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

Université Mohamed Khider – BISKRA
Faculté des Sciences et Sciences de l'ingénieur
Département d'informatique

Mémoire en vue de l'obtention du diplôme de

Magister en Informatique
Option : Intelligence artificielle et image

Rendu de texture volumique par une représentation à base de points

Réalisé par :
Ammar HAMIDA

Soutenue devant le jury composé de :

BATOUCHE Med Chaouki	Professeur	Université de Constantine	Président
DJEDI Nour Eddine	Maître de conférence	Université de Biskra	Rapporteur
KHOLLADI Med Khiereddine	Maître de conférence	Université de Constantine	Examinateur
BELlATAR Brahim	Maître de conférence	Université de Batna	Examinateur
BOUKERRAM Abdellah	Chargé de cours (Docteur N.T.)	Université de Setif	Examinateur

2004/2005

REMERCIEMENTS

Nous saisissons cette unique et honorable occasion pour exprimer nos vifs remerciements et chaleureux hommages à tous ceux qui ont contribués de près ou de loin à la réalisation et à l'amendement de la présente thèse. Plus particulièrement à nos courtois et éminents encadreurs, qui n'ont pas hésités un seul instant à nous apporter leur large, fructueux et indubitable concours, pour nous éclairer et orienter dans le bon sens.

- Mr DJEDI Nour Eddine.
- Mr M^{ed} Chaouki Babahenini.

Par la même et sans perdre de vue, les membres du jury auxquels nous leurs accordons nos souhaits les plus sincères pour leur sagesse, leur compréhension et leur largesse en matière de temps réservé à notre égard.

Merci, à tous ...

DIDICACES

Je dédie la présente thèse à mes chers et bien aimés parents, pour leur soutient moral et financier et à tous les membres de la famille Hamida (grands-mères, tentes et oncles); plus particulièrement mes frères et sœur : Fahed, Farouk, Hakim, Bilal, ma sœur Soraya et mon petit frère Youcef.

A tout mes amis ; particulièrement : Djelal, Ramzi, Halim, Beda, Hamza.

Ainsi qu'à tous ceux qui m'ont aidé à son élaboration.

Table des matières

Table des figures

Liste des tableaux

INTRODUCTION GENERALE

La synthèse d'image a connu ces dernières années un remarquable développement, ce qui lui a permis de s'insérer dans plusieurs disciplines et de conquérir de nombreux marchés industriels et commerciaux, plus particulièrement les domaines de la communication, de la simulation (les simulateurs de vol, d'opérations chirurgicales), de la CAO, du cinéma, des jeux vidéos, de la publicité ou de la visualisation scientifique.

Parmi les objectifs de la synthèse d'images est la reproduction visuelle de la réalité en un temps raisonnable. Par exemple au domaine de la post-production cinématographique, l'avantage principal de ces images est bien le moindre coût de production par rapport à leurs prédécesseurs.

Le processus de création d'images s'effectue en deux étapes : la *modélisation* et le *rendu*. La modélisation consiste à définir la scène que l'on souhaite visualiser en données interprétables par l'ordinateur. Ces données, telles que la position des surfaces et leur nature ou l'éclairement de la scène, sont utilisées lors de l'étape de rendu pour obtenir des *image*s.

Ces images sont la projection visuelle, compréhensible par l'utilisateur, de modèles mathématiques et physiques de la réalité. Dans ce cadre, le *réalisme* d'une image est dépendant de la précision avec laquelle on définit la scène, ainsi que de la précision avec laquelle on simule la propagation de la lumière. En d'autres termes, pour obtenir une image visuellement réaliste d'une scène, il faudra d'une part la définir le plus précisément possible et d'autre part appliquer les modèles physiques de propagation de la lumière les plus complets. Idéalement, le but à atteindre est le *photoréalisme*, c'est à dire une image de synthèse indiscernable d'une photographie.

Problématique et objet des travaux

De nos jours, la performance du matériel est en pleine mutation ; la demande du réalisme et la complexité des scènes ne cessent de s'accroître et dépasse souvent les capacités de calcul. Partant de cette constatation, pour résoudre cette insuffisance, récemment on a découvert d'autres approches plus compétitives : les textures volumiques et le *rendu à base de points*. Ces deux approches offrent une représentation alternative de la scène, permettant de reformuler l'expression de la complexité géométrique.

La textures volumique est la méthode la plus compétitive en matière de mimétisme géométrique. Elle consiste à modéliser un volume de référence appeler texel (représentant une géométrie complexe et répétitives), ensuite le plaquer sur une surface volumique appeler peau. Cette technique présente les inconvénients suivants :

- Elle est de moindre souplesse.
- Elle est coûteuse en matière de temps de rendu et en mémoire.

Ces dernières années est né un nouveau paradigme de modélisation. Il répond à cet inconvénient. Ce paradigme définit les objets 3D en nuages de points non structurés, donc sans topologie explicite. Chaque point est équipé d'une normale, éventuellement d'un rayon, et d'une liste de paramètres d'apparence (couleur, coordonnées de textures, etc.). Visualiser de tels objets reviennent à projeter simplement le nuage sur le plan de l'écran. Pour pallier l'existence de trous aux zones insuffisamment denses, on dessine une « tache » à l'endroit de la projection du point (*splatting*). La taille de cette tache dépend de la portion de surface associée au point (son *rayon*), on obtient ainsi des objets sans trous. Cette technique a les avantages suivants :

- Elle s'adapte mieux à l'affichage d'importants modèles géométriques.
- La mémoire utilisée est ainsi réduite et les techniques de niveau de détail (*LOD*, "Level Of Détails") sont d'autant plus faciles à mettre en œuvre que les algorithmes de simplification de maillage géométriques.
- D'autre part, le rendu effectif d'un point isolé est très rapide comparé à la "rastérisation" d'une primitive géométrique.

Le majeur et inhérent handicape de ce mode de représentation est qu'il n'offre aucune garantit à la cohérence topologique et géométrique (continuités) des modèles affichés. En particulier, des zones sous échantillonnées apparaissent si l'on se rapproche trop de la surface. D'autre part, il n'y a aucune information locale dépassant l'espace du point ce qui rend difficile l'utilisation de cette méthode dans un environnement physique virtuel.

Afin de palier à ce que nous venons de constater, il est indispensable et utile de concevoir une méthode de rendu hybride (amalgamer une texture volumique et un rendu à base de poins). Elle permet plus ou moins une combinaison des avantages de la texture volumique avec celles du rendu à base de points. Elle se traduit par une nouvelle représentation de texture volumique, satisfaisante et offre une modélisation approprier aux géométries complexes répétitives. Elle rend ainsi la méthode plus générale, plus rapide ; elle consomme moins d'espace mémoire. En d'autres termes une réduction à l'aliassage et aux distorsions.

Organisation du mémoire

Le présent ouvrage englobe cinq chapitres, où chacun d'eux comporte sa propre particularité, à savoir:

En premier lieu, nous faisons une représentation succincte de l'art des différentes techniques de traitement de scènes complexes, ainsi que les diverses solutions adoptées en synthèse d'image pour les traiter. Celle ci comporte deux classes :

- Les approches classiques : traitent l'accélération par le matériel et les techniques de simplification de maillage. Nous avons mis en relief leurs avantages et leurs inconvénients.
- Les approches émergentes : traitent les différentes méthodes pour contourner les problèmes de complexité. En finale on présente le principe des textures volumiques classiques.

Au second chapitre on a mis en revue une classification de rendu à base de points. Nous ventilons les diverses méthodes, que nous classons en fonction de leur type d'échantillonnage. Nous clôturons le chapitre par un bilan de différentes approches et d'une manière comparative.

Aux deux chapitres suivants nous avons exposé avec minutie le fondement des travaux de notre nouvelle représentation de textures volumiques. La méthode repose essentiellement sur la modélisation et le rendu de points, encodée dans chaque voxel du volume. Ces voxel forment l'échantillon de texture volumique, dont les instances répétées et déformées sont des *texels*, ils recouvrent la surface de l'objet à habiller, formant ainsi une couche volumique. La fabrication et le rendu de ces diverses entités sont ainsi décrites. Enfin ; nous avons clos chacun des chapitres par un résultat et un exposé sur inconvénient et avantages de la méthode.

Le chapitre final, est consacré à l'exposition analytique des résultats obtenus.

Finalement une conclusion et des perspectives d'avenir clôturent notre mémoire.

Chapitre 1 : Complexité en synthèse d'image

1 Introduction

Dès le début de l'infographie, la quête du réalisme a représenté une cible importante. Le terme de photo-réalisme décrit des techniques et des formes d'art qui tentent de créer des images de synthèse qui pourraient être confondues avec les images réelles.

La construction d'images de synthèse nécessite de grandes puissances de calcul surtout si l'on souhaite réaliser des applications interactives en temps réel. Malgré la montée en puissance des PC, la demande est toujours plus grande que les possibilités offertes. La recherche conduit donc à concevoir des méthodes intelligentes de construction de scènes dynamiques où l'on joue sur les algorithmes de visibilité ainsi que sur les méthodes de modélisation des objets (fractals, L-systèmes,... etc.).

Pour obtenir des simulations visuelles riches, il est nécessaire d'avoir suffisamment de détails géométriques, de textures et des effets d'éclairage. Plusieurs techniques existent pour y arriver. La méthode traditionnelle pour représenter ce type d'objet nécessite un nombre trop élevé de polygones, ce qui rend l'affichage beaucoup trop long. Une façon de résoudre ce problème est l'utilisation de méthodes de rendu où le nombre d'éléments utilisés pour l'affichage dépend du point de vue : la complexité géométrique est donc fortement diminuée.

Pour remédier aux problèmes du rendu polygonal, plusieurs techniques et approches ont été présentées. Ces dernières sont classées en deux grandes familles [Debunne04] :

- Les approches classiques.
- Les approches émergentes.

2 Les approches classiques

2.1 Accélération par le hardware graphique

Jusqu'aux nos jours une des premières solutions adoptées, afin d'éviter les problèmes de la représentation géométrique est l'utilisation de matériel graphique.

Dès l'apparition des stations graphiques[1], on a bénéficie de leur performance de calcul dû à leurs architectures parallèles et l'ensemble de processeurs spécialisés qu'elles contiennent pour diminuer le temps de calcul énorme nécessaire à une application graphique. Il faut noter que la plupart des stations graphiques utilisent le lissage de *Gouraud* et le tampon de profondeur pour le rendu. Elles supposent la plupart du temps que la scène soit décomposée en facettes polygonales planes, c'est à dire que la primitive géométrique de prédilection est le triangle. Les calculs de profondeur et d'éclairement sont donc réalisés aux sommets des polygones et les valeurs aux pixels entre ces sommets sont déterminées par interpolation. Ces stations sont très chères (coûteuses) ce qui a limité leur usage personnel.

Limites de l'approche

Les inconvénients de l'approche tiennent à la spécificité de certaines fonctionnalités, hypothéquant la pérennité et la portabilité des méthodes qui les utilisent, dans un contexte où la rapidité de l'évolution du matériel peut rendre trivial un problème difficile quelques mois plutôt, ou au contraire une fonctionnalité puissante mais peu utilisée par l'industrie peut disparaître (quand ce n'est pas le fabriquant de la carte...). D'autre part (quelle que soit l'amélioration du matériel), devoir passer par une API[2] suppose nécessairement des contraintes et des limites, qui n'existent pas lorsqu'on adopte une solution purement logicielle.

[1] Tel que les SGI de Silicon Graphics ou Alto de Xerox.
[2] API : Application Personnel Interface.

2.2 Simplification polygonale

La complexité des modèles géométriques (mesurée par le nombre de polygones) semble grandir plus rapidement que la capacité du matériel graphique pour les rendre en mode interactif. D'une autre manière, le nombre de polygones que l'on veut semble toujours excéder le nombre de polygones permis [Luebke01].

Les techniques de simplification polygonale offrent une solution pour la totalité des modèles complexes. Ces méthodes peuvent notamment être utilisées quand l'objet est petit sur l'écran, s'il est éloigné, s'il se déplace, s'il n'est pas dans la direction principale du regard de l'utilisateur, tout en cherchant à réduire le coût de rendu sans une perte significative dans le contenu visuel de la scène.

2.2.1 Opération de simplification

La littérature d'infographie est remplie d'algorithmes de simplification excellents. Les chercheurs ont proposé des douzaines d'approches, chacune avec des forces et des faiblesses, dont la majorité nécessite deux caractéristiques principales :

- Les maillages des objets soient *bien formés.*
- 2D-manifold : toute arête soit l'incidence de deux faces (Figure 1 et 2).

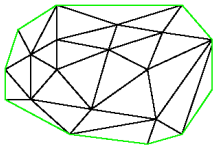


Figure 1 : Une 2D-manifold avec une frontière (bords de frontière en vert). [Luebke01]

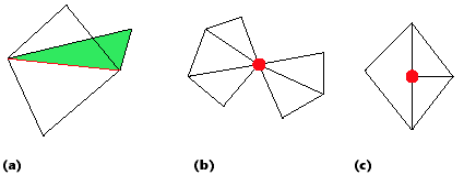


Figure 2 : Exemples des mailles de non-manifold : (a) Un bord partagé par trois triangles, (b) un sommet a partagé par deux ensembles de triangles autrement non liés, et (c) un T-Sommet. [Luebke01]

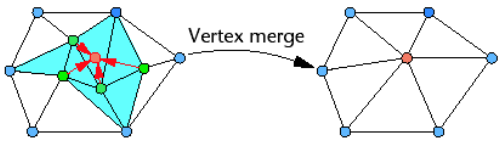


Figure 3 : Fusion de sommets (vert) dans un sommet représentatif simple (orange). [Luebke01]

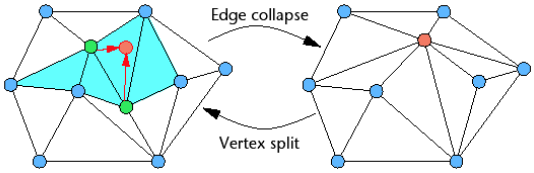


Figure 4 : Effondrement de bord (l'opération de fusion de deux sommets qui partagent un bord). [Luebke01]

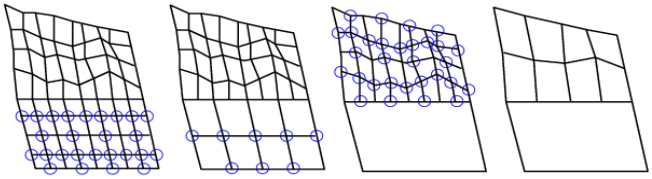
De plus, ces algorithmes présentent un certain nombre d'opérations de simplification grossière [Krus97] :

- Normalisation : procède à l'élimination des points et arêtes dégénérés ou définis plusieurs fois.
- Simplification des sommets : tous les points à l'intérieur d'un volume sont regroupés. Ainsi, les amas de points et les petites facettes sont combinés (Figure 3 et 4).
- Simplification des arêtes : chaque bord ayant une longueur inférieure à une longueur donnée est éliminé.
- Simplification basée sur les angles : les arêtes qui forment un angle très fermé sont éliminées. Par opposition, les arêtes qui sont alignées peuvent être fusionnées.
- Simplification basée sur la surface d'une facette : les facettes dont la surface est inférieure à une certaine valeur sont éliminées. Les trous qui en résultent peuvent être bouchés par une triangulation en utilisant, par exemple, le barycentre des points retirés.
- Simplification basée sur la normale à une facette : les facettes dont les normales sont presque parallèles sont éliminées (il faut aussi boucher le trou). Cela revient donc à fusionner les polygones coplanaires. Il faut cependant noter que ceci requiert la connaissance des relations de voisinage entre facettes.

2.2.2 Classification des méthodes de simplification

Il existe plusieurs manières de classer les algorithmes de simplification [Krus97, Luebke01] selon les différents critères adaptés. Nous avons choisi de présenter une classification selon les mécanismes de déplacement ou « *enlèvement* » de polygone de base, où il y a trois grandes classes de simplification :

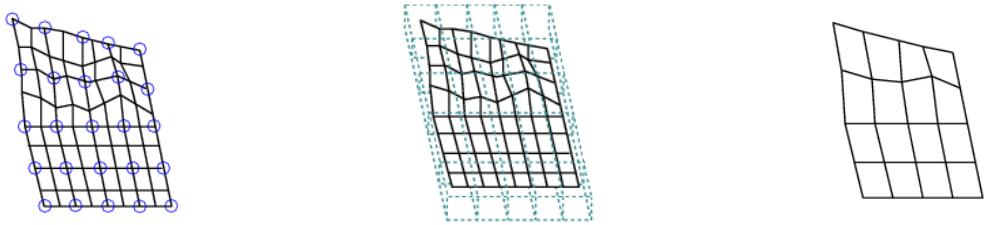
- La décimation : Les techniques de décimation enlèvent itérativement des sommets ou des faces de la maille, et re-triangulent le trou résultant après chaque pas (Figure 5). Le procédé de décimation s'arrête lorsqu'il ne peut plus retirer de géométrie et lorsque la représentation satisfait un degré d'approximation spécifié par l'utilisateur [Luebke01, Vincent02, Schwarze03]. Ces algorithmes sont relativement simples à coder et peuvent être très rapides. La plupart de ces algorithmes utilisent des changements strictement locaux (retirer une face ou un sommet) si cela ne cause pas de violations dans la topologie.



Modèle initial 1ère réduction 2ème réduction 3ème réduction

Figure 5 : Exemple de la décimation.[Krus97]

- **L'échantillonnage :** Cette méthode effectue un échantillonnage de la géométrie de l'objet, soit en choisissant arbitrairement un certain nombre de sommets à conserver, soit en englobant le modèle dans une grille tridimensionnelle et en échantillonnant chaque boîte de la grille (Figure 6). Ces approches sont les plus complexes et les plus difficiles à coder. Ils peuvent avoir des difficultés à réaliser la haute fidélité puisque les particularités de haute fréquence sont difficiles à les échantillonner avec exactitude. Ces algorithmes travaillent d'habitude mieux sur des formes organiques douces sans coins pointus.



Sélection aléatoire Subdivision spatiale Modèle simplifié

Figure 6 : Exemple D'échantillonnage. [Krus97]

- **La subdivision adaptative :** elle commence avec un modèle de base très simple et le subdivise récursivement en ajoutant des détails à certains endroits du modèle à chaque étape. L'algorithme s'arrête lorsque le modèle approxime le modèle original à un degré spécifier par l'utilisateur [Luebke01]. Cette méthode est assez peu utilisée car il n'est pas facile de construire la simplification initiale dans le cas général (Figure 7).

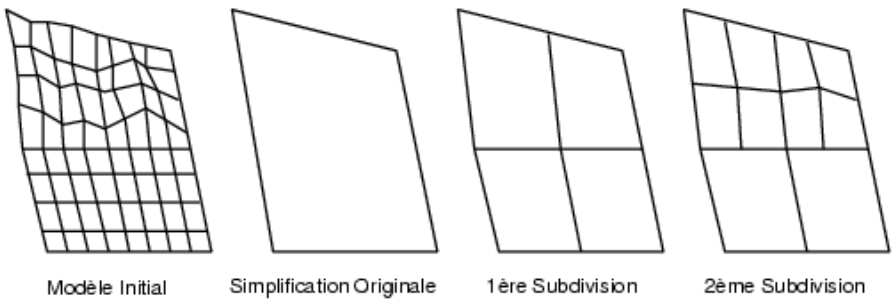


Figure 7 : Exemple de la subdivision adaptative. [Krus97]

Limites de l'approche

De nos jours, les algorithmes de décimation sont les plus utilisés dans le domaine de simplification, en dépit des inconvénients suivants :

- Le temps de calcul prohibitif engendré par deux processus successif : le processus de modélisation géométrique et le processus de décimation.
- Il n'existe pas d'algorithme général qui peut simplifier tous les objets.
- Le résultat de décimation n'est pas parfait ; il peut apparaître ainsi des trous de triangles hachurés. Ceci est dû généralement au difficulté de spécification des paramètres de contrôle d'erreur.

3 Les approches émergentes

3.1 Détermination de visibilité

Dès les débuts de la synthèse d'images on constate que les scènes développées sont de plus en plus complexes et contiennent souvent plus d'objets. Parallèlement à cette évolution, les algorithmes de visualisation et de rendu se sont perfectionnés. Très rapidement, on a cherché à regrouper les objets suivants différents critères afin d'optimiser les traitements. Les approches les plus anciennes, les plus connues sont les volumes englobants, les subdivisions spatiales régulières et les subdivisions spatiales hiérarchiques (octree). D'autres approches dédiées plus précisément à la radiosité ont été développées [Hasenfratz97].

3.1.1 Visibilité en général

On peut classer de différentes façons les algorithmes de visibilité [Leblanc00]. Certains déterminent des surfaces visibles ou potentiellement visibles d'un point de vue précis ou de vue ponctuelle (*pinhole camera*), et par conséquent elles sont difficilement applicables pour déterminer de façon spécifique la visibilité entre éléments pairs, comme c'est le cas dans un algorithme de radiosité.

Les différentes techniques de visibilité se divisent selon les catégories suivantes [Leblanc00] :

- **La visibilité approximative :** Plusieurs méthodes de visibilité se basent sur l'échantillonnage ; soit pour déterminer les surfaces visibles soit pour calculer un pourcentage d'occlusion. Parmi ces dernières, on peut citer les méthodes suivantes [Leblanc00] : La visibilité volumique et la carte d'ombre de *Williams*. La méthode la plus populaire consiste simplement à lancer un certain nombre de rayons entre une paire d'éléments, et à faire correspondre la visibilité à la fraction des rayons ayant atteint leur objectif sans blocage. Ces méthodes sont simples et efficaces.

- **La visibilité exacte :** Beaucoup de techniques classiques sont employées pour déterminer les surfaces visibles d'un point de vue. Elles retournent des résultats exacts mais à une précision fixée à l'avance par la taille de l'image désirée. Ces techniques [Leblanc00] comprennent entre autres le tampon de profondeur, le balayage des lignes, l'algorithme du peintre par arbre BSP et les algorithmes par subdivision d'aire. Les techniques de visibilité analytique 3D fournissent la solution complète et précise au problème de la visibilité. Malheureusement, la complexité algorithmique de ces techniques devient rapidement un souci majeur pour des scènes très complexes.

- **La visibilité conservatrice :** Les techniques mentionnées jusqu'à présent sont souvent inefficaces lorsque la scène contient un nombre élevé d'objets. Un groupe d'algorithmes dits conservateurs tentent d'accélérer les différentes requêtes de visibilité. Ces algorithmes retournent un ensemble de surfaces visibles d'un point de vue ou d'une région. Pour être efficaces, il est essentiel que « toutes » les surfaces visibles soient contenues dans l'ensemble final. Parmi les techniques utilisées on peut citer [Leblanc00] : Les portals, La pyramide de vue et celle utilisant l'occlusion provoquée par les surfaces rapprochées afin d'éliminer les surfaces cachées.

Généralement, on peut citer cinq techniques de détermination de la visibilité [Hasenfratz97] :

- La première consiste à disposer de plusieurs descriptions du même objet avec différentes résolutions. Ceci permet de choisir dynamiquement la meilleure résolution en fonction de la taille apparente de l'objet.
- La deuxième utilise des imposteurs pour remplacer des ensembles d'objets éloignés du point de vue par une texture de même apparence. On diminue ainsi le nombre de polygones à traiter en utilisant les algorithmes de placage de textures câblés sur la plupart des stations graphiques.

- La troisième solution consiste à "plonger" la scène dans une subdivision spatiale régulière (grille de voxels) et à construire, pour chaque voxel, la liste des voxels qui lui sont visibles. Cette approche est lourde et consomme trop de mémoire pour être utilisable.
- La quatrième solution est celle qui a inspiré une partie des travaux de *Hasenfratz* dans le cadre du lancer de faisceaux. Elle cherche à éliminer grossièrement les parties de la scène non visibles du point de vue parce qu'elles sont cachées par quelques objets imposants. On parle d'objets occultant ou d'occulteurs.
- La dernière utilise des faisceaux afin d'écarter rapidement les objets clairement non visibles depuis un point de vue donné.

3.2 Niveaux de détails (*LOD*)

Après avoir éliminé tous les objets qui ne sont pas dans le champ de vision, il est intéressant d'utiliser une géométrie plus simple pour les objets ayant une moindre importance visuelle. Cette technique fait appel à des niveaux de détails, ici appelés *LOD* (de l'anglais Level Of Detail).

Cette technique essaye d'améliorer la performance, d'accélérer le rendu et d'augmenter l'interactivité en simplifiant la scène polygonale générée. Pour cela, on représente les objets éloignés avec un *LOD* inférieur et les objets voisins avec un plus haut *LOD* (Figure 8). Le problème de l'utilisation des niveaux de détails se décompose en deux parties : La création et la gestion des *LOD*.

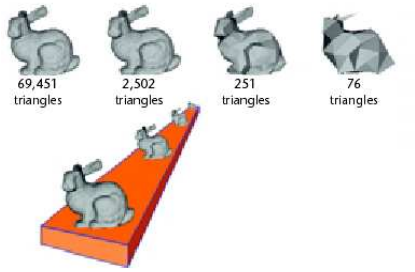


Figure 8 : Niveaux de détails dégradés en fonction de la distance. [Luebke01]

3.2.1 Création des LOD

La création des *LOD* est basée sur la simplification polygonale vue dans la section précédente. Il y a deux grandes classes de méthodes de création des *LOD* [Krus96] :

- La simplification orientée géométrie.
- La simplification orientée scène.

3.2.1.1 La simplification orientée géométrie

Elle est utilisée pour simplifier les objets polygonaux, il existe deux types de méthodes :

3.2.1.1.1 La simplification polygonale

Cette simplification est bien étudiée dans la section précédente (voir § 2.2).

3.2.1.1.2 Simplification structurelle et polygones texturés

Ces méthodes changent la structure de représentation des objets. Par exemple, il est possible de remplacer un objet polygonal par une boite englobante texturée à l'aide d'une image produite à partir d'une version détaillée de l'objet. Les *LOD* créés par cette méthode sont appelés des imposteurs [Schwarze03]. Ces méthodes produisent souvent des imposteurs qui dépendent du point de vue et posent des problèmes pour les calculs d'éclairement dans le cas où les projections d'ombres seraient nécessaires (le résultat est correct dans le cas d'un éclairage ambiant) [Krus97].

Certaines méthodes proposent de remplacer les objets par un polygone texturé à l'aide d'une image de l'objet original calculée avec un haut niveau de détails.

3.2.1.2 Simplification orientée scène

La plupart des algorithmes orientés scène cherche à simplifier des régions de la scène plutôt que les objets eux-mêmes. Ainsi, dans une phase de précalcul, ils décomposent récursivement la scène en zones 3D (en utilisant un octree ou une grille 3D), pour obtenir une description hiérarchique, chaque niveau étant plus détaillé que le

précédent. Ensuite, à chaque nœud de la hiérarchie, une représentation simplifiée de la sous hiérarchie est produite. Cette représentation sera utilisée si les critères de sélection sont satisfaits.

Toutes ces méthodes obligent à opérer, au sein de l'algorithme de sélection des *LOD*, un parcours d'arbre pour chaque image afin de déterminer le niveau de détails à utiliser. En plus, elles ne sont vraiment efficaces que sur des scènes très profondes, sans occlusion due par exemple à des murs [Krus97]. Parmi ces méthodes on peut citer les boîtes colorées [Krus97, Luebke01] et la fusion hiérarchique [Krus97].

Ces algorithmes sont assez primaires, et posent certains problèmes. Par exemple, que faire si un objet se déplace dans la scène ? De plus, ils ont tendance à produire des résultats peu satisfaisants du point de vue visuel, car ils peuvent produire des discontinuités entre les polygones et faire apparaître des trous, ce qui restreint leur usage.

Mais ils ont aussi des avantages : ils s'intègrent bien dans les algorithmes de gestion de scène au sens large, comme l'élimination des objets non visibles (object culling) et les z-buffers hiérarchiques. De plus, ils permettent de faire la gestion de *LOD* progressive, en dégradant les détails sur les parties éloignées d'un objet [Luebke01].

3.2.2 Gestion des LOD

3.2.2.1 Critères de sélection

L'art de l'utilisation des niveaux de détails consiste à trouver le juste équilibre entre le réalisme et la vitesse de calcul. Ainsi, on cherche à décrire l'importance d'un objet dans une scène afin de choisir la finesse de sa représentation.

En 1993, *Funkhouser* et *Séquin* ont signalé que la gestion des différents *LOD* doit être faite de manière prédictive[3], plutôt que réactive[4]. Ainsi, il est possible de garantir un temps de rendu inférieur à une certaine limite. A cette fin, les auteurs définissent un certain nombre de facteurs qui permettent de choisir un *LOD* parmi d'autres [Krus97] :

- **Distance :** Plus un objet est petit, moins ses détails sont perceptibles (Figure 8).
- **Incidence :** Dans certains cas, un objet peut avoir une forme particulière selon l'angle de vue. Par exemple, une porte ou un tube, vus de profil, se réduisent à des représentations très simples.
- **Vitesse :** Un objet en mouvement est moins bien perçu par le système visuel humain que celui d'un objet fixe. On pourra ainsi utiliser une représentation moins détaillée si un objet tourne rapidement sur lui-même ou si cet objet passe rapidement dans le champ de vision de l'utilisateur.
- **Tâche :** Il est envisageable d'adapter la présentation à la tâche en cours, par exemple, en utilisant des niveaux de détails dégradés pour les objets qui n'interviennent pas directement dans la tâche. Néanmoins, pour certaines tâches, il est important de présenter tous les objets avec le niveau de détails le plus élevé possible [Krus97].

3.2.2.2 Transition entre LODs

Si deux images successives utilisent deux représentations différentes d'un objet, il est possible d'échanger immédiatement les représentations entre les deux images. Toutefois, cela peut provoquer des sauts perceptibles par l'utilisateur si les deux représentations sont utilisées alternativement dans plusieurs images successives.

La meilleure solution consiste à utiliser, au sein de l'algorithme de sélection, un critère d'hystérésis qui produit une inertie dans les changements de *LOD*, en interdisant tout changement dans les *n* images suivant une transition.

Par ailleurs, certains algorithmes permettent de créer une suite continue de niveaux de détails, où ceux-ci s'adaptent en permanence aux contraintes d'affichage. Ces représentations sont appelées des *géomorphes*. Malheureusement, les temps de calcul restent prohibitifs pour une utilisation interactive.

Il subsiste néanmoins un problème : comment spécifier l'action de transition entre *LOD* ? Cette détermination ne peut être faite de manière automatique et nécessite encore une intervention humaine. Toutefois, certaines méthodes proposent d'utiliser le taux d'erreur ou la distance par rapport à l'original en faisant une correspondance avec la taille dans l'espace image.

[3] Basée sur la complexité de la scène.
[4] Basée sur le temps de calcul de l'image précédente.

Limites de l'approche (LOD)

Cette approche est cependant purement géométrique, et présente généralement deux inconvénients majeurs :

- Elle suppose que les détails plus petits qu'un pixel d'image ne contribuent pas à l'illumination, ce qui est faux : quelle que soit la taille apparente de ses ondulations, une tôle ondulée reflète globalement la lumière différemment qu'une tôle plane.
- L'effet de saut *« poping »* lors du passage d'une échelle de LOD à la suivante.

3.3 Les algorithmes spécialisés

3.3.1 Les cartes d'horizon

Les cartes d'horizon encodent, la manière avec laquelle de petites perturbations géométriques sur une surface portant des ombres sur la surface elle-même. Le but est de produire des cartes d'horizon consistant aux données acquises, et ensuite interpole et extrapole ces données [Rushmeier01].

Pour construire la carte d'horizon d'un objet, *Rushmeier* [Rushmeier01] a proposé une méthode qui consiste à acquérir huit images, une pour chacune des directions azimutales dans la carte d'horizon. On considère le cas d'un objet qui est essentiellement plat, avec des petites variations. Les huit images sont obtenues à partir des directions azimutales standard, à un angle de zénith d'environ 45°. Si on peut extrapoler cet ensemble d'ombres pour un angle de zénith à un domaine entier de 0° à 90°, on a les données complètes pour construire une carte d'horizon. La différence entre les ombres portées et propres ne peut pas être effectuée par une simple inspection d'une image unique. Cependant avec la carte des normales, les ombres propres sont identifiés dans chaque image en cherchant les pixels où le produit scalaire de la normale de la surface et la direction de la lumière est négative. Les pixels restants sont dans les ombres portées.

Bilan

La méthode de capture des cartes d'horizon, peut être employé avec un système de vision interactif pour afficher des ombres portées des surfaces inégales. Cette méthode exige un nombre restreint d'images, et n'exige aucune reconstruction pleine de la surface balayée. Elle prend en charge les bosses relativement pointues, toutefois elle produit des ombres plausibles qui sont conformes aux ombres observées dans les images d'entrée.

Bien que *Sloan* et *Cohen* aient proposé un algorithme en plusieurs passes permettant d'effectuer cette technique intégralement par la carte graphique, l'*horizon mapping* tel quel reste une technique coûteuse à utiliser car d'une part le rendu en plusieurs passes prend du temps, et d'autre part la carte de visibilité (d'horizon) occupe beaucoup d'espace mémoire [Rushmeier01, Porquet04].

3.3.2 Approches phénoménologiques

Parmi les objets les plus difficiles à modéliser et à animer par ordinateur ceux qui ne sont pas définis par une structure statique, rigide ou topologiquement simple. Beaucoup de ces formes complexes se rencontrent dans la nature, tels que l'eau, le nuage, la fumée, le feu et le vent qui présentent des difficultés pour déterminer leur mouvement.

L'approche phénoménologique permet de caractériser très efficacement les configurations et les comportements visibles, en évitant ainsi les problèmes de la simulation à partir d'un modèle purement physique :

- Difficulté de simuler à partir d'un système d'équation complexe.
- Coût de calcul très élevé.
- Difficulté de contrôler le résultat et les paramètres.

L'approche phénoménologique vise à reproduire directement la forme de l'objet et les phénomènes émergeants (plis, ondes, instabilités, formes d'équilibre) pour certaines familles. Elle a pour but d'obtenir une simulation phénoménologique visuellement réaliste, rapide et facilement contrôlable. Nous décrivant aux sections suivantes (les plants, les liquides et les gaz).

3.3.2.1 Plantes

Les plantes sont les seuls objets avec une surface bien définie; la complexité de modélisation dérive de leur structure d'embranchement et de leur processus de croissance.

La modélisation et l'animation des plantes représentent un secteur intéressant et stimulant pour l'animation par ordinateur. Les plantes semblent exposer la complexité arbitraire en possédant une structure contrainte. Ils

grandissent d'un seul point de source, développant une structure d'embranchement en quelque temps tandis que les branches s'allongent. Les plantes ont été modelées en employant les L-systèmes.

3.3.2.1.1 L-systèmes

Les L-systemes ont été introduit par *A. Lindenmayer* qui a considéré le développement des plantes comme l'exécution d'un « programme de développement », représenté par des règles d'une grammaire de réécriture parallèle où celle ci possède la structure suivante [Péroche90, Péroche98] :

- Un alphabet.
- Un axiome de départ.
- Un ensemble de règles de réécriture.

L'interprétation graphique des mots générés par la grammaire engendre les parties élémentaires d'un arbre ou d'une plante (branche, feuille, …) ainsi que leurs propriétés géométriques telles que la taille des branches et des feuilles.

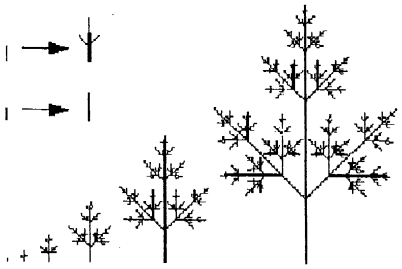


Figure 9 : Arbre désigné par un L-système. [Péroche98]

3.3.2.1.2 Simulation des déchirures

S. Lefebvre s'est intéressé essentiellement aux déchirures (écorce, pain) qui résultent d'un phénomène de croissance [Lefebvre01]. Il s'agit d'un habillage visuel couplant la géométrie et la texture, c'est pourquoi une approche phénoménologique est tout à fait appropriée. Une représentation en bandelettes parallèles aux efforts dus à la croissance, segmentées en zones d'écorce pleines ou fracturées, adjointe à un modèle de propagation de fractures, permet d'obtenir des surfaces convaincantes aux aspects modulables par l'utilisateur (taux et forme des fractures, voir emplacement), sans recourir à une simulation lourde.

Figure 10 : Exemple de déchirure. [Lefebvre01]

3.3.2.2 Liquides

3.3.2.2.1 L'animation interactive des vagues d'un océan

L'animation et l'affichage interactifs d'un océan illimité se fondent sur un modèle procédural de vagues. Ce modèle exprime les déplacements de points surfaciques comme des sommes de contributions de la forme de la vague active, modelée en tant que primitives d'animation indépendantes. Il permet de concentrer les calculs sur la surface d'océan et sur les longueurs des ondes qui sont actuellement visibles. La contribution principale de *Hinsinger* et *al.* [Hinsinger01] est un schéma pour l'adaptation dynamique de la résolution géométrique et de l'ensemble de primitives d'animation : l'échantillonnage de la surface est adapté au point de vue courant, alors que les trains de vagues qui ne correspondent pas aux fréquences évidentes sont filtrés.

Le modèle utilisé se base sur le modèle de bosse de *Gerstner* simulant des trochoïdes. Les trains de vague sont produits d'une manière dont il approche un spectre de vague connu. Ce modèle est consacré aux eaux profondes, et ne couvre pas ainsi la réfraction des vagues proche du rivage ni la rupture des vagues.

Concernant le rendu, *Hinsinger* et *al*. [Hinsinger01] appliquent une carte d'environnement sur la surface de l'océan, reflétant ainsi le soleil et le ciel. Le rendu et l'affichage sont ainsi exécutés à l'aide de matériel graphique.

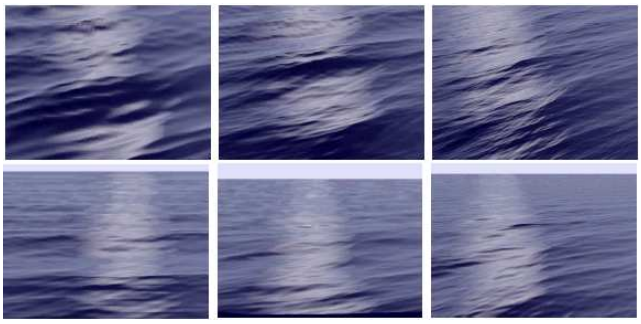


Figure 11 : A gauche : résolution de maille 50x50. Milieu : résolution de maille 100x100; A Droite : résolution de maille 500x500. [Hinsinger01]

Bilan

Un des avantages de cette approche est sa flexibilité : l'utilisation des trochoïdes permet de modeler un éventail de surfaces d'océan, de mers calmes aux orageuses. Puisque le déplacement d'un point témoin est calculé comme une somme de contributions de vagues, l'ajout d'effets supplémentaires tel que des vagues de bateau sont faciles. De plus, le rapport qualité/coût est réglable, ainsi des images de plus haute qualité peuvent être calculées en utilisant le même modèle.

3.3.2.2.2 Représentation de la lave

L'animation de la lave [Pierre99] coulant en pentes d'un volcan apporte plusieurs défis : la modélisation des particularités mécaniques de la lave[5] et le calcul, dans un temps raisonnable, des interactions à l'intérieur du flux, entre le flux et une base de données de terrain complexe; et finalement, rendant l'aspect visuel du flux.

Des flux de lave naturels présentent une large diversité de morphologies et de structures. Parmi d'autres facteurs, la topographie de terrain et le débit du cratère affectent la diffusion de la lave. Les études des particularités principales mécaniques de lave établissent que :

- La viscosité dépend de la composition chimique de la lave;
- Pour une composition chimique donnée, la viscosité δ s'augmente exponentiellement quand il y-a des diminutions de température.
- La densité massive de la lave reste constante avant qu'elle ne commute en solide.
- Le terrain est défini par un modèle de carte d'élévation digital (DEM), c'est-à-dire une grille 2D où les altitudes sont stockées. L'utilisation d'un DEM permet une détection plus facile de collision entre une particule et le terrain, puisqu'il exige seulement le calcul de l'altitude du terrain (par interpolation bilinéaire) lors de la projection de la particule sur le plan horizontal. En pratique, la détection de collision est seulement exécutée pour les particules qui sont à la surface du flux. Le calcul de ce sous-ensemble de particules est aussi nécessaire pour appliquer les équations de transfert de chaleur.

Les particules sont représentées par des sphères de rayon *ri*. Leur couleur varie avec la température. La modélisation des particules comme des sphères n'offre pas un rendu visuellement réaliste dont on a besoin.

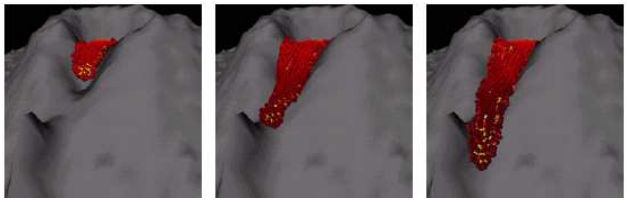


Figure 12 : Un exemple de flux de lave où les particules sont représentées par des sphères. Environ 3000 particules sont étendues du volcan dans la dernière image. [Pierre99]

[5] Comment elle se développe dans le temps selon la température.

Bilan

Les structures de données particulières proposées mènent à un temps linéaire en ce qui concerne le nombre de particules. Les avantages principaux du modèle de particules utilisé consistent en ce que l'utilisateur définit le comportement macroscopique du matériel par l'équation d'état. Un autre avantage de ce modèle consiste en ce qu'on est capable de contrôler la densité massive restée, dont la valeur est bien connue pour la lave.

3.3.2.3 Phénomènes gazeux

Le gaz n'a aucune définition géométrique ; sa modélisation, son rendu et son animation sont souvent en corrélation. En termes scientifiques, le gaz est d'habitude classé avec les liquides et leurs mouvements sont généralement mentionnés comme un calcul dynamique de fluides (CFD : *computational fluid dynamics*) [Parent02, Neyret97]. Le gaz est d'habitude traité comme *compressible*, ce qui signifie que la densité est variable dans l'espace et le changement de calcul de la densité fait partie du coût de calcul [Parent02].

3.3.2.3.1 Approches généraux de la modélisation du gaz

Il y a trois approches de la modélisation du gaz : les méthodes basées sur grille (formulations *Eulérienne*), les méthodes basées sur les particules (formulations *Lagrangienne*) et les méthodes hybrides. Les approches sont illustrées ici dans deux dimensions, mais l'extension à trois dimensions doit être évidente.

3.3.2.3.1.1 Méthode basée sur grille

Cette méthode décompose l'espace en cellules individuelles et le flux du gaz entrant et sortant de chaque cellule est calculée (Figure 13). De cette façon, la densité du gaz dans chaque cellule est mise à jour d'une manière périodique. La densité dans chaque cellule est employée pour déterminer la visibilité et l'illumination du gaz pendant le rendu. Les attributs du gaz dans une cellule, comme la vitesse, l'accélération et la densité peuvent être employés pour suivre la trace du gaz pendant qu'il voyage d'une cellule à une autre [Parent02].

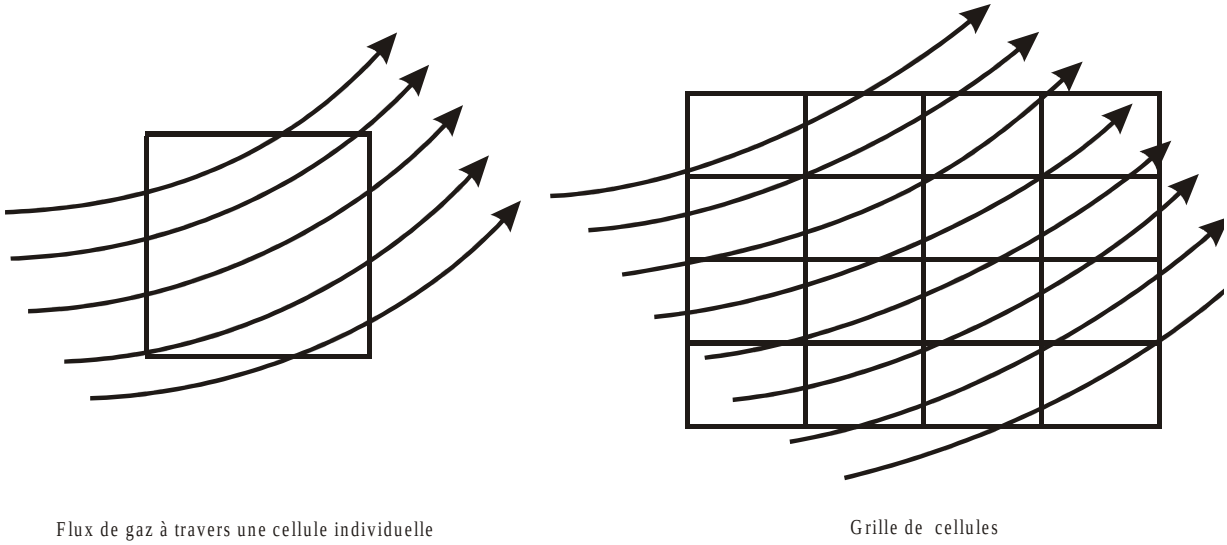


Figure 13 : Méthode basée sur grille. [Parent02]

Le flux sortant de la cellule peut être calculé en se basant sur la vélocité, la taille et la densité de la cellule. Le flux entrant dans une cellule est déterminé en distribuant la densité des cellules adjacentes. Des forces externes, comme le vent et des obstacles, sont employées pour modifier l'accélération des particules dans une cellule.

L'inconvénient de cette approche est le fait que l'utilisation d'une structure de données statique pour la décomposition cellulaire provoque des mesures pour l'initialisation des cellules nécessaires pendant le processus de la simulation.

