisinage d'ordre 3, les images perdent leur qualité (par rapport à l'image de référence) et le minimum MSE pouvant être obtenu est de 0.0048 (ou **0.48%**) ce qui correspond au voisinage d'ordre 3.

Comme on le voit sur les figures **Figure 63** et **Figure 64**, nous avons rendu le modèle de particules à partir de deux points de vue différents (en utilisant notre modèle de metaballs avec le voisinage d'ordre 3). Premièrement, nous calculons seulement la réfraction à la surface frontale. Ensuite, à la fois les réfractions des surfaces d'entrée et de sortie du milieu. Enfin, nous prenons en compte toutes les réfractions possibles (limitées au nombre maximal de rebonds, soit 8). **Figure 65b** montre le MSE mesuré entre ces trois images et la référence. Contrairement aux méthodes précédentes (Kanamori, Szego et Nishita 2008, Müller, Charypar et Gross 2003, Laan, Green et Sainz 2009, Wyman 2005, Xiao, Zhang et Yang 2017) qui simplifient les choses en calculant au plus deux réfractions, nous prenons en compte toutes les réfractions possibles afin d'obtenir le résultat le plus proche de la référence.

Comme illustré sur **Figure 66a**, notre approche basée sur les particules rend les particules tout en considérant et en calculant tous les événements (réflexion, réfractions multiples, diffusion multiple). Nous notons que nous avons créé une scène de référence en suivant le schéma montré dans **Figure 61**. Plus précisément, d'abord, nous transformons la densité de particules en une grille eulérienne, puis nous la rendons (avec la surface triangulée déjà reconstruite par Houdini) par le path tracing volumique sur le MitsubaRenderer (voir **Figure 66b**). Notez que les deux résultats sont très similaires en qualité et que le MSE mesuré est de **0.09%**. Néanmoins, notre approche nécessite plus de temps comme le montre **Tableau 3**. Cela est dû au processus de définition implicite de surface avec estimation de la normale sur la base du voisinage du 3ème ordre ainsi que le calcul d'intégrale de densité sur l'ensemble des particules.

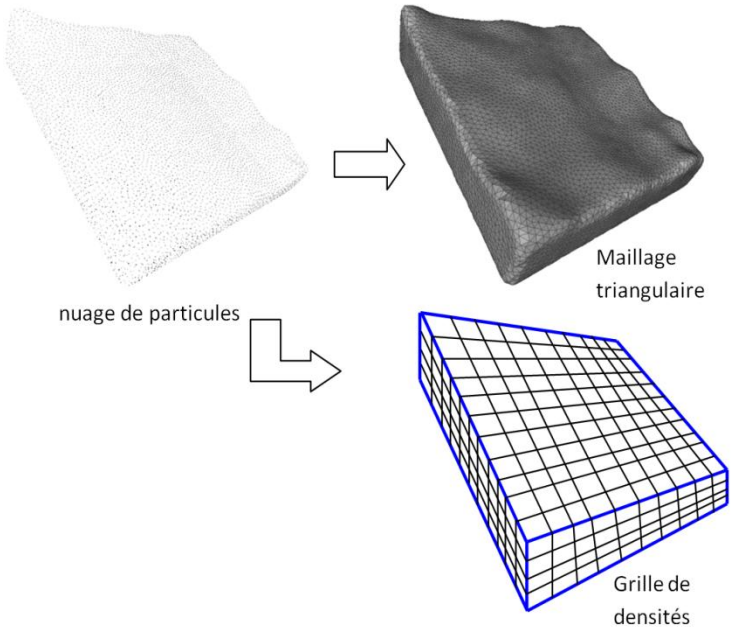


Figure 61. Préparation de la scène de référence.