3.3.2.3.1.2 Méthode basée sur les particules

Dans cette méthode, les particules sont suivies pendant qu'elles progressent dans l'espace. Les particules peuvent être rendues individuellement, ou elles peuvent être rendues comme des sphères de gaz avec une densité donnée [Parent02].

L'avantage de cette technique est qu'elle est semblable à la dynamique des corps rigides et donc les équations sont relativement simples et familières. Par contre, l'inconvénient de cette approche est qu'un grand nombre de particules est nécessaire pour simuler un gaz dense et expansif.

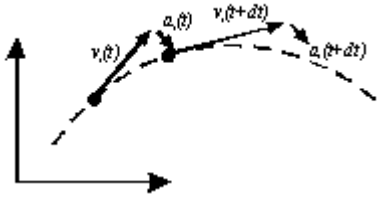


Figure 14 : Méthode basée sur les particules. [Parent02]

3.3.2.3.1.3 Méthode hybride

Quelques modèles de gaz lancent des particules à travers une grille spatiale. Les particules se déplacent d'une cellule vers une autre (Figure 15). Les attributs d'affichage de cellules individuelles sont déterminés par le nombre et le type de particules contenues dans la cellule au moment de l'affichage. Les particules sont employées pour porter et distribuer des attributs à travers la grille; ensuite la grille est employée pour produire l'affichage [Parent02].

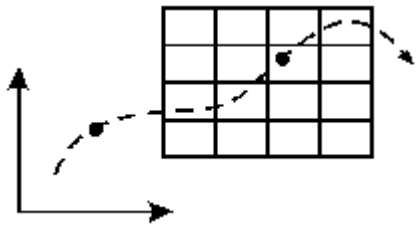


Figure 15 : Méthode hybride. [Parent02]

3.3.2.3.2 Représentation de nuages

La modélisation de nuages est une tâche très difficile en raison de leur complexité, vu leur amorphe, leur structure, leur du remplissage de l'espace. Dans ce cas il est facile de juger le réalisme d'un modèle de nuage.

Les premières approches ont employé des surfaces semi-transparentes pour produire les images de nuages convaincantes telle que [Parent02] :

- *R. Voss* a employé la synthèse du fractal des modèles de plans parallèles pour produire des images de nuages vus de loin.
- *Gardner* a utilisé la synthèse de *Fourier* pour contrôler la transparence des grands et creux ellipsoïdes. Cette approche emploie des groupes d'ellipsoïdes pour définir et animer la forme générale de nuages tout en employant la transparence de texture procédurale afin de produire l'aspect détaillé du nuage.
- Une autre approche semblable a été adoptée par *Kluyskens* pour produire des nuages dans des systèmes d'animation. Il emploie les opérations aléatoires et des recouvrements sur des sphères pour définir la forme générale du nuage. Une texture solide de nuage est alors employée pour colorer le nuage et contrôler la transparence des sphères. Finalement, *Kluyskens* augmente la transparence des sphères près de leurs bords, de sorte que la définition de forme ne soit pas apparente.

Bien que des techniques basées surface puissent produire des images réalistes de nuages vus d'une distance, ces modèles de nuage ne permettent pas à l'utilisateur de voyager à travers et d'inspecter leur intérieur. *Kajiya* a produit le premier modèle volumique du nuage dans l'infographie [Parent02].

En 1997, *Neyret* a produit quelques résultats préliminaires d'un modèle de nuage basé sur des caractéristiques physiques générales, telles que des processus de convection et de bullage [Neyret97]. Ce modèle semble prometteur pour simuler les nuages convectifs. Cependant, il a utilisé des surfaces (de grandes particules) pour modeler la structure de nuage. En 2000, *Neyret* [Neyret00] a proposé un modèle de rendu analytico-phénoménologique, permettant de construire très rapidement des images de haute qualité visuelle (effets et résolution). Il s'agit d'exploiter toutes les connaissances à priori disponibles sur les nuages convectifs[6].

[6] Ils sont très denses, d'albedo très proche de un, la zone de transition entre intérieur et extérieur et très fine, la surface est composée de forme sphérique.

Les systèmes de particule sont généralement employés pour simuler les gaz volumiques, comme la fumée, avec des résultats convaincants et fournissent un contrôle facile de l'animation. La difficulté d'utilisation d'un système de particule est le nombre massif des particules nécessaires pour simuler les nuages réalistes [Parent02].

Plusieurs auteurs ont employé l'idée de fonctions implicites rendues par volume pour la modélisation du nuage volumique [Parent02] :

- *Nishita, Nakamae* et *Dobashi* sont concentré sur des effets d'illumination. Ils ont employé ces fonctions comme un modèle de base dans leur travail.
- *Stam* a aussi employé les formes volumiques pour créer ces modèles de fumée et des nuages.
- *Ebert* a employé les volumes implicites combinés avec les systèmes de particule pour simuler la formation et la géométrie des nuages volumiques.

Les volumes implicites sont contrôlés par un système de particule modifié qui incorpore les simulations dynamiques de formation de nuage.

Malgré la complexité des processus physiques qui forment des nuages, la plupart de leurs aspects visuels importants ont été efficacement modelés par les chercheurs. Cependant, il reste des défis en termes de contrôle de mouvement de nuage ainsi que le rendu.

3.3.2.4 Représentation du feu

Le feu est un processus intensif et difficile à modéliser. Il possède toutes les complexités de fumée, des nuages, en plus de la complexité de changement des processus actifs internes produisant la lumière, le mouvement et la création des attributs d'affichage qui varient rapidement [Parent02].

3.3.2.4.1 Approche du système de particule

Cette approche emploie une hiérarchie de particules à deux niveaux. Le premier niveau de particules est placé au point d'impact pour simuler la détonation initiale. Le deuxième consiste en anneaux concentriques de particules, chronométrés pour progresser du point central vers l'extérieur, formant le mur de feu et des explosions [Parent02].

Chaque anneau de la hiérarchie de deuxième niveau se compose d'un nombre de systèmes de particules individuels qui sont placés en chevauchement sur l'anneau afin de former un anneau continu. Les systèmes de particule individuels sont modelés pour ressembler aux explosions (Figure 16). La position initiale pour une particule est aléatoirement choisie de la base circulaire du système de particule. La direction initiale de voyage de chaque particule est contrainte à dévier le moins possible l'angle d'éjection loin de la normale de surface.

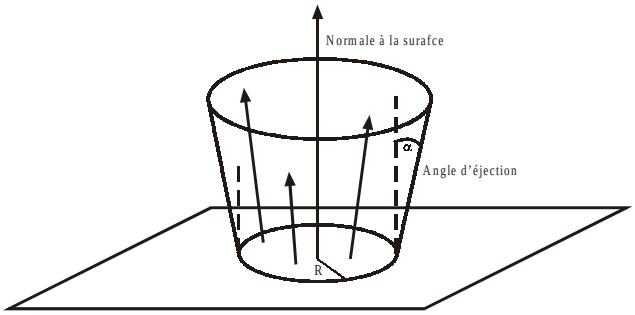


Figure 16 : Exemple de génération de feu. [Parent02]

3.3.2.4.2 D'autres approches

Plusieurs approches ont été employées avec des niveaux de succès variables [Parent02] :

- Des cartes de textures bidimensionnelles animées ont été employées pour créer l'effet du mouvement ascendant du gaz brûlant, mais de tels modèles sont seulement efficaces une fois vue d'une direction spécifique. L'utilisation d'une approche bidimensionnelle de plan multiple ajoute quelque profondeur au feu, mais les directions de vision sont encore limitées.
- *Stam* et *Fiume* présentent un système d'équation pour évaluer la densité et la température des champs. L'utilisateur contrôle la simulation en spécifiant un champ de vent. Les résultats sont efficaces, mais les mathématiques de base utilisées sont compliquées et il est difficile de contrôler le modèle.

Limites de l'approche

La modélisation et l'animation de la plupart des phénomènes naturels sont stimulantes. De plus, la plupart des techniques sont toujours le sujet d'efforts de recherche où on constate que les inconvénients majeurs de cette approche sont les suivants :

- On exploite les cas particuliers plutôt que de traiter le cas général, ce qui occasionne une simulation lourde.
- Il faut établir des ponts avec d'autres disciplines, pour pouvoir trouver ensuite une méthode applicable à un phénomène naturel particulier.
- Où lieu de développer une nouvelle représentation fonctionnelle, crédible et efficace, on suit une méthode standard basée sur un traitement atomique qui s'avère plus simple.

A cet effet, l'avenir et sans doute aux méthodes hybrides où la grande échelle est confiée aux modèles physiques et où des représentations complémentaires se chargent des petites échelles [Neyret01].

3.4 Les représentations alternatives

L'idée sous-jacente est de construire des représentations géométriques minimales selon un critère purement visuel. Il s'agit de recourir aussi bien à des modèles géométriques (facettes) qu'aux texturels, photométrique pour représenter les formes à diverses échelles.

L'objectif est d'améliorer l'efficacité (coût de rendu), la qualité. La représentation alternative se divise en 3 parties :

- Codage direct de comportement lumineux.
- Codage texturel.
- Codage volumique.

3.4.1 Codage direct du comportement lumineux

3.4.1.1 Les systèmes de particules

Reeves [Péroche98, Parent02] a introduit la modélisation par particules pour représenter les objets aux formes mouvantes : flammes, fumées, feux d'artifices, torrents, cascades, nuages, champs ou végétation sous le vent.

Figure 17 : Un exemple de paysage généré par des systèmes de particules. [Meyer01b]

L'idée initiale est de représenter des géométries complexes ou difficilement représentables avec de très grands nombres de particules : une particule est une primitive géométrique classique simple (point, polygone, sphère, …etc.), munie d'attributs tels qu'une position, une couleur, une transparence, une vitesse, une taille, une forme, une durée de vie.

Les particules évoluent dans l'espace en subissant différentes influences (gravité, lois de croissance, etc.) et forment ainsi des trajectoires. Il n'y a pas d'interaction entre les particules dans le modèle initial.

Reeves et *Blau* présentent en 1985 des applications de systèmes de particules pour la modélisation et le rendu d'éléments naturels : arbres, prairies, …etc. L'une des nouveautés est que les particules peuvent interagir entre elles (par exemple, collisions) ; donc la forme et la topologie des objets ainsi engendrées émergent du comportement d'ensemble des particules, et ne sont pas connus du logiciel [Parent02].

K.Sims a utilisé ces systèmes de particules pour réaliser des animations très réussies (de cascades par exemple). Par ailleurs une notion de particule orientée a été introduite pour modéliser des objets déformables ou réaliser des animations [Parent02].

Outre la spécificité de son champ d'application, une des limites de cette technique est son incapacité à modéliser une structure pour une espèce d'arbre donnée. En outre, la spécificité de son algorithme de rendu, qui dessine les particules comme des traits de crayon, diffère totalement de la représentation classique et rend assez difficile son intégration à une scène existante.

3.4.1.2 Rendu à base de points

M. Levoy et *T. Whitted* ont introduit, en 1985, une nouvelle représentation hybride située entre les représentations à base d'images et les représentations polygonales [Meyer01b]; dont l'idée est d'utiliser le point comme des particules pour rendre un objet solide, dans son article nommé « The use of points as display primitive » ils présentent les problèmes fondamentaux comme la reconstruction de la surface, la visibilité et le rendu de surfaces semi-transparentes. Mais ce n'est qu'en 1998 que *J.P. Grossman* et *Dally* reprirent cette idée et formalisèrent pour la première fois l'utilisation de points comme primitive de rendu.

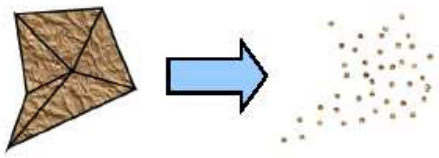


Figure 18 : Maillage triangulaire (avec texture) vers un nuage de point (échantillonnage).

Les objets sont représentés par un ensemble d'échantillons ponctuels appartenant à leur surface. En fait, ces échantillons sont communément appelés surfels pour *« élément de surface »* (*« surface element »* en anglais). Ce terme a été introduit mathématiquement par *Herman* en 1992, mais *Pfister* et *al.* [Pfister00] proposent une nouvelle définition plus adaptée. D'après *Pfister*, un surfel est un n-tuple de dimension 0 avec des attributs de forme et de matière qui approxime localement la surface d'un objet. De plus, les surfels ne contiennent aucune connectivité explicite avec leurs voisins, ce qui en fait une représentation très souple à manipuler. Le coût d'affichage d'un objet est proportionnel à sa taille à l'écran et non à sa complexité intrinsèque. Ce qui permet de conserver un taux de rafraîchissement de l'image constant, même avec beaucoup d'objets affichés.

Tandis que *Pfister* convertit les objets polygonaux en *surfels* en pré-traitement, *Stamminger* [Stamminger01] échantillonne à la volée des objets procéduraux, ce qui offre une plus grande souplesse dans le choix de la précision. Il montre aussi que cette technique est bien adaptée au rendu de phénomènes naturels variés comme une montagne, le mouvement de l'eau ou le rendu d'arbres.

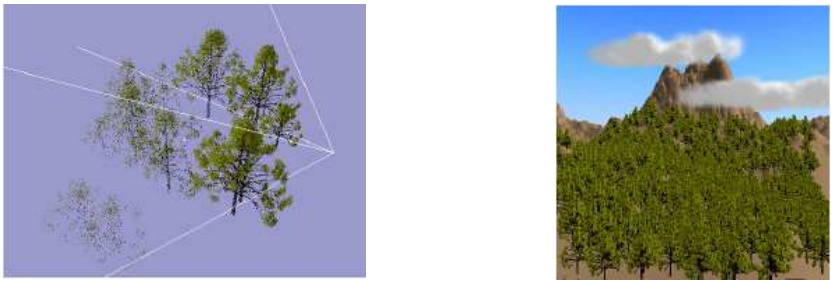


Figure 19 : À gauche : des arbres représentés par des points. Plus on s'éloigne de la caméra moins il y a de points affichés. À droite : un exemple de paysage rendu avec la technique des points. [Meyer01b]

Il existe deux manières d'aborder la reconstruction de la surface. Une première approche est de travailler dans l'espace image [Pauly01, Baar01]. Avec ce type d'algorithme, il est possible de prendre en compte des surfaces semi-transparentes (utilisation d'un A-buffer), par contre seule une implémentation logicielle est actuellement possible. Pour bénéficier d'une accélération par le matériel graphique il est nécessaire d'exprimer cette reconstruction dans l'espace objet [Rusinkiewicz00, Kalaiah01]. Cependant, aucune des techniques précédentes ne supporte un anti-aliassage pour les modèles ayant une texture complexe. Pour répondre à ce manque, *Pfister* et al. reprennent le concept du filtrage pondéré elliptique de *Heckbert* nommé EWA (Elliptical Weighted Average), et l'appliquent au *splatting* de surface en formulant les noyaux de reconstruction dans l'espace image [Baar01]. Récemment *Liu Ren* et *al.* [Ren02] parviennent à exprimer ces noyaux dans l'espace objet et tirent ainsi bénéfice d'une implémentation possible avec OpenGL.

Cette technique est jeune et comporte certains défauts : texturation limitée, objets transparents et semi-transparents difficilement représentables, le calcul par moyenne des normales lorsqu'on simplifie le modèle est contestable, etc. Néanmoins l'idée semble très prometteuse, pour preuve la quantité de travaux en cours sur ce sujet.

3.4.1.3 Shaders

Le *shader* est un modèle d'illumination analytique et hiérarchique destiné à représenter la nature microscopique d'un objet ou les phénomènes sous–pixel lors de rendu : Le détail d'un ensemble de primitives (par exemple : les feuilles d'un arbre, la fourrure d'un animal) n'est pas visible explicitement lorsque l'observateur est loin (*i.e.* lorsque l'ensemble des primitives ne recouvrent que quelques dizaines de pixels de l'image ou moins). Seule la forme et la couleur d'ensemble sont alors distinguées.

L'idée de *shader* est de représenter ce groupe de primitives par sa forme, associée à une formule analytique décrivant son comportement photométrique et son opacité globale (Figure 20), c'est-à-dire par un modèle d'illumination (*shader*). Ce *shader* est obtenu en intégrant un modèle d'illumination simple (*par exemple :* celui de *Phong*) sur l'ensemble des primitives du groupe en tenant compte de la visibilité (ceci se traduit par une restriction du domaine d'intégration), et en calculant leur opacité moyenne [Meyer01b].

Cette idée est présente dans beaucoup de modèles d'illumination de surface ou par exemple le modèle de *Goldman* ou celui de *Kajiya* pour la représentation de fourrure puisqu'il intègre le modèle d'illumination de *Phong* sur un cylindre.

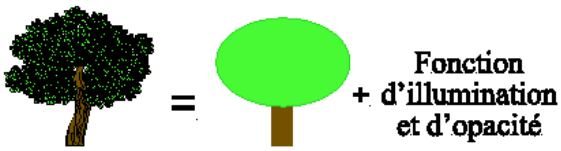


Figure 20 : Un objet vu de loin peut être représenté par sa forme et son modèle de shader. [Meyer01b]

Figure 21 : Un chien rendu par le shader de Goldman. [Debunne04]

Alexandre Meyer a introduit, en 1998, un pseudo *shader* 3D dont l'objectif est d'exploiter au mieux la connaissance à priori disponible pour certaines familles d'objets afin de calculer analytiquement une bonne approximation de l'intégrale de l'illumination dans la région de l'espace visible à travers un pixel [Meyer00, Meyer01a]. Ces travaux portent sur les forêts de pins, dans la mesure où la connaissance à priori sur la distribution des aiguilles est très forte. Il dérive une hiérarchie de trois *shaders* intégrant l'illumination (y compris les ombres et la transparence résiduelle) à l'échelle d'une aiguille, d'un cône d'aiguilles, et de toute une touffe. L'implémentation actuelle (peu optimisée), tourne environ 8 fois plus vite que *Rayshade* (un logiciel de rendu par ray-tracing), pour calculer l'image d'une forêt de sapins sans artefacts visuels.

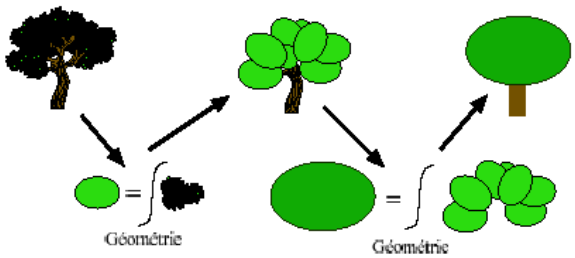


Figure 22 : Principe d'une hiérarchie de shaders analytique dans le cas d'un arbre. [Meyer01b]

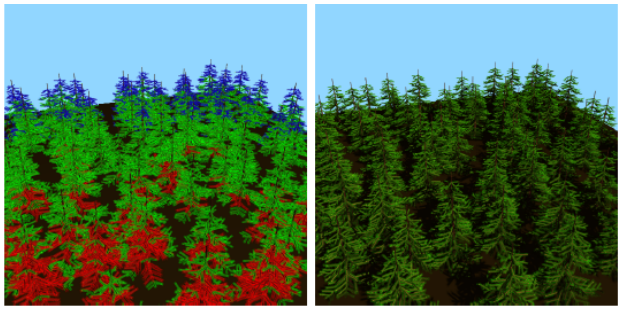


Figure 23 : À gauche : Les trois niveaux de détails (niveau un en rouge, niveau deux en vert et niveau trois en bleu). À droite : 80 sapins. [Meyer01b]

Gardner a proposé, en 1985, un autre pseudo *shader* pour synthétiser la complexité des surfaces naturelles. Il utilise une sommation d'un certain nombre de sinusoïdes d'amplitudes et de fréquences variées [Péroche98].

Cette méthode a permis de visualiser de manière efficace des arbres, des montagnes et plus particulièrement différents types de nuages.

3.4.1.4 Bump mapping

Le *bump Mapping*, appelé également *Embossing*, est une technique qui renforce le réalisme des images, des textures et des objets dans un environnement 3D, en donnant l'illusion de profondeur sur des surfaces apparemment lisses, grâce à des effets de relief.

Dans le monde réel, l'impression de profondeur est donnée par la quantité de lumière réfléchie par la surface de l'objet et perçue par l'œil : la quantité de lumière réfractée par l'objet et la direction qu'elle prend sont déterminées par le type de surface, lisse ou rugueuse, plate ou saillante, sur laquelle la lumière se reflète. Plus la direction de la lumière est perpendiculaire à l'objet, plus cette lumière va s'y refléter. Comparativement, moins la direction de la lumière est perpendiculaire à la surface, moins la lumière est reflétée.

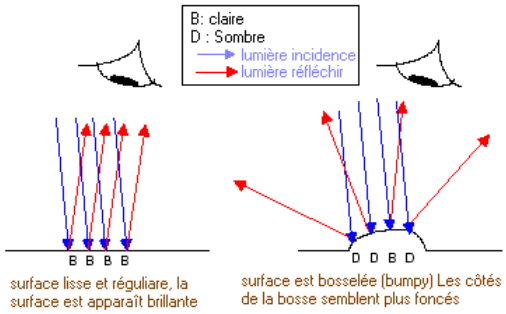


Figure 24 : Réflexion de la lumière sur des surfaces plates et bosselées.

Perlin, en 1985, propose de généraliser la notion de perturbation de la normale (Bump Mapping) aux textures 3D en définissant une fonction appelée « *Dnoise* » analogue à la fonction de bruit qui, au lieu de retourner un scalaire, retourne un vecteur pseudo-aléatoire. Ce vecteur sert à perturber la normale en (x, y, z) par simple addition au vecteur normal, N, en (x, y, z) [Péroche98]:

$$N' = N + Dnoise(x, y, z).$$

3.4.2 Codage texturel

3.4.2.1 Rendu à base d'images

Le coût (en temps et en calcul) des étapes de modélisation et de rendu ainsi que le faible réalisme des images produites ont encouragé le développement de techniques basées sur l'utilisation d'images de référence (photos, séquences vidéos,...) de la scène que l'on cherche à représenter. Ces nouvelles techniques de modélisation et de rendu, basé sur l'image (IBMR), ont pour but de recréer (ou de simuler) un environnement uniquement à partir de photos (séquences vidéos) prises dans cet environnement.

Les techniques de rendu basé sur des images ont été introduites pour essayer d'éviter l'étape de modélisation, ainsi que pour tenter d'accélérer et d'améliorer le rendu en utilisant l'information contenue dans des images capturées du monde réel. Essentiellement, toutes ces techniques se basent sur un échantillonnage de la lumière dans

une scène soit par voie d'images, ou encore à partir d'un échantillonnage dense du champ plénoptique ou de lumière [Guillou00].

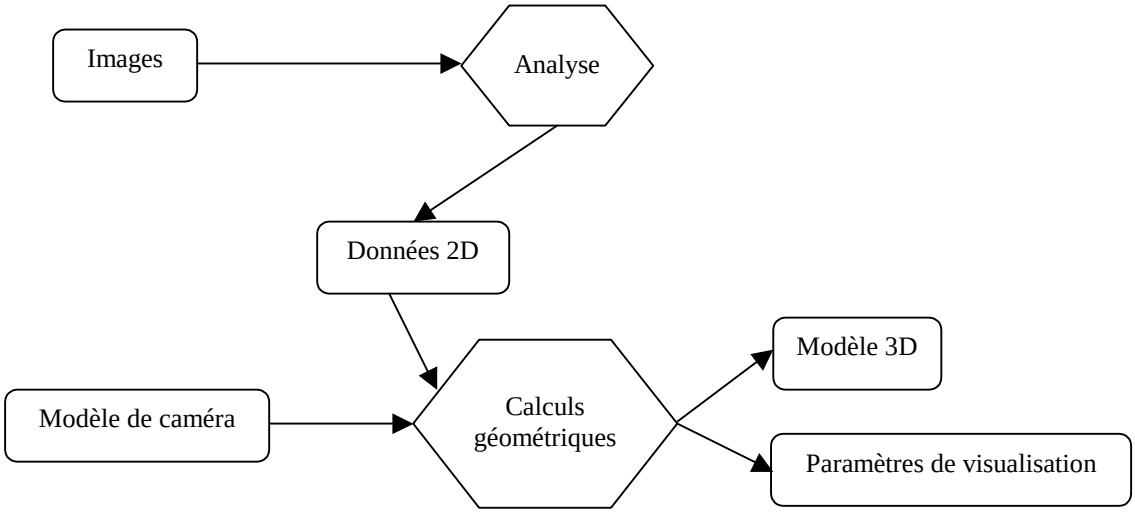


Figure 25 : Approche traditionnelle de la vision par ordinateur. [Guillou00]

La technique de rendu à base d'images la plus fréquemment utilisée en synthèse d'images est le plaquage de textures où la complexité apparente d'un objet est représentée par l'application d'une image sur sa surface. La forme grossière de l'objet est représentée par sa description géométrique, les détails étant décrits par les informations dérivées de l'image de la texture qui lui est appliquée. Plus récemment, des variantes plus intelligentes du plaquage de textures ont été développées, en particulier le plaquage de textures projectives permettant à un ensemble de primitives 3D de partager une même texture définie pour un point de vue particulier. Dans une certaine mesure, ce principe peut être étendu à l'utilisation d'une image panoramique unique pour représenter l'environnement. C'est le cas dans la technologie *Quicktime-VR* développée par Apple. Ce type de représentation ne permet qu'un faible nombre de mouvements différents, rotations autour de la position de l'observateur et changements de focale [Guillou00].

Des représentations alternatives de la scène peuvent être utilisées, en ajoutant aux images des informations supplémentaires telles que la profondeur ou le flot optique (décrivant le déplacement apparent des pixels entre deux images). L'utilisation de techniques de re-projection d'images permet alors d'obtenir de nouvelles vues en tenant compte des effets de parallaxe (dus à la projection perspective) et des occlusions entre les différents objets de la scène. La difficulté majeure des techniques précédentes concerne l'estimation des données additionnelles (profondeur, flot optique).

Lors de la re-projection des images, il se peut que des régions de la scène qui n'étaient pas visibles, le deviennent. Pour pallier ce problème, une méthode consiste à combiner les informations colorimétriques et géométriques de plusieurs images dans une seule image à plusieurs niveaux de profondeurs (appelée LDI). Pour chaque pixel de la LDI, les informations de couleur et de profondeur de chaque point 3D se projetant sur ce pixel sont conservées. Une autre alternative consiste à utiliser une image à plusieurs centres de projection [Guillou00].

Une alternative consiste à considérer un ensemble d'images comme une base de données de rayons traversant la scène. Pour calculer de nouvelles vues, chaque pixel est colorié en interrogeant la base de donnée par rapport au rayon correspondant à ce pixel (les rayons non présents dans la base de données étant interpolés). L'avantage de ces méthodes est qu'elles ne nécessitent qu'une interprétation minimale des données, mais supposent avoir accès à un grand nombre d'images [Guillou00].

Les différentes méthodes existantes sont :

- *Création de panoramas cylindriques.*
- *Quicktime VR.*
- *Modélisation Plénoptique.*
- *Lightfield, Lumigraph*.
- *Layer Depth Images* (LDI).

Bilan

Ces nouvelles techniques de rendu offrent l'avantage où la complexité des modèles est sans limite, et le temps de rendu est indépendant de la complexité (il dépend plutôt de la taille de l'image à produire).

Bien qu'en général moins précises du fait de la nature discrète des images utilisées, ces techniques offrent aussi l'avantage d'être assez rapide (temps constant), pour faire des manipulations sur les pixels d'images afin obtenir des effets tridimensionnels. Par Conséquent, un attrait du rendu basé sur des images repose dans la possibilité de faire des rendus à des vitesses interactives, en considérant par contre un coût important en terme d'utilisation de la mémoire [Blais98].

3.4.2.2 Les nuages de panneaux d'affichage (billborad clouds)

Un *nuage de panneaux d'affichage* est un ensemble de polygones (ou panneaux) texturés, partiellement transparents, avec des tailles, orientations et résolutions de texture indépendantes (Figure 26). Typiquement, les *billboards* sont employés pour représenter des objets tels que des nuages ou des arbres, très difficiles à modéliser explicitement [Décoret02, Porquet04].

Un tel nuage peut être employé pour accélérer l'exposition d'un modèle complexe, conduire des contrôles de collision, exécuter des simulations de transfert radiatives en utilisant des techniques de radiosité, ou dans n'importe quelle situation où la description d'un modèle comme un ensemble de primitives plates est utile [Décoret02].

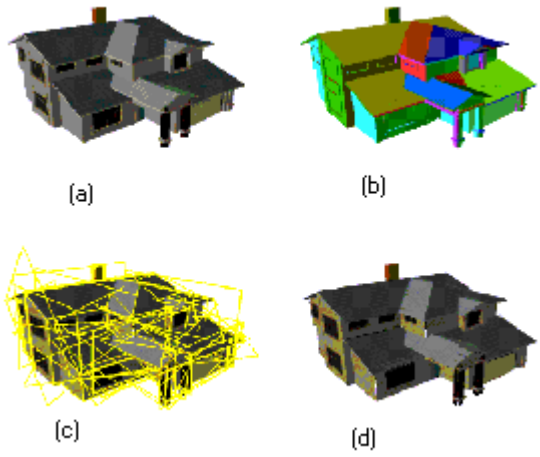


Figure 26 : Exemple d'un nuage de panneaux : (a) modèle original (1,960 polygones) (b) rendu faux coloré en utilisant une couleur par panneau pour montrer les faces qui ont été groupés sur celui-ci (c) Vue (automatiquement produite) de 52 panneaux texturés, avec mise en évidence de leur frontière (d) nuage de panneaux rendu. [Décoret02]

Le principe de la méthode de *Décoret* et al. [Décoret02, Porquet04] est de remplacer un objet par la combinaison de plusieurs *billboards* représentant cet objet sous différents angles. Leur méthode permet de répartir automatiquement un certain nombre (fixé par l'utilisateur) de plans recouvrant l'objet de façon optimale. Cet ensemble de *billboards* est simplement affiché lors du rendu, les plans se superposant et se combinant à l'aide de la composante de transparence des textures.

Beaucoup de techniques ont employé la notion de polygones multiples, partiellement transparents pour accélérer l'affichage de modèles complexes. En effet, ces techniques peuvent être vues comme des cas spéciaux des nuages de panneaux (les panneaux classiques, les couches d'imposteurs où tous les panneaux sont parallèles, les caches d'image où un panneau est associé à chaque région d'une partition spatiale, …etc.) [Décoret02].

Bilan

L'avantage principal de nuages de panneaux [Décoret02, Porquet04] consiste en ce qu'aucune information topologique (comme la connectivité de polygone) n'est exigée. De plus la complexité visuelle de nuage de panneaux résulte principalement de la texture, et aucune description complexe de frontière n'est nécessaire.

En travaillant dans l'espace supportant les plans, l'algorithme gagne beaucoup de flexibilité comparée aux techniques de simplification géométriques qui fonctionnent dans l'espace 3D régulier. Les nuages de panneaux sont fortement efficaces dans la simplification de modèles complexes avec des textures multiples vers quelques dizaines

de polygones texturés. La représentation de nuage de panneaux permet alors un rendu très rapide, ou un test de collision efficace. La qualité visuelle des modèles simplifiés est très haute et contrôlée par l'utilisateur.

Le problème principal de cette méthode est que l'éclairement ne peut varier, du fait de l'emploi de placage de texture dépendant du point de vue.

3.4.2.3 Couches d'imposteurs

Les couches d'imposteurs sont une généralisation d'imposteurs dynamiques de *Schaufler* [Porquet04] dans le sens suivant : un imposteur remplace un objet par un polygone transparent sur lequel l'image opaque de l'objet est plaquée. Quand le point de vue se déplace, la profondeur de l'objet est omise et la forme plate du polygone devient apparente. Un imposteur en couches consiste en beaucoup de tels polygones transparents. Sur chaque polygone tout les texels dessinés montrent les parties de la surface de l'objet qui sont à une distance semblable de l'observateur à un polygone. Autrement dit, chaque couche dans l'imposteur dépeint une tranche de l'objet à une distance appropriée à l'observateur (Figure 27). [Schaufler98]

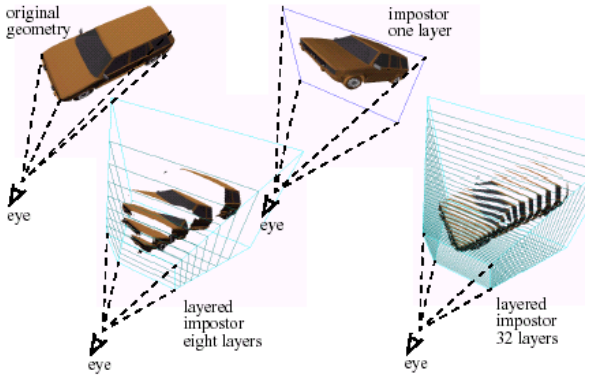


Figure 27 : Remplacement d'un objet (haut gauche) avec imposteur (haut droit) ou imposteur feuilleté (bas). [Schaufler98]

Quand l'image de l'objet est produite, au lieu du renoncement aux contenus de tampon de profondeur (l'information décrivant la forme tridimensionnelle de l'objet) les contenus de tampon de profondeur sont conservés et stockés ensemble avec l'information de couleur pour chaque pixel. De cette information RGBz une texture est définie dans le format RGBα (avec z incorporé dans la composante α) disponible sur les stations de travail graphiques d'aujourd'hui [Schaufler98].

Dans l'étape de rendu, plusieurs couches polygonales seront rendues comme une pile pyramidale ayant le point de vue l'apex (Figure 28). Si des intervalles de profondeur disjoints sont dessinés sur chaque couche, des résultats de trous étant visibles entre les couches pour chaque déviation du point de génération de texture. Ces trous sont évités en dessinant légèrement des intervalles de profondeur chevauchés sur chaque couche (Figure 28) [Schaufler98].

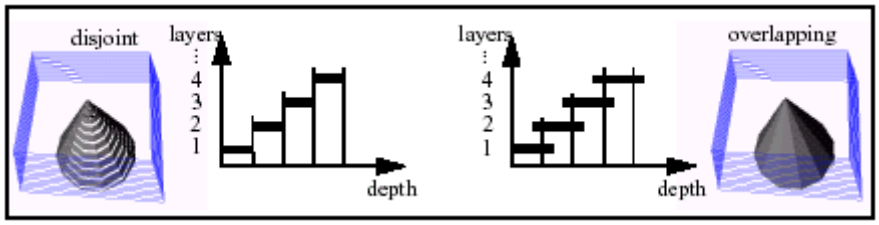


Figure 28 : Couches d'imposteur pour un cône approximé : des intervalles de profondeur recouvrant sont assignés à chaque couche pour éviter l'apparition des trous entre couches quand on déplace le point de vue. [Schaufler98]

3.4.2.4 Textures volumiques temps réel

Meyer et *Neyret* [Neyret98b] ont introduit une méthode permettant le rendu interactif de scènes complexes et répétitives. L'idée est d'adapter les textures volumiques introduites par *Neyret* [Neyret96] au z-buffer où contrairement au rendu volumique, la quantité de donnée volumique traitée est petite. Par exemple un volume de 64 tranches peut être rendu avec 64 faces texturées alors que le même modèle représenté avec des polygones en contiendrait plusieurs milliers.

Cette approche consiste à couper en tranches un morceau de géométrie 3D (un détail répétitif de l'objet complexe). Une tranche est un rectangle qui contient la géométrie de cette zone avec son illumination. Ces tranches

sont utilisées comme des textures transparentes, qui sont plaqués sur la surface sous-jacente (par exemple, une colline ou la peau d'un animal) avec un offset d'extrusion.

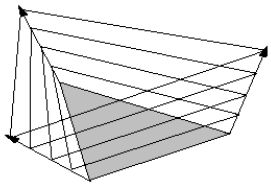


Figure 29 : Un texel est dessiné en rendant une superposition de faces texturées. [Neyret98b]

A la manière des textures volumiques utilisées en lancer de rayons [Neyret96] la spécification d'une surface est une triangulation contenant à chaque sommet des coordonnées textures (u, v) et un vecteur définissant la hauteur et la direction de la texture 3D.

La partie volumique de ce modèle est différente de celle utilisée en lancer de rayon : chaque voxel du volume de donnée contient quatre composantes (RGBA où A code la transparence, 0 pour transparent et 255 pour entièrement opaque), tous les voxels d'une même tranche forment ainsi une texture 2D.

Le rendu s'effectue en utilisant une carte graphique standard comportant des fonctionnalités 3D (OpenGL par exemple). On peut utilise le *mip-map* implanté en hardware pour supprimer l'effet d'aliassage quand la surface est lointaine ou bien que l'angle de vue soit rasant. Il est à noter que l'utilisation de texture, de la transparence et du *mip-map* ne coûte pratiquement que le rendu d'un polygone « nu » si ces fonctionnalités sont câblées sur la carte graphique. L'utilisation de la transparence et du z-buffer pose certains problèmes si on affiche les polygones dans un ordre quelconque. Lorsque l'on rend un pixel dont la transparence est semi-opaque on prend en compte les pixels se trouvant derrière seulement s'ils ont été rendus auparavant. Une solution pour résoudre ce problème est de rendre les tranches de l'arrière vers l'avant (un trie selon les *z* des texels).

Une fois que l'utilisateur a choisi un modèle pour représenter leurs détails, il passe à la construction du motif de référence pour que le modèle reproduise le même effet visuel que la représentation polygonale. Le procédé sera de la façon suivante :

- Couper en tranches le modèle 3D.
- Evaluer l'éclairement en chaque point.
- Remplir l'intérieur de l'objet.

Bilan

Comparé à la version utilisant le lancer de rayons, la qualité de rendu est inférieure (l'éclairage et les ombres sont précalculés). Mais comparé à la pauvreté visuelle des scènes existantes en réalité virtuelle; ce modèle a grandement augmenté la qualité des détails. D'un autre coté l'utilisation des processeurs graphiques apporte des limitations : l'éclairage et les ombres sont précalculés, ils ne seront pas remis à jour si la source de lumière se déplace.

3.4.2.5 Textures procédurales

Les textures procédurales ont été proposées par *Ken Perlin* en 1985. Celles-ci se basent sur l'idée où la plupart des textures naturelles se caractérisent avant tout par une certaine irrégularité et une grande quantité de détails distribuée plus ou moins aléatoirement. De cet effet, *Perlin* a introduit la notion de *bruit volumique* (*solide noise*) afin de générer des textures 3D d'apparence naturelle comme le marbre, nuage, bois.

Perlin et *peachey*, ont utilisé le *bruit*[7] de la manière suivante : la texture est décrite dans un premier temps par un modèle mathématique simplifié, appelé fonction de base ou modèle de base ; ensuite cette fonction de base est perturbée par un bruit ou une autre procédure stochastique pour lui donner un aspect plus irrégulier et plus naturel.

Perlin a introduit un autre type de fonction stochastique, nommée *turbulence* qui sert à perturber le modèle de base et elle est définie par une somme de bruit sur différents niveaux de fréquences (octaves) et d'amplitudes [Péroche98].

Le temps de calcul nécessaire pour générer une texture procédurale est grand, ce qui ne nous permet pas d'avoir un rendu des scènes animées en temps réel, à cet effet, *Antoine Miné* [Antoine01] a introduit, en 2001, « les textures procédurales en temps réel », pour cela il utilise la bibliothèque de rendu OpenGL de *Silicon Graphics*.

[7] Une fonction qui pour tout point de R^3, retourne une valeur pseudo aléatoire comprise entre –1 et 1.

L'inconvénient majeur de cette méthode est le choix des paramètres de texture ; ces derniers sont choisis par expérience pour obtenir un type de texture particulier.

En 1996, *Worley* a introduit les *textures cellulaires*, permettant de décrire des minéraux en cristaux, des peaux à écaille, des palmes et de simuler des surfaces ayant un aspect externe organique. Le principe de base consiste à générer des échantillons d'éléments géométriques sur une surface tout en simulant le développement de cellules biologiques. La surface où les différents échantillons sont posés, peut être définie par une fonction implicite et une base de données de volume ou un polygone.

Les *textures cellulaires* sont obtenues par l'interaction de différents éléments appelés cellules 3D liées à une surface et forme donc une texture 3D géométrique, orientée et colorée [Kenneth97]. Ce type de texture, peut être combiné avec d'autres systèmes tel que le système de particules ou les systèmes à réaction-diffusion, afin de former un seul modèle.

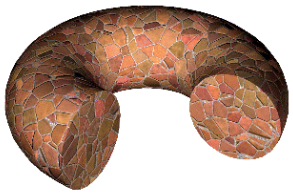


Figure 30 : Exemple de texture cellulaire. [Monks99]

Le modèle de texture par *réaction - diffusion* décrit une méthode biologique pour la génération de texture tout en gardant l'aspect géométrique des surfaces.

Réaction – Diffusion (RD) est un processus dans lequel deux produits chimiques ou plus se diffusent sur la surface d'un objet et réagissent entre eux pour former un échantillon stable [Péroche98].

Ce modèle est proposé pour la simulation de processus d'interaction locale non linéaire générant des échantillons biologiques. Ces derniers compensent les effets non uniformes de la paramétrisation des surfaces, ils sont dirigés par deux aspects concurrents [Péroche98] :

- Diffusion d'éléments à travers les tissus (la surface).
- Réaction de ces éléments à l'excitation ou à la négligence des autres éléments existant dans le même tissu. Le résultat de la réaction peut être une production ou une destruction de ces éléments.

Figure 31 : Exemple de texture Réaction-Diffusion. [Debunne04]

3.4.3 Codage volumique

Les textures volumiques ont été introduites dans les années quatre-vingt par *Kajiya et Kay* [Neyret96]. En premier lieu ce modèle était dédié à la construction des fourrures puis il s'est développé au fur et à mesure jusqu'à devenir un modèle plus général incluant la notion multi-échelle.

3.4.3.1 La texture volumique de KAJIYA et KAY

Introduite en 1989, cette texture a généré de l'intérêt, du fait qu'elle capture la complexité des surfaces en réduisant l'aliassage d'une façon importante. *Kajiya et Kay* ont introduit essentiellement cette texture pour proposer un modèle de fourrure [Neyret96].

Le principe consiste à fabriquer un échantillon de texture volumique; en l'occurrence un échantillon de fourrure, lequel est stocké en un seul exemplaire dans un *volume de référence*, dont les copies déformées, appelées « *texels* », seront plaquées sur une surface composée de patchs bilinéaires, constituant une peau épaisse continue à la surface d'un objet.

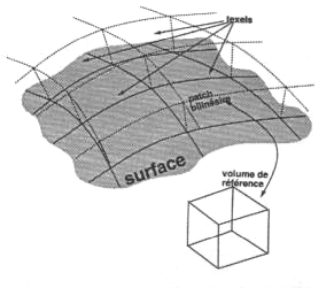


Figure 32 : Texel sur une surface. [Neyret96]

Le volume de référence comporte un ensemble d'éléments de volumes appelés « voxels ». Le contenu de chaque voxel est une probabilité d'occultation isotrope, plus un comportement photométrique local, qui consiste en un repère local et une fonction de réflectance analytique, permettant au texel de refléter la lumière comme s'il contenait vraiment un morceau de surface.

L'implémentation proposée pour le rendu de fourrure modélise la réflectance d'un poil par celle d'un cylindre. La fonction de réflectance correspond à une approximation de l'intégration du modèle d'illumination de *Phong* sur un cylindre.

Au rendu, un rayon intersecte la surface sous-jacente et la surface parallèle correspondent au plafond des texels. On se ramène alors dans l'espace de la texture où les texels s'identifient au volume de référence.

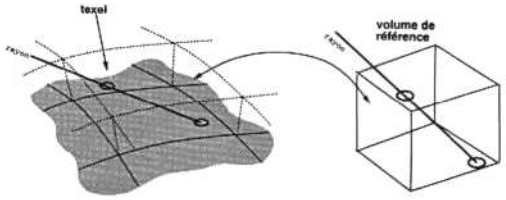


Figure 33 : Rendu de la texture. [Neyret96]

Un échantillonnage stochastique du rayon est réalisé afin d'alléger les calculs; ensuite, l'illumination locale est effectuée à l'aide du modèle de réflectance qui simule la présence d'un cylindre. Un rayon est envoyé vers la source de lumière afin de tester l'ombrage du voxel courant.

Ce modèle est le premier dans le domaine à avoir traiter les scènes complexes répétitives et a été proposé pour éviter les problèmes qu'auraient posés les modèles géométriques, il présente aussi plusieurs limites :

- Ce modèle reste un modèle dédié à la fourrure, le volume de référence ne code que la fonction d'occultation. La réflectance est factorisée car la surface est considérée comme isotrope.
- Le modèle ne tient pas en compte de la physique du matériau (opacité, illumination) ce qui nuit au réalisme du résultat.
- Le rendu est particulièrement lent.

3.4.3.2 Texture volumique de NEYRET

Afin de réaliser une représentation efficace des géométries complexes et partant des textures volumiques de *Kajiya et Kay*, *Neyret* a proposé en 1995 une autre approche hiérarchique, pour le rendu de la texture volumique [Neyret96].

Les principales caractéristiques de ce modèle sont :

- Utilisation de l'octree pour le codage volumique de l'échantillon de texture.
- Une représentation multi-échelle dans l'esprit du mip-map, ce qui suppose de savoir filtrer les données.
- Des informations géométriques (encodé par une probabilité d'occultation) et photométriques (encodé par une approximation de la distribution des normales).

En effet, cette méthode n'utilise qu'un seul texel de référence, qui est copié virtuellement pour éviter la duplication inutile des informations, où chaque copie ne conserve qu'un pointeur vers le texel de référence.

En ce qui concerne la modélisation de la réflectance, *Neyret* [Neyret98a] a choisi la BRDF qui caractérise l'anisotropie. Pour représenter cette dernière, il a considéré que la BRDF est issue de la distribution des normales locales (NDF : Fonction de Densité des Normales). La réflectance est donc obtenue en intégrant un modèle simple, typiquement celui de *Phong*, sur la distribution des normales.

Neyret a choisi l'ellipsoïde comme primitive de compromis pour encoder la réflectance puisqu'il permet de simuler des formes communes comme la sphère, le cylindre et l'élément de plan. De plus, l'opérateur d'addition est défini pour les ellipsoïdes, ce qui est une caractéristique très importante pour permettre la représentation multi-échelle. Il utilise donc les ellipsoïdes comme support des normales.

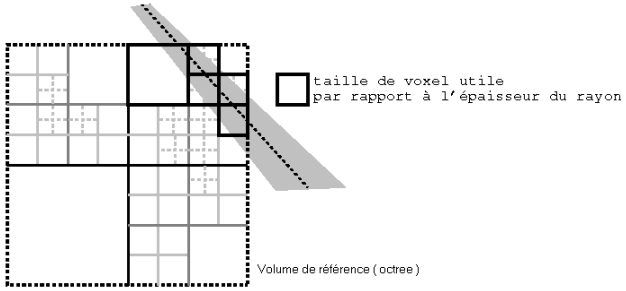


Figure 34 : Parcours du rayon. [Neyret96]

La manipulation des formes se fait à deux niveaux différents :

- La forme géométrique à grande échelle qui recouvre plusieurs voxels.
- La micro-primitive locale (forme locale) codée dans chaque voxel, qui représente la micro-géométrie (l'échelle microscopique).

Le rendu de la texture volumique se fait en deux passes :

- Rendu global qui se fait au niveau de chaque texel en utilisant un algorithme classique de lancer de cône en parcourant l'espace scène (ou géométrique), l'espace texel puis l'espace objet en utilisant l'interpolation trilinéaire ; où passage de l'espace scène à l'espace objet l'auteur suppose que le rayon subit une légère déformation qui va simuler la déformation du texel pour épouser la totalité de la boîte.
- Rendu local qui se fait au niveau de chaque voxel en intégrant le modèle d'illumination locale de *Phong*.

Cette méthode présente quelques limites, à savoir :

- Le coût inhérent à l'animation du texel de référence.
- Le stockage du volume de référence est considérable et exponentiel suivant le nivaux de profondeur utiliser lors de codage de l'octree.

4 Conclusion

Dans ce chapitre nous exposons l'art des différentes techniques de traitement ainsi que la complexité en synthèse d'image. Les différentes classes et méthodes que nous venons d'étudier ont pour but d'obtenir une grande complexité visuelle à moindre coût en terme de temps d'affichage et d'espace mémoire.

Dans cette optique, l'utilisation de polygones (même avec les dernières techniques de simplification et de LOD) n'est pas compétitive dans son ensemble; les chercheurs en synthèse d'image ont fourni cert de grands efforts en ce domaine pour arriver à ce stade mais, ils leurs restent beaucoup à faire. Ils doivent concevoir d'autres représentations plus performantes et à moindre coût afin qu'elles trouvent une large consommation au sein de la société.

Les modèles alternatifs sont souvent conçus et destinés à une famille d'objets définie, leur domaine d'application est alors réduit au minimum. Les plus usités sont :

- Les texture volumiques : Ils offrent une représentation compacte de scène répétitive.

- Le rendu à base de points : Il permet un affichage des modèles extrêmement complexes, mais il n'est pas intégrable tel quel dans un moteur de simulation d'environnement 3D.
- Le rendu à base d'images : Il génère des images du modèle sur la base d'échantillons visuels réels ou virtuels de celui-ci.

Finalement, La synthèse d'images naturelles pose de gros problèmes à cause de la grande diversité des objets (animaux, plantes, eaux, terres, ciel, … etc.) constituant la nature. Vu la grande diversité, il n'est pas envisageable pour le moment de trouver une solution globale à la problématique de la génération sans effets secondaires. Toutes les méthodes que nous venons de décortiquer jusqu'ici présentent un compromis entre les deux extrêmes. Ce compromis est basé en particulier sur les spécificités du matériel disponible à un instant donné.

Les recherches de ces dix dernières années sont toutes orientées vers les approches émergentes et plus particulièrement à la représentation à base de points. Cette dernière sera le sujet de notre prochain chapitre.

Chapitre 2 : Classification des méthodes de représentation et de rendu à base de points

1 Introduction

Aujourd'hui, il n'échappe à personne que l'infographie 3D est omniprésente en force et à la portée du consommateur même le plus démuni. Toutefois Il y a lieu de souligner qu'il a une prolifération des accélérateurs accessibles de matériels graphiques 3D, de postes de travail, de PC de haute qualité, de gamestations à bas prix ; la clef du succès est sans aucun doute les jeux électroniques interactifs qui sont apparus comme *l'application remarquable* du graphique 3D. Néanmoins, l'infographie interactive comme n'importe quelle science, elle évolue avec le temps, ce qui nous laisse affirmer qu'elle n'a pas encore atteint le niveau du réalisme lui permettant une vraie immersion dans le monde virtuel. Il y a des lacunes à remplir et des insuffisances à résoudre, à cela nous citons, les caractères typiques de premier plan dans les jeux, en temps réel sont extrêmement minimaliste des modèles de polygone qui montrent souvent des objets facettés, tels que les silhouettes angulaires.

Ainsi les diverses techniques de modélisation sophistiquées, telles que les surfaces implicites, NURBS ou les surfaces de subdivision, permettent la création de modèles graphiques 3D avec des formes de plus en plus complexes; sont en fin de compte décomposés en triangles avant d'être rendu par le sous-système graphique.

Actuellement, les triangles sont des primitives communes souvent employées au rendu, semblent rencontrer un bon équilibre entre la puissance descriptive et le fardeau de calcul. Reproduire des modèles naturels exige des formes fortement complexes avec plus de triangles. En effet, la géométrie est donnée de manière beaucoup trop explicite et de plus, il existe un lien de connectivité beaucoup trop fort entre les primitives représentant l'objet. Cela rend la réduction du nombre de polygones très délicate. Lorsque l'objet devient petit à l'écran, de nombreux polygones sont projetés sur un seul pixel, ce qui entraîne un surcoût de calcul.

Afin d'augmenter la complexité visuelle apparente des objets, un plaquage de texture a été présenté par *Catmull* et appliqué avec succès par d'autres. Les textures offrent plus de détail à l'intérieur d'un polygone et permettent ainsi l'emploi de grands triangles avec un nombre minime de triangles. Actuellement, des moteurs graphiques sont fortement adaptés pour le plaquage de texture à haute performance. Cependant, l'habillage de texture doit suivre une géométrie sous-jacente du modèle de polygone et mieux fonctionner sur les surfaces plates ou légèrement courbées. Néanmoins, les surfaces réalistes exigent fréquemment un grand nombre de textures qui doivent être appliquées dans des passages multiples lors de la rasterisation. Par ailleurs, il est difficile de reproduire les phénomènes tels que la fumée, le feu ou l'eau en utilisant des triangles texturés [Pfister00].

Les multiples problèmes qu'on vient de mettre en relief ont motivé les chercheurs à développer une nouvelle approche et élucider les points suivants :

- Quelle est la primitive la plus simple et la plus efficace que le triangle ?
- Quelle est la primitive de rendu pour la visualisation directe des modèles, sans nécessiter de produire un maillage triangulaire ?

C'est grâce aux éminents pionniers chercheurs *Levoy et al*, qui ont résolu les questions citées précédemment. Ils utilisent le point en tant que primitive de rendu (Figure 18), le nuage de points décrit la géométrie de la surface 3D et sa propriété de réflectance, ainsi il n'y a aucune information supplémentaire tel que la connectivité (i.e. : l'information explicite de voisinage entre les points) et le plaquage de texture ; le nuage de points est obtenu en employant un processus d'échantillonnage [Zwicker02a, Gross02] :

- Le nuage de point est le résultat d'un scanner 3D.
- Prendre le maillage triangulaire et ignorer la connectivité.
- Echantillonner le modèle polygonal.

Dans ce qui suit, nous présentons une classification des méthodes de rendu à base de points. Débutant par un ensemble de critère de classification, les méthodes de rendu à base de points sont classifient en trois classes :

L'approximation géométrique de premier ordre, L'approximation géométrique de deuxième ordre et les modèles hybrides. En suite on donnant un bilan récapitulatif contient une définition générale du paradigme, les différentes structures de modélisation et un schéma général de rendu. A la fin nous citons ces avantages et ces limites.

2 Critères de classification

Suite au travail innovateur de *Levoy et Whitted* [Levoy85] ; plusieurs chercheurs ont proposé récemment l'emploie de *points* en tant que primitive de base de rendu, au lieu du rendu traditionnel par des primitives triangulaires. Dans cette optique nous proposons une classification basée sur les critères suivants :

- Le contenu de l'échantillon de point : celui ci conserve deux types d'informations, information de réflectance (couleur diffuse) et information géométrique, qui divisent les méthodes en deux grandes classes :
  - Celle qui utilise l'approximation géométrique de premier ordre, tel que : la position, la normale, la taille des points.
  - Celle qui utilise l'approximation géométrique de deuxième ordre «ou géométrie différentielle », tel que : les surfaces régulières, les courbes.
- La source du nuage de points renferme deux grandes familles :
  - Echantillonnage de la géométrie (sampling the geometry) : le nuage de point est obtenu à partir d'un scanner 3D d'un objet réel, ou d'un échantillonnage de la géométrie d'un objet a modélisé
  - Le rendu d'échantillon de point (point sample rendering) : le nuage de point est existe déjà. Elle est obtenue, par exemple, à partir d'une photo numérique.

Enfin, une dernière classe « *le modèle hybride »* qui découle d'une combinaison du modèle polygonale et celui du rendu à base de points afin de réaliser certains critères[8].

3 L'approximation géométrique de premier ordre

3.1 Echantillonnage de la géométrie

3.1.1 L'utilisation du point comme une primitive d'affichage

En 1985, *Levoy et Whitted* [Levoy85] ont édifié une nouvelle technique. Celle-ci traite les problèmes de la croissante complexité d'image, causée par la modélisation géométrique. Ils ont proposé la notion de découpler la modélisation de la géométrie du processus de rendu ; en présentant la notion de points en tant que méta-primitif à zéro-dimension. Cette méthode se divise en deux parties essentielles (Figure 35):

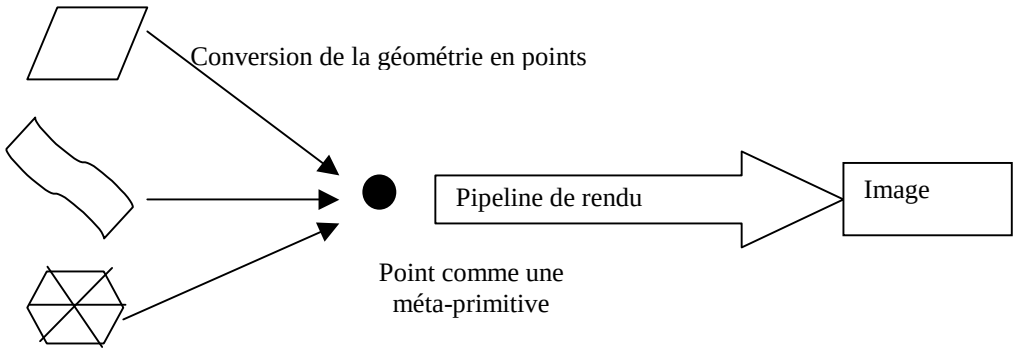


Figure 35 : Vu d'ensemble de l'algorithme. [Levoy85]

- **Définition d'un point :** Le *point* est défini comme un 7-tuple (x, y, z, r, g, b, α) chacune des valeurs dans le 7-tuple est appelée attribut. Les valeurs de x, y et z sont les attributs spatiaux, le reste est appelé attributs non-spatiaux.

[8] Les différents critères cités ont pour objectif de minimiser le temps de rendu et d'avoir une haute qualité d'image.

- **Conversion de la géométrie en points :** On a mis un ensemble de conditions minimums à toutes géométries, afin d'être traitée par un algorithme de rendu prévu :
  - Possibilité de décomposer la surface en points.
  - La surface doit être continue et différentiable dans un petit voisinage autour de chaque point.
  - En ce qui concerne le déterminant du Jacobien (voir pipeline de rendu phase **e**), on doit trouver deux vecteurs non colinéaires, qui se trouver tous les deux sur un même plan tangent et qui rapproche localement la surface au point. On peut également les employer pour trouver le vecteur normal de surface, exigé par le détecteur de replie.

  C'est la condition primordiale que doit satisfaire une surface pour être convertie en un nuage de points. Ces derniers sont sauvegardés dans un tableau. Néanmoins, rien n'est exigé soit de l'espacement ou la distribution des points.

- **Pipeline de rendu :** L'objectif du pipeline de rendu est de saisir un tableau de points, le projeter sur un écran de manière qu'ils apparaissent en une surface tridimensionnelle continue. A cet effet on doit satisfaire les trois critères suivants :
  - La texture à l'intérieur du tableau doit être correctement filtrée.
  - Le bord du tableau doit être correctement anti-aliassage.
  - Le tableau doit complètement obscurcir son arrière plan.

Le pipeline de rendu renferme sept étapes qui sont :

a) **Définition d'une grille initiale:** L'étape consiste à définir une *grille initiale* comme un treillis d'espacement régulier, rectangulaire, bidimensionnel de points sources portant les coordonnées paramétriques u et v. A cette étape, on admet que x = u et y = v. La grille est facilement stockée en tant que texture (c'est-à-dire dans un tableau bidimensionnel sans les valeurs de y et de x).

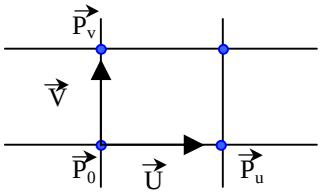
b) **Choix d'un point :** Chaque itération du processus de rendu repose sur le choix d'un point arbitraire de la grille initiale et l'envoyer par le pipeline de rendu. L'ordre de rendu peut être séquentiel dans quelque espace paramétrique, dirigé par l'exécution d'une procédure ou aléatoire.

c) **Perturbation de points :** Soit une perturbation définie comme opération qui change chacun des attributs associés à un point. Toute perturbation peut être appliquée au point aléatoirement choisi.

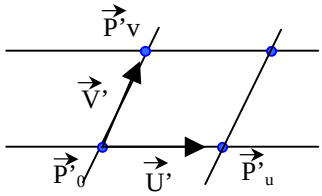
d) **Transformation et coupure (clipping) :** Considérant M comme matrice de transformation carrée qui inclue la projection perspective de dimension 4. L'étape de transformation consiste à exécuter la multiplication matricielle habituelle de [x, y, z, 1] par M et le suivi par la perspective. La division de z par w est supprimée pour que le z-coupure soit exécuté. La coupure d'un point repose sur la comparaison des coordonnées transformées de x, y et de z (du point à la vision frustum tridimensionnelle), rejeter le point s'il se trouve à l'extérieur des frontières du frustum.

e) **Calcul de la densité de points dans l'espace image :** La densité de points sources dans l'espace écran est inversement proportionnelle à la surface du parallélogramme formé par les vecteurs d'unité (Figure 36**Erreur ! Source du renvoi introuvable.**d). Cette surface est facile à calculer. Elle est égale à la valeur absolue du déterminant de la matrice Jacobien:

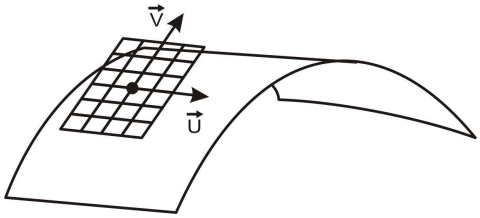
$$A = \left| \det \left[J_{\mathbf{F}}(\mathbf{p}_0') \right] \right| = \left| \det \begin{bmatrix} x_u' - x_0' & x_v' - x_0' \\ y_u' - y_0' & y_v' - y_0' \end{bmatrix} \right|$$



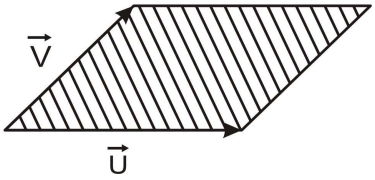
a) Vecteurs d'unité dans l'espace u-v : perturbé mais non transformé.



b) Vecteurs d'unité perturbé et transformé dans l'espace u-v.



c) Plan tangent sur la surface dans le petit voisinage



d) Aire de parallélogramme qui donne la densité des points source.

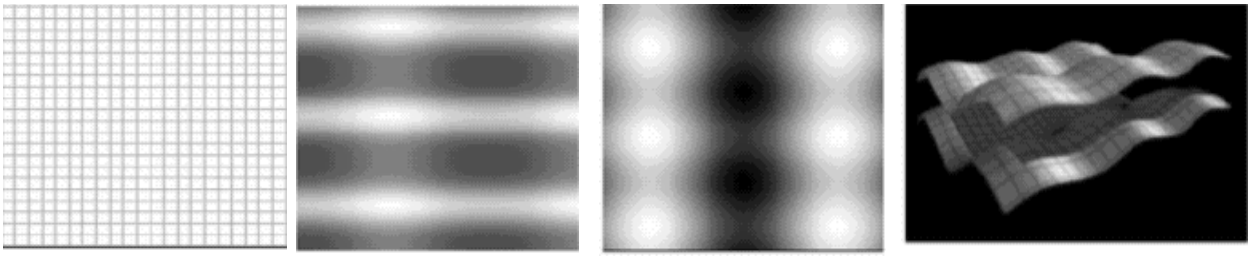
Figure 36 : Calcul de la densité de points sources. [Levoy85]

f) **Sélection du rayon de filtre approprié :** Le rayon efficace d'un point est calculé en fonction de la densité source et la densité d'échantillon d'affichage. Quand la transformation d'observation appelle à la minification (beaucoup de points source à un pixel d'affichage), le filtre doit être assez grand pour éviter l'aliassage de la fonction source. Au cas de magnification (beaucoup de pixels d'exposition à un point source), le filtre doit être assez grand pour éviter l'aliassage de la reconstruction. En pratique, le rayon efficace diminue à mesure que la densité source augmente, mais atteint un rayon minimum qui dépend de la résolution d'échantillon d'affichage. Le rendu utilise l'ellipse flou pour une haute qualité d'image.

g) **Elimination de surface cachée :** Pour se faire, on utilise l'algorithme A-buffer ; celui ci réuni toutes les contributions à une surface donnée. On exécute les calculs de mélange dans la mesure du possible afin de consolider les fragments superficiels et retarder le calcul de visibilité jusqu'à ce que toutes les contributions soient calculées.

En conclusion : la capacité de rendre les points dans un ordre aléatoire suscite que le rendu de chaque point soit indépendant du rendu d'un autre point. Cela suggère aussi l'utilisation du matériel parallèle. Les principaux avantages de cette approche sont :

- Un algorithme de rendu normalisé peut être employé pour afficher n'importe quelle géométrie. La personnalisation pour chaque nouvelle primitive de modélisation n'est pas nécessaire.
- La géométrie peut être rendue dans l'ordre d'objet. Il est particulièrement avantageux en cas d'objets définis procédurallement.
- L'algorithme est en mesure de rendre les tableaux de surfaces de points qui n'ont aucune géométrie sous-jacente. Cela fournit une solution simple au problème de bump mapping de silhouette.
- Le point est simple et n'exige aucune cohérence pour être rendu efficacement.



a b c d

Figure 37 : Vu de rendu d'un nuage de points. [Levoy85]

La Figure 37offre un exemple de rendu d'un polygone rectangulaire plat représenté comme un tableau de 170 x 170 points. La nuance de chaque point est multipliée d'abord par la texture de raie de goupille représentée sur la Figure 37a pour augmenter sa lisibilité, puis par la carte de réflectance représentée sur la Figure 37b. En conclusion, la coordonnée z de chaque point est perturbée vers le haut en utilisant le champ discret de taille représenté sur la Figure 37c. le résultat est décrit sur la Figure 37d.

3.1.2 Surfel

En 2001, *Pfister et al*. ont présenté, une nouvelle méthode nommée *surfels* (surface element). Un *surfel* est un n-tuple de zéro-dimension avec des attributs de forme et d'ombre qui rapprochent localement une surface d'objet [Pfister00]. L'idée principale est la représentation des objets avec des surfels, de les décrire dans une vue indépendante, de manière à centrer l'objet au lieu de centrer l'image.

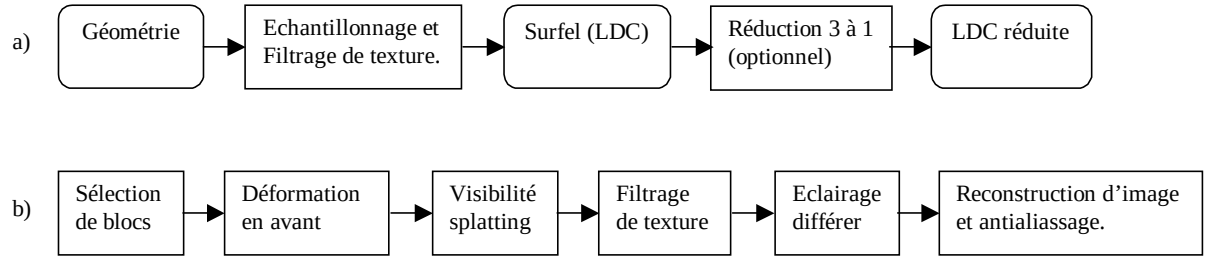


Figure 38 : Vu d'ensemble d'algorithme : a) prétraitement ; b) rendu de l'arbre LDC. [Pfister00]

Semblable à la méthode présentée par *Levoy et Whitted* [Levoy85], cette méthode est décrite par deux étapes principales (Figure 38**Erreur ! Source du renvoi introuvable.**) :

- **Processus d'échantillonnage :** Le processus convertit les objets géométriques et leurs textures en surfels. On emploie le lancer de rayon distribué pour créer trois couches d'images de profondeurs orthogonales (LDIs). Les LDIs [Shade98, Chang99] stockent de multiples surfels le long de chaque rayon, un pour chaque point d'intersection de rayon surface. *Lischinski et Rappaport* [Lischinski98] appellent cet arrangement de trois LDIs orthogonaux un *cube de couche de profondeur* (LDC).

  Un nouvel aspect important de cette méthode de prélèvement d'échantillons est la distinction entre le prélèvement d'échantillons de la forme (ou géométrie) et de l'ombre (ou couleur de texture). Un surfel stocke la forme (telle que la position et l'orientation de surface) et l'ombre (tel que les multiples niveaux de couleurs de textures préfiltrées). En raison des similitudes aux plaquages traditionnels de texture, on appelle cette information hiérarchique de couleur : *mipmap de surfel*.

  A partir du LDC on crée une structure de données hiérarchique efficace pour le rendu. *Pfister et al.* emploient une structure hiérarchique, divisent l'espace et stockent un LDC à chaque nœud de l'octree. Chaque nœud de LDC dans l'octree est appelé un *bloc*. La structure de données résultante est appelée «arbre *de LDC »*. Dans une étape facultative appelée réduction 3-à-1 on est amené à réduire le LDC en un LDI simple sur la base de bloc-par-bloc pour accélérer le temps de rendu.

- **Rendu de surfel ou pipeline de rendu :** Celui-ci consiste à projeter le niveau de blocs approprié à l'espace écran en utilisant la projection perspective :
  - Sélection de blocs : Lors de la traversée de l'arbre, on est amené à choisir le niveau de détail approprié, ce qui permet d'accélérer le rendu.
  - Déformation en avant : Elle consiste à reprojeter les surfels de LDI à l'espace écran, en estimant la densité de surfel projetée dans l'image de sortie afin de contrôler la vitesse de rendu et la qualité de la reconstruction d'image.
  - Détecter les trous : On utilise la combinaison d'un z-buffer conventionnel avec une nouvelle méthode nommée « *visibilité splatting »*, ceci facilite la résolution du problème de visibilité et la détection des trous.
  - Filtrage de texture et ombrage : On filtre les couleurs de texture des surfels visibles en utilisant l'interpolation linéaire entre les niveaux appropriés du mipmap de surfel. Chaque surfel visible est ombragé, pour se faire-on utilise par exemple, l'illumination de *Phong* et la réflexion de mipmap.

- Remplir les trous : L'étape finale exécute la reconstruction d'image des surfels visibles, en incluant le remplissage de trou et l'anti-aliassage. En général, la résolution de l'image de sortie et la résolution de z-buffer ne doivent pas être identiques.

Figure 39 : Exemple de rendu de surfel. [Pfister00]

Enfin, cette approche a les avantages suivants :

- Le pipeline de rendu de surfel vient compléter et forcer le pipeline matériel.
- Rendu rapide et de haute qualité d'image.

Notez bien que cette approche ne fournie pas une définition mathématique de surface lisse comparable à celle de « point set surface » de *Alexa* [Zwicker02a, Kobbelt04].

3.1.3 Qsplat

Rusinkiewicz et al. présentent un système, nommé *Qsplat*, pour la représentation et l'affichage progressif de grandes mailles qui combine une hiérarchie de multi-résolution basée sur les sphères englobantes avec un système de rendu basé sur des points [Rusinkiewicz00]. Ce système qui ne maintient pas la connectivité de la maille d'entrée est constitué en deux étapes :

- **Algorithme de prétraitement :** Qsplat utilise une hiérarchie de sphères englobantes compacte pour être employée à la sélection de visibilité, le contrôle de niveau de détails et le rendu rapide. Chaque nœud de l'arbre contient le centre et le rayon de la sphère, une normale, la largeur d'un cône normal et une couleur facultative.

  On pourrait produire une telle hiérarchie de sphères englobantes à partir de mailles triangulaires représentant le modèle qui doit être codé. La création de cette hiérarchie se fait comme suit :

  - Calculer la taille de la sphère feuille : Pour chaque sommet de maille la taille de la sphère qui l'englobe est égale à la taille maximale des sphères englobantes de tous les triangles qui se joignent à ce sommet ; cela est employé pour garantir l'élimination des trous lors du rendu [Rusinkiewicz00].
  - L'arbre est construit à partir de l'utilisation de la méthode de coupe médiane, c'est à dire de la même manière que KD-tree. On utilise l'algorithme suivant afin de créer le reste de l'arbre :

```
ConstruireArbre (sommet [debut.. fin])
{
        Si (debut = fin)
           return Sphère (sommet [debut ] )
        Sinon
           Milieu-du-point = Division_ Lelong_Pluslong_Axe (sommet [debut.. fin])
           sous-arbre_gauche = ConstruireArbre (sommet [debut.. Milieu-du-point])
           sous-arbre_droite = ConstruireArbre (sommet [Milieu-du-point +1.. Fin])
         return Sphère_englobante ( sous-arbre gauche, sous-arbre_droite)
}
```

  L'algorithme crée l'arbre en distribuant l'ensemble des sommets sur l'axe le plus long de sa boîte englobante afin de trouver le sommet du milieu ; récursivement on calcul les deux sous arbres pour découvrir la sphère englobante des deux sphères enfants [Kobbelt04]. De même que la création de l'arbre, on calcule les propriétés (comme la normale et la couleur) du nœud intérieur par la moyenne des propriétés de ses fils. Quand la répétition atteint un seul sommet, il suffit de concevoir une sphère dont le centre est la position du sommet. Du fait que la taille totale d'un arbre dépend du facteur d'embranchement à chaque nœud, on combine les nœuds dans l'arbre pour augmenter la moyenne

d'embranchement, le facteur est approximativement égale à 4. Cela réduit le nombre de nœuds intérieurs, réduisant ainsi les exigences de stockage pour l'arbre. Cette structure a les avantages suivants [Rusinkiewicz00]:

- La représentation est compacte.
- Le calcul est rapide.
- Elle est adéquate pour une grande masse de données.

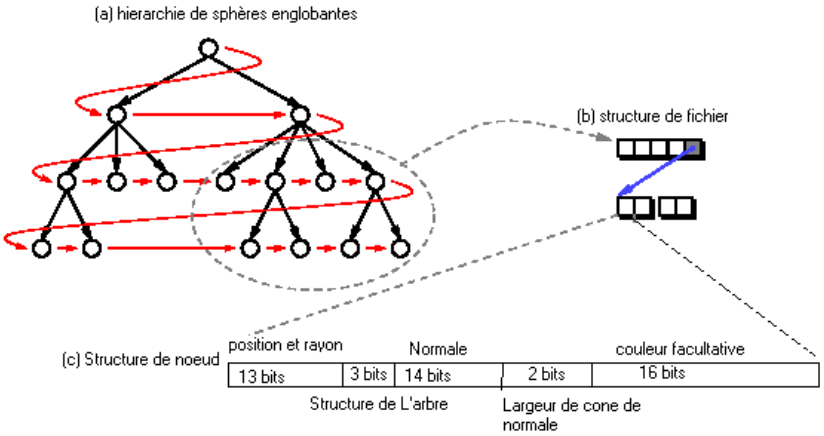


Figure 40 : La hiérarchie des sphères englobantes. [Rusinkiewicz00]

❖ **Algorithme de rendu :** Une fois que la hiérarchie a été construite, l'algorithme concerne la traversée de l'arbre en déterminant les trois phases suivantes :

➢ La sélection de visibilité : L'objectif de cette phase est d'éliminer les nœuds qui ne sont pas visibles, pour cela on utilise les deux méthodes suivantes :

- *La sélection frustum* : Consiste à évaluer la position de chaque sphère par rapport aux plans de la *vue frustum*. Si la sphère se trouve à l'extérieur, celle-ci et ses sous arbres sont négligés (i.e. nœud invisible). Si la sphère se trouve entièrement à l'intérieur du frustum (i.e. nœud visible), ce fait est noté et aucune nouvelle *sélection de frustum* n'est essayée sur les enfants du nœud.
- *L'élimination des faces arrières (backface)*: En utilisant la normale et le cône des normales stockés à chaque nœud. Si les faces du cône sont entièrement loin de la camera, le nœud et son sous arbre sont mis au rébus.

➢ Détermination de la traversée de l'arbre : L'heuristique employée par Qsplat pour décider la traversée est basée sur la taille du nœud projetée sur l'écran. C'est-à-dire un nœud est traversé, si l'aire de la sphère projetée sur le plan d'observation, excède un seuil.

➢ Dessiner le splat : Une fois qu'on a atteint un nœud de feuille ou qu'on a décidé d'arrêter la traversée, on dessine un splat représentant la sphère actuelle. La taille du splat est basée sur le diamètre projeté de la sphère actuelle ; sa couleur est obtenue avec un calcul d'éclairage basé sur la normale et la couleur de la sphère actuelle. Les splats sont accomplis par le z-buffer.

```
TraverserHiérarchie (noeud )
{       si (nœud est invisible)
         sautez cette branche de l'arbre
        sinon si (le nœud est un nœud feuille)
         dessinez un splat
        sinon si (l'avantage de la récursivité est trop bas)
        dessinez un splat
        sinon
         Pour chaque enfant de fils(nœud )
         TraverserHiérarchie (enfant)
}
```

- **Forme de splat**[9] : Il y a trois formes de primitive :
  - *Le carré plat* : C'est l'approximation grossière de la forme projetée d'un point. Il se caractérise par :
    - C'est la forme la plus rapide à dessiner et elle est supportée par le matériel (OpenGL : GL- POINT).
    - Cette forme de splat ne s'adapte pas à l'orientation de surface qui mène à l'épaississement des silhouettes d'objet.
    - La qualité inférieure d'image.
    - La visibilité est résolue en utilisant un z-buffer conventionnel.
    - La profondeur de primitive est constante, et égale au coordonnée z du point projeté.
  - *L'ellipse* : C'est la forme qui correspond parfaitement à la forme réelle du splat, elle est caractérisée par :
    - L'adaptation à l'orientation de surface, c'est à dire il n'y a aucun épaississement près des silhouettes.
    - L'ellipse fournis une haute qualité d'image que le carré plat.
    - La profondeur de la primitive n'est pas constante, mais déduite de la projection du plan tangent du point; Ceci mène à une meilleure résolution de visibilité.
    - Les ellipses peuvent laisser des trous près des silhouettes. Ceci peut être résolu en limitant le rapport entre le rayon principal et le rayon secondaire de l'ellipse par une valeur maximum.
    - Les ellipses sont plus lentes que les carrés plats et ne sont pas directement soutenues dans le matériel.
  - *Splat flou* : L'inconvénient commun des formes précédentes est qu'il n'y a aucune interpolation de couleur entre les splats. Ceci mène aux artefacts et indique la forme de splat utilisée à l'observateur. Une meilleure résolution de ce problème est l'emploi du splat flou (Figure 41), ce dernier consiste à convoluer la forme de splat par un masque alpha (fonction gaussienne 2D).

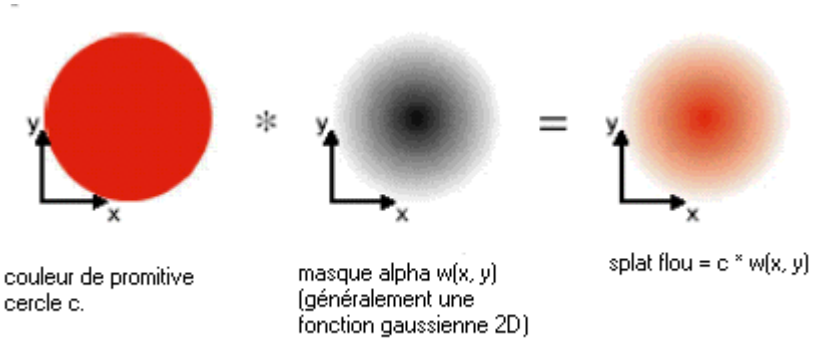


Figure 41 : Calcul d'une primitive de splat gaussien.

Remarque : cette technique permet de rendre de 200k à 300k de points par frames, ce qui fournit une utilisation réelle sur les PCs, où l'étape de prétraitement est beaucoup plus rapide que les algorithmes de LOD.

3.1.4 Algorithme z-buffer aléatoire

Wand et al. [Wand00], présentent un nouvel algorithme de rendu *sortie-sensible*, nommée *z-buffer aléatoire*, capable de rendre des scènes fortement complexes aux taux frames interactifs, car sa complexité de temps dépend seulement et faiblement de la complexité de la scène d'entrée.

[9] projeter chaque voxel du volume de subdivision dans le plan image. Littéraire : splat = tache.

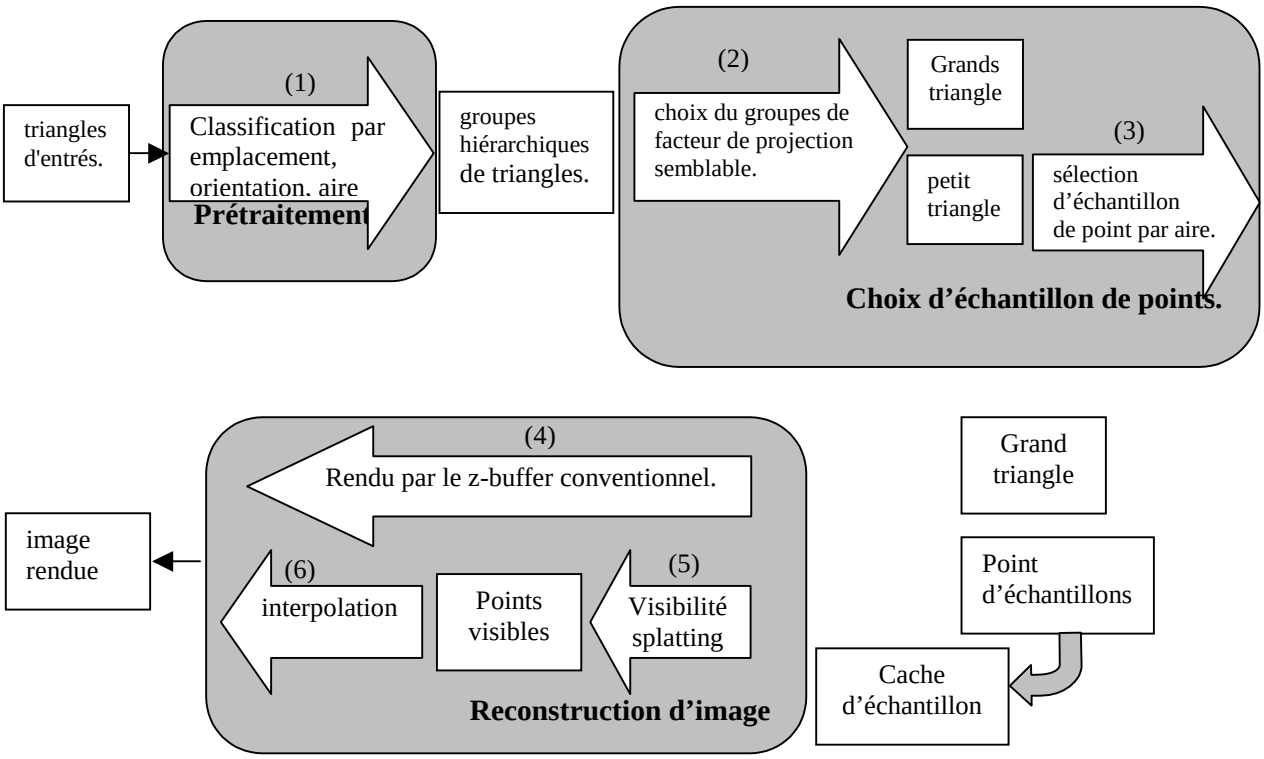


Figure 42 : Plan de l'algorithme. [Wand00]

L'idée principale de l'algorithme est de produire l'image d'une scène en la reconstruisant à partir d'un ensemble d'échantillonnage de points de surfaces aléatoires choisi dynamiquement. Les points d'échantillons représentent la géométrie de la scène complexe, par conséquent chaque triangle ne doit pas être manipulé séparément pendant le rendu. L'algorithme se compose de deux étapes principales et une étape de prétraitement (Figure 42) :

- **Prétraitement :** Cette étape consiste à reproduire une structure hiérarchique (généralement octree) à partir de *n* triangles représentant l'image d'entrée, avec une fonction d'illumination tel que *Phong*, texture et cartes d'environnement pour déterminer la couleur en chaque point.
  - Construction de l'octree : La construction de l'octree ordinaire est modifiée aux points suivants :
    - Tous les nœuds intérieurs, qui ont seulement un nœud enfant, ne sont pas stockés explicitement.
    - Chaque nœud a une boîte étendue, qui est la boîte du nœud agrandi par un facteur constant.
    - Les triangles, qui croisent la grille d'octree, sont stockés dans le nœud intérieur, le plus petit qui a une boîte étendue qui contient entièrement le triangle.
    - Chaque nœud dans l'octree tient une liste de distribution des aires de tous les triangles contenus dans ce nœud.
- **Choix de points d'échantillon :** Cette étape consiste à extraire les points d'échantillon à partir des petits triangles ; de plus ces points doivent être distribués uniformément sur les projections des objets dans le plan d'image. Ainsi, pour le choix de points d'échantillon on doit avoir :
  - Une *densité de probabilité*, proportionnelle à l'aire projetée des surfaces des objets, doit être calculée en utilisant un *algorithme d'approximation*.
  - Pour chaque point de vue, un ensemble de groupe de triangle est choisi dynamiquement de la hiérarchie (Figure 42, (2)). On utilise la *densité de probabilité*, ce groupe de triangle est classé soit dans un groupe de large triangle ou dans un autre groupe de petit triangle.
  - Le *facteur de projection* maximal est calculé pour déterminer le nombre de points d'échantillon de chaque groupe (Figure 42, (3)): Dans la plupart des cas, il est suffisant de considérer seulement la distance entre un groupe et la camera (ou l'orientation des triangles) pour estimer le facteur de projection.

Suite au choix d'un ensemble de points d'échantillon pour un groupe d'objets, deux techniques d'accélération sont utilisées :

- *Cache d'échantillon* : L'ensemble des points d'échantillon est stocké dans la *cache* du matériel graphique pour réduire au minimum le coût de *sélection d'échantillon*.
- *Z-buffer conventionnel* : Il est utilisé pour le rendu du grand triangle où le rendu est plus rapide que le *z-buffer aléatoire* (Figure 42, (4)).

- **Reconstruction d'image :** Pour reconstruire une image à partir d'échantillon de points on passe par les étapes suivantes:
  - Les points occultés doivent être enlevés de l'ensemble d'échantillon (Figure 42, (5)).
  - L'image est obtenue par l'interpolation entre les points d'échantillon visible (Figure 42**Erreur ! Source du renvoi introuvable.**, (6)).
  - Techniques d'amélioration : l'utilisation des plus grands splats de couleur constante accélère la reconstruction. Les résultats de haute qualité peuvent être obtenus en utilisant un filtre gaussien. De cette façon, le bruit et les artefacts peuvent être évités.

Avantages de cette approche :

- L'algorithme est g*énéral.* Il prend des modèles arbitraires, se composent de triangles comme entrée.
- Le temps de rendu et les résultats sont indépendants de la topologie d'entrée.
- Le temps de rendu est O (*a* log n). Pour une scène constituée de *n* triangles couvrant une zone projetée sur écran de *a* pixels, permet de rendre des scènes fortement complexes.
- Une haute qualité d'image.
- Temps interactifs d'ordre O (t), permet la mise à jour locale de la scène. Où ***t*** est la hauteur d'un octree construit pour les objets de la scène.

En comparaison avec Qsplat et Surfels, cette méthode tient compte de l'insertion dynamique efficace et le déplacement d'objets. Ainsi, il peut être employé dans une large gamme d'applications.

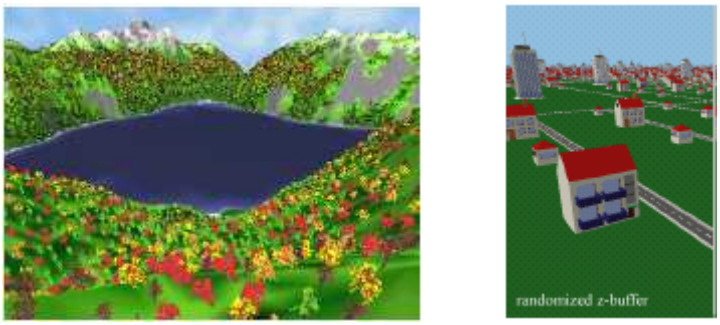


Figure 43 : Exemple de scènes générées par cette méthode. [Wand00]

3.1.5 Rendu flexible à base de points sur plates-formes mobiles

Duguet et Drettakis présentent, en 2003, une nouvelle approche qui combine le pipeline de rendu de Qsplat, et la structure hiérarchique de stockage compact (*Botsch et al.*) [Botsch02] ; celle ci rend les scènes complexes, renfermant des centaines de milliers et même des millions de primitives sur des plates-formes mobiles[10]. Pour se faire, l'approche se subdivise en deux étapes [Duguet03] :

- **P-grille :** C'est une représentation d'échantillon de points codés dans une structure hiérarchique. L'avantage de cette structure est sa flexibilité, en termes de stockage et de rendu. C'est des grilles récursives avec une subdivision régulière et uniforme à chaque niveau ; les p-grille peuvent être vus (avec une subdivision de p^3 pour chaque cellule) comme une généralisation de l'octree (subdivisé dans 2^3 pour chaque cellule). En combinant la stratégie de prélèvement d'échantillons et le processus de création de p-grille on obtient un nouvel algorithme simple et général [Duguet03, Botsch02] :

[10] Les plate-formes mobiles ont typiquement un seul processeur et des capacités de calculs en virgule flottante limitées, peu de mémoire, un affichage réduit et aucun processeur dédié au graphisme.

- Chaque cellule est subdivisée en *p* sous cellules.
- Pour chaque cellule de la hiérarchie intermédiaire ou feuille, on échantillonne la géométrie au point de l'objet le plus proche au centre de la cellule.
- Si la cellule ne contient aucun objet, la cellule est vide.
- Si la cellule entrecoupe l'objet (la cellule est pleine), alors on stocke le centre de la cellule (point d'entrée) ainsi que ses attributs d'échantillonnage intermédiaire (i.e. : la normale, les couleurs ou les attributs de matériel).
- Dans cette structure de grille récursive, on stocke les attributs des échantillons intermédiaires pour les nœuds intérieurs (c'est-à-dire, non feuille) de la hiérarchie. Ces échantillons intermédiaires permettent d'effectuer un rendu *multi-niveaux* efficace et flexible avec des ressources de traitement limité.

Enfin, ils ont choisi l'utilisation de tri-grille (i.e. : p=3) pour un meilleur compromis entre la taille de mémoire compacte et le temps de rendu nécessaire.

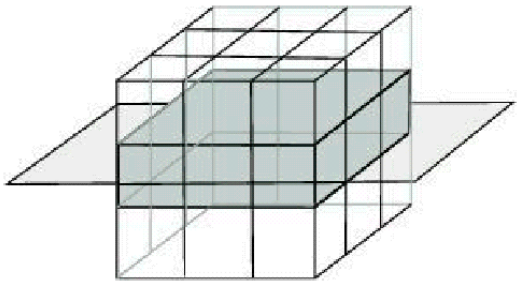


Figure 44 : Illustration de dimension pour un plan. [Duguet03]

- **Le rendu flexible :** Cet algorithme de rendu est similaire à celui de Qsplat ; il se compose en deux étapes qui sont :
  - Sélection de points visible : quand on traverse la hiérarchie de p-grille, on sélectionne les nœuds qui doivent être rendu, en utilisant la sélection frustum.
  - Splatting : Une fois qu'on a atteint un nœud de feuille, ou on décide de dessiner un splat représentant le voxel actuel ; La taille du splat est basée sur le diamètre projeté du voxel actuel, sa couleur est obtenue par l'utilisation d'une carte d'ombrage.