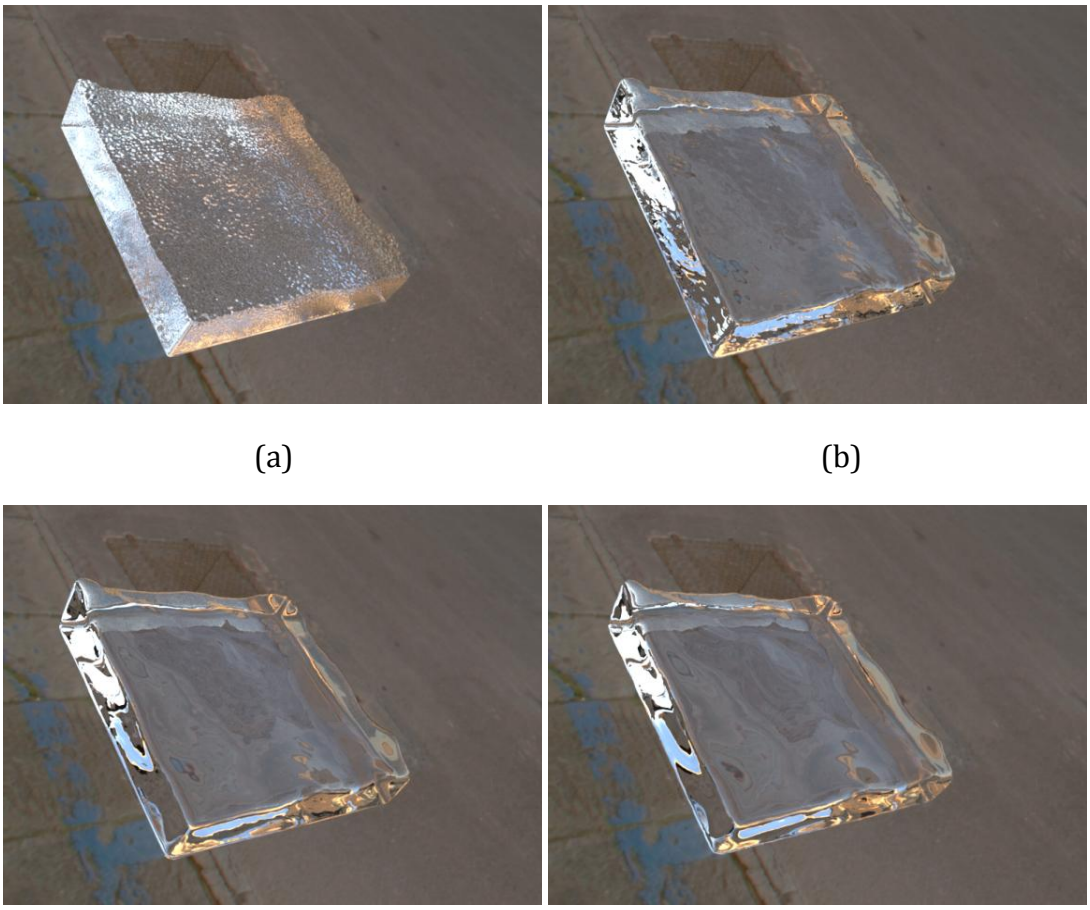
Les quatre types de particules (eau, mousse, bulles et sable) sont présents sur **Figure 67**. Nous avons également testé notre path tracer SPH en rendant les données de simulation d'un autre solveur (Fluides v.3 (Rama s.d.)) qui contiennent 65K particules comme montré sur **Figure 68**. En fait, **Figure 68a** montre des particules transparentes de type eau qui ont un matériau BSDF, alors que **Figure 68b** démontre l'existence de deux fluides de couleurs différentes (rouge et bleu). En ce qui concerne **Figure 68c,** elle affiche des particules de type bulle et de type sable parmi de nombreuses particules transparentes de type eau. Enfin, un ensemble de particules de type mousse est localisé sur le dessus du fluide dans **Figure 68d**.

Les résultats expérimentaux, obtenus en rendant plusieurs données de simulation fournies par différents solveurs, démontrent clairement l'efficacité de notre approche en termes de (1) qualité de surface implicite (incluant l'estimation normale) et (2) effets de transport de lumière (plus spécifiquement multiples réfractions et diffusion).

Les images produites par notre approche, qui considère l'effet physique des réfractions multiples, sont plus réalistes que les approches précédentes qui prennent en compte seulement une ou deux réfractions (Kanamori, Szego et Nishita 2008, Müller, Charypar et Gross 2003, Laan, Green et Sainz 2009, Wyman 2005, Xiao, Zhang et Yang 2017). Même en considérant quatre réfractions, comme présenté dans (Imai, Kanamori et Mitani 2016), les images résultantes ne peuvent pas être proches de la référence (Imai, Kanamori et Mitani 2016).

Un autre avantage de notre approche entièrement basée sur les particules réside dans le fait que les bulles, le sable et la mousse sont directement visualisés en rendant les particules (après modification de certaines propriétés) sans faire recours à des techniques supplémentaires (texturation) comme des approches précédentes (Takahashi, et al. 2003, Laan, Green et Sainz 2009).