Figure 45 : La structure employée pour le rendu hiérarchique basé sur point d'un modèle de 4.7M polygones à 2.1 fps sur un 200Mhz IPAQ, échantillonné à 1.3 M de points. L'approche de multi-niveau limite le nombre de points rendus selon la vue. [Duguet03]

3.2 Le rendu d'échantillons de points

3.2.1 Rendu d'échantillon de points

En 1998 le rendu à base de point a été révisé par *Grossman et Dally* [Grossman98b]. Cette fois le but est de rendre les objets complexes en temps réel sans l'utilisation de matériel coûteux, qui soutiendraient l'éclairage dynamique [Grossman98a]. Le travail en question s'est inspiré du développement du rendu à base d'image. Il se compose en deux étapes :

- **Modélisation des objets** : Les objets sont modélisés tel un dense ensemble d'échantillons de points de surface, obtenus à partir de vues orthographiques, stockées avec l'information de couleur, de profondeur et la normale, permettant la composition en z-buffer, l'illumination de *Phong*, et autres effets tels que les ombres.
  - Prélèvement d'échantillons : Il se réalise en échantillonnant des vues orthographiques sur un treillis de triangle équilatéral où la longueur du coté est égale à ds.cosθ [11] (Figure 46) afin d'avoir un échantillon de points réguliers. Ce treillis permet de maintenir la vitesse de rendu et un stockage efficace de points. Ensuite on cherche les points échantillonnant la surface de l'objet.

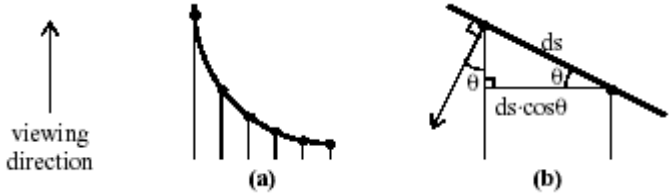


Figure 46 : (a) Les échantillons sur la surface de l'objet peuvent être arbitrairement irréguliers. (b) Utilisation d'un angle de tolérance pour contraindre la distance entre échantillons adjacents. [Grossman98a]

- **Rendu :** Connaissant la topologie de la surface, les échantillons de points sont rendus directement et indépendamment. Le pipeline de rendu d'échantillon de point [Grossman98a], renferme cinq étapes qui sont :
  - Test de visibilité : Afin de rendre seulement les blocs visibles, on utilise la *sélection frustum*. On utilise aussi deux structures de données complémentaires pour accélérer le rendu où l'on n'affiche que les points visibles dans un bloc. Ces structures de données sont : le *cône de visibilité* et le *masque de visibilité.*
  - Déformation de Bloc : Les points dans chaque bloc visible doivent être transformés à l'espace écran et plaqués aux pixels de l'image. La procédure utilise des calculs progressifs ; elle est très rapide [Grossman98b].
  - Reconstruction d'image : La précédente étape génère des trous dans l'image résultante. Le phénomène est dû aux effets d'agrandissement et de projection perspective. Pour résoudre ce problème on utilise une technique qui s'effectue deux étapes :
    - La détection de trou : Elle consiste à identifier les points de fond, les supprimer du tampon d'image, et leurs pixel sont comptabilisés comme trous [Grossman98a]. L'idée est d'avoir une hiérarchie de z-buffers a une résolution décroissante (Figure 47).
      - Le tampon le plus bas dans la hiérarchie possède la même résolution que l'image cible.
      - Chaque tampon dans la hiérarchie a la moitié de la résolution du tampon précédent.

[11] **ds** : est égale la longueur de coté de pixel à la résolution cible

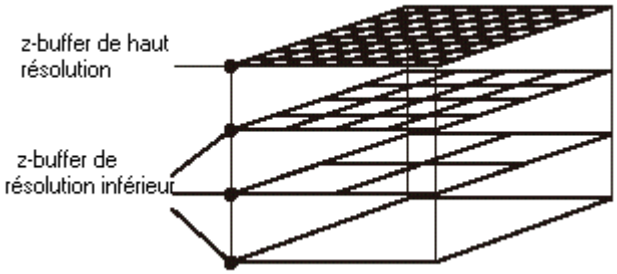


Figure 47 : Hiérarchie de z-buffer.

L'utilisation de cette hiérarchie implique une modification du précédent procédé de déformation, pour se faire: à chaque point déformé à l'espace écran, on fait d'abord un test de profondeur z entre le point courant et le pixel qui lui correspond, se trouvant au z-buffer de résolution d'image. Si le test échoue, le point est jeté, si non on réalise le même test de profondeur sur le z-buffer de résolution inférieur impliquant l'inexistence de trous (c.-à-d. plus l'objet est agrandi, plus le z-buffer de résolution inférieur est utilisé, étant donnée que la densité de point est inférieure). Si le second test est vérifié. La couleur du point est copiée au tampon de couleur. Notez que les informations sur la densité du point sont exigées pour choisir un z-buffer de résolution inférieure appropriée.

Cette technique de base produit l'artefact nommé *effet de bloc*, d'où une méthode plus complexe est réellement employée à l'implémentation. Elle consiste d'assigner aux pixels des poids. Ces poids sont employés pour une interpolation de couleur à l'étape suivante [Grossman98a, Grossman98b].

- Remplissage de trous : Le but est de remplir les pixels comptabilisés en trous par la moyenne des couleurs de leurs voisins. On utilise l'algorithme d'approximation de données dispersées en deux phases "*pull-push*" [Gortler96].Dans la première « *pull* » on calcule une série d'images approximées de résolution décroissante qui serviront à la seconde phase « *push* » au calcul d'une deuxième série d'images de résolution croissante. On rempli au fur et à mesure les trous.

➢ Illumination : Chaque pixel de premier plan est éclairé à la manière de *Phong*. L'éclairage est exécuté dans l'espace écran suite à la déformation des blocs.

Le rendu de points d'échantillons a les avantages et les inconvénients suivants :

➢ La méthode est rapide du fait de la simplicité de la procédure de déformation de points.

➢ Le temps de rendu est indépendant de la complexité du modèle

➢ Une limitation de l'algorithme est la difficulté d'échantillonner correctement les structures minces de l'ordre d'un pixel.

➢ Telles structures paraissent souvent loqueteuses dans les images de rendus.

➢ Le défi fondamental de rendu d'échantillons de points est la reconstruction sur écran de surfaces continues.

3.2.2 EWA splatting surface

La surface splatting (*Zwicker et al.*) combine les idées de *Levoy et al.* [Levoy85] avec le filtre EWA de *Heckbert* utilisé dans le rendu polygonal pour le filtrage de texture [Heckbert89]. Son but est d'avoir une image de haute qualité qui comporte le filtrage de texture anisotrope, anticrénelage de bord et la transparence. La structure générale de cette méthode repose sur les étapes suivantes :

❖ **Prétraitement :** A partir d'un nuage de points, on détermine les fonctions de base r_k et les coefficients w_k pour l'acquisition de textures.

➢ Objet à base de points : L'objet est représenté comme un ensemble de points irrégulièrement espacés $\{P_k\}$ dans l'espace objet tridimensionnel. Un point p_k a une position et une normale. Il est associé à une fonction de base r_k et des coefficients w^r_k, w^g_k, w^b_k, représentent des fonctions continues pour les composants de couleurs rouge, vert et bleu. Aux formules suivantes, on utilise un coefficient scalaire w_k. à la place des coefficients de couleur.

➢ Une fonction continue de texture à la surface représentée par l'ensemble de points (Figure 48) et la paramétrisation locale, est définie par :

$$f_c(u)=\sum_{k\in N} w_k r_k(u-u_k) \qquad \textbf{(2.1)}$$

Où u_k sont les coordonnées locales du point p_k, la fonction f_c (u) donne la couleur du point q.

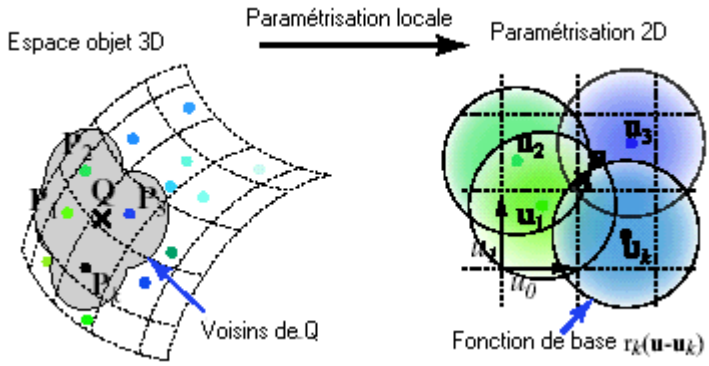


Figure 48 : Définition d'une fonction de texture sur la surface d'un objet basé sur point. [Zwicker02c]

❖ **Le Rendu :** Le rendu d'un objet à base de points dont la texture est définie par l'équation (2.1) ; celle ci doit être plaquée à l'espace écran. Pour se faire, on utilise le rééchantillonnage de *Heckbert* [Heckbert89] qui emprunte les étapes suivantes:

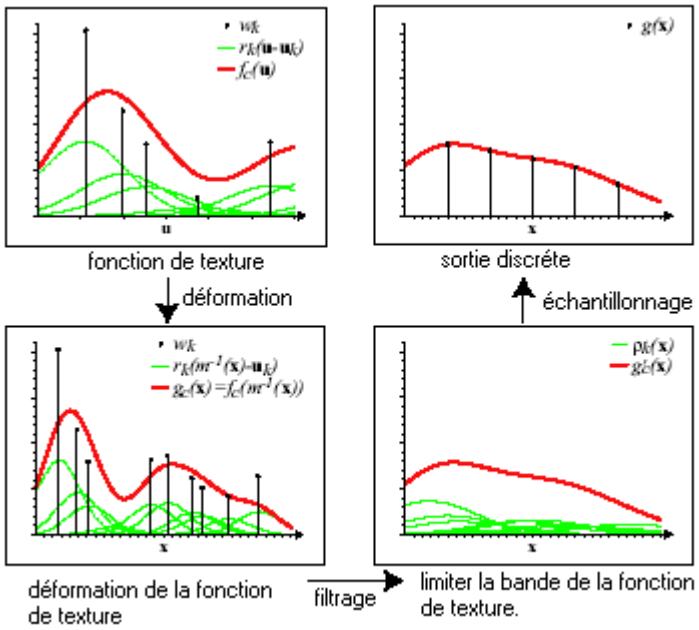


Figure 49 : Déformation, filtrage et prélèvement d'échantillons de la fonction de texture. [Zwicker02c]

- Déformer f_c *(u)* à l'espace écran : La fonction f_c est déformée à l'espace écran. On utilise une approximation affine de plaquage de l'espace objet à l'espace écran.
- Limiter la bande de signal à l'espace écran : La fonction déformée f_c est convoluée avec le préfiltre *h.* aboutissant à la fonction continue g'_c *(x).*
- Echantillonner la fonction continue de production en la multipliant avec un train d'impulsion pour produire la fonction discrète g (x).

Enchaînant les trois premières étapes, la fonction de sortie $g'_c(x)$ est :

$$g'_c(x)=\sum_{k\in N} w_k \rho_k(x) \qquad \textbf{(2.2)}$$

$$\text{Où} \quad \rho_k(x)=(r'_k \otimes h)(x-m_k(u_k)) \qquad \textbf{(2.3)}$$

Ici, le filtre de rééchantillonnage $\rho_k(x)$ est écrit comme une convolution *d'une fonction de déformation de base* $r'_k(x)= r_k(m^{-1}(x))$ et un pré-filtre $h(\boldsymbol{x})$. Pour simplifier l'évaluation de $\rho_k(x)$ à chaque point u_k en employant une approximation locale affine $x= m_k(x)$ de plaquage projectif $\boldsymbol{x}= m(\boldsymbol{u})$ de la paramétrisation de surface locale à l'espace écran.

Notez bien : L*a déformation de la fonction de base* dans le contexte de l'EWA splatting, est similaire à ce que nous avons appelé dans les sections précédentes *« projection d'un surfel à l'espace écran»* (Figure 50).

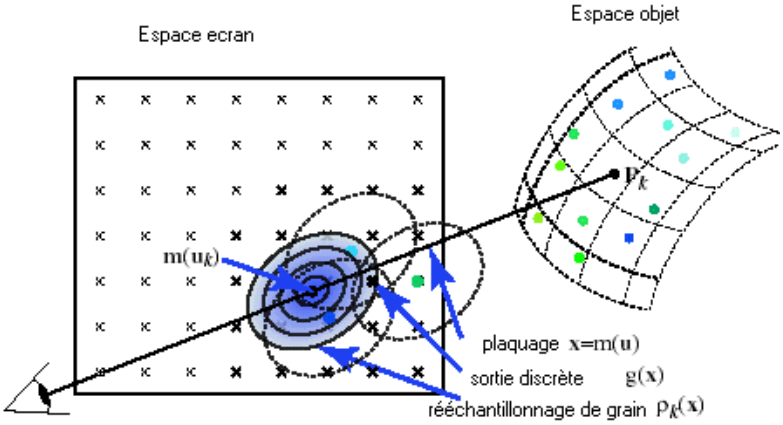


Figure 50 : Le rendu par la surface splatting, le reprélèvement d'échantillons de grains est accumulé dans l'espace écran. [Zwicker02c]

Afin de simplifier et de faciliter l'évaluation de grain de rééchantillonnage $\rho_k(x)$. On remplace les fonctions de base r_k et le préfiltre h par une fonction *gaussienne elliptique.*

La fonction *gaussienne elliptique* 2D $G_V(\boldsymbol{x})$ est défini par :

$$G_V(x) = \frac{1}{2\pi |V|^{\frac{1}{2}}} e^{-\frac{1}{2} x^t V^{-1} x} \qquad \textbf{(2.4)}$$

Où V est la matrice de variance, $|V|$ le déterminant de V. on dénote les matrices de variance de fonctions de base r_k et de préfiltre h avec V^r_k et V^h (V^h est généralement la *matrice d'identité I*), par conséquent $r_k = G_{V^{rk}}$ et $h = G_{V^h}$.

En substituant la fonction de base et le préfiltre gaussien dans l'équation (2.3), on obtient le grain de rééchantillonnage EWA à l'espace écran :

$$\rho_k(x)=\frac{1}{|J_k^{-1}|} G_{J_k V_k^r J_k^t + I}(x-m_k(u_k)). \qquad \textbf{(2.5)}$$

J_k : dénote le Jacobian de plaquage. Pour le calcul de J_k voir [Ren02, Zwicker02c].

❖ **L'algorithme de rendu :** Il ne diffère pas d'algorithme de splatting général (Figure 51). Au début, chaque point P_k est plaqué à la position $m(u_k)$ sur l'écran. Ensuite le grain de rééchantillonnage est centré à $m(u_k)$ et est évalué pour chaque pixel, c'est à dire l'équation 2.5 est employée au calcul de forme de splat. Autrement dit, les contributions de tous les points sont splatté dans un *tampon d'accumulation* (A-buffer).

```
Pour chaque point P [k] {
        Projetez P [k] à l'espace écran;
        Déterminez ρk(x) ;
        Splat ρk(x) ;
}
Pour chaque pixel x dans le tampon de frame {
        Illuminez x;
}
```

Figure 51 : L'algorithme de rendu. [Zwicker02c]

- Résolution de la visibilité : Afin de déterminer les surfaces visibles on utilise une méthode semblable à la *visibilité splatting*.

  Pour se faire on utilise la valeur z du plan tangent à P_k, calculée à chaque pixel couvert par le grain. Pour déterminer si une nouvelle contribution appartient à la même surface déjà stockée dans un pixel : la différence entre la nouvelle valeur z et la valeur de z stockée dans le tampon de frame est comparée à un seuil. Si la différence est inférieure au seuil, la contribution est ajoutée au pixel. Si non, les données de tampon de frame sont remplacées par la nouvelle contribution.

- L'éclairage différé : Suite à la projection de tous les points d'une scène, on illumine le tampon d'image. Cela évite l'éclairage des points invisibles. On emploie un filtre normal pour ombrer chaque pixel. Les paramètres de l'ombre sont accessibles via un index à une table avec des propriétés matérielles. Des méthodes d'ombrage de pixel avancé peuvent être facilement appliquées (comme la réflexion mapping).

Cette méthode a les avantages et les inconvénients suivants :

- Permet de rendre des surfaces transparentes.
- Fournit un anti-aliassage de bord puissant.
- Fournit une transition douce de minimisation au grossissement.
- Plus lente que le splat flou.
- N'est pas supportée par le matériel graphique.

Enfin, L'EWA splatting a été étendu au rendu de volume [Zwicker01] ainsi qu'à la combinaison d'EWA de volume et d'EWA splatting surface vue en [Zwicker02c et Baar01].

3.2.3 EWA splatting surface à l'espace objet

En 2002, *Liu Ren et al.* [Ren02], proposent une nouvelle approche de multi-passage. Ils exécutent le EWA splatting surface sur un matériel graphique de PC, appelé EWA splatting à l'espace objet. L'idée est de dériver une formulation du filtre EWA à l'espace objet (Figure 52). Susceptible à l'accélération par un matériel graphique conventionnel basé sur triangle. L'algorithme de rendu du filtre EWA à l'espace objet est constitué de deux passages:

- Au premier passage de rendu, la *visibilité splatting* est exécutée en décalant les polygones opaques de surfel vers l'arrière, le long des rayons d'observation.
- Au deuxième passage de rendu le préfiltrage EWA est exécuté en déformant la texture plaquée sur les polygones de surfel.

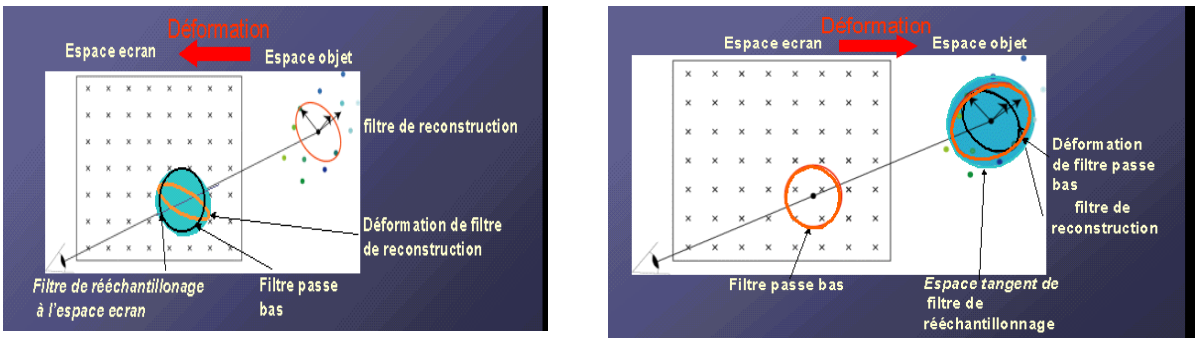


Figure 52 : A gauche, schéma du filtre EWA dans l'espace écran, à droite schéma du filtre EWA dans l'espace objet. [Ren02]

Pour faciliter le processus de splatting, on emploie le plaquage de texture et le mélange alpha. Pour l'éclairage l'algorithme exploite efficacement les capacités des unités de traitement graphique (GPUs), en utilisant le vertex programmable et le pixel shader.

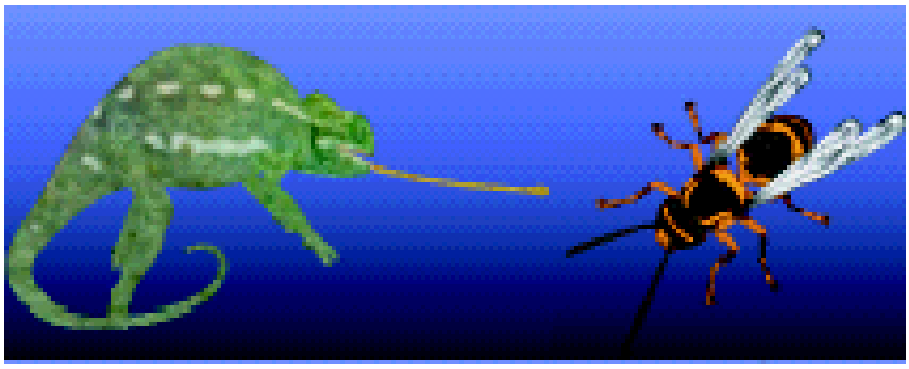


Figure 53 : Exemple de scène générée. [Ren02]

3.2.4 Pointshop 3D

Pointshop 3D est un système de retouche de photo interactif 3D qui est entièrement basé sur les points. Ce système fournit une structure unifiée conceptuelle pour éditer des modèles 3D. Il offre une grande variété d'outils individuels pour changer la géométrie et l'apparition de géométrie irrégulière de point échantillonnée. La portée d'opérations possibles va bien au-delà de la fonctionnalité de systèmes de retouche de photo conventionnels 2D [Zwicker02b]. De plus il supporte les effets de la retexture, sculpture et le filtrage. En général ce système combine l'efficacité de retouche de photo 2D avec la fonctionnalité de systèmes sculpture 3D. le traitement s'accomplit en deux étapes:

- **La transition de l'image à la surface :** En généralisant les pixels d'image 2D vers des points de surface 3D (surfels), donc on remplace l'image discrète I par une surface S basée sur point. De là, on représente un objet 3D comme un ensemble d'échantillons irréguliers de sa surface $S=\{s_i\}$. Puisque les échantillons sont une extension directe de pixels d'image, on les appelle aussi surfels [Reunanen04].
- **Manipulation surface basée sur point S :** Le procédé des opérateurs de manipulation de la surface S, donné par la formule (2.6), est décrit comme suit (Figure 54) :

$$I' = \Omega(\Psi(\Phi(S)), \Psi(B)) \qquad \textbf{(2.6)}$$

Où I' est l'image résultante de ces opérations.

  - L'image de pinceau (brosse) B : Elle est employée comme un outil général pour modifier l'image originale. Selon l'opération considérée, elle peut être interprétée comme un pinceau de peinture ou un filtre discret [Reunanen04].
  - Paramétrage interactif Φ : Il consiste à définir les paramètres de plaquage sur la surface S (i.e. zone de travaille), tout en minimisant la distorsion du nuage de points.

- Rééchantillons dynamique Ψ : La distribution de prélèvement d'échantillons de surfaces est en générale irrégulière ; cette phase consiste à rééchantillonner les points pour avoir un échantillon de point régulier [Baar01 et Zwicker02b].
- Rédaction Ω : Une fois que le paramétrage est établi et le reprélèvement d'échantillons a été exécuté, tous les calculs ont lieu sur des échantillons discrets dans le domaine du paramètre 2D.

Enfin, on utilise le carré plat pour la reconstruction d'image et l'antialiassage.

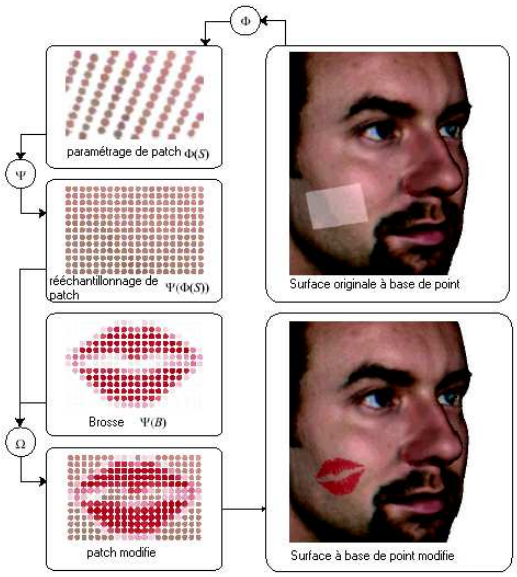


Figure 54 : Vu d'ensemble de la structure d'opérateur pour la rédaction de surface basée sur point. [Zwicker02b]

4 L'approximation géométrique de deuxième ordre

Les techniques de rendu de points avec leur géométrie locale exigent la connaissance de la variation superficielle à n'importe quel point donné. Pour comprendre la variation superficielle à un point, on doit revoir la géométrie différentielle classique qui nous offre un modèle mathématique. Le calcul de courbure sur des surfaces paramétriques a un modèle mathématique robuste. Diverses techniques ont été conçues pour estimer la courbure de représentations discrètes échantillonnées.

4.1 Surface d'un ensemble de points

Cette technique développée par *Marc Alexa et al*, utilise la *géométrie différentielle.* Une surface lisse est caractérisée par l'existence de cartes locales lisses à n'importe quel point. Elle minimise l'erreur géométrique de l'approximation [Alexa01], on rapproche localement la surface avec des polynômes employant le MLS (Moving Least Squares). Elle se compose de trois principales étapes :

- **Définition de la surface :** L'idée principale est la définition d'une procédure de projection, qui projette n'importe quel point près de l'ensemble de point sur la surface. La surface MLS est définie par la projection de points sur eux-mêmes. Cette projection se compose de (Figure 55) :
  - Un calcul du plan de référence local *H* pour le point violet r.
  - Une projection de *r* sur *H* définissant son origine *q* (point rouge).
  - Le calcul de l'approximation du polynôme local *g* aux hauteurs f_i de points p_i sur *H*.
  - La projection de *r* sur *g* (point bleu) est le résultat de la procédure de projection MLS.

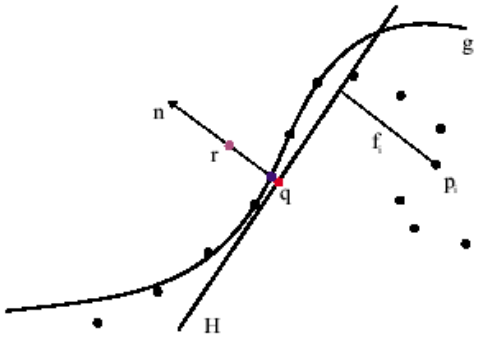


Figure 55 : La procédure de projection MLS. [Alexa01]

❖ **Production d'un ensemble de point de représentation :** La génération de points sur la surface d'une forme est un processus de prélèvement d'échantillons. Le nombre de points est ajusté en *hauts échantillons* ou *bas échantillons* de la représentation [Kobbelt04].

Ce paradigme général est illustré en 2D (Figure 56): les points *P* représentés en violet, définissent une courbe S_P (en violet). S_P est rééchantillonné avec des points $r_i \in S_P$ (points rouges). En général cet ensemble de point appelé *points de représentation* définit maintenant la courbe rouge S_R qui rapproche le *Sp*.

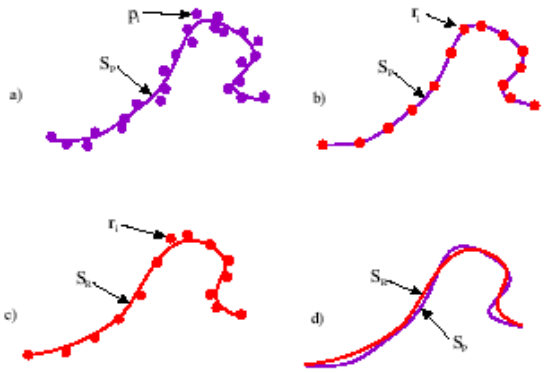


Figure 56 : Illustration du paradigme. [Alexa01]

La technique qui définit et rééchantillonne S_P fournit les propriétés importantes suivantes [Alexa01] :

- Surface lisse et manifold.
- Réduction du bruit.
- Réduction de redondance.

Le résultat de cette étape est une structure hiérarchique semblable à celle de Qsplat. En plus de cette dernière la présente structure stockent aux nœuds feuilles l'orientation de la surface et les coefficients du polynôme associé.

❖ **Le rendu :** Pour afficher la surface d'un ensemble de point, une nouvelle technique de rendu de point est appliquée. L'idée est d'évaluer les cartes locales selon la résolution d'image. Ce qui aboutit à [Alexa01] :

- Une haute qualité de rendu aux taux d'image interactifs.
- Une procédure simple où aucun trou ne doit être rempli dans une étape de post-traitement.

Le rendu est décrit comme une traversée simple de l'octree. On sélectionne les nœuds suivant la sélection hiérarchique de backface [Kumar96] et la vue frustum. Si la traversée atteint un nœud d'une aire projetée à une taille inférieur à un pixel, ce nœud est projeté au tampon de frame, sans traverser son sous arbre. Quand la traversée atteint un nœud de feuille et la mesure de sa projection de sphère englobante est supérieure à un pixel dans l'espace écran, des points complémentaires seront générés [Alexa03].

Enfin, une nouvelle amélioration proposée par *Fleishman et al*, en 2003, appelée surfaces d'ensemble de points progressives [Fleishman03]. L'idée consiste à combiner les cartes de déplacement scalaire multi-niveaux et la représentation de surface à base de points.

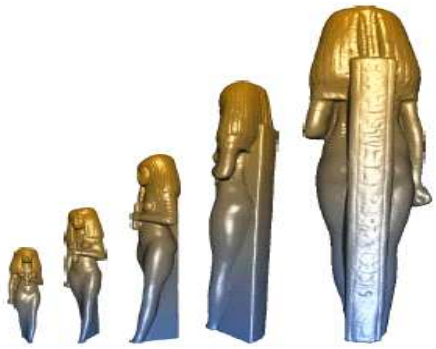


Figure 57 : Série progressive du modèle d'Isis, côté gauche un modèle avec 19K points progressivement raffinés ; vers le haut de 189K points dans le modèle du côté droit. [Alexa01]

4.2 Modélisation et rendu de point avec la géométrie locale

Cette méthode emploi une nouvelle primitive de rendu. Elle combine la brièveté de la modélisation de point avec l'efficacité de polygone. Elle est appelée *point différentiel*. La surface est représentée par un nuage de *points différentiels* (DP), où chaque point possède une information de courbure incorporée ; celle ci capture la géométrie locale différentielle aux alentours de ce point. Le rendu est accéléré par le matériel graphique en utilisant l'ombre de pixel.

- **Points Différentiels :** Un point différentiel (DP) *p*, est une primitive de rendu, construit d'un point d'échantillon ; elle a les paramètres suivants [Kalaiah03]:
  - x_p : position du point.
  - λ_{up} *et* λ_{vp} : Courbures principales.
  - $\hat{u}_p$ et $\hat{v}_p$: Directions principales.

A partir de ces données, on extrait la normale d'unité $\hat{n}_p$ et le plan tangent τ_p de *p*. qui représente l'information de deuxième ordre à chaque DP [Kalaiah03]. On extrapole cette information pour définir une surface S_p, qui sera utilisée pour approcher le voisinage de x_p.

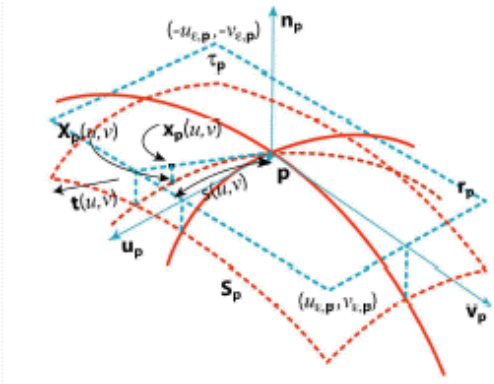


Figure 58 : Paramétrage du plan tangent d'un point différentiel : le plan tangent tp est paramétré par les coordonnées (u, v). Sp emploie les mêmes paramètres qui le rapproche de tp. [Kalaiah03]

En d'autres termes, un point différentiel est défini par la surface S_p, la distribution normale $N_p(u, v)$ et le rectangle r_p. Il est souhaitable de rendre *p* en utilisant *Sp*, simplement une telle primitive de rendu n'est pas soutenue actuellement par le matériel graphique. Au lieu de cela *p* est rendu en employant r_p [Kalaiah01].

- **Prélèvement d'échantillons :** Le modèle 3D est d'abord échantillonné par des points. On utilise les propriétés inhérentes de la représentation de surface, on extrait l'information différentielle de la géométrie à chaque points échantillonnés [Kalaiah03].
  - Si la surface est paramétrique (NURBS). Elle est échantillonnée uniformément dans le domaine paramétrique et des techniques standards décrites à la littérature de la géométrie différentielle, sont employées pour extraire l'information appropriée à chaque points échantillonnés.
  - Si la surface est un maillage triangulaire. On emploie les sommets de la maille comme des points échantillons. On extraie l'information différentielle pour chaque point en employant les techniques développées par Taubin. Ainsi les DPs obtenus du maillage triangulaire ont les mêmes propriétés que ceux obtenus en échantillonnant une surface NURBS.
- **Simplification :** Suite à l'étape précédente, la surface est sur-échantillonnée afin que le rectangle de chaque point différentiel se chevauche suffisamment avec ses voisins sans laisser des trous à la couverture de surface. Cela offre une représentation de surface complète mais contient des échantillons superflus [Kalaiah03]. Donc on suit l'échantillonnage de point par un processus de simplification pour réduire la redondance de DPs.
- **Rendu :** Afin d'afficher un point différentiel p, on utilise le rectangle r_p au lieu de S_p. L'algorithme de rendu est constitué des étapes suivantes [Kalaiah03]:
  - Calcul de la distribution de la normale et la carte normale dans l'espace écran pour être exploité par le matériel
  - Illumination : Pour calculer l'éclairage spéculaire et diffuse à chaque pixel, on a besoin de la distribution de la normale locale, et de la moitié de la distribution vectorielle locale. Cette dernière est obtenue en employant la carte de vecteur du cube offerte par le matériel graphique.
  - Affichage de r_p.

Figure 59 : Illumination et ombre par pixel. [Kalaiah03]

Les avantages de cette représentation sont :

- Comparé à d'autres primitives de point, la surface peut être représentée clairsemé.
- Avec un matériel robuste, elle atteint une accélération.
- Elle offre une nouvelle technique de simplification basée sur point.
- Elle offre une meilleure qualité de rendu qu'une approche purement à base de splat.
- Le temps de rendu de DPs est deux fois plus rapide et exige (environ 75 %) moins d'espace disque par rapport aux primitifs splatting.

L'imperfection de DPs résulte d'une complexité des frontières. Cette dernière limite le calcul de S_p. Elle mène au problème de sur échantillonnage. Une approche de détermination de largeur employant l'information différentielle de troisième ordre peut traiter efficacement ce problème.

5 Les modèles hybrides

5.1 Simplification hybride : combinaison entre les polygones de multi-résolution et le rendu de point

La hiérarchie de multi-résolution des polygones et récemment les points sont les outils familiers et utiles pour réaliser des taux de rendu interactifs. Dans cet esprit *Cohen et al.* présentent (en 2001), un algorithme qui intègre étroitement les deux dans une structure de données hiérarchique. Cette approche est basée sur : une simplification ascendante du processus comportant [Cohen01] :

- Les opérations de simplification de polygone.
- Le remplacement de polygone par des points.
- Les opérations de simplification de point.

L'algorithme produit une hiérarchie hybride comportant des primitives de polygone et de point à partir d'une ou plusieurs mailles de surface. La hiérarchie peut être optimisée selon les caractéristiques relatives aux types de primitives sur la plate-forme de rendu prévue [Cohen01]; à cela deux approches sont prévues pour la construction :

- Approche conservatrice : A certains endroits, elle produit une meilleure hiérarchie qu'une hiérarchie purement polygonale. Elle est approximativement égale dans d'autres.
- Approche moins conservatrice : Elle peut réduire la complexité d'un objet de visionnement lointain ou proches.

La technique permet de réaliser un taux de rendu interactif, où le temps de rendu est inférieur (qui varie entre 1,3 et 4,7) à ceux de la simplification de triangle [Cohen01].

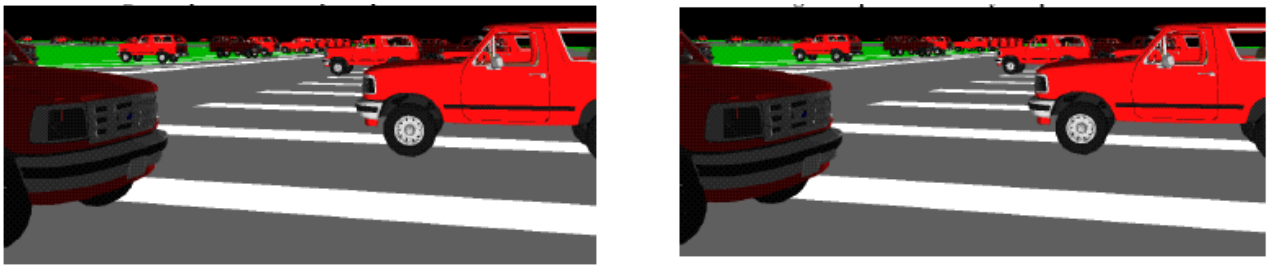


Figure 60 : A gauche la simplification de triangle ; à droite la simplification hybride. [Cohen01]

5.2 POP : un système de rendu hybride entre le point et le polygone pour une grande donnée

Chen et Nguyen [Chen01] présentent (en 2001), une extension simple, efficace au système de rendu de point existant. Au lieu d'utiliser uniquement des points, ils emploient des points et des polygones pour représenter et rendre de grands modèles de maille. D'abord on crée à partir des triangles comme nœuds de feuille une structure arborescente hiérarchique. Ils sont accumulés vers des nœuds intermédiaires comme points [Chen01].

Durant le rendu, le système détermine l'emploi de point (d'un certain nœud de niveau intermédiaire) ou de triangle (d'un nœud de feuille) pour l'affichage. Ceci selon la contribution de chaque nœud à l'écran.

Cette méthode accélère le rendu de grands modèles. Elle compromet peu la qualité de l'image. A cet effet les points sont employés pour accélérer le rendu des objets éloignés ; les triangles sont employés pour assurer la qualité des objets proches. Enfin, la méthode facilite l'anti-aliassage efficace pour le plaquage de texture [Chen01].

6 Bilan

Le rendu à base de points n'est qu'une technique parmi d'autres qui embrasse un large domaine d'infographie. Serte cette science n'a pas encore atteint le but escompté ; néanmoins elle a connu un grand développement tant au domaine technique qu'industrielle.

Dans cette optique, pour en arriver à ce stade, plusieurs chercheurs ont contribué, chacun à sa manière, au développement d'une approche propre à lui. Toutefois, il y a lieu de vous souligner que ces approches se complètent les unes aux autres et comme on peut le constater elles ont toutes leurs qualités et leurs défauts.

En ce qui concerne notre étude. Le rendu à base de points peut être considéré comme une *reconstruction de surface à partir* de ses échantillons de points (Figure 61). D'un ensemble de point, l'algorithme de rendu dessine une douce surface représentée par l'ensemble de points. Cette reconstruction dépend du point de vision et doit être répétée pour chaque changement de vue.

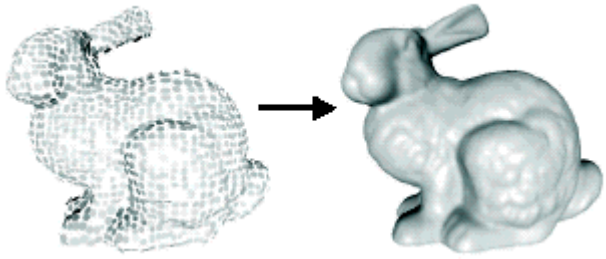


Figure 61 : Rendu à base de points: à gauche nuage de points; à droite la reconstruction de surface. [Krivanek03]

Ils existent diverses approches pour afficher des objets directement à partir d'un nuage de points. La majorité des algorithmes du rendu de point procèdent dans *l'ordre d'objet.* Ces approches se caractérisent par :

- **Echantillonnage d'objet :** A l'aide d'un scanner 3D on obtient un nuage de points en scannant un objet réel [Gross02, Matusik02a, Matusik02b] ou en échantillonnant un objet synthétisé avec un algorithme. Ces derniers sont repartis en quatre catégories:
  - *L'échantillonnage aléatoire :* On choisit d'une manière aléatoire les points sur les surfaces. La distribution est réalisée selon une certaine fonction de densité de probabilité.
  - *L'échantillonnage déterministe :* Il inclut les techniques tel que l'échantillonnage sur treillis régulier, l'échantillonnage de plusieurs vues en utilisant un appareil-photo perspectif ou en prenant seulement les sommets du maillage triangulaire.
  - *L'échantillonnage en prétraitement :* Il fournit un ensemble de vue indépendant d'échantillons qui sont stockés dans une structure de données spéciale et utiliser plus tard au rendu.
  - *L'échantillonnage* dynamique : Afin d'assurer une densité des points prescrite à l'espace écran. Selon le point de vue, on échantillonne les objets en temps réel durant le rendu.
  - *L'échantillonnage sensible de sortie :* Il échantillonne les objets en une résolution égale à la résolution de l'image de sortie prévue comme dans un rendu à base d'image.
  - *L'échantillonnage sensible d'entrée :* Il fournit des échantillons de point indépendamment de la résolution de sortie prévue.
- **Attributs d'échantillon de point :** Chaque point d'échantillon a une position, une normale à la surface et une couleur. Il y a deux types d'échantillon de points :
  - Surfel ou point : Si nous assignons une a*ire* au point, celui ci devient un élément de surface [Pfister00].
  - Points différentiels : Il sauvegarde des attributs supplémentaires par point, ces derniers décrivent les propriétés différentielles locales de surface [Kalaiah01, Kalaiah03].
- **Sélection de visibilité :** Elle est employée presque dans tous les systèmes de rendu. Pour une bonne exécution elle s'avère indispensable. Toutefois elle n'est pas une étape nécessaire dans un système de rendu à base de points.

L'exécution de sélection sur un nuage de points n'est pas très efficace, étant donnée qu'on traite un nombre de points élevé. Elle est donc appliquée sur un groupe de points ou sur une structure hiérarchique. On rencontre trois genres de sélection:

- *Sélection de vue frustum :* elle élimine la géométrie à l'extérieur de la vue frustum.
- *Elimination des faces arrières « backface » :* elle élimine la géométrie orientée loin de la vision (ex : cône de normale).
- *Elimination de l'occlusion :* elle élimine la géométrie occultée par une autre géométrie (ex : les masques de visibilité de *Grossman* et de *Dally*).

❖ **Structures de données :** On distingue trois structures hiérarchiques qui permettent la multi-résolution et la compression:

- *La hiérarchie des sphères englobantes* : souhaitable pour un nuage de points arbitraires.
- *Le cube de couche de profondeur (LDC)* : utilisé pour un nuage de point régulier. Il permet une déformation par incrément.
- *P-grille* : Il est utilisé pour les dispositifs d'affichage réduit. Nuage de point approximé.

❖ **Le pipeline de rendu :** C'est le noyau de n'importe quel système de rendu à base de points (Figure 62). Il accepte des points comme entrée et produit des images comme résultat. Le rendu de point est très efficace, favorable à l'implémentation sur matériel [Krivanek03].

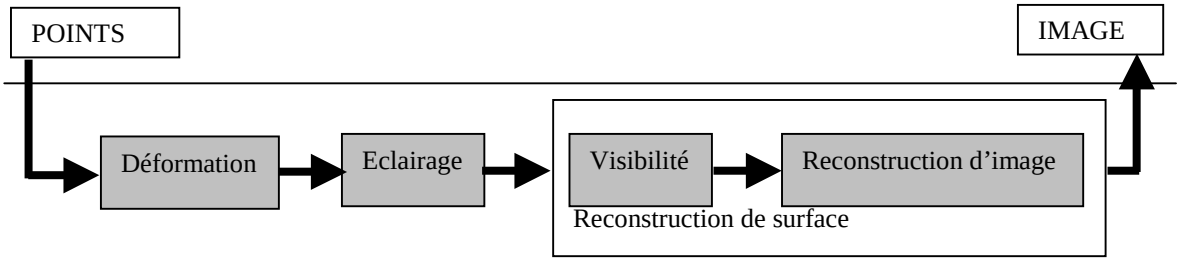


Figure 62 : Le pipeline de rendu de point. [Krivanek03]

- *Etape* de *déformation* : Elle projette chaque point à l'espace écran en utilisant la projection perspective. Elle est réalisée avec le produit homogène de la matrice-vecteur, plus efficacement par incrément [Grossman98a, Grossman98b, Pfister00] ou par la déformation hiérarchique [Rusinkiewicz00].
- *Etape d'éclairage* : Elle est exécutée point par point. A cet effet n'importe quel modèle d'illumination locale peut être appliquer. L'éclairage est souvent exécuté suite à la *visibilité* (seul les points visibles sont éclairés), *ou* après la *reconstruction d'image* (par pixel shading).
- *Reconstruction de surface* : son rôle est d'afficher un nuage de point comme une surface lisse sans aucuns trous et aliassage. Elle peut inclure un filtre d'anti-aliassage. Les différentes techniques de reconstruction de surface suivantes partagent une propriété commune [Krivanek03]: « *la nécessité d'avoir la densité de point d'échantillon* ».
  - Ignorer les trous : est souhaitable à la géométrie non structurée (voir [Max 95 et Duguet 03]).
  - Détection de trous et reconstruction d'image : Il y a une séparation entre l'étape de visibilité et la reconstruction d'image (voir [Grossman 98a] et la visibilité splatting [Pfister 00]).
  - Splatting : la visibilité et la reconstruction d'image forment une seule étape. Elle contient : Le carré plat, l'ellipse, le cercle, le splat flou, normalisation par surfel [Ren02], EWA surface splatting.
  - Haut-échantillonnage : elle consiste à générer dynamiquement des points d'échantillon durant le rendu. Elle contient : la surface MLS[Alexa01], le $\sqrt{5}$ -échantillonnage [Stamminger01] et le z-buffer aléatoire [Wand01].

7 Conclusion

7.1 Avantages des points.

7.1.1 En tant que primitive de rendu

- Une primitive unique, très rapide à exécuter (même dans le logiciel).
- Rendu efficace des objets et des environnements complexes.
- Algorithme de rendu sensible de sortie.
- Pipeline de rendu simple, efficace et facilement parallélisable.
- Contrôle simple et continue du LOD.
- Offre un bon compromis entre la vitesse de rendu et la qualité d'image.
- Simplification du maillage dynamique simple et efficace [Stamminger01].