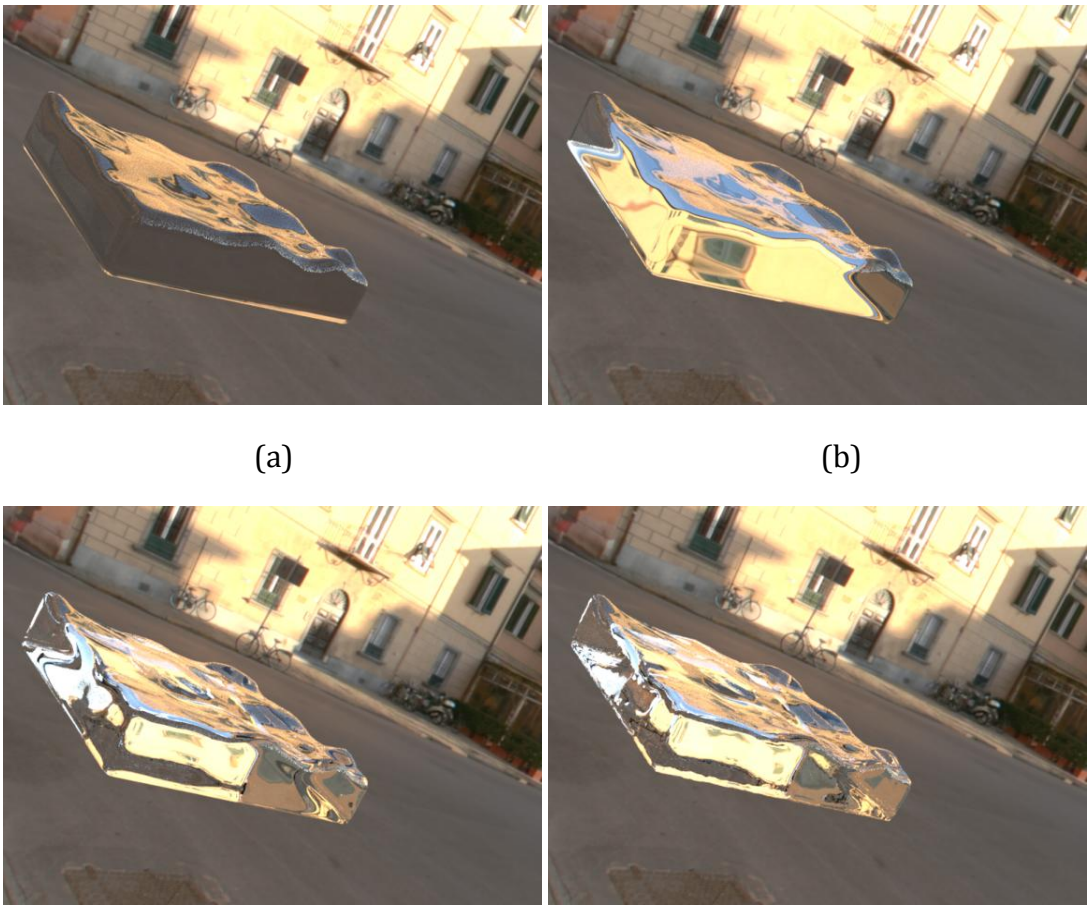
Notre proposition a l'avantage de sauter les étapes intermédiaires de l'extraction de surface et de la transformation des particules, et par conséquent, les données brutes produites par les solveurs de fluide sont directement rendues. Cependant, comme toute autre technique de rendu physique, la solution proposée nécessite du temps parce que le calcul de la surface implicite (qui inclut l'estimation de normale) et le traitement des effets de transport de lumière (en particulier les réfractions multiples) nécessitent un temps de calcul supplémentaire comme représenté dans **Tableau 1** et **Tableau 2**.



(a) (b)

(c) (d)

Figure 62. Barrage d'eau rendu avec estimation de normal:(a) 1er ordre. (b) 2ème ordre. (c) du 3ème ordre. (d) image de référence.

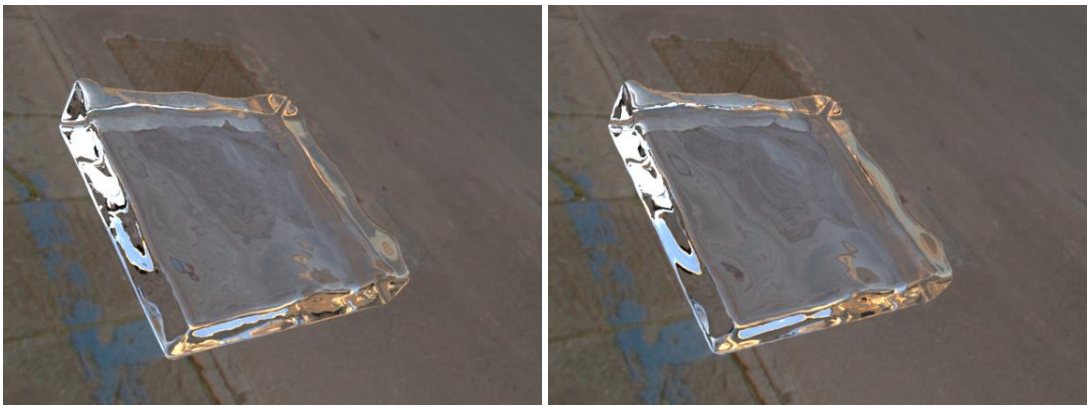


(a) (b)

(c) (d)

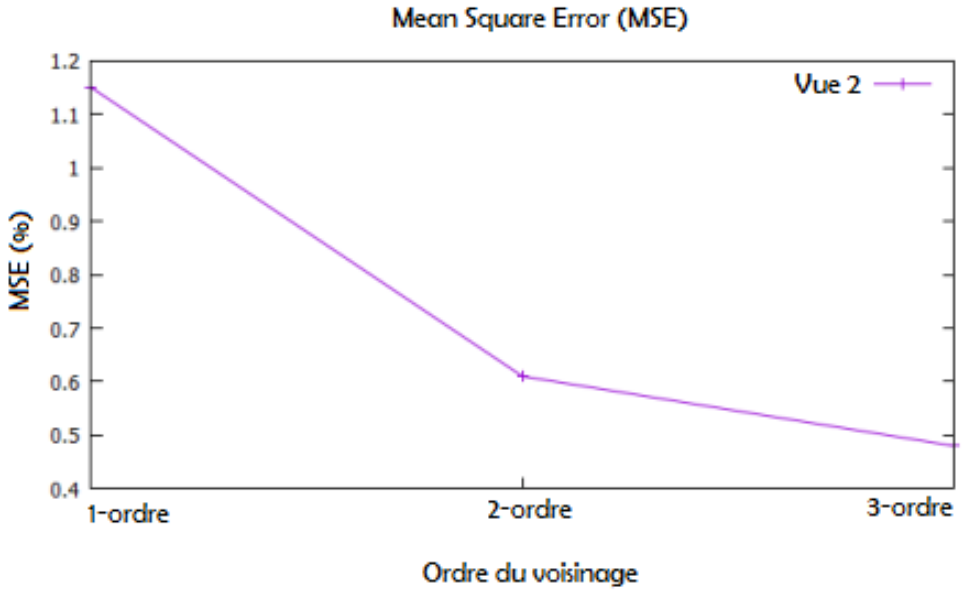
Figure 63. Barrage d'eau rendu (vue 1): (a) avec une réfraction. (b) avec deux réfractions. (c) avec multiples réfractions. (d) par lancer de rayon sur le maillage (image de référence).