7.1.2 En tant que primitive de modélisation

- Modification simple de topologie (aucune topologie n'est explicitement stockée).
- Le changement potentiel de la topologie de surface est simple et clair avec les points lors du raffinement adaptatif. Avec un maillage l'exécution de ceci est complexe et souffre des problèmes robustes [Stamminger01].
- Les échantillons de point d'un objet se génèrent de haut en bas, fournissant rapidement des représentations grossières [Stamminger01].

7.2 Inconvénients des points

Ici on se contente de citer les inconvénients des points qu'on considère inhérents et graves aux techniques à base de points :

- La représentation et le rendu des surfaces plates sont inefficaces (aucune cohérence n'est exploitée).
- La représentation des discontinuités des normales de la surface (i.e. plis) est inefficace.
- Le taux d'éclairage d'échantillon n'est pas indépendant du taux d'éclairage de la géométrie.
- Aliassage : malgré les techniques de filtrage adopté au PBR l'aliassage reste irrésolu.
- Manque de connectivité : elle est nécessaire pour exprimer certains effets d'animation, de déformation ou même de plaquage.
- La complexité d'un PBR est indépendante de la simplicité de forme : on génère trop de point pour une simple forme.
- Toutes les techniques énumérées auparavant utilisent l'illumination locale. Aucune d'elle ne propose le calcul de l'illumination globale.
- Le rendu à base de points souffre d'une résolution limitée du nombre de points d'échantillons représentant le modèle. A une certaine distance, la résolution d'espace écran est relativement grande par rapport à la résolution de point d'échantillon, qui causent le sous-échantillonnage.

7.3 Conclusion

Dans ce chapitre nous avons essayé de mettre successivement en relief et en valeur et avec dextérité, une vue globale sur les différentes techniques de rendu à base de points (PBR : Point Based Rendering) récemment développées (tableaux 1). Les points sont très efficaces à la modélisation et le rendu des environnements complexes. Ils sont simples au niveau conceptuel, donc facile à étudier. Un des objectifs principaux de l'infographie est de représenter et visualiser la réalité sans décrire explicitement sa structure ; Les points sont une grande étape vers cette extrémité.

La réalisation particulière doit couvrir par des avantages, les insuffisances. Afin d'acquérir plus de profits nous optons pour l'utilisation d'une représentation hybride. Dans cette optique, il est souhaitable de préconiser une représentation hybride entre les textures volumiques et le rendu à base de points, afin de bénéficier de leurs avantages et minimiser dans la mesure du possible les inconvénients érigés.

Article	**Classe**	**Echantillonnage**	**Structure de données**	**Rendu**	**Purement logiciel**	**Accélération matérielle**	**Filtrage de texture**	**Note**
Levoy et Whitted 1985.	EG	- Déterministe, en prétraitement, sensible de sortie.	Grille régulière de point.	- Splatting gaussien.	O	N	O	- Modification de A-buffer. - Anti-alliassage de bord.
Grossman et Dally 1998.	REP	- 32 vues orthographiques sur un treillis régulier. - Déterministe, en prétraitement, sensible de sortie.	Hiérarchie de bloc de 8*8 points.	- Détection de trous + reconstruction d'image.	O	N	O/N	- Rendu rapide, exige seulement les ressources modestes. - Vitesse de rendu : plus de 5 MPS.
Pfister et al. 2000.	EG	- 3 vues orthographiques. - Déterministe, en prétraitement, sensible de sortie.	Arbre de LDC.	- Détection de trous + reconstruction d'image (i.e. visibilité spaltting).	O	N	O	- Anti-aliassage dans l'espace objet. - La résolution est limitée. - Rendu <10 fps pour un modèle complexe de 10^5 polygones.
Rusinkiewicz et Levoy 2000.	EG	- Scanner 3D. - Déterministe, en prétraitement, sensible d'entrée.	Arbre de sphères englobantes.	- Carré splatting. - Gaussian splatting	N	O	O/N	- Antia-liassage dans l'espace objet : OpenGL : GL-Point. - La résolution est limitée. - Visualisation des scènes complexes de 10^8-10^9 polygones. - Vitesse de rendu 200k-300k pps - Interactif : 8 fps.
Wand et al. 2001.	EG	- Des triangles. - Aléatoire, dynamique, sensible de sortie.	Octree de groupe de triangles.	- Carré splatting: OpenGL, GL-Point ().	N	O	N	- Visualisation des scènes complexes > 10^{14} triangles. - Modification de z-buffer. - Plus rapide que le z-buffer traditionnel.
Zwicker et al. 2002 (Pointshop 3D).	REP	- Déterministe, en prétraitement, sensible de sortie.	Grille régulière.	- Carré splatting. - EWA splatting.	O	N	---	- Logiciel de traitement d'image.
Duguet et al 2003.	EG	- A partir de polygone. - Déterministe, en prétraitement, sensible d'entrée.	P-grille.	- Carré splatting.	O	N	N	- Vitesse de rendu 2,1 fps. - Généralisation de la méthode de Botch.

Zwicker et al. 2001	REP	- De texture et de polygone. - Déterministe, en prétraitement, sensible de sortie.	---	- EWA splat dans l'espace écran.	O	N	O	- Rendu indépendamment de vu. - Vitesse de rendu : 500K pps. - Afficher des surfaces transparentes. - Anti-alliassage de bord.
Ren et al. 2002	REP	- Echantillonnage non régulier. - Déterministe, en prétraitement, sensible de sortie.	---	- EWA splat dans l'espace objet.	N	O	O	- Rendu indépendamment de vu. - Vitesse de rendu : 3 Mpps . - Utilise le plaquage de texture et l'ombre de vertex.
Alexa et al. 2001,2003.	REP	- Scanner 3D. - Déterministe, dynamique, sensible de sortie.	Hiérarchie de sphère englobante.	-Haut-échantillonnage + carré plat.	O	N	O	- Vitesse : 1500- 3500 pps. - Rendu interactif. - Utilisation de polynôme d'ordre 3 et 4.
Kalaiah et Varshney 2001,2003.	REP	- NURBS ou maillage triangulaire. - Déterministe, en prétraitement, sensible d'entrée.	Liste des points Différentiels et leur voisinage.	- Splatting des points différentiels (i.e : r_p).	N	O	N	- Vitesse de rendu : 330k de DPs par seconde.

Tableau 1: Classification des méthodes de rendu à base de points pure.

Légende :

- REP : Rendu de points d'échantillons.
- EG : Echantillonnage de la géométrie.
- N : Non.
- O : Oui.
- --- : Indéfini.
- Pps : Points par seconde.
- Fps : frames par seconde.

Chapitre 3 : Nouvelle représentation de textures volumiques

1 Introduction

Depuis une décennie, la texture volumique a constitué un pivot fondamental de la recherche au sein la communauté de l'infographie. En effet, elle permet une modélisation et une animation efficace des scènes complexes, tel que fourrure, chevelure, forêt ou d'herbe caractérisées par un phénomène de répétitivité d'un échantillon.

Si cette approche a connu un notable succès; elle présente constamment un certain défi (cf. chapitre 1 section 3.4.3). Pour pallier cet handicape ; nous allons développer une étude de méthode hybride plus performante. Elle consiste en un mariage entre un rendu à base de points et une texture volumique. Afin d'aboutir au but escompté, nous allons mettre en oeuvre un traitement probant de plusieurs aspect pratiques avant d'obtenir une méthode réellement applicable. A cet effet, des questions émergent :

- De quelle manière sera spécifiée le contenu des texels, ainsi que la conversion des données existantes en texels (cf. section 3), tout en utilisant une représentation à base de points ?
- Comment peut on décrire une peau volumique (cf. section 4) ?
- Comment plaquer la texture sur une surface (cf. chapitre 4 section 4.3), tout en tenant compte de la déformation au rendu (cf. chapitre 4 section 3) ?
- De quelle manière intègre t'on l'ensemble dans une plate-forme de rendu à base de points (cf. chapitre suivant) ?

Dans ce chapitre nous allons exposer d'une manière profonde et détaillée de notre représentation, tout en essayant de répondre aux deux premières questions (le reste sera pris en compte au chapitre suivant). Nous allons donc entreprendre la description à la section 2 les motivations qui nous ont conduit à l'adoption d'une nouvelle représentation. Nous examinerons par la suite et en section 3 la méthode de construction d'un volume de référence en utilisant une représentation hiérarchique à base de points compact et multi-résolution (LDC tree). A la section 4 nous verrons l'impossibilité à convertir la peau volumique en un nuage de points.

2 Motivation d'une nouvelle représentation

La base d'une texture volumique est *le volume de référence* qui capture la complexité des objets. Ce *volume de référence* est encodé par une structure volumique pleine (hiérarchique et multi-échelle) appelé *octree*. Elle comporte un ensemble d'élément de volume appelé *voxels* [Ratib97]. Le contenu de chaque voxel est une *probabilité d'occultation* anisotrope représentant l'information géométrique. De plus il renferme un comportement photométrique local, qui consiste en un repère local et une *fonction de réflectance bidirectionnelle (BRDF)*, qui permettent au texel de refléter la lumière comme s'il contenait vraiment un morceau de surface.

En effet, cette méthode n'utilise qu'un seul texel de référence, qui est copié virtuellement afin d'éviter la duplication inutile des informations où chaque copie ne conserve qu'un pointeur vers le texel de référence [Neyret 96]. En dépit de cette technique d'optimisation la méthode rencontre des problèmes tel que:

- L'utilisation de différentes techniques de construction de l'échantillon de texture. Chaque technique est adaptée à un type précis de modélisation géométrique (exemple : La scan-conversion 3D pour la CSG, Les fonctions implicites. La voxalisation systématique pour les hypertextures) [Neyret96].
- L'occupation d'un grand espace mémoire traduit par :
  - Un stockage des informations géométriques représentées par l'ellipsoïde comme une micro-primitive (i.e. le stockage des matrices quadratiques, les valeurs propres et les vecteurs propres qui représentent l'ellipsoïde) [Neyret96].
  - Un stockage de l'information photométrique.
  - Un stockage des données inutiles tel que le volume intérieur de l'objet, si celui-ci est un objet solide ou opaque.

- L'animation du texel de référence est quasi impossible [Neyret98a].

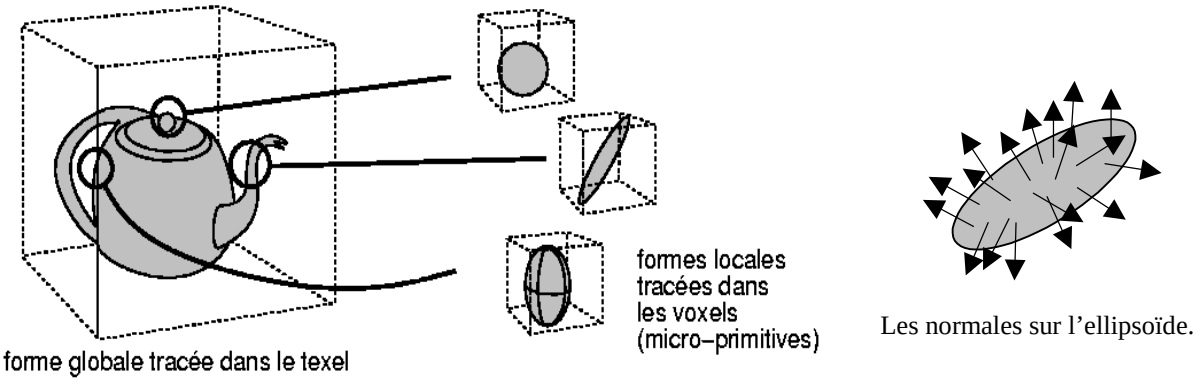


Figure 63 : A gauche : forme globale ; au milieu : forme locale ; à droite : Les informations photométriques. [Neyret96]

Afin de limiter ces inconvénients on a recoure à la représentation à base de points. Celle ci nous permet :

- D'échantillonner la surface de l'objet au lieu de son volume tout en minimisant le taux de stockage.
- De représenter l'information géométrique par un simple surfels (point à zéro- dimension), au lieu d'une probabilité d'occultation (Figure 64).
- De représenter l'information photométrique par l'utilisation des normales du point ; au lieu de calculer la fonction de distribution de normale sur l'ellipsoïde (NDF) pour avoir la BRDF.

De ce qui suit, nous allons faire en détail l'étude sur le choix d'une représentation à base de points en tant que structure de données utilisée pour le stockage et le processus d'échantillonnage associé.

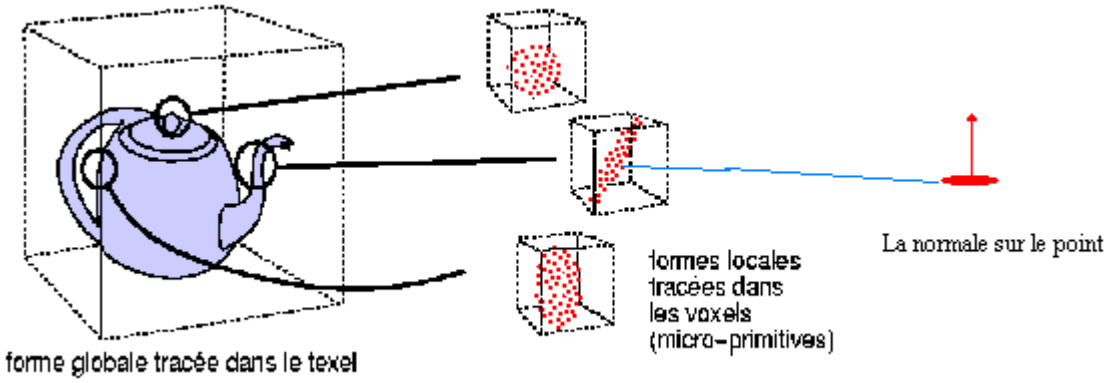


Figure 64 : Représentation de volume de référence par une représentation à base de points.

3 Construction de l'échantillon de texture

Elle consiste à convertir les représentations existantes en nuage de points. Afin de définir une nouvelle méthode de construction du volume de référence ; cette partie repose sur le choix d'une représentation à base de points répondant à certains critères.

3.1 Choix d'une représentation à base de points

Au précédent chapitre, nous avons fait l'étude des différentes structures de données, développées pour le stockage de points. Il s'est avéré que trois structures émergent du lot par leurs aspects de multi-résolution, hiérarchique, ces deux caractéristiques sont indispensables à la gestion d'un modèle complexe. Il s'agit de :

- *P-grille* : C'est la hiérarchie la plus compacte, la plus rapide à construire (cf. 3.1.5) ; du fait quelle utilise une *approximation grossière* des échantillons de points, ce qui la rend, de préférence utilisable aux plates-

formes mobiles. On l'évite pour les images de taille assez grande sur les PC à cause des trous. En sus, selon le modèle d'objet le processus d'échantillonnage change.

- *La hiérarchie de sphères englobantes :* Développée au système Qsplat (cf. 3.1.2), cette hiérarchie a les avantages d'être une représentation multi-échelle, rapide et facile à implémenter. Malheureusement, le coût mémoire d'une telle représentation est assez important car on crée à chaque niveau des surfels fictifs représentant la moyenne des surfels de niveau bas. De plus, elle ne tient pas compte de la méthode d'échantillonnage de points (Les points sont acquis à partir d'un scanner 3D).
- *LDC tree* (Layered Depth Cube Tree) : Il a les avantages d'être une représentation multi-échelle, facile à implémenter puisqu'il décrit une méthode d'échantillonnage de points adéquate à n'importe quel type d'objet (cf. 3.1.1). Il nécessite un espace mémoire réduit ; En sus, le temps d'échantillonnage de la surface est proportionnel à la complexité de l'objet.

Pour des raisons particulières et positives, nous avons fait le choix de la hiérarchie de représentation LDC tree. Celles ci sont :

- LDC tree est plus compact que la hiérarchie de sphères englobantes du fait que plusieurs attributs liés aux surfels stockés deviennent implicites, aussi, la profondeur de l'arbre (il s'agit de structures hiérarchiques) est moins importante dans le cas du LDC tree.
- LDC tree offre une projection par incrément des surfels (on utilise les indices discrets de la grille régulière).
- Possibilité d'utiliser un matériel graphique à la projection des surfels.
- Elle utilise une seule technique d'échantillonnage qui est l'échantillonnage par le lancer de rayon.

3.2 Primitive de base : Le surfel

Le terme de *surfel* traduit l'abréviation *« élément de surface»* ou *« voxel de surface »*, en rendu de volume et en littérature de topologie discrète. *Herman* définit un surfel en tant qu'objet orienté de dimension (n -1) dans $\boldsymbol{R^n}$ [Herman 92]. Pour n = 3, ceci correspond à un carré d'unité orienté (visage de voxel).

Autrement dit, *Un surfel* est une collection de points (de n-tuple de zéro-dimension) avec des attributs de forme et d'ombre qui approxime localement une surface d'objet (Figure 65) [Pfister00], sans aucune information de connectivité avec les autres surfels qui représentent la surface de l'objet. Il existe deux types de surfel [Zwicker02a]:

- Surfel de base : Il comporte la position du point et sa couleur.
- Surfel étendu : Il tient compte du rendu de haute qualité et des effets d'ombre avancé. Il stocke des attributs supplémentaires, tels que la normale, le rayon, bump, BRDF,…etc.

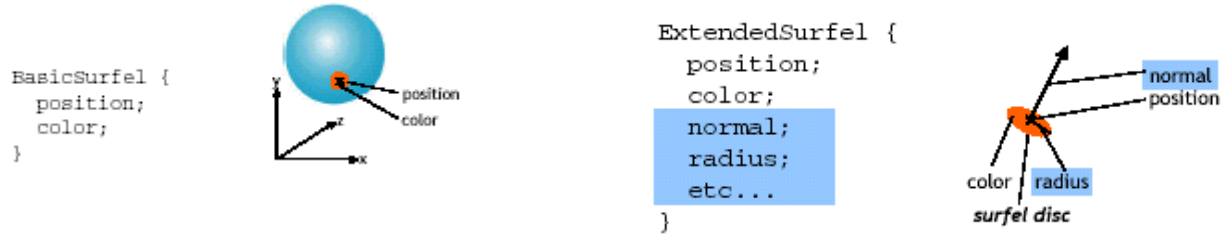


Figure 65 : Type de surfel : à gauche le surfel de base, à droite le surfel étendu. [Zwicker02a]

Pour notre cas, un surfel est représenté par un disque tangent à la surface de l'objet. Donc, il doit stocker une information de position (*x, y, z*) et d'orientation, c'est à dire la normale à la surface de l'objet en ce point. En ce qui concerne l'aspect géométrique, il y a lieu de stocker aussi le diamètre du disque.

Aussi, il y a lieu de souligner que le surfel stocke un index de matériau et une couleur de texture. Les attributs du matériau indexés dépendent du modèle d'illumination que l'utilisateur souhaite utiliser. On peut, par exemple, caractériser un matériau par une couleur et un exposant spéculaires.

3.3 Prélèvement d'échantillon : LDC

Le but du prélèvement d'échantillon est l'obtention d'une représentation optimale de la géométrie avec un minimum de redondance. La plupart des méthodes de prélèvement effectuent la discrétisation d'objet en fonction des paramètres géométriques de la surface, tels que la courbure ou les silhouettes. Cette discrétisation de *l'espace objet* mène typiquement à un nombre élevé ou minime de primitifs pour le rendu.

Pour notre cas, le processus de prélèvement d'échantillon consiste à convertir les objets géométriques et leurs textures en surfels. On emploie la méthode « lancer de rayon distribué » pour créer trois couches d'images de profondeur orthogonales (LDIs) [Pfister00]. Les LDIs stockent de multiples surfels le long de chaque rayon ; un pour chaque point d'intersection de rayon-surface. *Lischinski et Rappaport* [Lischinski98] appellent cet arrangement de trois LDIs orthogonales un cube de couche en profondeur (LDC). L'originalité de cette méthode est la distinction entre le prélèvement de *la forme* et *l'ombre*.

3.3.1 Couche d'image de profondeur (LDI):

LDI est une abréviation de « Layered Depth Image » [Shade98], il s'agit d'images stockées en couches ; où chaque pixel de LDI contient une liste *d'échantillons de scène* correspondant à tous les points de surface dans la scène intersectée par le rayon de projection à travers le centre du pixel (Figure 66). Chaque élément de la liste contient une information de profondeur. Cette information de profondeur est relative au modèle de la caméra liée au plan image. Pour notre cas, il s'agit d'une caméra à projection parallèle [Lischinski98].

Un LDI est donc une matrice 2D où chaque pixel stocke la liste des intersections entre le rayon lui correspondant et la scène (Figure 66). D'où la notion de couche [Shade98]. Pour nous, il s'agit d'une liste de surfels. La structure d'un LDI est récapitulée par la représentation conceptuelle suivante :

```
Surfel = { Position, Couleur, Rayon, SplatIndex,…}
LayeredDepthPixel ={ NumLayers: integer
      Layers[0..numlayers-1]: array of Surfel}
LayeredDepthImage ={
      Camera: camera
      Pixels[0..l,0..h]: array of LayeredDepthPixel}
```

Cependant, un LDI est caractérisé par sa largeur en pixel *l*, sa hauteur *h* et l'inverse de sa résolution *r*, qui correspond à la taille réelle d'un pixel (Figure 67).

Ce stockage offre de multiples avantages. Certains attributs de surfels deviennent implicites, comme leurs coordonnées (x, y) et leur diamètre ; ce qui entraîne une certaine *compacité*. Un second point implique *l'efficacité*. En effet, le stockage des surfels dans une grille régulière permet d'avoir une *approche incrémentale* de la projection [Grossman98a].

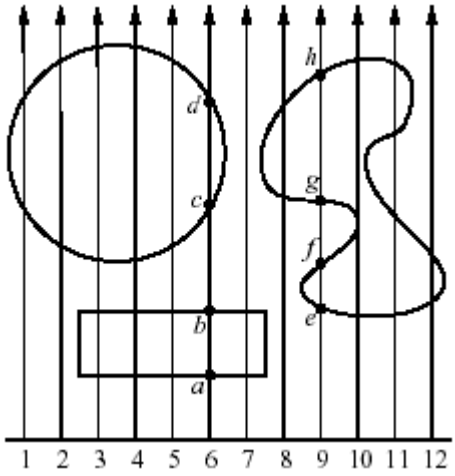


Figure 66 : Un LDI parallèle d'une scène 2D. Le pixel 6 dans le LDI stocke les échantillons de la scène a, b, c, d et le pixel 9 stocke les échantillons de la scène e, f, g, h. [Lischinski98]

3.3.2 Cube de couche de profondeur (LDC) :

Une seule LDI ne permet pas de représenter correctement la totalité de l'objet. En effet, les zones de la surface tangentes par rapport à la direction *z* de LDI sont très mal échantillonnées. Une éventuelle résolution consiste à employer trois LDIs correspondants à trois directions orthogonales. Cet arrangement est appelé LDC, abréviation de « Layered Depth Cube » illustré à la Figure 67 [Lischinski98]. Un LDC est aussi nommé bloc.

En outre, le prélèvement d'échantillonnage LDC nous permet d'établir facilement une structure de données hiérarchique. En plus les caractéristiques du LDI, Le LDC est caractérisé par la profondeur *p* du bloc et d'un repère principal qui est le repère d'un des trois LDI (Figure 67).

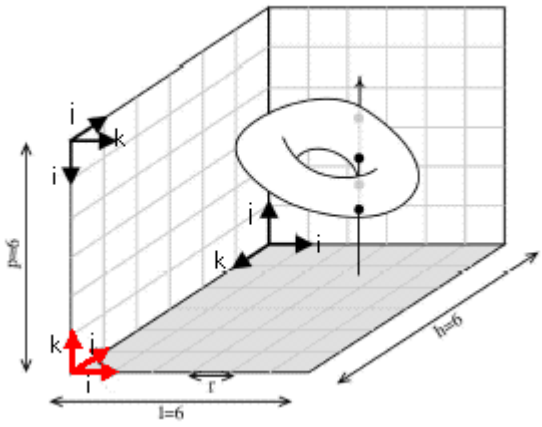


Figure 67 : Un LDC composé de 3 LDI avec leur repère (i, j, k) respectif, le repère rouge et le repère principal. Le rayon associé au pixel (3, 4) du LDI grisée intersecte 4 fois l'objet, ce pixel contient donc 4 surfels. [Guennbaud 02]

3.4 Structure de donnée

L'emploi de LDC tree, qui est une structure de données hiérarchique efficace et afin de stocker LDC acquis pendant le prélèvement d'échantillon, celle ci permet :

- ❖ D'estimer rapidement le nombre de surfels projetés par pixel.
- ❖ Un meilleur compromis entre la rapidité de rendu et la haute qualité d'image.

De ce qui suit nous allons exposer en détail les différentes définitions et techniques d'élaboration de cette hiérarchie. Ensuite, nous procéderons au développement d'une nouvelle technique de construction.

3.4.1 LDC tree de Pfister

Le LDC tree est une combinaison de structure de données « LDC », une hiérarchie de structure de subdivision de l'espace « LDI-tree » ; Il peut être aperçu comme une sorte d'octree où on stocke le LDC à chaque nœud dans l'octree ; ce sont des versions sous échantillonnées de la résolution la plus élevée de LDC. La construction d'une telle hiérarchie se compose de [Pfister 00] :

- ➢ L'octree est construit récursivement de bas vers le haut, sa profondeur est choisie par l'utilisateur.
- ➢ A un niveau plus bas **n = 0 :** on stocke la résolution la plus élevée de LDC acquise pendant l'échantillonnage de la géométrie :
  - Le LDC est subdivisé en blocs de dimension **b** spécifiée par l'utilisateur, c-à-d, les LDIs dans un bloc ont b^3 les couches de profondeur de pixel.
  - Ces blocks sont stockés aux feuilles.
  - **b** est la même à tous les niveaux de l'arbre.
- ➢ Si la résolution la plus élevée de LDC a un espacement de pixel de h_0, alors le LDC au niveau **n** a un espacement de pixel de $h_n=h_0 2^n$.
- ➢ Aux niveaux les plus élevés **n** $\neq$ **0 :** Les blocs de l'octree sont construits par le sous-échantillonnage de leurs enfants par un facteur de deux.

- Noter que les surfels aux niveaux les plus élevés de l'octree mettent en référence les surfels dans le LDC de niveau 0, c-à-d, les surfels qui apparaissent dans plusieurs blocs de la hiérarchie sont stockés seulement une fois et partagés entre les blocs.

➢ Les blocs vides ne sont pas stockés.

La Figure 68a représente le niveau n = 0 de LDC tree avec b = 4 en utilisant un schéma 2D. Dans la figure, les blocs voisins sont différemment ombragés, et les blocs blancs sont vides. Les Figure 68b et 68c montrent respectivement le niveau n = 1 et n = 2 de l'arbre de LDC. Le LDC tree est représenté par la Figure 68d.

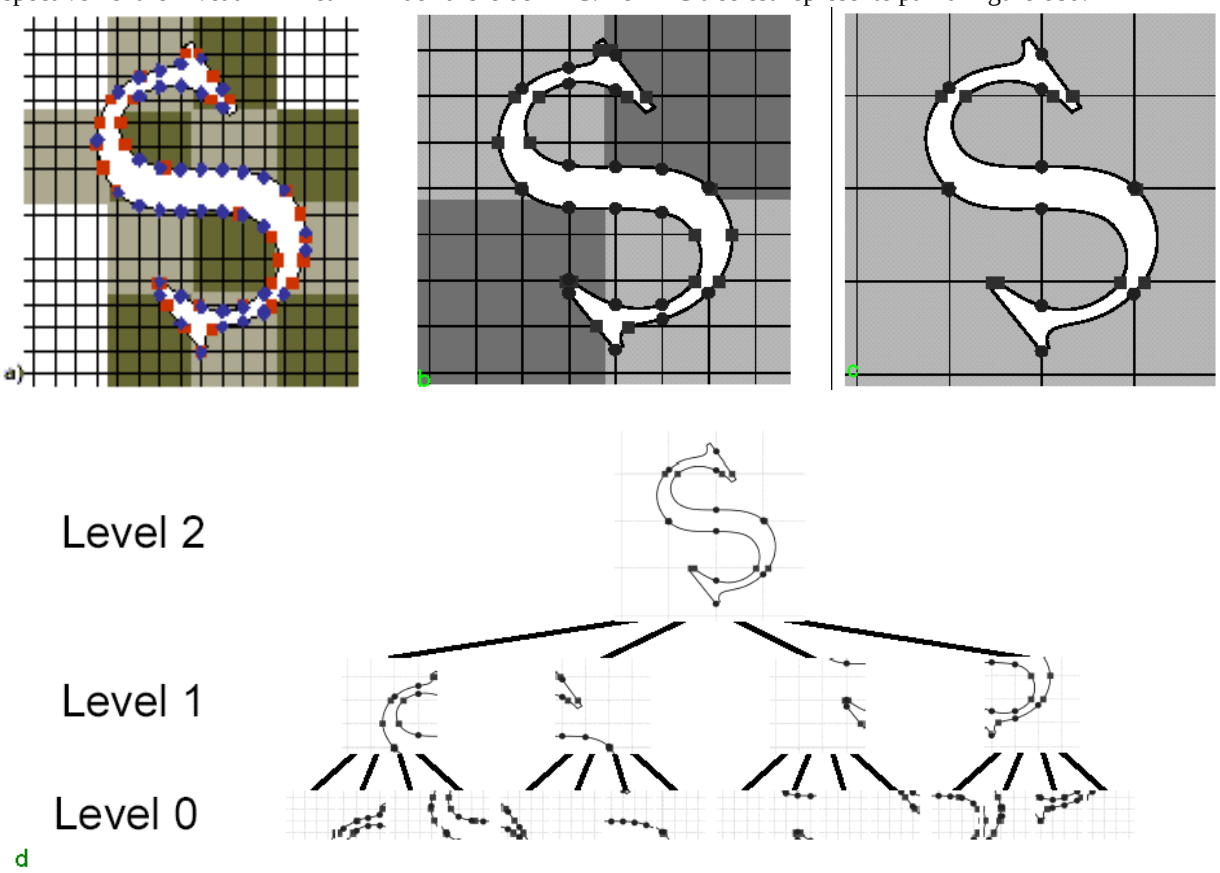


Figure 68 : Trois niveaux de LDC tree représentés en 2D. [Pfister 00]

Conclusion : si on laisse le choix de valeur de profondeur de l'arbre et la dimension b du bloc au service de l'utilisateur; on risque de rencontrer les problèmes suivants :

❖ Perte de l'échelle : la racine ainsi que certains niveaux de LDC tree ne renferment aucune information (voxels vides) : ceci se traduit lorsque l'utilisateur choisi une profondeur élevée ; ce qui implique un espacement entre le pixel h_n à un certain niveau dépasse la taille de LDC. Ces voxels vides seront supprimés ce qui induit un temps de traitement supplémentaire, deux cas possibles émergent:

  ➢ Si on supprime le nœud et ses fils, alors on obtient un arbre vide.

  ➢ Si on supprime seulement le nœud vide sans suppression de ses fils, on obtient un arbre de profondeur proportionnel à la dimension du bloc b, où h_n est inférieur à la taille du LDC.

❖ Perte de la hiérarchie : au cas ou l'utilisateur choisit une basse profondeur, il se trouve devant une situation où la construction du LDC tree est impossible. On cite comme exemple, si l'utilisateur utilise une profondeur = 1 et une dimension du bloc = 4 (Figure 68d) ; il se trouve en face d'une situation où la génération de la racine de l'arbre est impossible (un échec à la construction).

Remarque: On déduit qu'on a une étroite relation entre la profondeur et la dimension du bloc, cela nécessite un nouveau critère d'où : h_n est inférieur à la taille du LDC.[12]

[12] Dans le LDI tree il n'y a aucune relation entre la profondeur de l'arbre et la dimension du bloc; ce qui nécessite un algorithme de splatting lors de la construction du LDI tree.

3.4.2 LDC tree de Guennbaud

La méthode de *Guennbaud* est une variation de celle de Pfister, où LDC tree peut être vu comme une sorte d'octree où chaque nœud est un LDC ; Pour élaborer un LDC tree, on doit suivre les étapes suivantes :

- Construire un LDC réunissant entièrement l'objet voulu.
- Subdivisé celui ci en un nombre de petits blocs de taille b^3 (pour simplifier, on suppose que les blocs sont cubiques, donc $l=h=p=b$).
- Au niveau le plus bas n=0 : Ces blocs forment les feuilles de l'octree.
- Au niveau le plus élevé n≠0 : Le niveau supérieur est construit en fusionnant 8 blocs adjacents (fils) pour former un seul bloc (père) de même taille b^3 mais dont l'espace r entre les pixels est double [Guennbaud02]. Pour la fusion des fils :
  - Un bloc temporaire de taille ($2\ b^3$) est crée par concaténation des huit fils.
  - La réduction de la résolution est réalisée, en faisant la moyenne des pixels des LDI quatre par quatre. Mais ici, les pixels contiennent une liste de surfels.
    - Les surfels des quatre listes à fusionner sont d'abord mis en correspondance (en comparant leur valeur z)
    - puis fusionnés en faisant les moyennes de leurs attributs (profondeur, couleur, normale)

Les niveaux supérieurs de la hiérarchie sont construits au fur et à mesure jusqu'à obtenir un seul bloc, la racine, représentant à lui seul tout l'objet mais à une très faible résolution.

Cette méthode présente deux inconvénients. En premier lieu, durant la création des surfels fictives on augmente la taille de la mémoire nécessaire pour le stockage ; en second, les surfels fictives ne représentent vraiment plus la surface échantillonnée.

3.4.3 L'amélioration que nous apportons à LDC tree

La méthode améliorée dont nous lui réservons une importance particulière et dont nous étudions avec plus de minutie est une approche déjà connue. C'est une amélioration de l'approche de *Pfister*.

L'arbre LDC est un octree comportant un LDC attaché à chaque cellule de l'octree. Ce dernier est adopté vu sa simplicité ; chaque cellule de l'octree comporte en plus une boîte englobante et des indicateurs sur ses huit cellules enfants. La racine de l'octree renferme une boîte englobante de l'objet. Le pseudo code suivant représente la structure de données :

```
LDC_tree_node ={
    Bounding_box [X.. Z, Min.. Max] : tableau de réels;
    fils [0.. 7] : tableau de pointeur sur LDC_tree_node;
    LDC : Layered_depth_cube
}
```

Tous les LDCs dans l'arbre LDC ont la même résolution (Figure 68), qui peuvent être mise arbitrairement. La profondeur de l'arbre LDC s'adaptera aux différents choix de résolution. En général, une résolution de qualité supérieure aboutit à plusieurs niveaux dans l'arbre LDC. La construction d'une telle hiérarchie renferme les étapes suivantes :

- Construire un LDC contenant entièrement l'objet voulu.
- Subdivisé celui ci en un certain nombre de petits blocs de dimension **b** spécifiée par l'utilisateur, c-à-d, les LDIs dans un bloc ont la dimension $b^{3.}$
- L'octree est construit récursivement du haut vers le bas, sa profondeur n est calculée par la formule : n = log(NVF)/log(8).

  Où, NVF est le nombre de voxel fils au niveau le plus bas de l'octree ; il équivaut à $(l/(b*h_0))^3$.
- Durant la construction de l'octree on vérifie si le voxel renferme un ou plusieurs bloc non vide de LDC ; sinon on supprime ce voxel. Afin de réaliser ceci on utilise le domaine de définition du voxel courant.

- Aux niveaux le plus élevés **n ≠ 0 :** les blocs de l'octree sont construits par le sous-échantillonnage de leurs enfants par un facteur de deux.
  - Si la résolution la plus élevée (LDC) a un espacement de pixel de h_0, alors le LDC au niveau **n** a un espacement de pixel de $h_n=h_0 2^n$ et h_n est inférieur à la taille de LDC.
  - Noter : que les surfels qui apparaissent dans plusieurs blocs de la hiérarchie sont stockés une fois seulement et partagés entre les blocs.
- Au niveau le plus bas **n = 0 :** on stocke la résolution la plus élevé de LDC acquis pendant l'échantillonnage de la géométrie :
  - Les blocks résultant de la subdivision de LDC (étape 2) sont stockés à ce niveau (i.e. niveau feuilles).
- Lors de la récursion et afin d'obtenir l'élément de référence on normalise l'arbre LDC. Pour la réalisation, on divise le domaine de définition des blocs et la position des surfels sur la résolution d'image d'échantillonnage.

3.5 Réduction du LDC

Il est souvent utile de réduire les LDC en un seul LDI sur la base de bloc par bloc où les surfels des deux autres LDI sont alors insérés dans le LDI principale. L'idée est de *rééchantillonner* les surfels aux endroits entiers de la grille générée par les intersections de rayon (Figure 69) ; tout en arrondissant la valeur z des surfels (cela entraîne une dégradation de la représentation, mains pratiquement cela n'est pas trop gênant pour les objets ne présentant pas des détails géométriques fins). Afin de réaliser ceci, on emploie l'interpolation du plus proche voisin afin de trouver les nouveaux surfels. Les avantages de cette méthode sont [Pfister00] :

- Réduire le stockage par la diminution de redondance induite par les trois LDI.
- Réduire le temps de rendu par un facteur de trois, du fait qu'il est moins coûteux de projeter un seul LDI au lieu de trois.

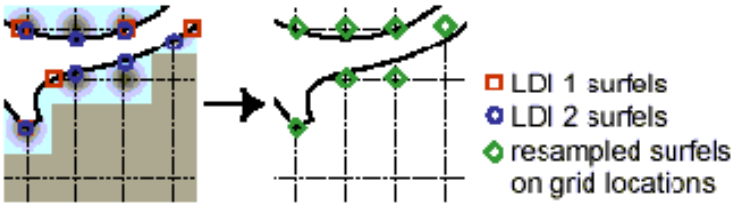


Figure 69 : Exemple de réduction d'un LDC en un LDI. [Pfister 00]

Le pseudo code suivant représente une structure de données réduite :

```
LDC_tree_node ={
    Bounding_box [X.. Z, Min.. Max] : tableau de réels;
    fils [0.. 7] : tableau de pointeur sur LDC_tree_node;
    LDI : Layered_depth_Image
}
```

3.5.1 Interpolation du plus proche voisin

L'interpolation du plus proche voisin est une méthode simple pour la reconstruction d'une fonction (image) ; elle prend pour chaque position la valeur du point d'échantillon le plus proche. Ceci a comme conséquence une fonction par morceaux constants (qui est discontinue, à moins que la fonction elle-même soit constante). La couleur d'un pixel dans la nouvelle image est la couleur du point d'échantillon le plus proche de l'image initiale [Theubl99].

Comme tout type d'interpolation, l'interpolation du plus proche voisin est fondamentalement un procédé de filtrage linéaire. En d'autres termes, Elle correspond à la convolution des points échantillonnés à une fonction rectangulaire (Figure 70) :

- L'interpolation du plus proche voisin est exprimée par :

$$f_1(x, y) = g_s \text{ (auteur de (x), auteur (y))}. \quad \textbf{(3.1)}$$

Qui assigne au point (x, y) la valeur de brillance du plus proche point g dans le rastre discret. Où $f_1(x, y)$ est le résultat d'interpolation, et l'indice 1 indique l'interpolation du plus proche voisin.

- Le grain d'interpolation (filtre) h_1 est défini via

$$h_1^1(t) = \begin{cases} 1, & \text{si } t \in [-0.5, 0.5] \\ 0, & \text{si non} \end{cases} \quad \textbf{(3.2)}$$

$$par \quad h_1(x,y) = h_1^1(x) h_1^1(y). \quad \textbf{(3.3)}$$

Les avantages de cette méthode sont :

- L'interpolation du plus proche voisin est la méthode la plus simple et la plus facile à implémenter.
- Elle peut être utilisée pour réduire ou augmenter le nombre de points d'échantillons : par exemple, La couleur d'un pixel dans la nouvelle image est la couleur du pixel le plus proche de l'image initiale. Si on agrandit l'image à 200%, un pixel sera agrandi à une zone de 4 pixel avec la même couleur que le pixel initial.
- La plupart des logiciels de traitement d'image emploient ce type d'interpolation pour agrandir une image digitale pour un examen plus étroit ; du fait qu'il ne change pas l'information de couleur de l'image et ne présente aucun *anticrénelage.*

En conclusion : Cette technique présente une quantité d'erreur de position ; celle ci équivaut à la moitié d'un pixel. Cette erreur est perceptible sur des objets qui ont des frontières en ligne droites qui peuvent apparaître pareilles ou non après la transformation.

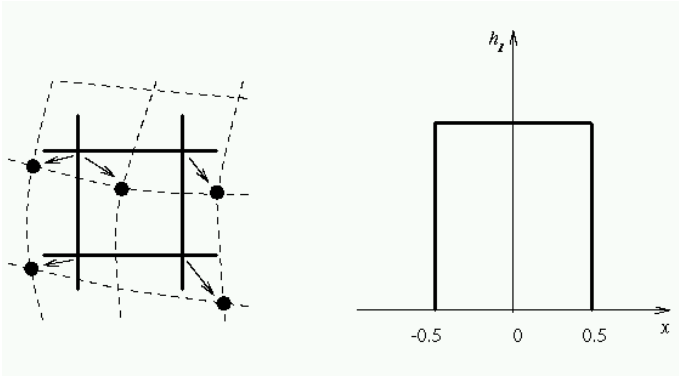


Figure 70 : Le côté droit de la figure montre comment la nouvelle brillance est assignée. Les lignes tirées montrent comment la transformation inverse planaire qui plaque la grille de l'image de production dans l'image d'entrée. Les lignes solides montrent la grille de l'image d'entrée. [Theubl99]

4 Modélisation de la surface sous-jacente (la peau)

La surface sous-jacente est le support sur laquelle on plaque les texels. Celle-ci est composée soit de patches bilinéaires (cas *Kajiya*) ou de quadrangles modélisés à partir d'un maillage quadratique de la surface, plus un vecteur hauteur (d'épaisseur) donné en chaque sommet de la surface pour contrôler le *coiffage* des texels, formant ainsi un ensemble de boites constituant une peau épaisse et continue à la surface de l'objet (Figure 71).

Traditionnellement, nous construisons une liste de surfaces, une liste de faces et une liste de sommets. Les adjacences (entre faces) sont prétraitées de sorte que chaque face pointe vers ses voisines. La structure liée à un point contient ses coordonnées spatiales, ses coordonnées textures et un vecteur hauteur. On associe également une sphère englobante à la surface entière et une sphère englobante pour chaque boite afin d'accélérer la vue frustum.

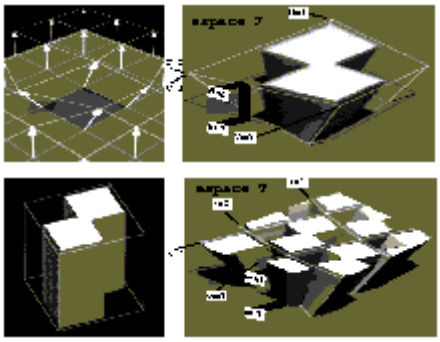


Figure 71 : Système de coordonnées de l'espace liée à l'objet. [Neyret96]

Malheureusement, on ne peut pas modéliser la surface sous-jacente avec une représentation à base de points à cause de la nécessité de l'information géométrique entre les sommets qui nous permettent de coiffer les texels et manipuler la surface sous-jacente (Figure 72).

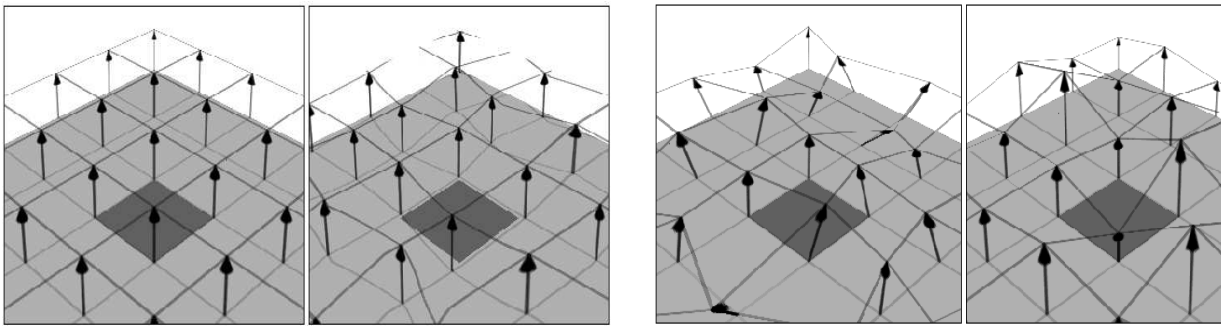
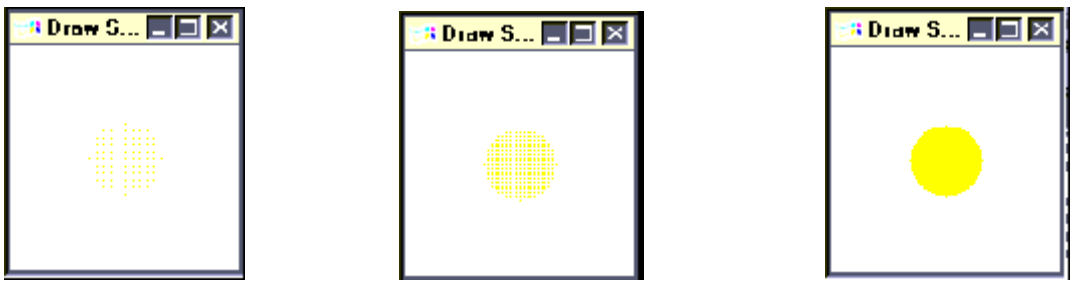


Figure 72 : Perturbations continues : placage non perturbé ; perturbation des coordonnées texture, de l'orientation des vecteurs, de leur amplitude. [Neyret96]

5 Résultats et discussion

Les résultats suivants sont obtenus en utilisant le langage C++ et la bibliothèque graphique OpenGL. Nous employons une résolution de prélèvement d'échantillons de 128^3 pour le LDC. Le rendu à base de points est réalisé par GL_POINT qui permet d'afficher 5 MPS (Million de point par second). L'élimination des parties cachées est réalisée par le z-buffer de OpenGL ; plus l'éclairage.