(a) (b)

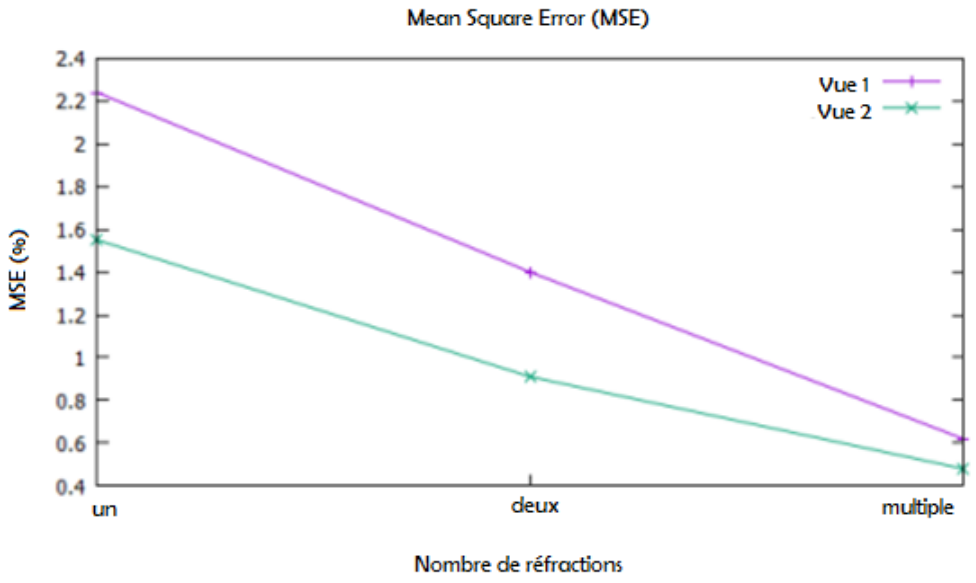


(c) (d)

Figure 64. Barrage d'eau rendu (vue 2): (a) avec une réfraction. (b) avec deux réfraction. (c) avec multiples réfractions. (d) par lancer de rayon sur le maillage (image de référence).



(a)

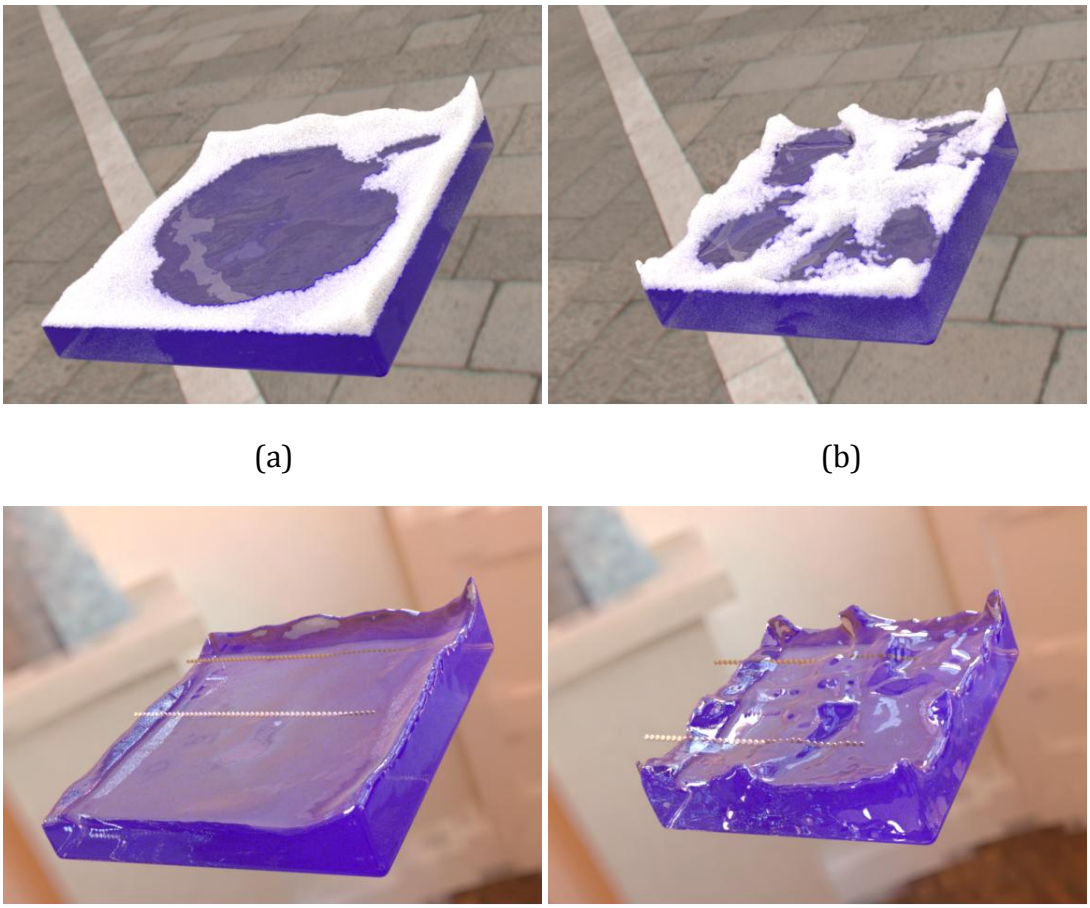


(b)

Figure 65. MSE entre: (a) les images calculées avec plusieurs ordres de voisinage, et l'image de référence. (b) les images calculées avec plusieurs réfractions, et l'image de référence.