5.1 Le sous échantillons

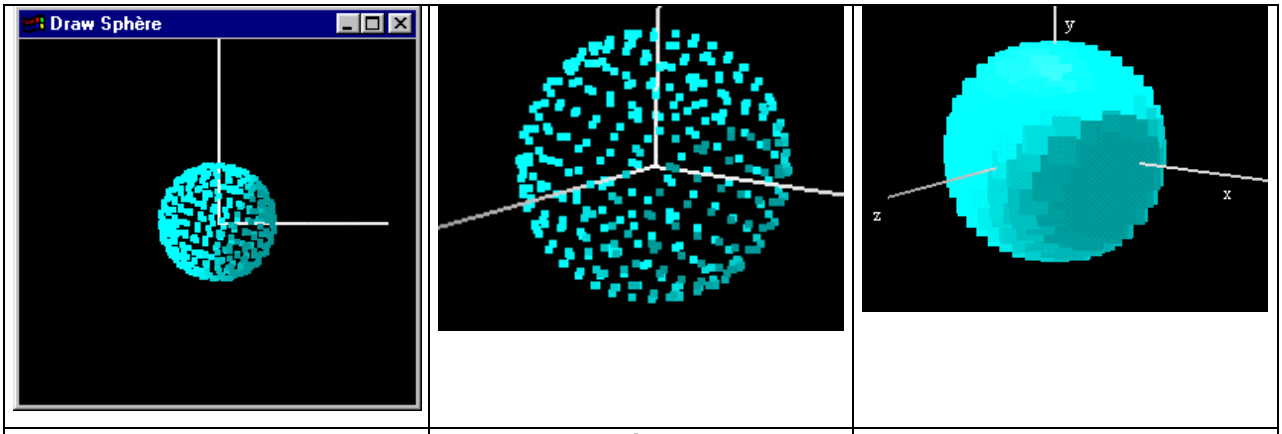


a b c

Figure 73 : Effet de trous[13] : les figures a, b, c montrent le rendu de LDC tree (a : la racine, b : le niveau 1, c : les feuilles « niveau 0 ») ; dans les figures a et b apparaît l'effet de trous du fait qu'on a qu'un sous-ensemble d'échantillons total ; dans la figure c : la sphère est pleine sans aucun trous.

[13] On inverse les couleurs de l'image pour une meilleure impression.

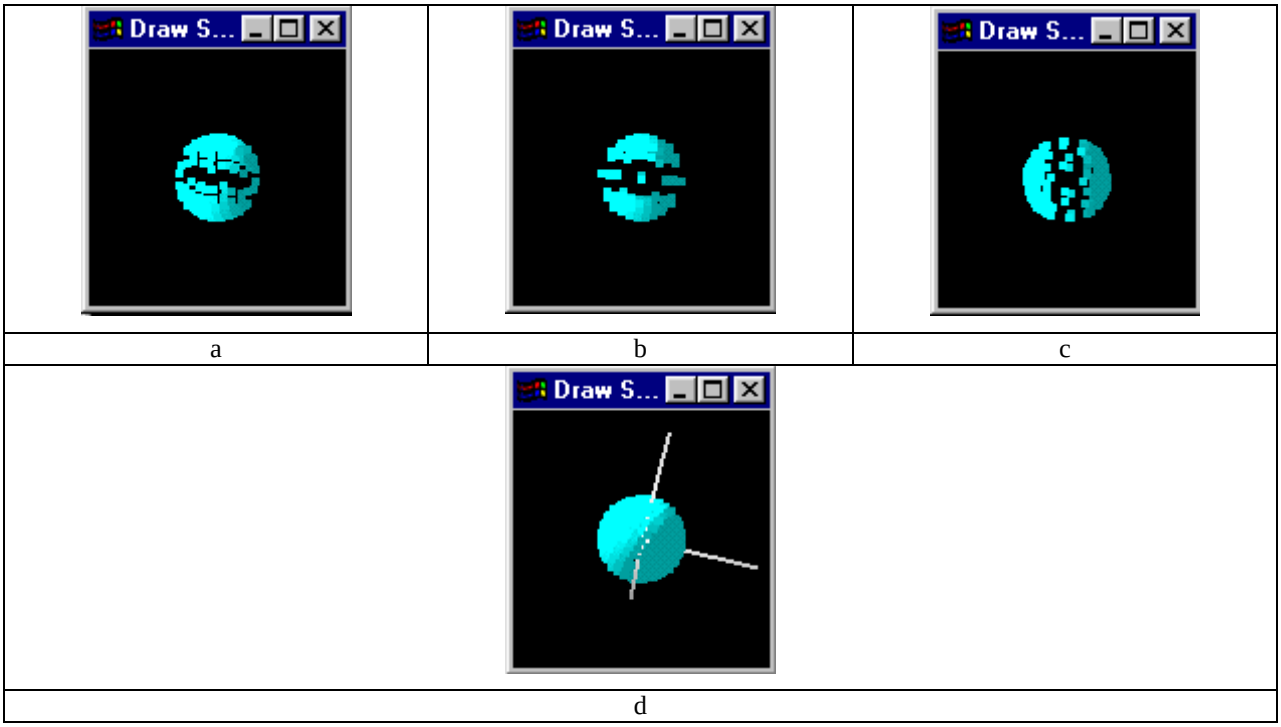
5.2 L'effet de projection et d'agrandissement



a	b	c

Figure 74 : Effet de projection perspective et d'agrandissement : Ces figures sont les mêmes que les figures a, b, c précédentes ; sauf que la projection orthographique est remplacée par une projection perspective et l'unité d'agrandissement est 4 fois l'unité d'agrandissement de la construction de LDC. Dans la figure c : en perçois bien cet effet sur l'effet de trous.

5.3 Effet tangent



a	b	c
d		

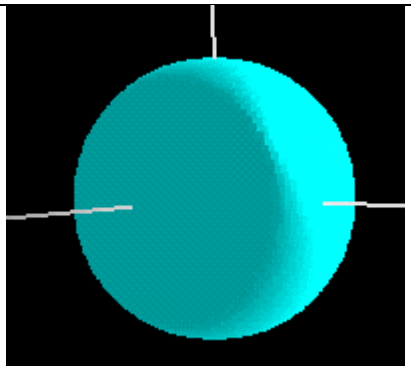
Figure 75 : Effet tangent : les figures a, b, c montrent le rendu de LDI selon la direction (a : l'axe de x, b : l'axe des y, c : l'axe des z), où la surface tangente par rapport à la direction (x, y, z) respectivement de LDI sont très mal échantillonnée. (d) le rendu de LDC correspondant.

5.4 Comparaison entre le volume de référence généré par Octree et par LDC tree :

5.4.1 Temps de construction et espace mémoire

Le Tableau 2 et 3 montrent que le volume de référence généré par LDC tree est plus compact en espace mémoire et nécessite un temps de construction très réduit par rapport à ceux générés par l'octree. On perçoit une rapidité lors de l'affichage.

	Statistique	
LDC	Nombre de surfels	7506
	Espace mémoire occupé	271,21 ko
	Temps de construction	320 ms
LDC Tree	Nombre de voxel	16
	Nombre de feuilles	8
	Espace mémoire occupé	271,61 ko
	Temps de construction	410 ms
	Temps de rendu	160 ms
	Dimension de bloc	32
	Niveau de résolution	2
LDC Tree réduite	Nombre de surfels	4182
	Espace mémoire occupé	151.33 ko
	Temps de construction	2s 190ms
	Temps de rendu	110 ms



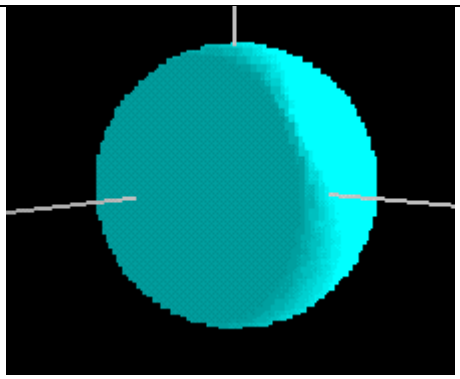


Tableau 2: Statistiques d'un volume de référence généré par LDC Tree

Statistiques	
Nombre de voxels	8136
Nombre de feuilles	5952
Temps de construction	550 ms
Temps de rendu	22 s 410 ms
Espace mémoire occupé	3.28 MO
Niveau de résolution	5

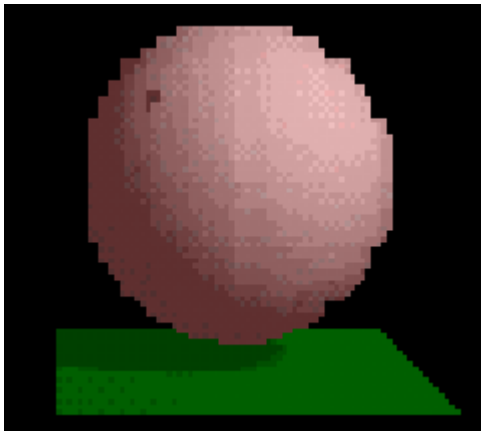


Tableau 3 : Rendu d'un volume de référence généré par l'octree.

5.4.2 Qualité d'image

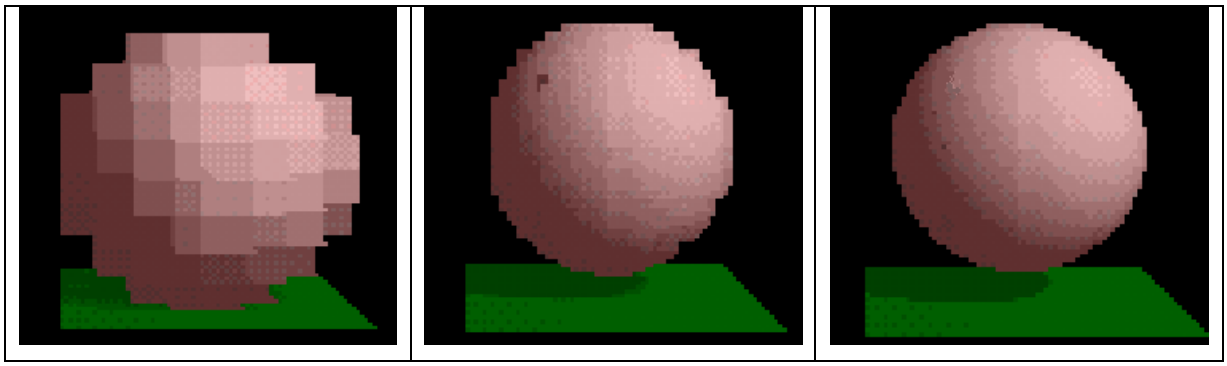


Figure 76 : Le volume de référence généré par l'octree : Texel à un niveau de résolution de 3 (à gauche), de 5 (au milieu) et de 7 (à droite).

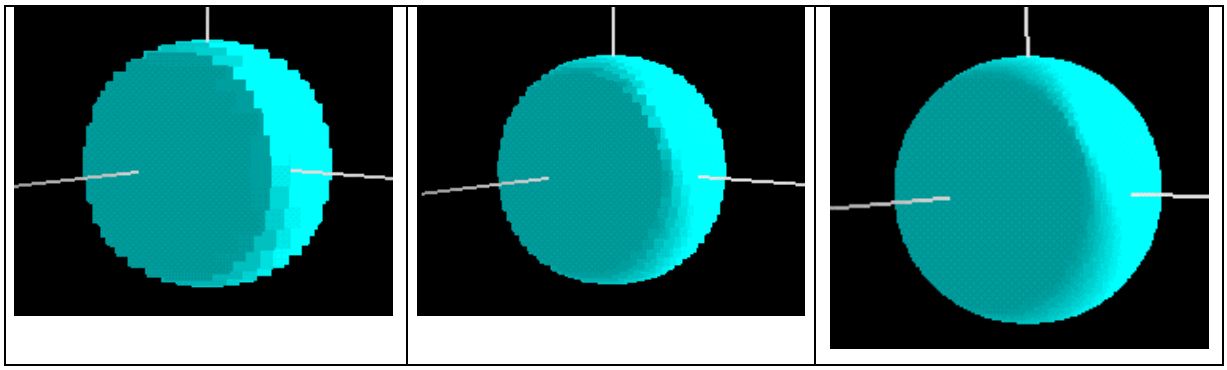


Figure 77 : Le volume de référence généré par LDC tree. L'arbre de profondeur égal à 2, à gauche : Rendu de la racine, au milieu rendu de niveau 1 ; à droite rendu de feuilles (niveau 0).

On remarque (Figure 76 et 77) que pour avoir une image de haute qualité, on doit construire au préalable un octree de profondeur élevé. Paradoxalement en LDC tree il faut avoir un échantillonnage adéquat pour aboutir à un meilleur résultat.

5.4.3 Relation entre espace mémoire et niveau de résolution

Au Tableau 4, on constate qu'il y a une étroite relation entre le niveau de résolution et l'occupation de l'espace mémoire ; ainsi l'octree occupe un important espace mémoire. Au contraire l'espace mémoire n'augmente pas proportionnellement avec celui du niveau de résolution (Tableau 5 et 6).

Niveau de résolution	Temps de construction	Temps de rendu	Nombre de feuilles	Nombre de voxels	Espace mémoire
3	10 ms	16 s 40 ms	384	456	0.18 Mo
4	60 ms	20 s 910 ms	1728	2184	0.88 Mo
5	550 ms	22 s 410 ms	5952	8136	3.28 Mo
6	3 s 310 ms	39 s 270 ms	21696	29832	12.06 Mo
7	19 s 940 ms	3 m 5 s	89472	119304	48.24 Mo

Tableau 4 : Statistiques de rendu d'une sphère généré par la texture volumique.

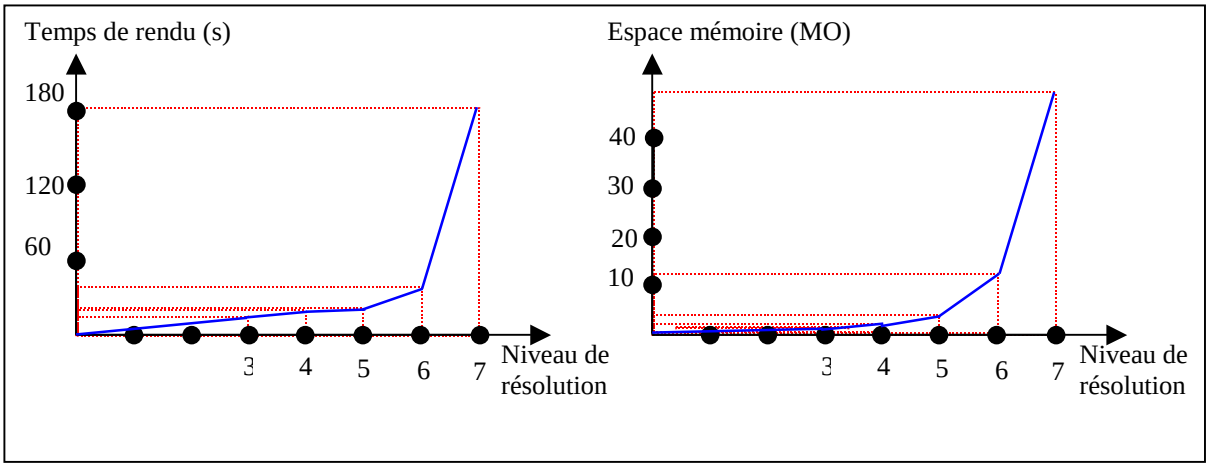


Figure 78 : Texture volumique classique ; Relation entre résolution du volume de référence et le temps de rendu (gauche), entre la résolution et l'espace mémoire occupé (droite).

Le graphique de droite (Figure 78) montre une mémoire utilisée ; afin de conserver le volume de référence en fonction du niveau de résolution choisie. Le graphe de gauche indique le temps de rendu en fonction du niveau de résolution du texel.

Niveau de résolution	Temps de construction	Temps de rendu	Nombre de feuilles	Nombre de voxels	Espace mémoire
2	410 ms	160 ms	8	16	271,61 ko
3	485 ms	170 ms	32	48	272,22 ko
4	530 ms	165 ms	116	164	273,61 ko
5	710 ms	160 ms	378	523	277,90 ko
6	1 s 160 ms	160 ms	1078	783	281,01 ko

Tableau 5 : Statistiques de rendu d'une sphère générée par le rendu à base de points (LDC tree).

Niveau de résolution	Temps de construction	Temps de rendu	Nombre de feuilles	Nombre de voxels	Espace mémoire
2	2s 190 ms	110 ms	8	16	151,33 ko
3	2s 430 ms	110 ms	32	48	151,71 ko
4	2s 990 ms	110 ms	116	164	153,10 ko
5	3s 100 ms	110 ms	378	523	157,39 ko
6	3s 710 ms	110 ms	1078	783	160,50 ko

Tableau 6 : Statistiques de rendu d'une sphère générée par le rendu à base de points (LDC tree réduit).

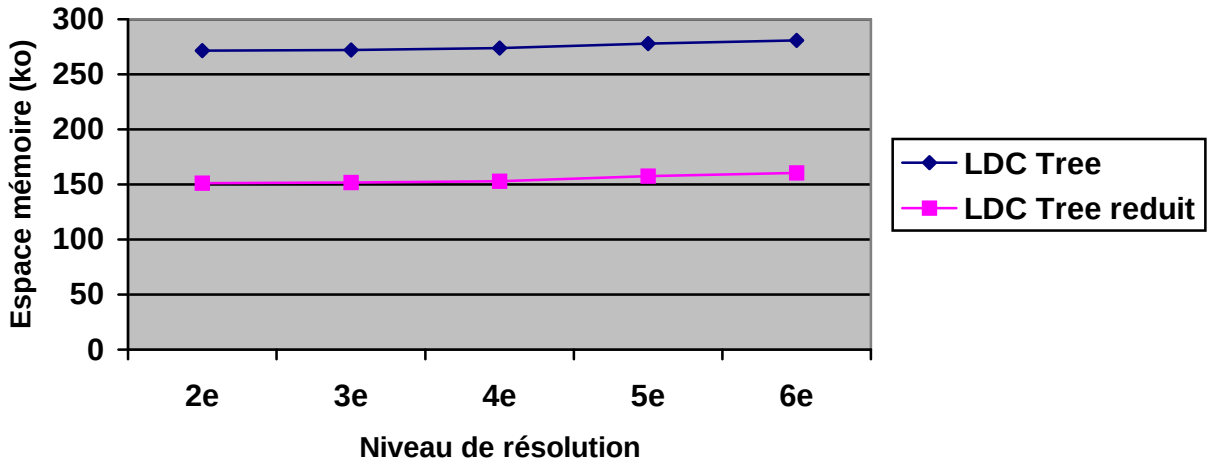


Figure 79 : Comparaison entre LDC tree et LDC tree reduite en espace mémoire.

La figure 79, montre que LDC tree réduite occupe moins d'espace mémoire que LDC tree ; ceci est justifié par le processus de réduction « bloc par bloc ».

5.4.4 Relation entre temps de rendu et niveau de résolution

On remarque que le rendu de LDC tree réduite est plus rapide que celui de LDC tree. Ainsi, le temps de rendu ne dépend pas de niveau de résolution, mais plutôt du nombre de points. Finalement le rendu de LDC tree est plus rapide que celui de texture volumique

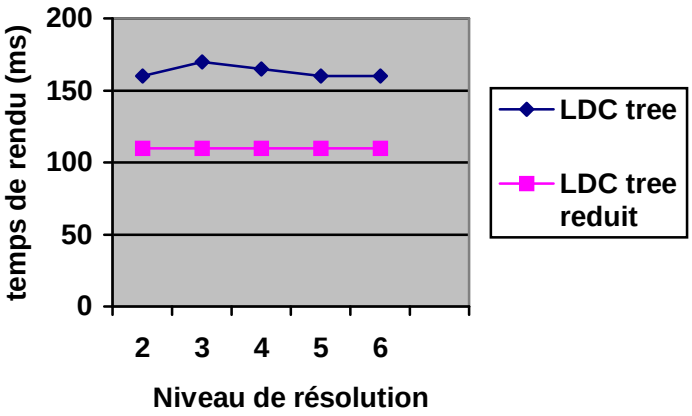


Figure 80 : Comparaison entre LDC tree et LDC tree réduite en temps de rendu.

6 Conclusion

Ce chapitre avait pour but d'étendre le noyau de texture volumique classique, tout en conservant la notion de multi-échelle et le mip-mapping. De plus, en réalisant une représentation souple de différent objet à modéliser en moindre coût de temps de calcul et espace mémoire. Afin de réaliser ceci, on utilise le LDC-tree normaliser comme volume de référence, ce qui permet d'avoir :

- A chaque voxel, on stocke un LDC au lieu d'une fonction de réflectance représentée par la distribution de normales d'un ellipsoïde, ceci permet de réduire le stockage de volume de référence.
- Ces voxels sont organisés en LDC-tree, ce qui permet à la fois de comprimer fortement le volume et de représenter les données à plusieurs échelles. Celui-ci constitue le volume de référence de texture.
- La possibilité d'utiliser le LDC-tree comme le mip-mapping en 2D.

Le chapitre suivant nous permettre d'intégrer le texture volumique dans une plate-forme de rendu à base de points.

Chapitre 4 : Le processus de visualisation

1 Introduction

Le pipeline du rendu classique consiste à prendre des surfels LDC tree et les afficher. Pour se faire on emploie la sélection de visibilité hiérarchique et la déformation des blocs. Cette étude va nous permettre d'examiner en détail la manière d'adapter un rendu classique LDC-tree (Lorsque LDC-tree représente un texel) à la texture volumique (Figure 81). Celle ci consiste à dupliquer et à plaquer le volume de référence au sein de la surface d'un objet, de même qu'une texture 2D. La figure 82 décrit succinctement les cinq étapes intervenant lors d'un rendu d'échantillon de points :

1. **Test de visibilité :** D'une manière conservatrice, on sélectionne un ensemble de blocs visibles par la vue frustum tout en exploitant le multi-échelle de chaque bloc.
2. **Déformation de bloc :** Chaque texel va prendre la forme de l'une des boites constituant la peau volumique, ainsi que la position du lieu où il sera plaqué.
3. **Projection :** Chaque bloc potentiellement visible est projeté dans le tampon image et le z-buffer hiérarchique. A cette étape les points ne sont pas éclairés; ils sont simplement copiés au tampon image.
4. **Construction d'image :** Elle consiste à réaliser les étapes suivantes.
   a. **Découverte de Trous :** Suite à la projection de tous les blocs. Afin de détecter les trous, on emploie les profondeurs se trouvant au z-buffer hiérarchique.
   b. **Anti-aliassage de bord :** A cette étape, on élimine les effets d'aliassage qui apparaissent au niveau de la frontière d'une surface.
   c. **Remplissage de Trous :** Finalement, les trous détectés à la première étape (découverte de trous) doivent être colorés.
5. **L'éclairage :** est exécuté à l'espace image (écran). Chaque pixel du premier plan possédant un poids différent de zéro est illuminé selon le modèle d'éclairage de *Phong*.

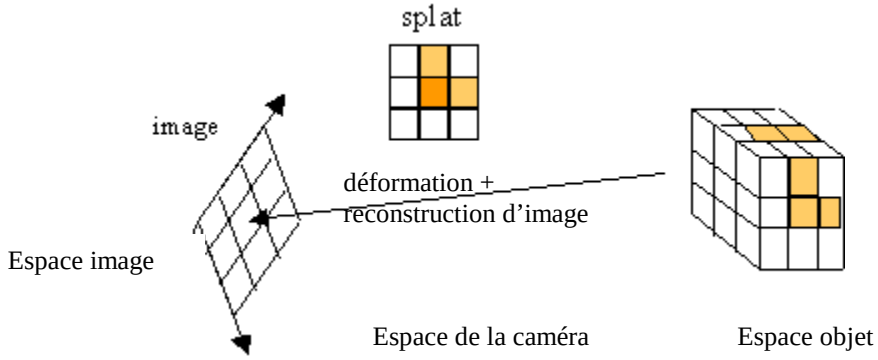


Figure 81 : Le rendu classique de surfels.

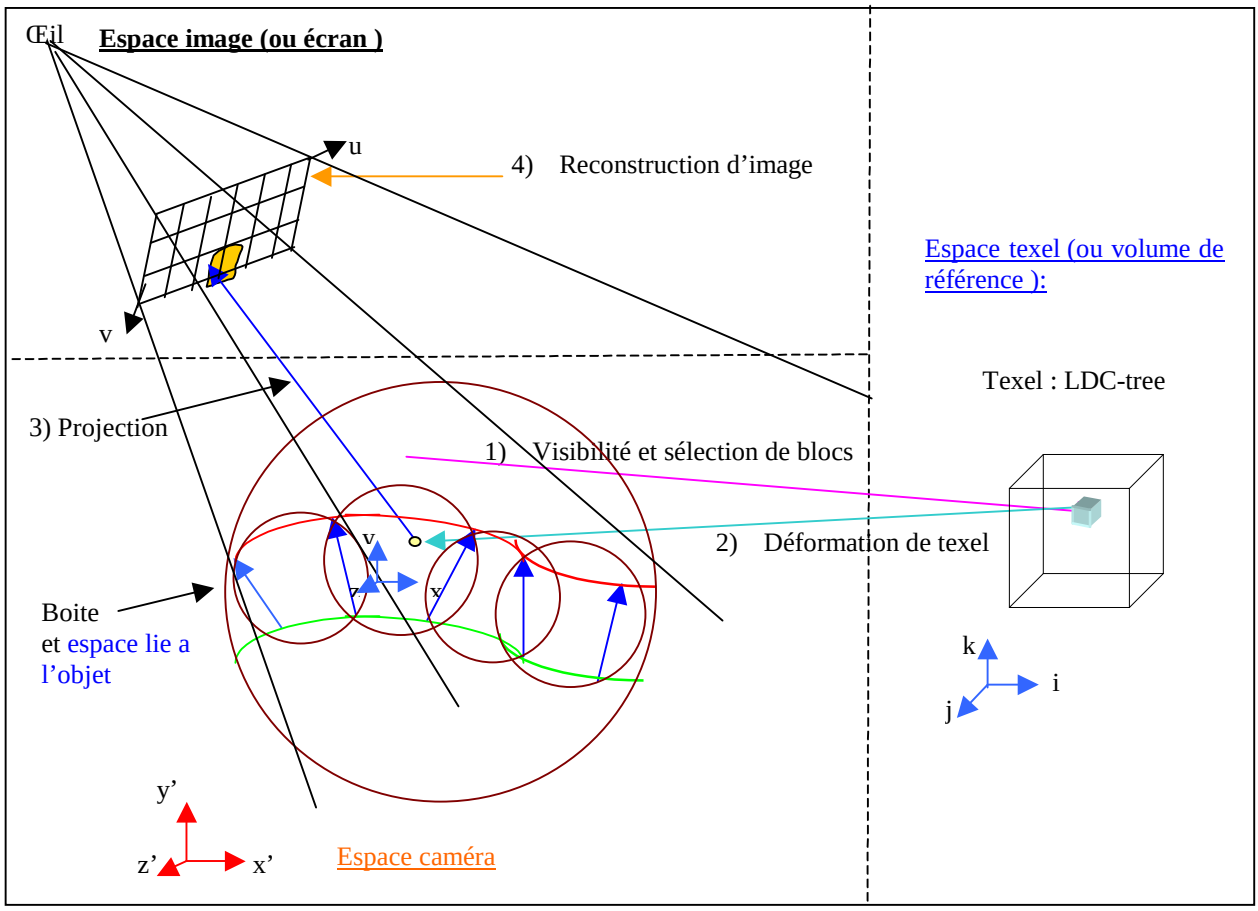


Figure 82 : Le rendu de surfels adapté aux textures volumiques.

2 Visibilité et sélection de blocs

La façon la plus simple pour édifier une image d'une représentation d'échantillon de point est d'afficher tous les blocs. Cependant, du point de vue de donné, un sous-ensemble seulement des blocs est visible. Si on arrive à identifier ce sous-ensemble on pourra alors réduire le temps de rendu, en projetant seulement ces blocs contribuant à l'image finale. Ainsi, il y a lieu de souligner que le calcul exact de cet ensemble est généralement impossible, ceci nous mène à estimer d'une manière rapide et conservatrice l'ensemble des blocs visibles en utilisant quelques astuces de test de visibilité.

2.1 Sélection de la vue Frustum

La première méthode qu'on utilise pour tester la visibilité d'un bloc est de chercher si ce dernier se trouve à l'intérieur de la vue frustum[14] [Grossman98a]. Généralement, la vue frustum est définie par un *plan proche*, un *plan lointain*, ainsi que les *quatre plans* qui passent par la caméra et les côtés de l'écran (Figure 83). Enfin la sélection de la vue frustum est simplement un processus de détermination qui définit si les blocs sont visibles dans cette région [Assarsson00].

L'avantage majeur de cette méthode découle de l'utilisation d'une approche hiérarchique de subdivision de l'espace. Si nous rejetons un nœud de niveau supérieur, nous rejetons automatiquement ces fils. Item si nous gardons un nœud de niveau supérieur (soit visible), nous ne testons pas ces fils, du fait qu'ils sont automatiquement visibles [Assarsson99].

Cette méthode consiste donc à calculer la distance entre chacun des six plans et le centre de la sphère englobante d'un bloc afin de déterminer si le bloc se trouve entièrement à l'extérieur, à l'intérieur ou intersecte un de ces plans. Enfin de cette distance trois possibilités sont envisagées :

[14] région de l'espace qui est dans le champ de vision de la caméra.

- Si la valeur absolue de la distance est inférieure au rayon de la sphère, ceci implique que la sphère croise le plan (Figure 84a).
- Si la distance est supérieure à 0 (Figure 84b), ceci implique que la sphère se trouve devant le plan (probablement à l'intérieur du frustum).
- Si elle est inférieure à 0 (Figure 84c), la sphère est derrière le plan et certainement à l'extérieur du frustum.

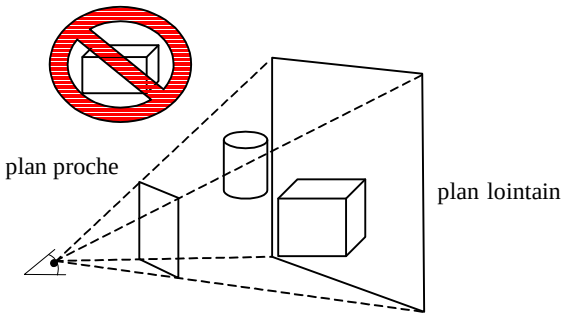
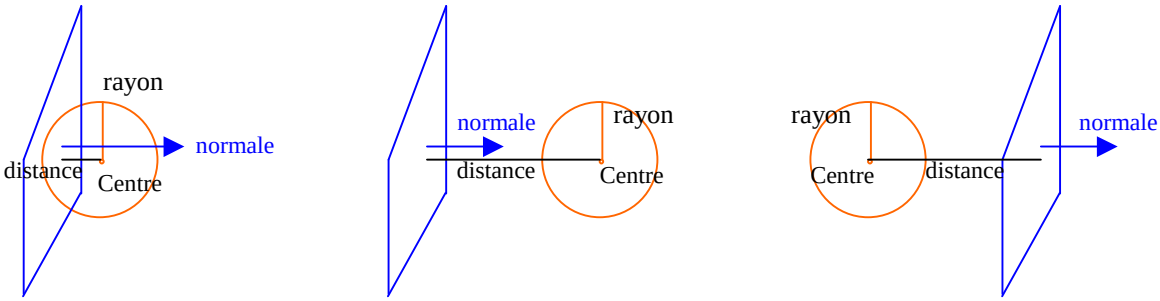


Figure 83 : Vue frustum.

De plus cette méthode permet d'éliminer les parties de la surface volumique se trouvant à l'extérieur de la vue frustum, de même que précédemment on utilise les sphères englobantes des boites de la surface volumique.



a) Sphère croise le plan b) Sphère devant le plan c) Sphère derrière le plan

Figure 84 : Position d'une sphère par rapport à un plan.

2.2 Optimisations

On remarque que le code d'intersection occupe souvent un assez grand pourcentage de notre unité centrale. Cela dépend du type application, le nombre d'objets, la profondeur de notre LDC tree et divers autres facteurs. De plus, nous pouvons remarquer que les méthodes d'intersection sont assez intensives. Même la méthode la plus rapide pour évaluer la sphère contre le frustum exige la mise à l'épreuve de chaque sphère contre les 6 plans du frustum. A cette fin, on propose deux techniques d'optimisation :

- Test d'intersection sphère-sphère : C'est une méthode facile et rapide à mettre en œuvre. L'idée est de construire une sphère autour du frustum (Figure 85), puis on fait un test d'intersection des deux sphères (si l'une est dans l'autre, s'il y a une intersection ou non).
- Test d'intersection sphère-cône : cette méthode est utilisée pour déterminer le croisement d'une sphère et un cône. La construction du cône qui entoure le frustum est simple. Cependant, le cône est défini par un sommet (l'origine du cône), un rayon d'axe et un angle d'ouverture du cône [Eberly02]:
  - Le sommet du cône est la position de la caméra.
  - Le rayon d'axe du cône est la direction de vision de la caméra.
  - L'angle du cône peut être égal à l'ouverture du champ de vision du frustum, donc nous créons un cône qui coupe les coins du frustum.

En conclusion on construit cette sphère et cône, afin de rejeter rapidement les nœuds du LDC tree en testant d'abord l'intersection entre la sphère englobante du nœud courant et la sphère du frustum puis avec le cône du frustum, avant de mettre en œuvre la vue frustum (puisque ses tests sont rapides).

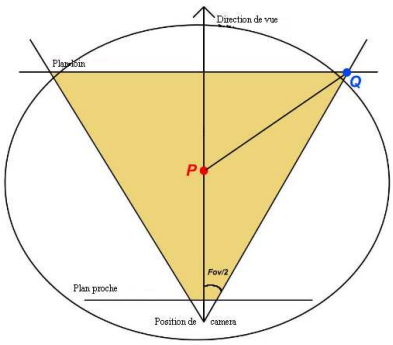


Figure 85 : Sphère frustum.

3 Déformation du texel

L'étude des surfels exposée précédemment ne prend en compte ni la projection de surfels dans la peau volumique, ni la déformation globale du volume de référence et donc de chacun de ces blocs. Cependant, la texture volumique prend en charge ce problème en utilisant l'interpolation trilinéaire et le lancer de rayon courburé (ce dernier génère un temps de rendu exorbitant). Néanmoins, nous allons étudier en détail la substitution d'une interpolation trilinéaire complexe par une interpolation bilinéaire simple.

En premier lieu, nous allons mettre en lumière et avec précision le fond du problème. La forme et la position d'un volume de référence, appliqué à la surface d'un objet est complètement défini par la position des quatre coins du texel en contact avec la surface de l'objet et des quatre vecteurs hauteurs en ces points (Figure 86), c'est à dire défini par la forme et la position de boite formant la peau.

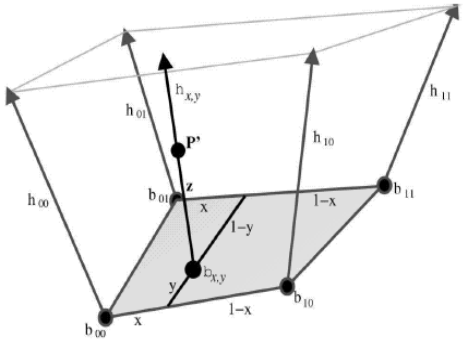


Figure 86 : Paramétrisation de la déformation d'un texel par les quatre sommets bi,j et les vecteurs hauteurs hi,j.

En second lieu, l'interpolation trilinéaire utilisée au sein de texture volumique est définie par :

$$p'(x', y', z') = (1-z)*((1-y)*((1-x)*b000 + x*b100)+ y*((1-x)*b010 + x* b110)) + z* ((1-y)*((1-x)*b001 + x*b101)+ y*((1-x)*b011 + x* b111)). \quad \textbf{(4.1)}$$

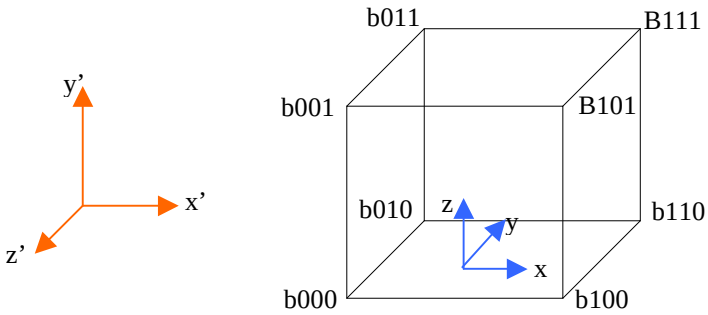


Figure 87 : Le passage de l'espace texl à l'espace scène par interpolation trilinéaire.

A partir de ces données, on remarque que les quatre points de la face haute de la boite peuvent être calculés à partir des quatre points de la surface sous-jacente et les quatre vecteurs hauteurs à ces points, on a donc:

$$b00= b000,\ b10= b100,\ b01= b010,\ b11= b110. \quad \textbf{(4.2)}$$

$$b001=b00+ h00,\ b101=b10+ h10,\ b011=b01+ h01,\ b111=b11+ h11. \quad \textbf{(4.3)}$$

En substituant dans l'équation 4.1 l'équation 4.2 et 4.3 on obtient :

p'(x', y', z') = (1-z)*[(1-y)*((1-x)*b00 + x*b10)+ y*((1-x)*b01 + x* b11)] + z*[(1-y)*((1-x)*b00 + x*b10)+ y*((1-x)*b01 + x* b11) + (1-y)*((1-x)*h00 + x*h10)+ y*((1-x)*h01 + x* h11)].

Ici on met : $b_{x,y,z}$ =(1-y)*((1-x)*b00 + x*b10)+ y*((1-x)*b01 + x* b11).

$h_{x,y,z}$ =(1-y)*((1-x)*h00 + x*h10)+ y*((1-x)*h01 + x* h11).

On obtient : p'(x', y', z') = (1-z)* $b_{x,y,z}$ + z*[$b_{x,y,z}$ + $h_{x,y,z}$].

= $b_{x,y,z}$ + z* $h_{x,y,z}$.

En résumé, Soit *p* un point du motif sans déformation de coordonnées *(x, y, z)* définies dans le repère du volume de référence. Soit *p'* l'image de *p* par déformation, de coordonnées *(x', y', z')* définies dans le repère objet. Le calcul de *p'* peut être effectué comme suit:

- Calcul de $b_{x,y,z}$ résultat de l'interpolation bilinéaire de b_{00}, b_{10}, b_{01} et b_{11} par les coefficients *x* et *y* :
$$b_{x,y,z} = lerp(\ y,\ lerp(\ x,\ b_{00},\ b_{10}),\ lerp(\ x,\ b_{01},\ b_{11}) \quad \textbf{(4.4)}$$
- Calcul de $h_{x,y,z}$ résultat de l'interpolation bilinéaire de h_{00}, h_{10}, h_{01} et h_{11} par les coefficients *x* et *y* :
$$h_{x,y,z} = lerp(\ y,\ lerp(x,\ h_{00},\ h_{10}),\ lerp(\ x,\ h_{01},\ h_{11}) \quad \textbf{(4.5)}$$

Où *lerp* est la fonction d'interpolation linéaire : $lerp(t, A, B)=(1-t)A+tB$. On a alors :

$$p' = b_{x,y,z} + z.h_{x,y,z}. \quad \textbf{(4.6)}$$

4 Projection

Dans cette section nous allons étudier l'algorithme de rendu à base de points (décrit par *Grossman* [Grossman98a]) qui projette chaque surfels au pixel le plus proche dans l'image finale. Nous allons aussi démontrer la manière d'optimiser les calculs en introduisant le calcul incrémental et son adaptation à la déformation du volume de référence. Cependant, la projection est réalisée sur un z-buffer ordinaire[15]. Finalement nous allons faire l'étude de l'utilisation de cet algorithme au plaquage de texture volumique (cf. § 4.3).

4.1 Choix de niveau de L'octree

Durant le rendu, l'arbre LDC est traversé de la racine aux feuilles. Afin de sélectionner le niveau de l'octree à projeter, on choisit un nombre de surfels par pixel pour chaque bloc. A cela, deux possibilités émergent [Pfister00]:

- Le choix d'un surfel par pixel pour un rendu rapide (Figure 88a).
- Ou un choix de plusieurs surfels par pixel pour une haute qualité d'image (Figure 88b).

Pour chaque bloc au niveau *n* de l'arbre, le nombre de surfels par pixel est prédéterminé par la distance maximale entre les surfels adjacents dans l'espace image i^n_{max}. On calcule i^n_{max}, on divise la longueur maximale des

[15] On ignore pour l'instant le z-buffer hiérarchique qui sera développé en section 5.1.

quatre diagonales principales projetées de la boîte englobante par la dimension *b* du bloc. Ce calcul est correct pour une projection orthographique. Cependant, si on emploie la projection perspective on rencontre une petite erreur qui n'a aucune influence car le bloc est projeté typiquement sur un nombre restreint de pixel.

A chaque bloc, i^n_{max} est comparé au rayon r'_{rec} du filtre désiré de reconstruction de pixel. r'_{rec} est typiquement $\frac{\sqrt{2}}{2}s_0$, où s_0 est la longueur du coté d'un pixel. Si i^n_{max} du bloc courant est supérieur à r'_{rec} donc ses enfants sont traversés. Nous projetons le bloc dont le i^n_{max} est inférieur à r'_{rec}, nous affichons approximativement un surfel par pixel. Il est à noter que le nombre de surfels par pixel peut être augmenté si i^n_{max} est une fraction de r'_{rec} [Pfister00].

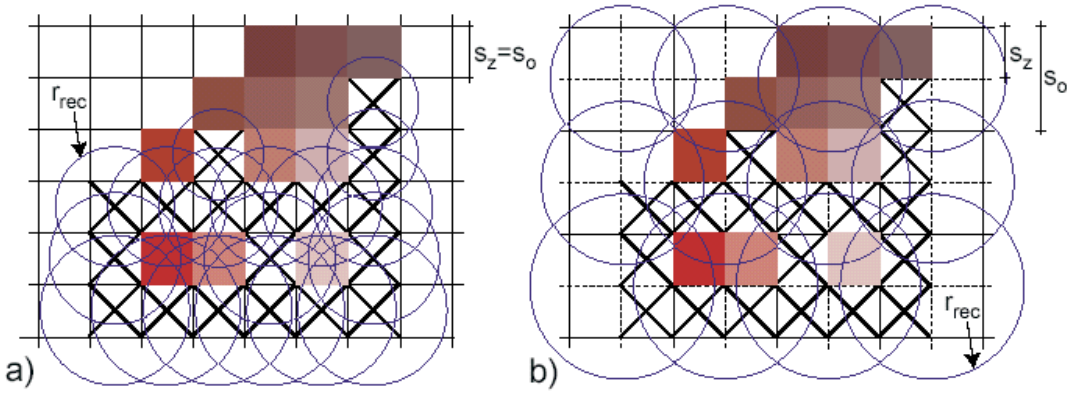


Figure 88 : Choix de surfels. [Pfister00]

4.2 Projection de base

Il est à noter que les points stockés dans un bloc sont obtenus à partir d'une projection orthographique. On commence alors par le calcul de la transformation de *l'espace objet* (LDC-tree) à *l'espace caméra*, pour chaque projection.

On appelle *espace caméra* le système de coordonnée par lequel la caméra est à l'origine, l'écran est le carré de côté 2 centré et orthogonal sur l'axe *z*, en $z = 1$ [Grossman98a] (Figure 89).

Soit *o* l'origine du Bloc courant dans le repère de LDC-tree, *r* la taille d'un pixel et $\delta_{i,j,k}$ la profondeur du $k^{ième}$ point en *(i,j)*. Soit $p_{i,j,k}$ le $k^{ième}$ point stocké en *(i, j)*, dans le repère de LDC-tree (cf. chapitre 3, Figure 67et Figure 68).

$$p_{i,j,k} = o + r \times \begin{bmatrix} i \\ j \\ \delta_{i,j,k} \end{bmatrix} \qquad \textbf{(4.7)}$$

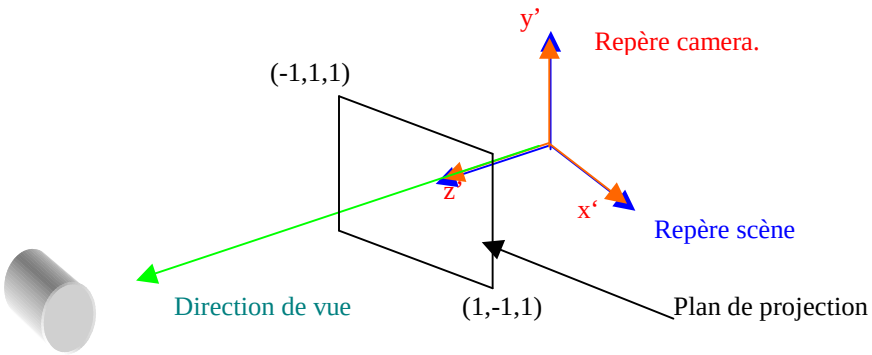


Figure 89 : Définition de plan de projection.