(a) (b)

Figure 66. Rendu du fluide avec le path tracing volumique : (a) notre approche. (b) utilisant le maillage triangulé (image de référence).



(a) (b)

(c) (d)

Figure 67. (a), (b) Particules de mousse. (c), (d) particules de bulles et sable.

(a) (b)

(c) (d)

Figure 68. (a)Particules continues avec un matériau BSDF. (b) deux fluides: rouge et bleu. (c) bulles et sable. (d) mousse.

Scène	Isosurface (voisinage d'ordre 1)	Isosurface (voisinage d'ordre 2)	Isosurface (voisinage d'ordre 3)	Surface maillée
Barrage d'eau (vue 2)	2m.25s.	5m.02s.	7m.45s.	1.70s.

Tableau 1. Temps de rendu avec estimation de la normal à partir de différents voisinages.

Scène	Isosurface (1 réfrac.)	Isosurface (2 réfrac.)	Isosurface (multi-réfrac.)	Surface maillée
Barrage d'eau (vue 1)	4m.52s.	6m.30s.	8m.22s.	1.81s.
Barrage d'eau (vue 2)	4m.08s.	6m.01s.	7m.45s.	1.70s.

Tableau 2. Temps de rendu avec différents nombres de réfractions.

Scène	Notre approche	Surface maillée + grille de densité
Barrage d'eau (vue 2)	7m.07s.	5.1s.

Tableau 3. Temps de rendu avec tous les effets de transport de lumière.

5.5. Conclusion

Dans ce travail, nous avons proposé une approche de path tracing volumique pour les données de simulation SPH. Cette solution est capable de rendre les fluides à une haute qualité. Comme montrent les résultats de

l'évaluation, des effets de transport de lumière (tels que la réflexion, la réfraction multiple, l'absorption et la diffusion multiple) ont été efficacement produits. Plus précisément, notre approche est une approche entièrement basée sur les particules dans laquelle (1) une surface lisse est implicitement définie, (2) la normale est fixée sur la base du voisinage spatial, et (3) le rayon traversant le milieu est échantillonné en fonction de la densité intégrée des particules. En conséquence, l'approche proposée est non seulement capable de rendre plusieurs fluides avec des caractéristiques différentes (couleur, fonction de phase, etc.), mais aussi plusieurs types de surface (continue / discrète) avec des matériaux différents (BRDF / BSDF). Il est à noter que ce travail peut être intégré dans le MitsubaRenderer en tant que nouveau module (intégrateur) afin de rendre directement les données de simulation SPH sans phases de prétraitement (reconstruction de surface et transformation de particules).

Même si la solution proposée est implémentée sur le MitsubaRenderer qui est multithread, elle peut être encore accélérée en utilisant des capacités matérielles modernes comme CUDA pour un rendu rapide/efficace.

Conclusion générale

Malgré que l'utilisation des points comme primitive de modélisation pour la synthèse d'images a clairement prouvé son efficacité (grâce aux progrès technologiques incarnés sous la forme de scanners 3D capables d'acquérir de nombreux modèles 3D du monde réel avec leur apparence d'une manière rapide et efficace). Le rendu des points d'une manière réaliste reste un champ de recherche, où les techniques d'illumination connues pour les scènes classiques, doivent être dérivées et adaptées au cas des points discrets qui n'ont aucune connectivité entre eux.

Travaux achevés

Le cadre dans lequel nous avons proposé notre approche de rendu physique est les fluides. Un fluide SPH est un ensemble de particules qui peuvent être considérées comme des points 3D associés à des attributs. Grâce aux techniques de rendu des nuages de points, nous avons traité les particules comme des points en définissant implicitement une surface qu'on va rendre par la suite, par une technique d'illumination globale et précisément le path tracing. Cela a été fait en suivant les étapes suivantes :

- Développement d'un lancer de rayon sur les points qui sont reconstruits localement par des splats.
- Accélération de ce lancer de rayon par des techniques et structures accélératrices (telles que les subdivisions spatiales et les volumes englobants).
- Amélioration de la qualité visuelle de la surface rendue (surtout sur la silhouette) par ce lancer de rayon, en appliquant certaines règles qui ont enlevé les conflits dus au chevauchement inter-splats.
- Transformation du lancer de rayon à un path tracing stochastique de Monte Carlo pour faire du rendu par illumination globale pour les particules que l'on a traité comme des points (positions des particules). Ces derniers nous ont servis pour reconstruire une surface lisse implicite utilisant notre modèle développé pour les particules SPH et qui se base sur le modèle classique des metaballs. Dans notre implémentation du path tracing, nous avons pris en compte des phénomènes lumineux compliqués (réflexion, diffusion multiple, absorption, **TIR:** Total Internal Reflection) afin d'avoir un rendu le plus réaliste possible.

Synthèse des résultats

Nous avons montré la crédibilité de notre approche de rendu physiquement réaliste de fluides à base de particules SPH, qui donne des résultats similaires à ceux donnés par le rendu de la surface reconstruite explicitement (par remaillage), de point de vue: qualité surfacique, effets de transport de lumière (réflexion, réfraction multiple, diffusion multiple, absorption, **TIR**: Total Internal Reflection).

Notre solution a été étendue pour un cas d'étude qui est l'eau de mer, où nous avons pu rendre des types de particules additionnels d'une manière physique (en définissant pour chaque type de particules, son comportement lumineux adéquat), à savoir : le sable, la mousse, et les bulles (en plus de l'eau elle-même).

Perspectives

La technique que nous avons proposée offre plusieurs perspectives de recherche qui devront l'améliorer d'un point de vue qualitatif et quantitatif.

Dans un premier temps, il nous semble intéressant de réfléchir sur l'amélioration de la qualité visuelle du rendu de fluide. Le problème de généralisation de notre technique à d'autres liquides reste en grande partie ouvert.

Enfin, à plus long terme, il nous semble intéressant de se pencher à l'amélioration du temps de calcul du rendu par l'intégration d'un processus de parallélisation grâce aux récents progrès technologiques qu'a connu le matériel graphique, le CPU et le GPU pourront être impliqués dans ce contexte pour coopérer plus efficacement.

Bibliographie

Adamson, Anders, et Marc Alexa. «Ray tracing point set surfaces.» *Shape Modeling International, 2003.* 2003. 272-279.

Alexa, Marc, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, et Claudio T. Silva. «Point Set Surfaces.» *Proceedings of the Conference on Visualization '01.* Washington, DC: IEEE Computer Society, 2001. 21-28.

Amanatides, John. «Ray tracing with cones.» *ACM SIGGRAPH Computer Graphics.* 1984. 129-135.

Beddiaf, Ali, et Med Chaouki Babahenini. «An improved splat-based ray tracing for point-based objects.» *Programming and Systems (ISPS), 2015 12th International Symposium on.* 2015. 1-5.

Beddiaf, Ali, et Med Chaouki Babahenini. «Efficient accelerated ray tracing of point-based objects.» *Multimedia Computing and Systems (ICMCS), 2014 International Conference on.* 2014. 15-19.

Blasi, Philippe, Bertrand Saec, et Christophe Schlick. «A rendering algorithm for discrete volume density objects.» *Computer Graphics Forum.* 1993. 201-210.

Blinn, James F. «A generalization of algebraic surface drawing.» *ACM transactions on graphics (TOG)* (ACM) 1 (1982): 235-256.

Blinn, James F. «Light reflection functions for simulation of clouds and dusty surfaces.» *ACM SIGGRAPH Computer Graphics.* 1982. 21-29.

Botsch, Mario, Michael Spernat, et Leif Kobbelt. «Phong Splatting.» *Proceedings of the First Eurographics Conference on Point-Based Graphics.* Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004. 25-32.

Brousset, Mathias, Emmanuelle Darles, Daniel Meneveaux, Pierre Poulin, et Benoît Crespin. «Simulation and control of breaking waves using an external force model.» *Computers & Graphics* (Elsevier) 57 (2016): 102-111.

Bunnell, Michael. «Dynamic ambient occlusion and indirect lighting.» *Gpu gems* (Citeseer) 2 (2005): 223-233.

Christensen, Per. «Point-based approximate color bleeding.» *Pixar Technical Notes* 2 (2008): 6.

Co, Christopher S., Bernd Hamann, et Kenneth I. Joy. «Iso-splatting: A point-based alternative to isosurface visualization.» *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on.* 2003. 325-334.

Cohen, Michael F., et Donald P. Greenberg. «The hemi-cube: A radiosity solution for complex environments.» *ACM SIGGRAPH Computer Graphics.* 1985. 31-40.

Csuri, Charles, Ron Hackathorn, Richard Parent, Wayne Carlson, et Marc Howard. «Towards an interactive high visual complexity animation system.» *ACM SIGGRAPH Computer Graphics.* 1979. 289-299.

Dachsbacher, Carsten, et Marc Stamminger. «Reflective shadow maps.» *Proceedings of the 2005 symposium on Interactive 3D graphics and games.* 2005. 203-231.

Desbrun, Mathieu, et Marie-Paule Gascuel. «Smoothed particles: A new paradigm for animating highly deformable bodies.» Dans *Computer Animation and Simulation'96*, 61-76. Springer, 1996.

Dobrev, Petar, Paul Rosenthal, et Lars Linsen. «Interactive image-space point cloud rendering with transparency and shadows.» (Václav Skala-UNION Agency) 2010.

Drebin, Robert A., Loren Carpenter, et Pat Hanrahan. «Volume rendering.» *ACM Siggraph Computer Graphics.* 1988. 65-74.

Enright, Douglas, Ronald Fedkiw, Joel Ferziger, et Ian Mitchell. «A hybrid particle level set method for improved interface capturing.» *Journal of Computational physics* (Elsevier) 183 (2002): 83-116.