Il est donc intéressant d'insérer la translation par *o* et la mise à l'échelle par *r* dans la matrice de modélisation[16] active ; soit *M* la matrice résultante. Cette matrice 4x4 peut se décomposer en une matrice 3x3 orthogonale A (représentant une rotation et mise à l'échelle) et un vecteur T représentant une translation. Cette transformation se traduit de la manière suivante:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{A}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + T = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \qquad \textbf{(4.8)}$$

Où (*x, y,* z) sont les coordonnées d'un point dans l'espace objet, et (*x', y', z'*) ses coordonnées dans l'espace caméra.

De même, on calcule les coordonnées entières écran (*u,* v) résultant de la projection perspective de ce point sur le plan image (écran) [Grossman98a]. Le résultat est comme suit:

$$u = \left\lfloor \frac{N}{2}\left(\frac{x'}{z'}+1\right)\right\rfloor \qquad v = \left\lfloor \frac{N}{2}\left(\frac{y'}{z'}+1\right)\right\rfloor \qquad \textbf{(4.9)}$$

Où $\lfloor \cdot \rfloor$ dénote la partie entière et N est la largeur de l'image. La transformation complète de (*x, y,* z) à (*u,* v) est ainsi :

$$\begin{aligned} u &= \left\lfloor \frac{N}{2}\left(\frac{A_{xx}x + A_{xy}y + A_{xz}z + t_x}{A_{zx}x + A_{zy}y + A_{zz}z + t_z}+1\right)\right\rfloor \\ v &= \left\lfloor \frac{N}{2}\left(\frac{A_{yx}x + A_{yy}y + A_{yz}z + t_y}{A_{zx}x + A_{zy}y + A_{zz}z + t_z}+1\right)\right\rfloor \end{aligned} \qquad \textbf{(4.10)}$$

Si nous utilisons l'approche directe de calcul (la transformation complète pour chaque point), nous avons besoin de 9 additions et 9 multiplications afin de calculer x', y', z', et une inversion pour calculer 1/z' et 2 additions supplémentaires et 4 multiplications supplémentaires afin de calculer u, v ; pour un total de 11 additions, 13 multiplications et une inversion pour chaque point. On peut réduire ceci à 11, 11 et 1 en absorbant le facteur de N/2 dans les coefficients pré calculés de la matrice [Grossman98a].

L'image finale est stockée en z-buffer. Suite à la projection de tous les blocs, l'éclairage est établi dans l'espace écran, ainsi les points doivent être copiés comme ils sont dans le z-buffer (à chaque pixel d'image nous stockons la couleur diffuse, la couleur spéculaire, la normale et le brillant d'un point), ceci sera étudié au paragraphe 6.

4.2.1 Calcul incrémental

En employant le calcul incrémental de la projection, nous avons la possibilité de réduire le nombre d'opérations indispensables tout en faisant un bénéfice ; étant donnée que les points sont échantillonnés selon une grille régulière [Grossman98a]. Soit ($x_{i,j,k}$, $y_{i,j,k}$, $z_{i,j,k}$) les coordonnées du $k^{ième}$ surfel en *(i,j)* d'un bloc. Afin de simplifier l'écriture des équations qui vont suivre, on considère un surfel par couple *(i,j)* et on se place dans une ligne j donnée, ses coordonnées s'écrivent simplement : (x_i, y_i, z_i) les coordonnées de l'espace lié à l'objet ; soit (x_i', y_i', z_i') les coordonnées de l'espace caméra et (u_i, v_i) ses coordonnées entières projetées à l'espace image. Si $(x_{i-1}, y_{i-1}, z_{i-1})$ et (x_i, y_i, z_i) sont les points adjacents sur la même ligne d'un bloc, donc on a :

$$x_i = x_{i-1} + dx \qquad et \qquad y_i = y_{i-1} \qquad \textbf{(4.11)}$$

Où *dx* est l'espacement entre deux points adjacents ; c'est à dire l'inverse de la résolution[17] *r* (Figure 90). Il faut donc prendre *dx =1*. Soit px_i, py_i, pz_i tel que :

$$\begin{aligned} px_i &= A_{xx}x_i + A_{xy}y_i + t_x \\ py_i &= A_{yx}x_i + A_{yy}y_i + t_y \\ pz_i &= A_{zx}x_i + A_{zy}y_i + t_z \end{aligned} \qquad \textbf{(4.12)}$$

[16] la matrice de modélisation contient deux types de transformation : la transformation de visualisation (permet de fixer la position et l'orientation de la caméra de visualisation) et la transformation de modélisation (permet de créer la scène à afficher par création, placement et orientation des objets qui la composent).

[17] r est déjà absorbé par les coefficients matriciels

Alors on peut calculer (u_i, v_i) de la manière suivante:

$$\begin{gathered} px_i = px_{i-1} + A_{xx}dx \qquad py_i = py_{i-1} + A_{yx}dx \qquad pz_i = pz_{i-1} + A_{zx}dx \\ x_i' = px_i + A_{xz}z_i \qquad y_i' = py_i + A_{yz}z_i \qquad z_i' = pz_i + A_{zz}z_i \\ u_i = \left\lfloor \frac{N}{2}\left(\frac{x_i'}{z_i'}+1\right)\right\rfloor \qquad v_i = \left\lfloor \frac{N}{2}\left(\frac{y_i'}{z_i'}+1\right)\right\rfloor \end{gathered} \tag{4.13}$$

Ceci exige maintenant 8 additions, 5 multiplications et 1 inversion pour chaque surfel en (i, j) du LDI, on a une amélioration significative. On remarque que $A_{xx}dx$, $A_{yx}dx$, et $A_{zx}dx$ ne sont pas pris en compte comme des multiplications car ces valeurs peuvent être précalculées.

Lors d'un incrément selon j (saut de ligne) on a :

$$\begin{aligned} px_{0,j} &= px_{0,j-1} + A_{xy}d_y \\ py_{0,j} &= py_{0,j-1} + A_{yy}d_y \\ pz_{0,j} &= pz_{0,j-1} + A_{zy}d_y \end{aligned} \tag{4.14}$$

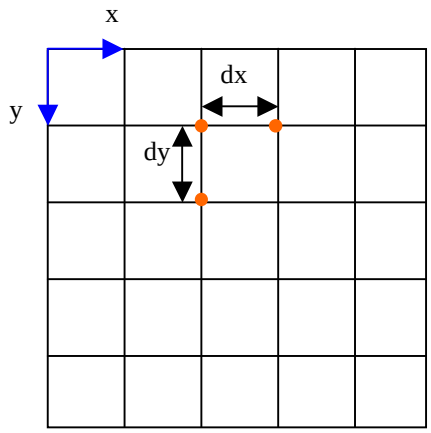


Figure 90 : Calcul incrémental sur la grille régulière du LDI d'un bloc donné.

4.2.2 Optimisation de calcul incrémental

En effet, on peut calculer (u_i, v_i) en utilisant moins d'opération. A cet effet, nous réécrivons l'équation (4.10) en utilisant les définitions (4.12), puis nous ajoutons et nous retirons une constante à chaque fraction :

$$\begin{aligned} u_i &= \left\lfloor \frac{N}{2}\left(\frac{px_i + A_{xz}z_i}{pz_i + A_{zz}z_i}+1\right)\right\rfloor = \left\lfloor \frac{N}{2}\left(\frac{px_i - \frac{A_{xz}}{A_{zz}}pz_i}{pz_i + A_{zz}z_i} + \frac{A_{xz}}{A_{zz}}+1\right)\right\rfloor \\ v_i &= \left\lfloor \frac{N}{2}\left(\frac{py_i + A_{yz}z_i}{pz_i + A_{zz}z_i}+1\right)\right\rfloor = \left\lfloor \frac{N}{2}\left(\frac{py_i - \frac{A_{yz}}{A_{zz}}pz_i}{pz_i + A_{zz}z_i} + \frac{A_{yz}}{A_{zz}}+1\right)\right\rfloor \end{aligned} \tag{4.15}$$

Ainsi, si nous définissons :

$$\begin{aligned} px_i' &= px_i - \frac{A_{xz}}{A_{zz}}pz_i \\ py_i' &= py_i - \frac{A_{yz}}{A_{zz}}pz_i \\ pz_i' &= pz_i \end{aligned} \tag{4.16}$$

Alors nous pouvons réécrire (4.13) comme suit:

$$px_i' = px_{i-1}' + \left(A_{xx} - \frac{A_{xz}A_{zx}}{A_{zz}} \right) dx \qquad py_i' = py_{i-1}' + \left(A_{yx} - \frac{A_{yz}A_{zx}}{A_{zz}} \right) \quad pz_i' = pz_{i-1}' + A_{zx}dx$$

$$z_i' = pz_i' + A_{zz}z_i \qquad \textbf{(4.17)}$$

$$u_i = \left\lfloor \frac{N}{2} \left(\frac{px_i'}{z_i'} + \frac{A_{xz}}{A_{zz}} + 1 \right) \right\rfloor \qquad v_i = \left\lfloor \frac{N}{2} \left(\frac{py_i'}{z_i'} + \frac{A_{yz}}{A_{zz}} + 1 \right) \right\rfloor$$

Ceci exige 6 additions, 3 multiplications et 1 inversion pour chaque surfel en (i, j) du LDI. Supposant une fois encore que les facteurs N/2 sont absorbés par les constantes précalculées. Le passage d'une ligne à une autre se fait de la même manière que précédemment.

4.3 Projection du texel

Nous allons étudier le rendu d'un bloc LDC tree lorsque celui ci représente un motif. Au début on remarque que la *forme* d'un bloc au sein du motif déformé est similaire à celui du motif représenté en entier Figure 86. Le processus de projection du texel est défini de la manière suivante :

a) Calcul des paramètres de forme $b'_{i,j}$ et $h'_{i,j}$ (bloc déformé): La position des quatre sommets $b'_{i,j}$ est facilement obtenue grâce aux équations (4.6) puisqu'on connaît la position de l'origine du bloc dans le motif ainsi que sa largeur, sa hauteur et sa résolution. Les vecteurs $h'_{i,j}$ sont obtenus en multipliant les vecteurs $h_{x,y}$ calculés précédemment par la profondeur multipliée par l'inverse de la résolution du bloc. En résumer on a :

$$\begin{aligned}
b'_{00} &= b_{ox,oy} + o_z h_{ox,oy} \\
b'_{10} &= b_{ox+r.l,oy} + o_z h_{ox+r.l,oy} \\
b'_{01} &= b_{ox,oy+r.h} + o_z h_{ox,oy+r.h} \\
b'_{11} &= b_{ox+r.l,oy+r.h} + o_z h_{ox+r.l,oy+r.h} \\
h'_{00} &= r.p.h_{ox,oy} \\
h'_{10} &= r.p.h_{ox+r.l,oy} \\
h'_{01} &= r.p.h_{ox,oy+r.h} \\
h'_{11} &= r.p.h_{ox+r.l,oyr.h}
\end{aligned} \qquad \textbf{(4.18)}$$

Où, (o_x, o_y, o_z) sont les coordonnées d'origine du bloc par rapport au LDC tree, l sa largeur, h sa hauteur, p sa profondeur et r la taille d'un pixel (inverse de la résolution).

b) Projection des surfels d'un bloc : Soit (x, y, z) les coordonnées du k^{ième} surfel $S_{i,j,k}$ stocké en (i, j) dans LDI. A partir de l'équation (4.6), on obtient (x', y', z'), coordonnées de $S_{i,j,k}$ dans le repère objet et après déformation. Il ne nous reste plus qu'à appliquer les équations (4.10) (transformation de modélisation et projection en perspective conique), pour obtenir les coordonnées (u, v) de $S_{i,j,k}$ dans l'espace image. Malheureusement, l'implémentation directe de cette méthode n'est pas vraiment envisageable, du fait quelle demande un nombre trop élevé d'opérations par surfel à projeter.

Aux sections précédentes, nous avons démontré qu'il était possible de tirer bénéfice de la cohérence spatiale entre les surfels due à leur stockage dans une grille régulière. Nous allons donc tenter la même approche, mais en tenant compte de la déformation du bloc. En plus, nous allons considérer uniquement le LDI des blocs (LDC) dont la référence correspond au carreau $(b'_{00}, b'_{10}, b'_{01}, b'_{11})$. Pour les autres, il suffit de faire subir une rotation à la Figure 86, le carreau $(b_{00}, b_{10}, b_{01}, b_{11})$ ne se retrouve plus appliqué à la surface de l'objet.

Afin de simplifier l'écriture des formules, nous allons nous placer dans une ligne i et considérer un seul surfel par «pixel» (i, j). Nous considérons donc le surfel S_i de coordonnées (x_i, y_i, z_i) dans le bloc. On pose :

$$b_i = b_{xi,yi,zi}$$

$$h_i = h_{xi,yi,zi}$$

D'après l'équation (4.10) et (4.6), on a :

$$u_i = \left\lfloor \frac{N}{2}\left(1+\frac{A_{xx}((b_i)_x + z_i(h_i)_x)+A_{xy}((b_i)_y + z_i(h_i)_y)+A_{xz}((b_i)_z + z_i(h_i)_z)+t_x}{A_{zx}((b_i)_x + z_i(h_i)_x)+A_{zy}((b_i)_y + z_i(h_i)_y)+A_{zz}((b_i)_z + z_i(h_i)_z)+t_z}\right)\right\rfloor$$
$$v_i = \left\lfloor \frac{N}{2}\left(1+\frac{A_{yx}((b_i)_x + z_i(h_i)_x)+A_{yy}((b_i)_y + z_i(h_i)_y)+A_{yz}((b_i)_z + z_i(h_i)_z)+t_y}{A_{zx}((b_i)_x + z_i(h_i)_x)+A_{zy}((b_i)_y + z_i(h_i)_y)+A_{zz}((b_i)_z + z_i(h_i)_z)+t_z}\right)\right\rfloor \quad \textbf{(4.19)}$$

On pose :

$$\begin{aligned}
(p_i)_x &= A_{xx}(b_i)_x + A_{xy}(b_i)_y + A_{xz}(b_i)_z + t_x = A_x \times b_i + t_x.\\
(p_i)_y &= A_{yx}(b_i)_x + A_{yy}(b_i)_y + A_{yz}(b_i)_z + t_y = A_y \times b_i + t_y.\\
(p_i)_z &= A_{zx}(b_i)_x + A_{zy}(b_i)_y + A_{zz}(b_i)_z + t_z = A_z \times b_i + t_z.\\
(q_i)_x &= A_{xx}(h_i)_x + A_{xy}(h_i)_y + A_{xz}(h_i)_z = A_x \times h_i.\\
(q_i)_y &= A_{yx}(h_i)_x + A_{yy}(h_i)_y + A_{yz}(h_i)_z = A_y \times h_i.\\
(q_i)_z &= A_{zx}(h_i)_x + A_{zy}(h_i)_y + A_{zz}(h_i)_z = A_z \times h_i.
\end{aligned} \quad \textbf{(4.20)}$$

Où A_x est une matrice 1x3 correspondant à la première ligne de la matrice A. simplement, on peut écrire :

$$\begin{aligned} p_i &= A \times b_i + T\\ q_i &= A \times h_i \end{aligned} \quad \textbf{(4.21)}$$

Les équations (4.19) deviennent alors :

$$u_i = \left\lfloor \frac{N}{2}\left(1+\frac{px_i + z_i qx_i}{pz_i + z_i qz_i}\right)\right\rfloor$$
$$v_i = \left\lfloor \frac{N}{2}\left(1+\frac{py_i + z_i qy_i}{pz_i + z_i qz_i}\right)\right\rfloor \quad \textbf{(4.22)}$$

De ces équations, on remarque que chaque point dans le bloc déformé est défini par q_i et p_i, donc pour passer d'un surfel au suivant sur la grille régulière, on doit trouver leurs incréments et les relations de passage :

- Lors du parcours d'une ligne j, on a :

$$\begin{aligned} p_i &= A \times b_i + T = A \times (b_{i-1} + db_j) + T = A \times b_{i-1} + T + A \times db_j = p_{i-1} + dp_j.\\ q_i &= A \times h_i = A \times (h_{i-1} + dh_j) + T = A \times h_{i-1} + T + A \times dh_j = q_{i-1} + dq_j. \end{aligned} \quad \textbf{(4.23)}$$

  Donc on doit incrémenter p_i et q_i par dp_j et dq_j :

$$\begin{aligned} p_i &= p_{i-1} + dp_j.\\ q_i &= q_{i-1} + dq_j. \end{aligned} \quad \textbf{(4.24)}$$

  Où ces incréments sont égaux à :

$$\begin{aligned} dp_j &= A \times db_j.\\ dq_j &= A \times dh_j. \end{aligned} \quad \textbf{(4.25)}$$

  Noter que les incréments db_j et dh_j permettent de passer de b_{i-1} à b_i et de h_{i-1} à h_i. De plus, Ils dépendent de la ligne j parcourue :

  - db_j constituent une suite arithmétique de raison ddbx et de premier terme db_{j0} (i. e. j=0) :

    Le terme général est : $db_j = db_{j0} + j \cdot ddbx$ **(4.26)**

    Où $db_{j0} = \frac{b_{10} - b_{00}}{l}$ et $ddbx = \frac{\frac{b_{11} - b_{01}}{l} - \frac{b_{10} - b_{00}}{l}}{h} = \frac{b_{11} - b_{01} - b_{10} + b_{00}}{l * h}$

  - dh_j constituent une suite arithmétique de raison ddhx et de premier terme dh_{j0} (i. e. j=0) :

    Le terme général est : $dh_j = dh_{j0} + j \cdot ddhx$ **(4.27)**

    Où $dh_{j0} = \frac{h_{10} - h_{00}}{l}$ et $ddhx = \frac{\frac{h_{11} - h_{01}}{l} - \frac{h_{10} - h_{00}}{l}}{h} = \frac{h_{11} - h_{01} - h_{10} + h_{00}}{l * h}$

- De même, pour le parcours de la colonne i, on a :

$$p_j = A\times b_j+T = A\times(b_{j-1}+db_i)+T = A\times b_{j-1}+T+A\times db_i = p_{j-1}+dp_i.$$
$$q_j = A\times h_j = A\times(h_{j-1}+dh_i)+T = A\times h_{j-1}+T+A\times dh_i = q_{j-1}+dq_i. \quad \textbf{(4.28)}$$

donc on doit incrémenter p_j et q_j par dp_i et dq_i :

$$p_j = p_{j-1}+dp_i.$$
$$q_j = q_{j-1}+dq_i. \quad \textbf{(4.29)}$$

Où ces incréments sont égaux à :

$$dp_i = A\times db_i.$$
$$dq_i = A\times dh_i. \quad \textbf{(4.30)}$$

Noter que les incréments db_i et dh_i permettent de passer de b_{j-1} à b_j et de h_{j-1} à h_j. De plus, Ils dépendent de la colonne i parcourue :

- db_i constituent une suite arithmétique de raison ddby et de premier terme db_{i0} (i. e. i=0) :

  Le terme général est : $db_i = db_{i0} + i\cdot ddby$ **(4.31)**

  Où : $db_{i0} = \frac{b_{01}-b_{00}}{h}$ et $ddby = \frac{\frac{b_{11}-b_{10}}{h} - \frac{b_{01}-b_{00}}{h}}{l} = \frac{b_{11}-b_{10}-b_{01}+b_{00}}{h*l}$

- dh_i constituent une suite arithmétique de raison ddby et de premier terme db_{i0}(i. e. i=0) :

  Le terme général est : $dh_i = dh_{i0} + i\cdot ddhy$ **(4.32)**

  Où : $dh_{i0} = \frac{h_{01}-h_{00}}{h}$ et $ddhy = \frac{\frac{h_{11}-h_{10}}{h} - \frac{h_{01}-h_{00}}{h}}{l} = \frac{h_{11}-h_{10}-h_{01}+h_{00}}{h*l}$

- Lors du saut de ligne : comme d'habitude, la grille est parcourue ligne par ligne, ceci nous conduit à trouver la manière de passer d'une ligne à l'autre.

  - D'abord, on doit calculer le premier point dans la nouvelle ligne en utilisant l'équation (4.28), Afin de réaliser ceci, on calcule les incréments dp_i et dq_i de saut de ligne, comme suit :

    on substitue L'équation (4.31) et (4.32) dans l'équation (4.30), on trouve :

    $$dp_i = A\times db_i = A\times (db_{i0} + i*ddby) = dp_{i-1} + A\times ddby.$$
    $$dq_i = A\times dh_i = A\times (dh_{i0} + i*ddhy) = dq_{i-1} + A\times ddhy. \quad \textbf{(4.33)}$$

    Où ddby et ddhy représentent les incréments de l'incrément db_i et dh_i.

  - Maintenant, il ne reste plus qu'incrémenter les incréments dp_j et dq_j, afin de trouver le reste des points sur cette nouvelle ligne. On substitue l'équation (4.26) et (4.27) dans l'équation (4.25), on trouve :

    $$dp_j = dp_{j-1} + A\times ddbx.$$
    $$dq_j = dq_{j-1} + A\times ddhx. \quad \textbf{(4.34)}$$

    Où ddbx et ddhx représentent les incréments de l'incrément db_j et dh_j.

Pour illustrer tous ces incréments on peut se référer à la Figure 91.

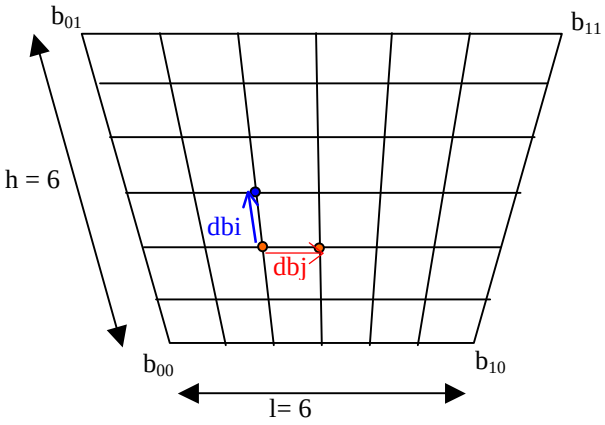


Figure 91 : Illustration des différents incréments et rappel des définitions de la largeur (l), hauteur (h). Ce schéma s'applique aussi bien au vecteur b, qu'aux vecteurs h, q et q.

En résumer et après initialisation des différentes constantes et variables, le rendu d'un bloc s'effectue simplement et à peu de frais. Un incrément selon i est réalisé par les équations (4.24) qui nécessitent 6 additions (2 additions vectorielles). La projection d'un surfel (équations (4.22)) nécessite 5 multiplications (2 multiplications vectorielles), 1 inversion et 5 additions (2 additions vectorielles). Le passage d'une ligne à la suivante est réalisé par les équations (4.29) et (4.33) pour un total de 12 additions (4 additions vectorielles). Bien sur, la prise en compte de la déformation ajoute un coût de calcul supplémentaire mais, en final, reste tout à fait raisonnable.

5 Reconstruction d'image

Autrefois, durant le rendu lorsqu'un pixel est atteint par des points de surfaces multiples, un z-buffer classique est employé pour résoudre la visibilité. Mais la difficulté consiste en général à ce que, tous les pixels appartenant à une surface seront atteints par quelque point de cette surface ; exemple, la Figure 92 montre des trous dans une surface sombre car certains des pixels ont *'manqué'*. Plus loin une surface claire est visible par ces trous.

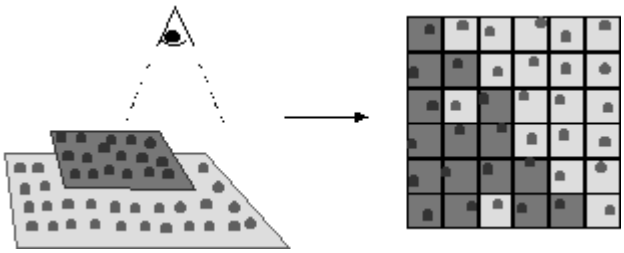


Figure 92 : La surface claire apparaît à travers les trous de la surface sombre. [Grossman98a]

D'où, d'une façon ou d'une autre, il est nécessaire de reconstruire ces surfaces afin qu'ils apparaissent continus lors du rendu. Plusieurs solutions ont été discutées dans la littérature ; parmi les plus utilisées on trouve :

- *Mark et al.* présentent une approche au problème de reconstruction de surface. L'idée est de traiter l'échantillon de points comme des sommets d'une maille triangulaire qui peut être scan-conversion[18] [Mark97]. Cette approche a les inconvénients suivants :
  - Nécessite un grand nombre d'opérations par échantillon de points. Elle est lente.
  - Difficile de produire correctement la maille sans certaines connaissances de la topologie superficielle de l'objet; il n'existe aucune façon exacte pour déterminer quel point doit être connecté aux triangles.
  - Il est extrêmement difficile de vérifier si vraiment la surface entière de l'objet est couverte par l'union de tous les triangles, de toutes les vues.
  - Quand une surface concave est échantillonnée par des vues multiples, les triangles formés par des points dans une vue peuvent obscurcir des échantillons de points d'autres vues (Figure 93a). Cela dégrade la qualité de l'image finale, car les pixels seront rempli par une couleur moins précise, obtenue de l'interpolation de triangle plutôt que celle d'échantillon de point.
  - De considérables artefacts, résultent quand on essaye de combiner les triangles de vues multiples dans une seule image. Les triangles de vues différentes apparaissent incohérents, comme indiqué dans la Figure 93b.

[18] Scan-conversion consiste à balayer la scène à la manière d'un scan-line.

(a) **(b)** **(c)**

Figure 93 : (a) Une surface courbée est échantillonnée par deux vues (montré dans une dimension). Quand les vues sont combinées, les triangles d'une vue (connectant les points sombres) obscures les échantillons de point de l'autre (les points blancs). (b) Partie d'un objet consiste en surface blanche intersectant une surface sombre à 90 ° (montré dans les nuances de gris). Ce coin est échantillonné par deux vues différentes. Quand les triangles de ces deux vues sont combinés par scan-conversion eux dans une image de z-buffer simple (montré dans les nuances de gris plus sombres), les triangles apparaissent incohérents en raison des taux différents auxquels ils interpolent du blanc au sombre. (c) 'Dépassement' dans splatting. [Grossman98a]

- Technique de splatting : Un seul point est plaqué aux multiples pixels sur l'écran, la couleur d'un pixel est la moyenne pondérée des couleurs des points contribuants. Cette méthode a été employée par *Levoy et Whitted* [Levoy85]; elle a été aussi employée dans le contexte de rendu volumique et dans le surfels de *Pfister* [Pfister00]. Cependant, cette méthode fait ressortir les inconvénients suivants :
  - Nécessite un grand nombre d'opérations par point lors de la projection. Elle est lente.
  - Afin d'assurer qu'il n'existe aucun trous dans n'importe quelle surface ; le choix de la taille et de la forme du splat est un problème extrêmement difficile. Une idée intéressante est l'emploie des normales d'échantillon de points pour dessiner de petits cercles ou des quadrilatères orientés, mais ceci a l'inconvénient de faire apparaître des « dépassements » près des coins comme indiqué à la Figure 93c.
- *Grossman* présente une hiérarchie de z-buffer : Lors du rendu, chaque partie de l'objet est projeté dans un tampon de profondeur à une résolution assez basse tel que les points ne peuvent pas se propager et laisser des trous. Quand toutes les parties de l'objet ont été rendu, nous aurons une image qui renfermera des trous. Cependant, il est possible actuellement de détecter et éliminer ces trous en comparant les profondeurs de pixels dans l'image par rapport aux profondeurs de pixels dans la hiérarchie de tampon de profondeur (Figure 94). Cette méthode présente les avantages suivants :
  - Rapide puisqu'elle exige un petit nombre d'opération par point lors de rendu.
  - Simple astuce pour détecter les trous.
  - Eclairage déféré.
  - Simple idée de traitement d'artefacts « bloc », d'où une haute qualité d'image.

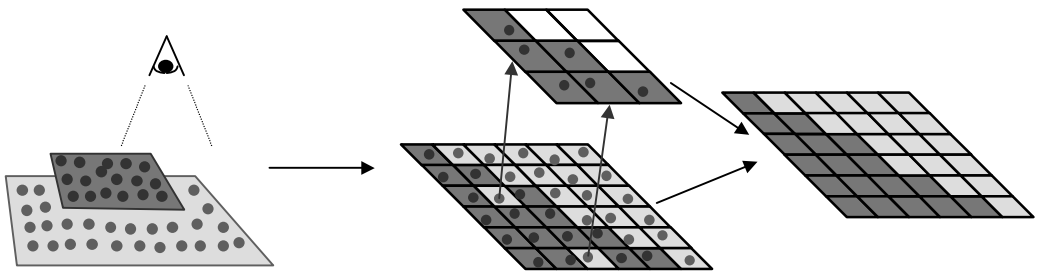


Figure 94 : La surface sombre est rendue en employant un tampon de profondeur de niveau plus haut dans lequel il n'y a aucun trou. Ce tampon est employé pour détecter et éliminer les trous dans l'image. [Grossman98a]

On s'est orienté à cette dernière méthode étant donnée quelle offre des avantages non négligeables. Une étude détaillée du processus est prise en compte aux sections qui suivent.

5.1 Z-Buffer hiérarchique

C'est une approche de visibilité à l'espace image. Elle permet ainsi de conclure rapidement si un visage de cube ou une primitive est cachée, en rendant le parcours de l'image pixel par pixel inutile [Greene93]. La construction d'un z-pyramide se traduit par :

- La base de la pyramide est un z-buffer ordinaire et a une même résolution que l'image cible.
- Chaque tampon dans la hiérarchie a la moitié de la résolution du tampon ci-dessous.

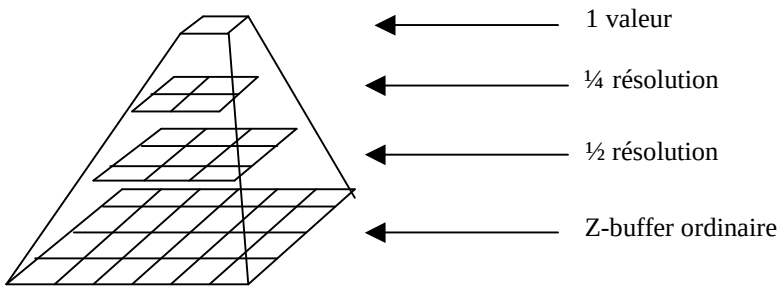


Figure 95 : La hiérarchie de z-buffer (z-pyramide). [Greene93]

Afin de mettre en œuvre cet algorithme de reconstruction de surface, chaque bloc doit être projeté dans la hiérarchie de z-buffer à un tampon de résolution assez basse tel que ses points ne peuvent pas se propager et laisser des trous. Si la résolution d'image est $N \times N$, alors le k^{ième} tampon de profondeur a une résolution $\left\lceil \frac{N}{2^k} \right\rceil \times \left\lceil \frac{N}{2^k} \right\rceil$ où $\lceil \cdot \rceil$ représente la partie entière. Pour choisir un tampon de profondeur approprié au bloc déformé, nous devons évaluer sa taille en espace image. Pour cela, on utilise la sphère englobant ce bloc.

Soit θ l'angle d'ouverture de la camera, d la distance entre la caméra et la sphère englobant le bloc et R le rayon de la sphère englobante ; la taille du bloc projeter est :

$$k = \frac{N \times R}{2 \times d \times \tan\frac{\theta}{2}} \qquad \textbf{(4.35)}$$

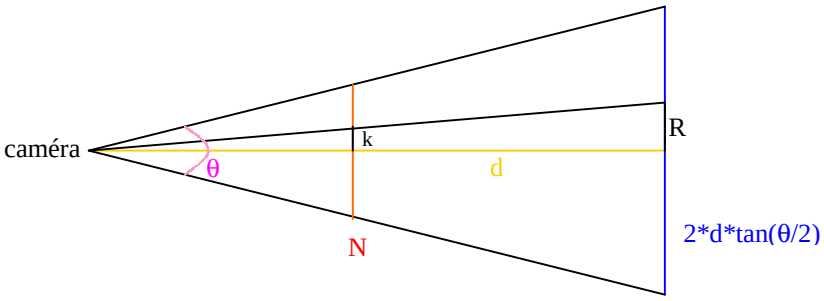


Figure 96 : Calcul de k

Si $k = 0$, dans ce cas le surfel est projeté seulement dans le z-buffer ordinaire où les coordonnées (u, v) ont été étudiées précédemment; si $k > 0$, alors on doit projeter le surfel au k^{ième} tampon de profondeur où les coordonnées (u' = u/2^k, v' = v/2^k) et au z-buffer ordinaire.

Grossman a proposé l'ajout d'un petit seuil à la valeur z du point puis le comparer avec la valeur dans le k^{ième} tampon de profondeur. La raison de ceci est que le tampon de profondeur sera employé pour filtrer les points qui se trouvent derrière la surface de premier plan; l'utilisation d'un seuil évite que les points qui se trouvent sur la surface de premier plan d'être accidentellement rejeter.

Comme exemple, on suppose, que trois points appartiennent à la surface de premier plan et un point situé en arrière plan, sont projetés sur quatre pixels différents dans l'image mais, sur un seul dans le tampon de profondeur de faible résolution. Si on n'utilise pas le seuil, seul le point le plus en avant serait conservé et les trois autres seront mis au rébus (Figure 97a). L'utilisation d'un seuil empêche ce cas de se produire (Figure 97b). La difficulté repose

essentiellement sur le choix du seuil. Un seuil plus petit produit des images de mauvaise qualité (élimine des points qui sont bien au premier plan). Un seuil plus grand génère une difficulté à la distinction entre les surfaces de premier plan et les surfaces d'arrière plan. *Grossman* suggère qu'un bon seuil doit être égal à trois fois la largeur de pixel d'image.

Figure 97 : Quatre points sont projetés sur quatre pixels différents dans l'image mais sur un seul dans le tampon de profondeur de faible résolution. (a) Aucun seuil de profondeur – seul le point le plus proche est conservé. (b) l'utilisation d'un petit seuil protège les points appartenant à la même surface que le point le plus proche d'être rejeter. [Grossman98a]

5.2 Recherche de trous

L'idée fondamentale de détection de trous est de comparer la profondeur de chacun des pixels de l'image avec la profondeur stockée dans les z-buffers de faible résolution. Pour cela on traite un pixel dans $k^{ième}$ tampon de profondeur comme un carré opaque qui couvre exactement 4^k pixel de l'image. En ce moment, on peut prendre une décision binaire pour chaque pixel en déterminant s'il se trouve derrière un certain pixel de tampon de profondeur. En fin de compte deux situations apparaissent :

- Si un pixel est couvert par un pixel du tampon de profondeur de faible résolution (c'est à dire ayant une moindre profondeur). On suppose qu'il n'est pas sur la surface de premier plan, donc c'est un trou. Dans l'image finale ce pixel doit être recoloré en interpolant ses voisins les plus proches.
- Si un pixel n'est couvert par aucun pixels du tampon de profondeur. On suppose qu'il se trouve sur la surface de premier plan, sa couleur est inchangée dans l'image finale.

Cette méthode est rapide, facile à mettre en œuvre ; simplement elle produit les artefacts d'aliassage appelés « effets de bloc », particulièrement aux bords, représenté à la Figure 98.

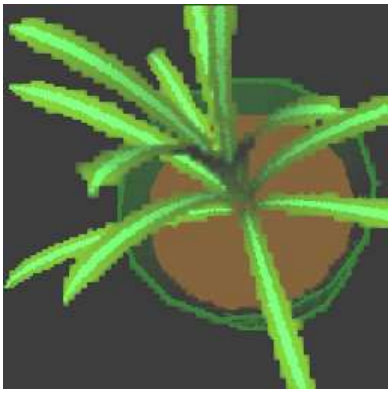
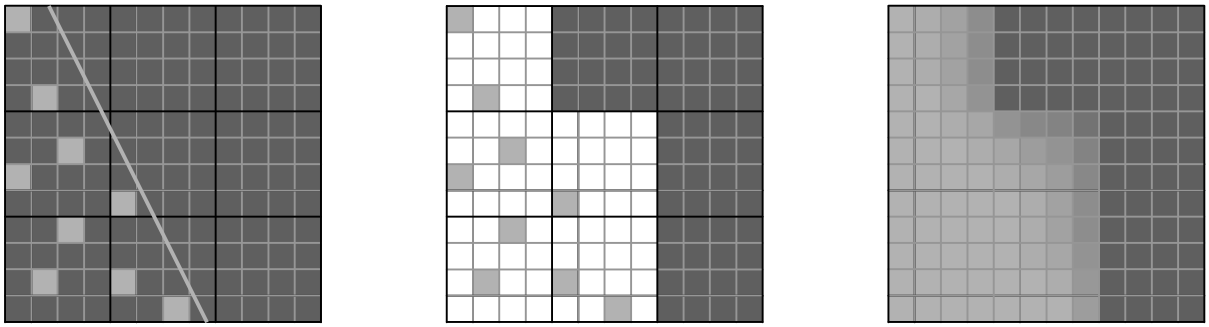


Figure 98 : Traitement des pixels du tampon de profondeur comme carrés opaques ; produisant les artefacts de bloc. [Grossman98a]

Un exemple typique (Figure 99) expose la manière dont le problème est posé. Le bord d'une surface de premier plan blanc est rendu au-dessus d'un fond sombre dans le tampon de profondeur k=2 (Figure 99a). Chaque pixel du tampon de profondeur qui est atteint par un certain échantillon de points devient un carré opaque de 4x4, ainsi tous les pixels de fond qui se trouvent derrière ces carrés sont traités comme des trous et reconstruit (Figure 99b). Cela crée de grands trous dans l'image; on suppose que plusieurs pixels sont des trous, bien qu'en réalité ils soient loin du bord de la surface. Quand ces trous sont remplis en utilisant des couleurs interpolées, ceci génère d'énormes artefacts.



(a) (b) (c)

Figure 99 : (a) Une surface blanche de premier plan claire est rendue dans le tampon de profondeur k = 2. Les pixels d'image et les pixels du tampon de profondeur sont représentés respectivement par des droites claires et noires. La droite claire diagonale représente le bord réel de la surface de premier plan et les carrés clairs indiquent les pixels qui sont atteints par les échantillons de points. (b) les pixels d'image qui se trouvent derrière les pixels du tampon de profondeur sont reconstruits. (c) image Finale : après remplissage de trous. [Grossman98a]

5.3 Antialiassage de bord

Pour éviter le problème en question on substitue la décision binaire par une décision floue et continu. On affecte à chaque pixel un indicateur de confiance appelé *poids*, qui appartient à l'intervalle [0, 1] et qui ont les significations suivantes :

- Un poids égale 1 : implique que le pixel est situé certainement dans le premier plan ; sa couleur reste inchangée.
- Un poids égale 0 : implique que le pixel représente certainement un trou ; sa couleur doit être ignorée.
- Un poids intermédiaire : implique que le pixel n'est pas entièrement recouvert par un pixel du tampon de profondeur, notamment aux frontières ; sa couleur va être mélangée avec la couleur interpolée.

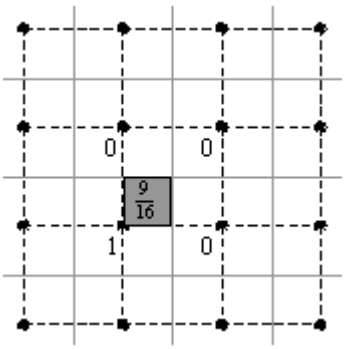
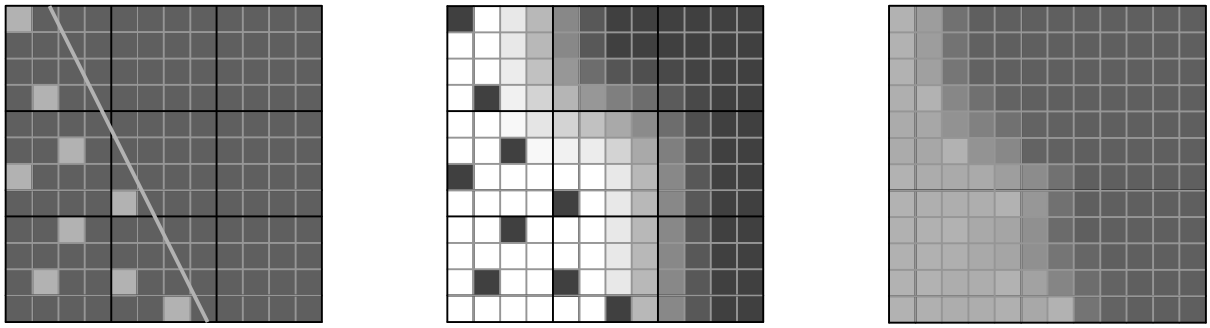


Figure 100 : Calcul du taux de recouvrement d'un pixel d'image. Les pixels du tampon de profondeur sont matérialisés par des lignes grises et le maillage résultant par des lignes noires. Le pixel image grisé est recouvert par un seul des pixels voisins du tampon de profondeur. Par interpolation bilinéaire, le taux de recouvrement est de 9/16. [Grossman98a]

Il est donc primordial de considérer plusieurs pixel du tampon de profondeur afin de calculer le poids d'un seul pixel d'image. Afin de calculer ces poids, on traite le $k^{ième}$ tampon de profondeur comme un maillage carré où les sommets sont centrés par rapport aux pixels. On calcule alors la quantité par laquelle le $k^{ième}$ tampon de profondeur couvre un pixel d'image. Il se traduit par :

- Le centre du pixel d'image se trouve à l'intérieur d'un des carrés du maillage.
- La profondeur à chaque coin du carré est comparée à la profondeur du pixel d'image; un coin est assigné à un poids égale à 1 s'il se trouve devant le pixel et 0 autrement.

- Finalement, le taux de recouvrement du pixel est obtenu par l'interpolation bilinéaire de ces quatre couvertures de coins. Le taux de recouvrement total d'un pixel est alors la somme des taux de recouvrement de chacun des tampons de profondeur (inférieur à 1) et le poids égale à 1 moins le taux de recouvrement (Figure 100).



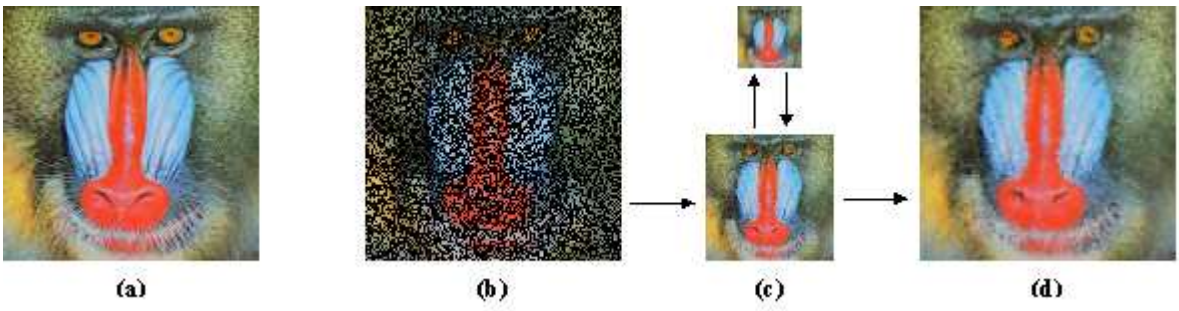
(a) **(b)** **(c)**

Figure 101 : (a) Rendu de la même surface qu'à la Figure 99a. (b) Poids assignés aux pixels d'image (ceux du grands poids sont plus sombres). (c) image Finale : les artefacts sont moins considérables. [Grossman98a]

La Figure 101 est un exemple qui décrit le même traitement que celui de la Figure 100, on utilise le calcul des poids vu précédemment. L'emploi de ces poids au remplissage des trous est traité à la prochaine section.

5.4 Remplissage de trous

Durant cette phase, on est en mesure de remplir les trous dans l'image finale. On utilise le calcul d'une couleur pour tous les pixels ayant un poids inférieur à 1. Le remplissage de trous n'est pas un problème spécifique pour le rendu à base de points; toutefois c'est un problème général de reconstruction d'image à partir d'information incomplète. La solution retenue est une variation de l'algorithme « Pull-Push » décrit par Gortler [Gortler96]. Celui ci s'accomplit en deux phases. Durant la première « Pull » on calcule une série d'image approximée de résolution décroissante (chacune renferme la moitié de la résolution du précédent), ces derniers vont servir dans la seconde phase « Push » au calcul d'une seconde série d'image de résolution croissante en remplissant au fur et à mesure les trous (Figure 102). L'algorithme de « Pull-Push » est défini comme suit :



(a) **(b)** **(c)** **(d)**

Figure 102 : L'algorithme de « Pull-Push ». (a) Image initiale. (b) Image inachevée; 50% des pixels ont été ignorés. (c) les approximations de résolution inférieure. (d) Reconstruction. [Grossman98a]

- **Première phase « Pull »** : En partant de l'image à reconstruire, Les images approximées de résolution décroissante sont calculées en réalisant successivement une moyenne pondérée des blocs 2x2 pixels. Les poids utilisés pour cette moyenne sont ceux calculés dans la section précédente. Les poids sont également fusionnés et stockés dans la nouvelle image. Spécifiquement, si $c^k_{x,y}$ et $w^k_{x,y}$ sont la couleur et le poids du pixel (x, y) de la k^{ième} image, on a alors :

$$c^{k+1}_{x,y} = \frac{w^k_{2x,2y}c^k_{2x,2y} + w^k_{2x+1,2y}c^k_{2x+1,2y} + w^k_{2x,2y+1}c^k_{2x,2y+1} + w^k_{2x+1,2y+1}c^k_{2x+1,2y+1}}{w^k_{2x,2y} + w^k_{2x+1,2y} + w^k_{2x,2y+1} + w^k_{2x+1,2y+1}}$$

$$w_{x,y}^{k+1} = MIN(1,\ \ w_{2x,2y}^{k} + w_{2x+1,2y}^{k} + w_{2x,2y+1}^{k} + w_{2x+1,2y+1}^{k})$$

- **La deuxième phase «Push» :** Durant cette phase, on calcule la couleur finale $c'^{k}_{x,y}$ du pixel (x, y) de la k[ième] image, on examine son poids $w^{k}_{x,y}$. Si son poids est $w^{k}_{x,y} = 1$, on conserve sa couleur $c'^{k}_{x,y} = c^{k}_{x,y}$. Autrement, On calcule une couleur interpolée $\tilde{c}^{k}_{x,y}$ en utilisant comme interpolants les pixels de la (k+1)[ième] image. On a alors :

$$c'^{k}_{x,y} = w^{k}_{x,y} c^{k}_{x,y} + (1 - w^{k}_{x,y})\tilde{c}^{k}_{x,y}$$

Si on doit employer l'interpolation bilinéaire pour calculer $\tilde{c}^{k}_{x,y}$ alors nous interpolerions les quatre pixels les plus étroits de la (k +1) [ième] image avec les poids 9/16, 3/16, 3/16 et 1/16 (puisque l'interpolation est toujours d'un niveau bas vers le haut dans la hiérarchie, donc les poids ne changent jamais). Cependant, Le pixel possédant le poids 1/16 contribue faiblement dans la couleur finale ; donc nous avons la possibilité d'ignorer ce pixel afin de réduire le coût de calcul sans aucune dégradation apparente de la qualité d'image. Par conséquent, en interpolant les trois pixels les plus étroits de la (k +1) [ième] image par les poids ½, ¼ et ¼ (Figure 103). On prend comme exemple :

$$\tilde{c}^{k}_{2x+1,2y+1} = \frac{1}{2} c'^{k+1}_{x,y} + \frac{1}{4} c'^{k+1}_{x+1,y} + \frac{1}{4} c'^{k+1}_{x,y+1}$$

Avec des expressions similaires pour : $\tilde{c}^{k}_{2x,2y}$, $\tilde{c}^{k}_{2x+1,2y}$, et $\tilde{c}^{k}_{2x,2y+1}$.