Esperanca, Claudio, Antonio Oliveira, et others. «Efficient HPR-Based Rendering of Point Clouds.» *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on.* 2012. 126-133.

Fedkiw, Ronald, Jos Stam, et Henrik Wann Jensen. «Visual simulation of smoke.» *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 2001. 15-22.

Gatzke, Timothy D., et Cindy M. Grimm. «Estimating curvature on triangular meshes.» *International journal of shape modeling* (World Scientific) 12 (2006): 1-28.

Gingold, Robert A., et Joseph J. Monaghan. «Smoothed particle hydrodynamics: theory and application to non-spherical stars.» *Monthly notices of the royal astronomical society* (Oxford University Press) 181 (1977): 375-389.

Goradia, Rhushabh, Sriram Kashyap, Parag Chaudhuri, et Sharat Chandran. «Gpu-based ray tracing of splats.» *Computer Graphics and Applications (PG), 2010 18th Pacific Conference on.* 2010. 101-108.

Gortler, Steven J., Radek Grzeszczuk, Richard Szeliski, et Michael F. Cohen. «The lumigraph.» *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques.* 1996. 43-54.

Goswami, Prashant, Philipp Schlegel, Barbara Solenthaler, et Renato Pajarola. «Interactive SPH simulation and rendering on the GPU.» *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 2010. 55-64.

Grossman, J. P., et William J. Dally. *Point sample rendering.* Springer, 1998.

Hadwiger, Markus, Christian Sigg, Henning Scharsach, Khatja Bühler, et Markus Gross. «Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces.» *Computer Graphics Forum.* 2005. 303-312.

Holzschuch, Nicolas. «Accurate computation of single scattering in participating media with refractive boundaries.» *Computer Graphics Forum.* 2015. 48-59.

Imai, Takuya, Yoshihiro Kanamori, et Jun Mitani. «Real-time screen-space liquid rendering with complex refractions.» *Computer Animation and Virtual Worlds* (Wiley Online Library) 27 (2016): 425-434.

Jakob, Wenzel. «Mitsuba renderer.» 2010.

Jarosz, Wojciech, Derek Nowrouzezahrai, Iman Sadeghi, et Henrik Wann Jensen. «A comprehensive theory of volumetric radiance estimation using photon points and beams.» *ACM Transactions on Graphics (TOG)* (ACM) 30 (2011): 5.

Jensen, Henrik Wann. «Global illumination using photon maps.» Dans *Rendering Techniques' 96*, 21-30. Springer, 1996.

Jensen, Henrik Wann. *Realistic image synthesis using photon mapping.* AK Peters, Ltd., 2001.

Jensen, Henrik Wann, et Per H. Christensen. «Efficient simulation of light transport in scenes with participating media using photon maps.» *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* 1998. 311-320.

Kajiya, James T. «The rendering equation.» *ACM Siggraph Computer Graphics.* 1986. 143-150.

Kanamori, Yoshihiro, Zoltan Szego, et Tomoyuki Nishita. «GPU-based Fast Ray Casting for a Large Number of Metaballs.» *Computer Graphics Forum.* 2008. 351-360.

Katz, Sagi, Ayellet Tal, et Ronen Basri. «Direct visibility of point sets.» *ACM Transactions on Graphics (TOG)* (ACM) 26 (2007): 24.

Kavan, Ladislav, Ivana Kolingerova, et Jiri Zara. «Fast approximation of convex hull.» (ACST) 6, n° 101-104 (2006).

Keller, Alexander. «Instant radiosity.» *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* 1997. 49-56.

Kulla, Christopher, et Marcos Fajardo. «Importance sampling techniques for path tracing in participating media.» *Computer Graphics Forum.* 2012. 1519-1528.

Laan, Wladimir J., Simon Green, et Miguel Sainz. «Screen space fluid rendering with curvature flow.» *Proceedings of the 2009 symposium on Interactive 3D graphics and games.* 2009. 91-98.

Lafortune, Eric P., et Yves D. Willems. «Rendering participating media with bidirectional path tracing.» Dans *Rendering Techniques' 96*, 91-100. Springer, 1996.

Levoy, Marc, et Turner Whitted. *The use of points as a display primitive.* University of North Carolina, Department of Computer Science, 1985.

Linsen, Lars, Karsten Müller, et Paul Rosenthal. «Splat-based Ray Tracing of Point Clouds.» *Journal of {WSCG}* 15 (2007): 51-58.

Lorensen, William E., et Harvey E. Cline. «Marching cubes: A high resolution 3D surface construction algorithm.» *ACM siggraph computer graphics.* 1987. 163-169.

Losasso, Frank, Frédéric Gibou, et Ron Fedkiw. «Simulating water and smoke with an octree data structure.» *ACM Transactions on Graphics (TOG).* 2004. 457-462.

Machado e Silva, R., Claudio Esperança, Ricardo Marroquim, et Antonio A. F. Oliveira. «Image Space Rendering of Point Clouds Using the HPR Operator.» *Computer Graphics Forum.* 2014. 178-189.

Marroquim, Ricardo, Martin Kraus, et Paulo Roma Cavalcanti. «Efficient Point-Based Rendering Using Image Reconstruction.» *SPBG.* 2007. 101-108.

Monaghan, Joe J. «Smoothed particle hydrodynamics.» *Annual review of astronomy and astrophysics* (Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA) 30 (1992): 543-574.

Müller, Matthias, David Charypar, et Markus Gross. «Particle-based fluid simulation for interactive applications.» *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* 2003. 154-159.

Pfister, Hanspeter, Matthias Zwicker, Jeroen Baar, et Markus Gross. «Surfels: Surface Elements As Rendering Primitives.» *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques.* New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. 335-342.

Rama, C. Hoetzlein. «Fluids v. 3-a large-scale, open source fluid simulator., december 2012.» *URL http://fluids3. com* 1.

Reeves, William T. «Particle systems—a technique for modeling a class of fuzzy objects.» *ACM Transactions on Graphics (TOG)* (ACM) 2 (1983): 91-108.

Ritschel, Tobias, Thorsten Grosch, Min H. Kim, H.-P. Seidel, Carsten Dachsbacher, et Jan Kautz. «Imperfect shadow maps for efficient computation of indirect illumination.» *ACM Transactions on Graphics (TOG)* (ACM) 27 (2008): 129.

Rusinkiewicz, Szymon, et Marc Levoy. «QSplat: A multiresolution point rendering system for large meshes.» *Proceedings of the 27th annual conference on Computer graphics and interactive techniques.* 2000. 343-352.

Schaufler, Gernot, et Henrik Wann Jensen. «Ray Tracing Point Sampled Geometry.» *Proceedings of the Eurographics Workshop on Rendering Techniques 2000.* London, UK: Springer-Verlag, 2000. 319-328.

Shirmun, Leon A., et Salim S. Abi-Ezzi. «The cone of normals technique for fast processing of curved patches.» *Computer Graphics Forum.* 1993. 261-272.

Smith, Alvy Ray. «Plants, fractals, and formal languages.» *ACM SIGGRAPH Computer Graphics* (ACM) 18 (1984): 1-10.

Stam, Jos, et Eugene Fiume. «Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes.» *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques.* New York, NY, USA: ACM, 1995. 129-136.

Takahashi, Tsunemi, et al. «Realistic animation of fluid with splash and foam.» *Computer Graphics Forum.* 2003. 391-400.

Thürey, Nils, Filip Sadlo, Simon Schirm, Matthias Müller-Fischer, et Markus Gross. «Real-time simulations of bubbles and foam within a shallow water framework.» *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation.* 2007. 191-198.

Wald, Ingo, et Hans-Peter Seidel. «Interactive ray tracing of point-based models.» *Point-Based Graphics, 2005. Eurographics/IEEE VGTC Symposium Proceedings.* 2005. 9-16.

Wald, Ingo, et Vlastimil Havran. «On building fast kd-trees for ray tracing, and on doing that in O (N log N).» *Interactive Ray Tracing 2006, IEEE Symposium on.* 2006. 61-69.

Wald, Ingo, Philipp Slusallek, Carsten Benthin, et Markus Wagner. «Interactive rendering with coherent ray tracing.» *Computer graphics forum.* 2001. 153-165.

Wand, Michael, et Wolfgang Straßer. «Multi-Resolution Point-Sample Raytracing.» *Graphics Interface.* 2003. 139-148.

Wang, Beibei, Jean-Dominique Gascuel, et Nicolas Nolzschuch. «Point-based Light Transport for Participating Media with Refractive Boundaries.» *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations.* Goslar, Germany: Eurographics Association, 2016. 109-119.

Wang, Rui, Rui Wang, Kun Zhou, Minghao Pan, et Hujun Bao. «An efficient GPU-based approach for interactive global illumination.» *ACM Transactions on Graphics (TOG)* (ACM) 28 (2009): 91.

Ward, Gregory J., Francis M. Rubinstein, et Robert D. Clear. «A ray tracing solution for diffuse interreflection.» *ACM SIGGRAPH Computer Graphics* (ACM) 22 (1988): 85-92.

Whitted, Turner. «An Improved Illumination Model for Shaded Display.» *Commun. ACM* (ACM) 23 (6 1980): 343-349.

Wu, Jianhua, et Leif Kobbelt. «Optimized Sub-Sampling of Point Sets for Surface Splatting.» *Computer Graphics Forum* 23 (2004): 643-652.

Wyman, Chris. «An approximate image-space approach for interactive refraction.» *ACM Transactions on Graphics (TOG).* 2005. 1050-1053.

Xiao, Xiangyun, Shuai Zhang, et Xubo Yang. «Real-time high-quality surface rendering for large scale particle-based fluids.» *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* 2017. 12.

Yu, Jihun, et Greg Turk. «Reconstructing surfaces of particle-based fluids using anisotropic kernels.» *ACM Transactions on Graphics (TOG)* (ACM) 32 (2013): 5.

Zhou, Kun, Qiming Hou, Rui Wang, et Baining Guo. «Real-time KD-tree construction on graphics hardware.» *ACM Transactions on Graphics (TOG)* (ACM) 27 (2008): 126.

Zhu, Yongning, et Robert Bridson. «Animating sand as a fluid.» *ACM Transactions on Graphics (TOG).* 2005. 965-972.

Zwicker, Matthias, Hanspeter Pfister, Jeroen Van Baar, et Markus Gross. «Surface splatting.» *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 2001. 371-378.