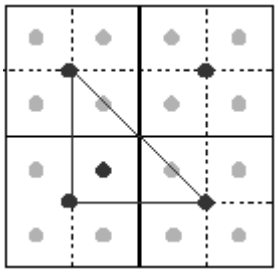


Figure 103 : Une couleur interpolée est calculée à partir des trois pixels de l'image de plus faible résolution entourant le pixel courant.

6 Eclairement différé

Afin d'afficher l'image finale, on doit lui attribuer un certain aspect de réalisme en simulant les effets de lumière dans la scène. On constate que le calcul de l'éclairage se réalise en trois moments différents :

- Lors de la projection des surfels : A chaque surfel projeté, on calcule son illumination, et on stocke uniquement la couleur résultante dans le tampon d'image. Cette idée implique beaucoup de calculs inutiles, puisque les surfels d'arrière plan seront supprimés durant l'étape de la hiérarchie du z-buffer.
- Eclairement différé : Cette opération consiste à stocker en chaque pixel de l'image toutes les informations nécessaires au calcul d'éclairage (couleur, normal, et index de matériaux). Elle s'accomplit en deux temps:
  - Entre le calcul des poids et le remplissage des trous : En ce moment, on calcule l'éclairage uniquement sur les pixels contribuant à la reconstruction de l'image, c'est à dire les points ayant le poids > 0 et n'appartient pas aux surfaces cachées. Cette solution s'avère plus rapide que la précédente.
  - Après la phase de reconstruction d'image : Pour réaliser l'éclairage en ce moment, lors du remplissage des trous on prépare les informations nécessaires à ce calcul, c'est à dire il faut non seulement interpoler les couleurs mais aussi les normales. Pour l'indice des matériaux, l'indice du plus proche voisin est utilisé. Cette solution calcule l'éclairage pour un nombre de points plus grands que précédemment et la phase de remplissage de trous et plus coûteuse.

On a choisi d'évaluer le calcul d'illumination dans l'espace écran (entre le calcul des poids et le remplissage de trous) en utilisant le modèle d'éclairage de *Phong*, puisqu'il apparaît le plus rapide d'entre eux.

Puisque les normales d'échantillon de point sont enregistrées dans l'espace objet. On exécute le calcul d'illumination dans l'espace objet pour éviter de transformer les normales à un autre système de coordonnée. A cet effet on doit récupérer les coordonnées de l'espace objet des points contenus dans les pixel d'image. Etant donnée qu'on n'enregistre pas les coordonnées exactes de x' et y' (de l'espace caméra) des points quand ils sont copiés aux pixel dans le tampon image ; on suppose qu'ils sont situés au centre de pixel. En utilisant les coordonnées z' stockées dans le z-buffer d'image, nous pouvons alors transformer les points de l'espace caméra à l'espace objet, ainsi cette transformation peut être exécutée rapidement en utilisant le calcul incrémental.

6.1 Elimination des points d'arrière plan

Etant donnée que les normales des points ne sont pas examinées lors du copiage au tampon image. En ce moment il est entièrement possible que certains d'entre eux se dirigent loin de la caméra. Pour les objets solides, ces points se trouvent nécessairement derrière une surface de premier plan et seront ainsi filtrés (Figure 104a). Néanmoins, les points qui se trouvent près d'un coin peuvent être omis par le processus de filtrage à cause du petit seuil employé pour éviter de mettre au rébus les points sur la surface de premier plan (Figure 104b). Cela peut générer une inexactitude de pixels colorés apparaissant dans l'image finale (Figure 105a). Pour résoudre ce problème, on doit éliminer les points des faces arrières en leur assignant un poids égale zéro (Figure 105b). Ceci est exécuté durant l'éclairage, lorsque les normales des points sont extraites.

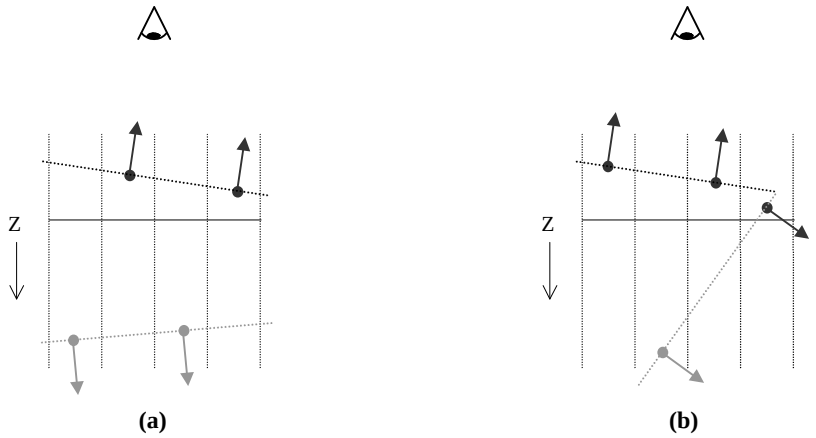


Figure 104 : (a) La plupart des points d'arrière plan sont filtrés puisqu'ils se trouvent derrière une surface de premier plan. (b) Les points d'arrière plan près d'un coin peuvent apparaître. [Grossman98a]

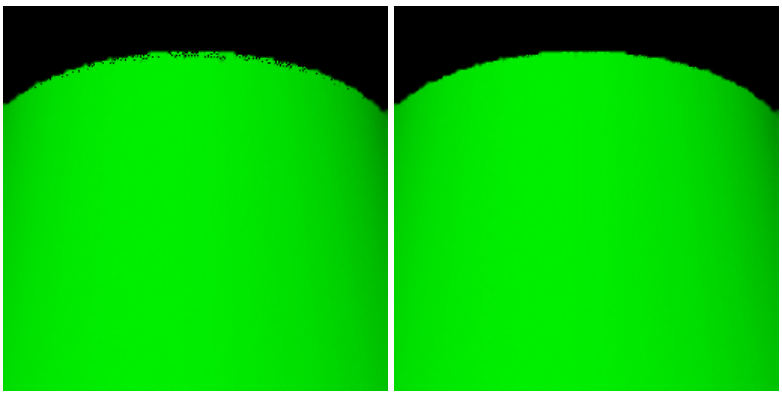


Figure 105 : De près du bord d'un cylindre. (a) On n'élimine pas les points d'arrière plan; quelques pixels sont inexactement colorés. (b) On élimine les points d'arrière plan.

7 Rendu de surface sous-jacente et ombre porté

La surface sous jacente est modélisée par un maillage polygonal, où leur visualisation sera réalisée par un rendu classique de l'API graphique OpenGL.

Afin de réaliser ceci, on doit initialiser les différentes variables d'environnement tel que la position de l'observateur, de la source de lumière. On mélange le z-buffer (utilisé dans le rendu de nuage de points) avec le depth-buffer d'OpenGL, tout en éliminant les surfaces ou points cachés.

Finalement, avant d'afficher l'image on utilise le tampon stencil d'OpenGL pour afficher les ombres portés sur la surface sous jacente.

8 Résultats et discussion

Les résultats suivants sont obtenus en utilisant le langage C++ et la bibliothèque graphique OpenGL. On emploie une résolution de prélèvement d'échantillons de 128^3 pour le LDC. On note que les résultats suivants sont générés à partir de LDC tree du chapitre précédent.

8.1 Effet du calcul incrémental

Les figures 106, 107 et 108 montrent l'habillage d'une peau volumique constituée respectivement d'une boite, de 4 boites et de 16 boites. Ces dernières définissent la forme et la position du motif d'habillage.

Les images de gauche des figures 106, 107 et 108 sont générées par une combinaison de calcul incrémental et déformation lors d'étape de projection. Par contre, les figures de droite sont générées par une séparation entre l'étape de déformation et la projection ordinaire matrice-vecteur.

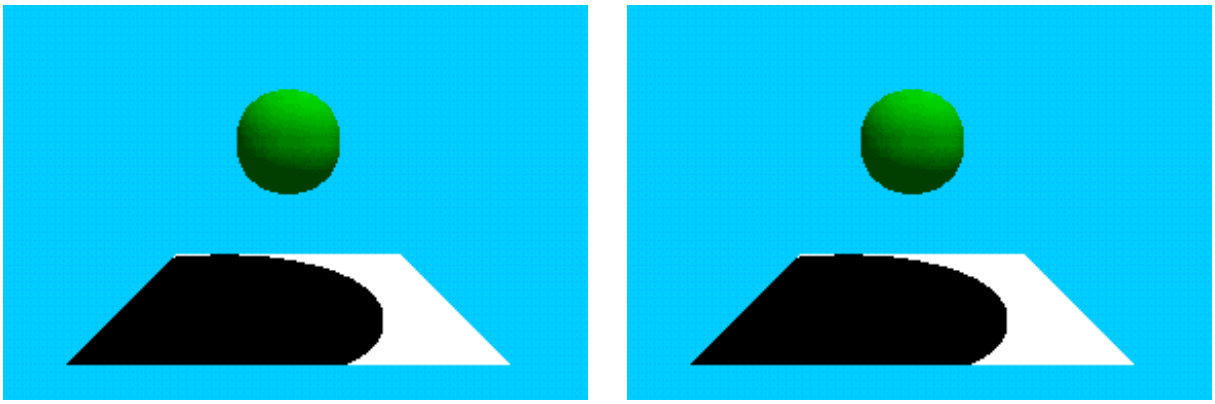


Figure 106 : Rendu d'une seule boite.

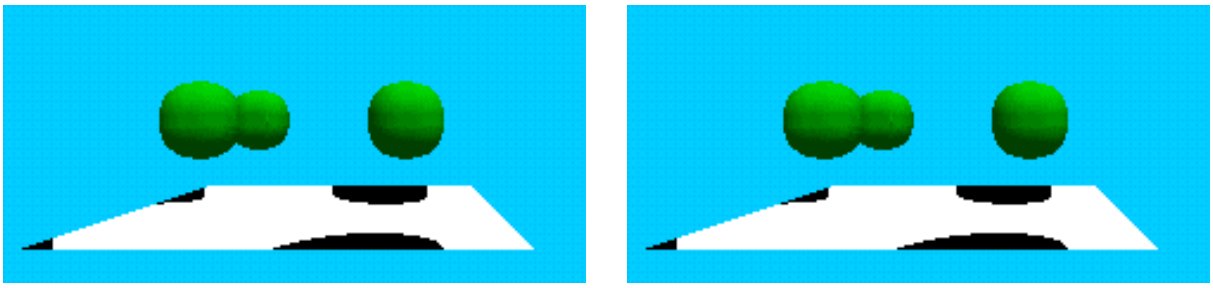


Figure 107 : Rendu de 4 boites.

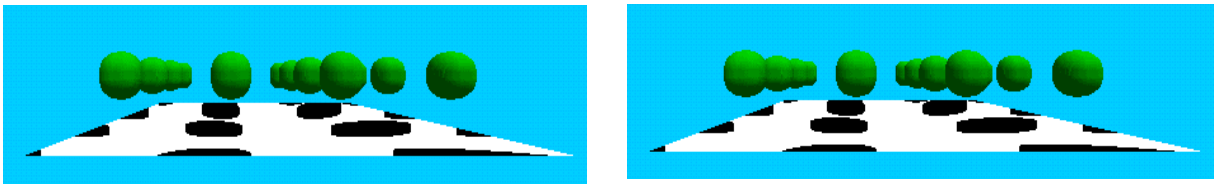


Figure 108 : Rendu de 16 boites.

	Gauche : Calcul incrémental			**Droite : Séparation entre la projection et la déformation**		
	Figure 106	**Figure 107**	**Figure 108**	**Figure 106**	**Figure 107**	**Figure 108**
Temps de frustum	0sc 0ms	0sc 0ms	0sc 0ms	0sc 0ms	0sc 0ms	0sc 0ms
Temps de déformation	0sc 20ms	0sc 120ms	1sc 640ms	3sc 530ms	9sc 830sc	56sc 670ms
Temps de projection	0sc 80ms	0sc 230ms	2sc 640ms	7sc 90ms	12 540sc	58sc 780ms
Temps de reconstruction d'image	0sc 21ms	0sc 57ms	0sc 660ms	0sc 21ms	0sc 57ms	0sc 660ms
Temps d'illumination	3sc 200ms	14sc 430ms	50sc 325ms	3sc 200ms	14sc 430ms	50sc 325ms
Temps total	4sc 450ms	15sc 600ms	1mn 04sc 750ms	13sc 810ms	42sc 690ms	3mn 20sc 470ms

Tableau 7: Statistique de rendu des images précédentes. A gauche rendu par optimisation, à droite rendu par séparation entre déformation et projection.

Le tableau 7 montre le temps consommé par les déférentes étapes de rendu. A cet instant, on remarque que :

- A cause des méthodes d'optimisation de la sélection de vue frustum, le temps consommé par cette dernière est toujours inférieur à 1ms (i.e. d'ordre nano- seconde).
- La combinaison de l'étape de déformation avec l'étape projection (calcul incrémental) en une seule étape *« Projection de texel »*, nous permet de réduire le temps total de rendu (cf. tableau 7 et figure 109).
- Le temps de reconstruction d'image est acceptable, toutefois le temps d'illumination et très grand par rapport au temps nécessaire aux autres étapes. Il occupe plus de 90% du temps global de rendu dans notre méthode. Ceci est justifié par le calcul empirique, de l'intensité spéculaire et diffuse nécessaire lors du calcul d'illumination pour chaque surfels (figure 110). D'autre part, il occupe moins de 30% du temps nécessaire dans le cas ou en sépare l'étape de déformation de celle de projection (figure 111).

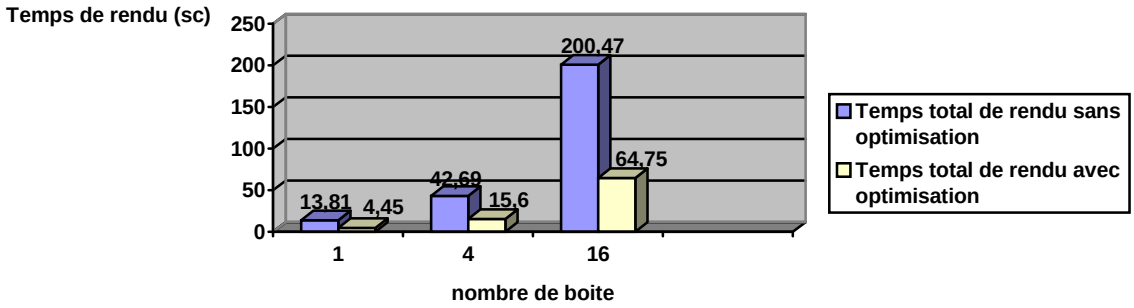


Figure 109 : Effet de d'optimisation de la projection de texel sur le temps total de rendu.

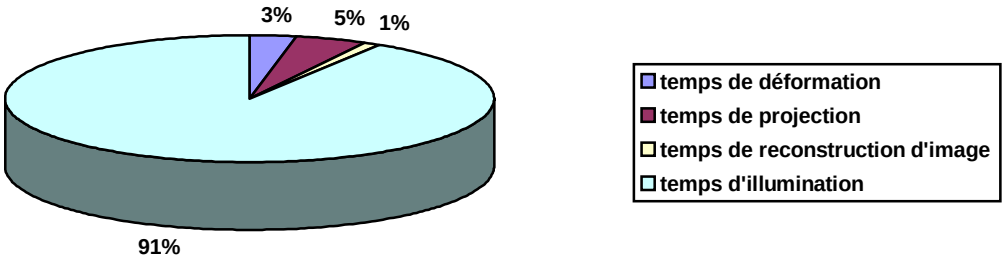


Figure 110 : Temps d'illumination nécessaire pour un rendu avec optimisation est plus de 95% du temps global.

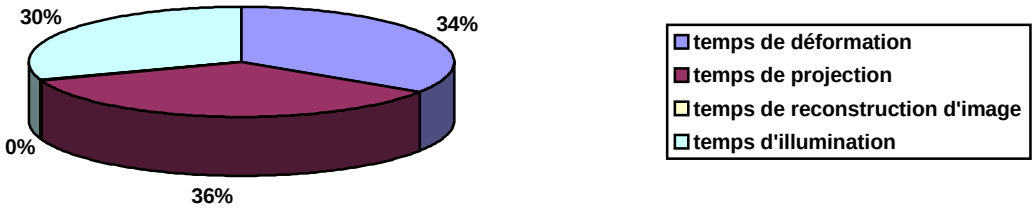


Figure 111 : Temps d'illumination nécessaire pour un rendu sans optimisation et moins de 30% du temps global.

8.2 Comparaison entre la texture volumique et notre méthode

8.2.1 Qualité d'image

Visuellement, les figures 112 et 113 font ressortir que la qualité d'images générée par notre méthode est meilleur que celle de générée par la texture volumique, de même que pour la déformation du motif.

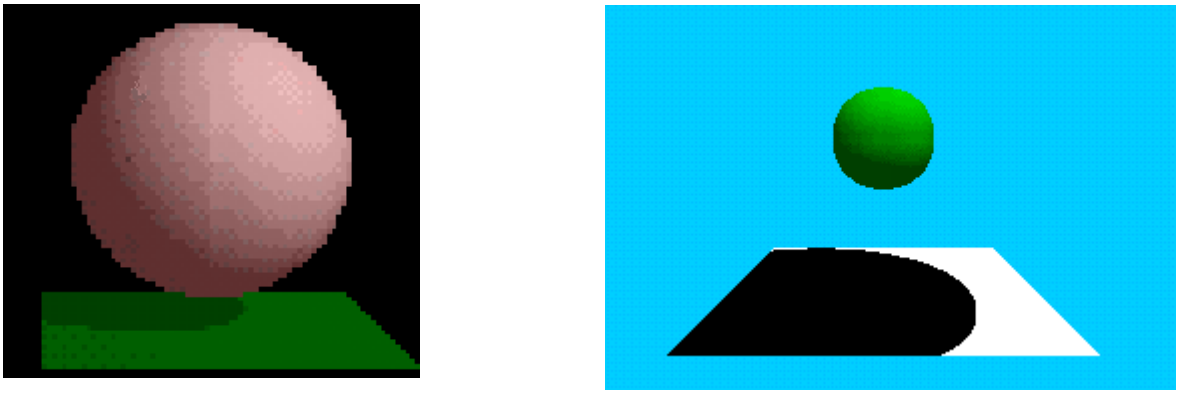


Figure 112 : Habillage d'une boite ; à gauche par la texture volumique ; à droite par notre méthode.

Enfin, La texture volumique offre un rendu multi-échelle selon la distance de la boite à l'observateur; il en est de même pour notre méthode préconisée.

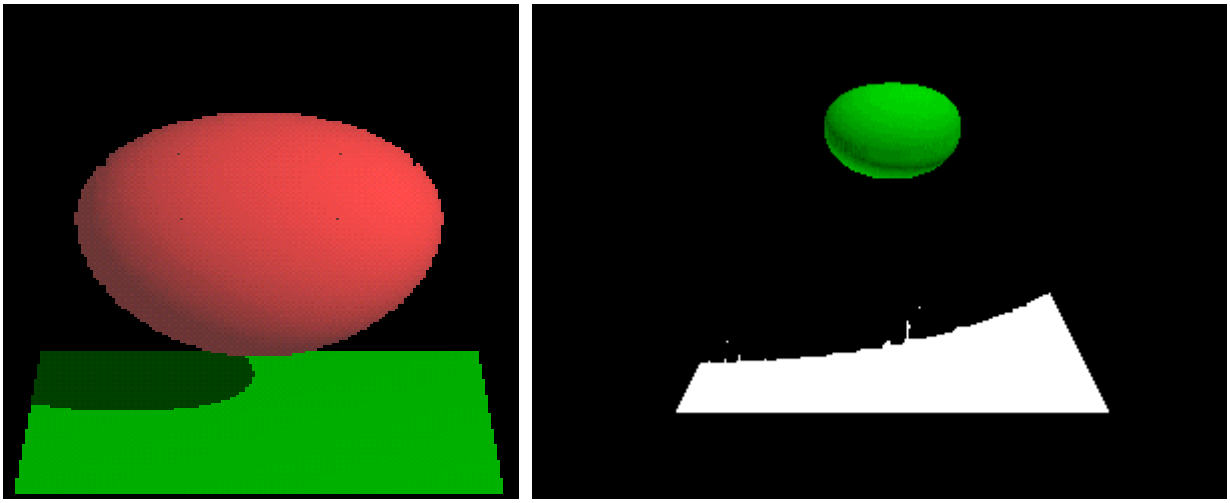


Figure 113 : Qualité de déformation du motif. A gauche selon la texture volumique, à droite selon notre méthode.

8.2.2 Temps de rendu

Le temps de rendu de texture volumique est beaucoup plus grand que le temps de rendu de notre méthode. Ceci pour les raisons suivantes :

- Le rendu est réalisé par un lancer de rayon nécessitant un temps prohibitif (passage d'un espace à l'autre, temps d'intersection, …). Entre autre, le rendu de notre méthode se réalise par une optimisation de calcul incrémental et une déformation, plus une reconstruction d'image.
- Le passage de l'espace texel à l'espace boite s'effectue par une interpolation trilinéaire dans la texture volumique ; et par une interpolation bilinéaire pour notre cas.
- La déformation est réalisée par un lancer de rayon courburé. Pour notre méthode, la définition d'un nouveau rayon de disque tangent est suffisante.
- L'éclairage se réalise par le lancer de rayon donc un modèle d'illumination globale. Par contre, nous utilisons un modèle local (celui de *Phong*).
- L'ombre porté est réaliser par le lancer de rayon. Pour notre cas, on utilise le tampon stencil d'OpenGL plus rapide.
- Le parcours du volume de référence : notre méthode utilise un texel de 3 niveaux, au lieu de 7 niveaux pour le texel de la texture volumique (voir chapitre 3, figure 76) ; ceci diminue le temps de rendu global par diminution du temps de parcours.

9 Conclusion

Ce chapitre avait pour but de rendre le processus de visualisation de texture volumique plus souple et plus interactif, tout en conservant la notion de déformation et de plaquage « mapping ». Afin de réaliser ceci, on greffe la texture volumique sur une plate-forme de rendu à base de points, ce qui nous permet d'avoirs :

- Plaquer le texel dans la peau volumique.
- Déformer le motif selon la forme des boites constituant cette peau.
- Minimiser l'aliassage de bord et la distorsion du motif.
- Combinaison simple et facile avec le rendu classique (OpenGL) : Afin d'afficher l'image finale avec la surface sous-jacente et l'ombre porté.

Notre méthode offre une bonne qualité d'image est un temps de rendu favorable par rapport à la texture volumique. Cette méthode présente des insuffisances se traduisant en deux inconvénients majeurs vis à vis de la texture volumique à savoir:

- Un modèle d'illumination locale (celui de *Phong*) faible par rapport au lancer de rayon qui est un modèle global.
- Le rendu des objets transparents et semi-transparents : est supporté par la texture volumique, rejeté par la notre du fait qu'on utilise un z-buffer.

Chapitre 5 : Résultats

1 Introduction

Dans cette partie nous exposons un ensemble de résultats qui permettent de valider le modèle de fourrure utilisé au placage d'une texture volumique sur une surface tridimensionnelle. Les résultats sont obtenus suite à l'utilisation d'un Pentium II 300 MHz équipé d'une mémoire 32 Mo SDRAM.

2 Volume de référence

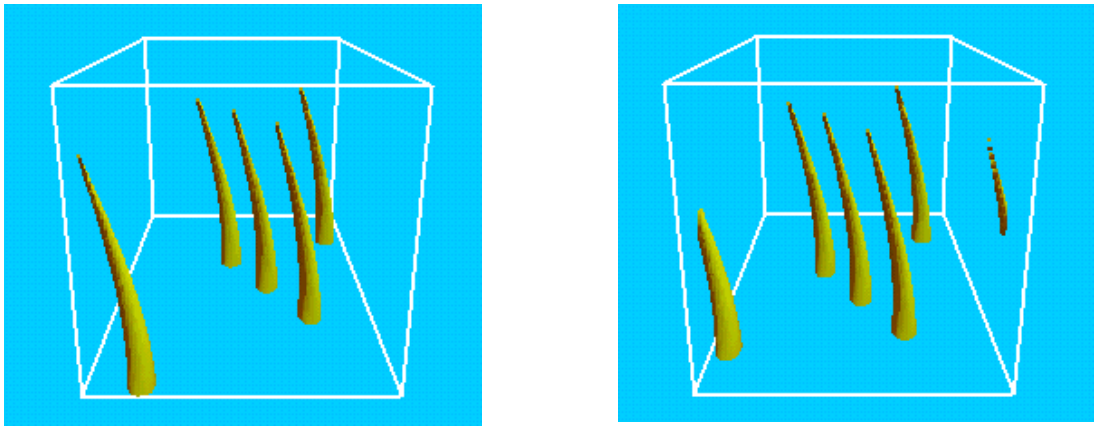


Figure 114 : Volume de référence : à gauche volume de référence normal, à droite un volume de référence qui assure la continuité des fourrures entre les boites.

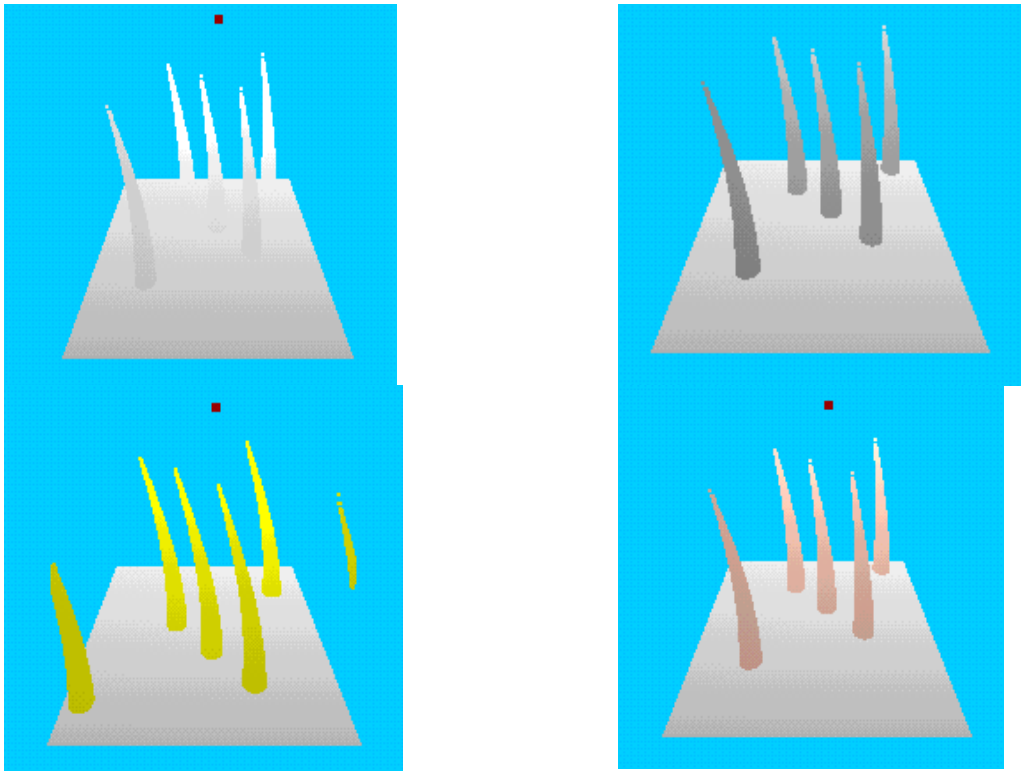


Figure 115 : Stockage de différents volumes de références, à cause de leurs différence en couleur. Le petit carré réprésente la source de la lumière.

On constate aux Figure 114 et 115,une modification de position d'un cheveu et/ou leur couleur nous oblige à rééchantillonner le modèle , le sauvegarder comme un nouveau modèle différent des autres.

3 Habillage d'une surface plate

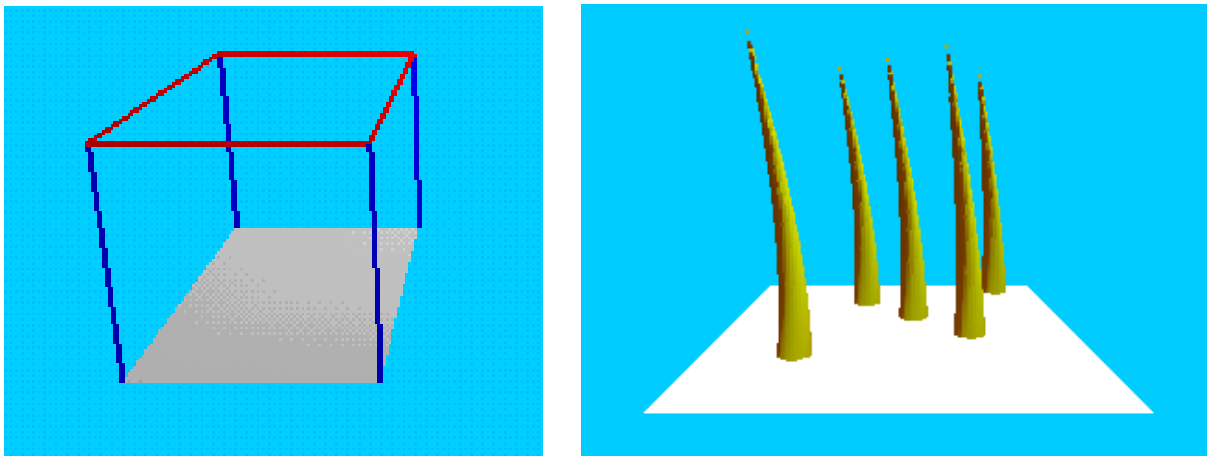


Figure 116 : A gauche une boite vide, à droite leur habillage.

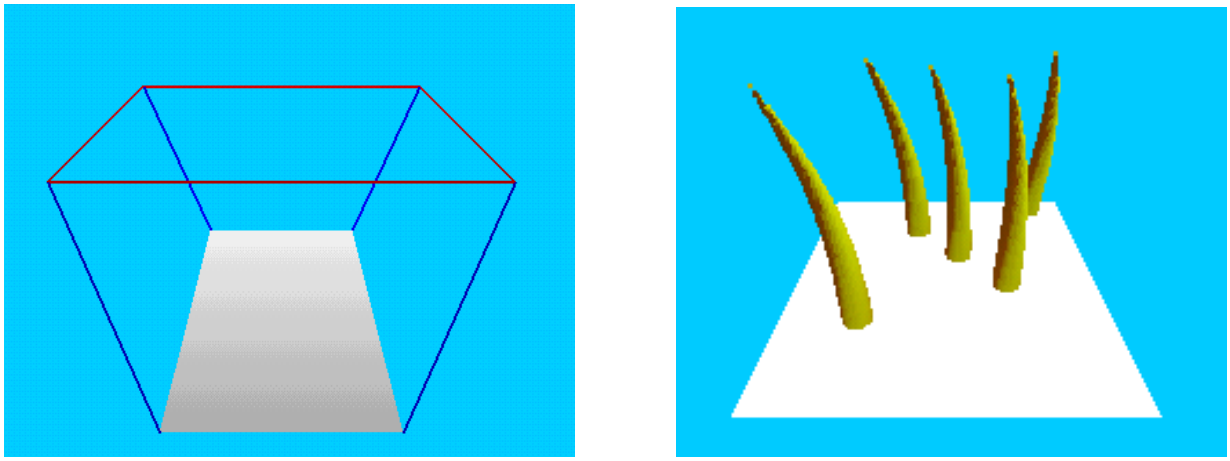


Figure 117 : A gauche une boite déformée, à droite leur habillage.

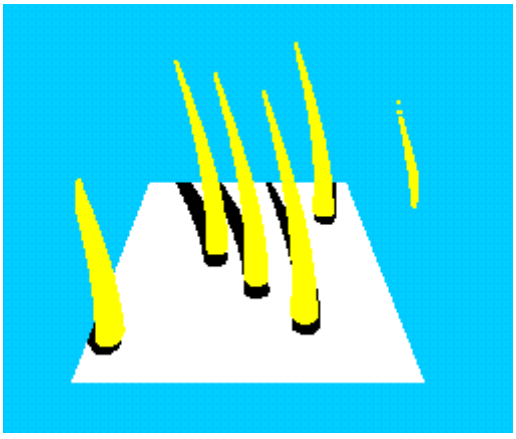


Figure 118 : Habillage avec ombre portée.

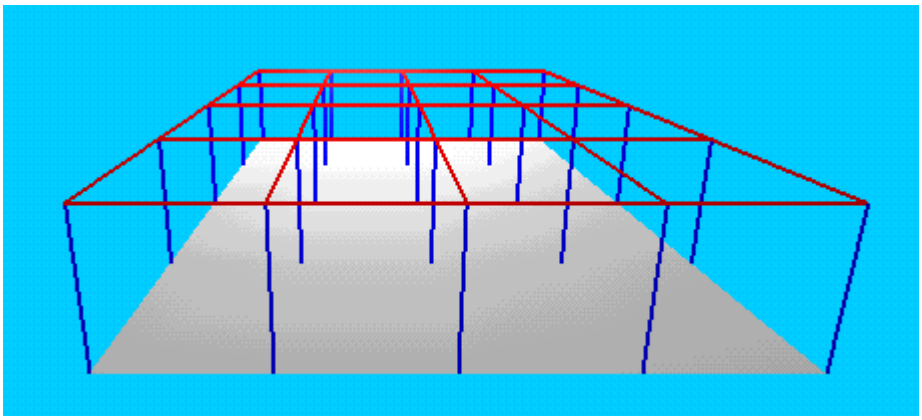


Figure 119: Peau volumique de 16 boites.

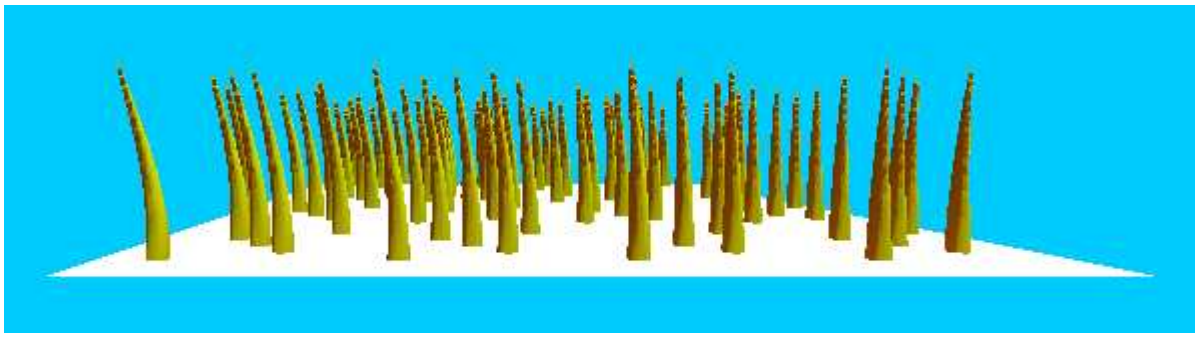


Figure 120 : Habillage de peau de la figure 119 par le texel normal. Il y a une discontinuité entre les boites.

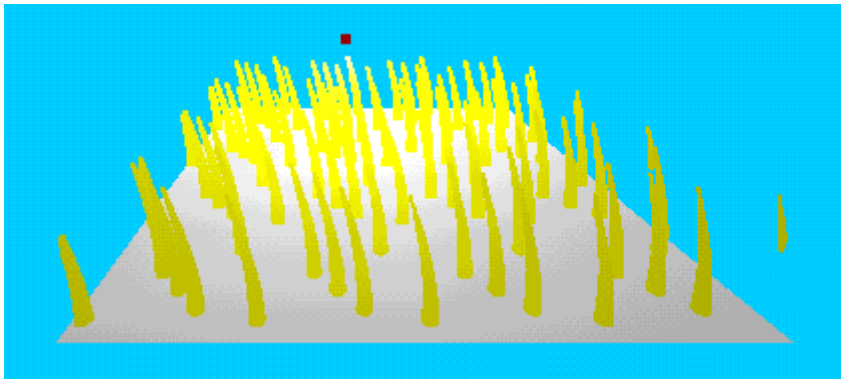


Figure 121 : Habillage de peau de la figure 119 par le texel qui assure la continuité.

4 Habillage d'une surface déformée

4.1 Perturbation de la surface plate selon l'axe y par une fonction sinusoïdale

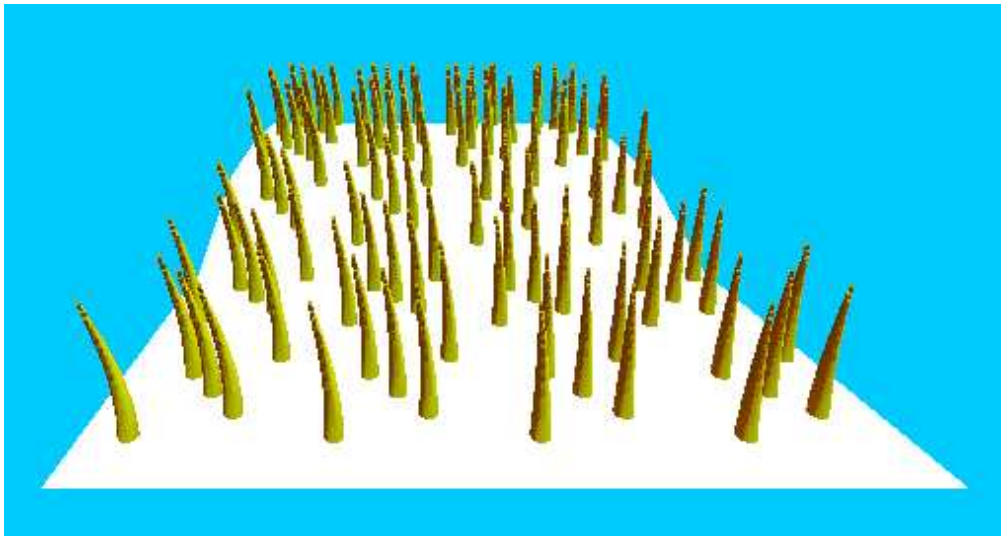


Figure 122 : Habillage d'une surface déformée (60 boites) par le texel normal pour montrer la discontinuité.

4.2 Perturbation de la surface plate selon les axes x et y par une fonction sinusoïdale

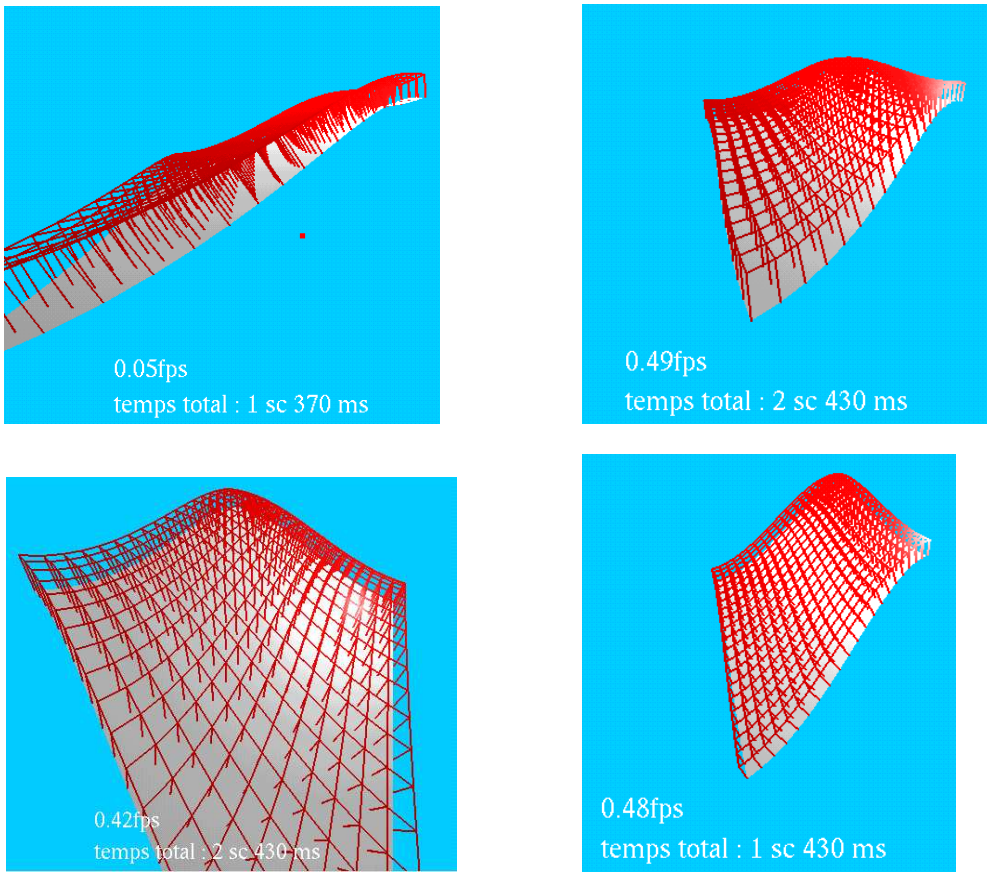


Figure 123 : Différents points de vues pour une surface déformée constituée de 1600 boites.

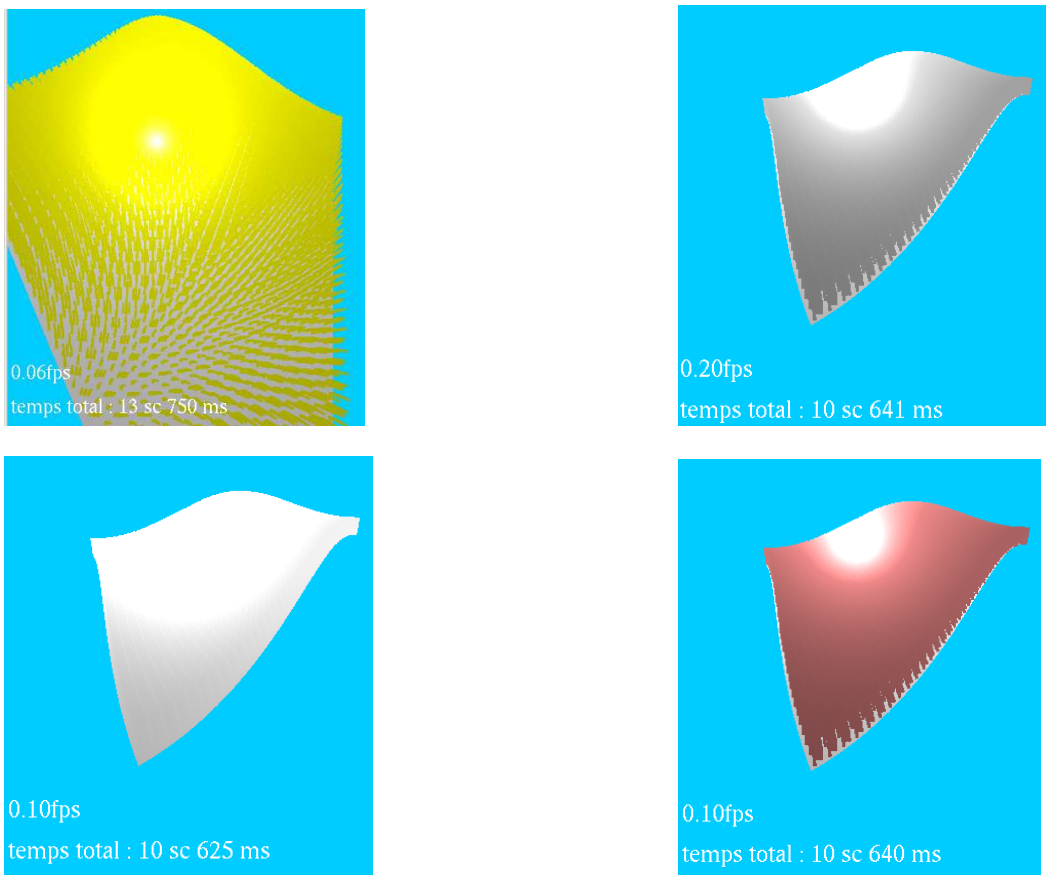


Figure 124 : Habillage de la surface précédente par la fourrure.

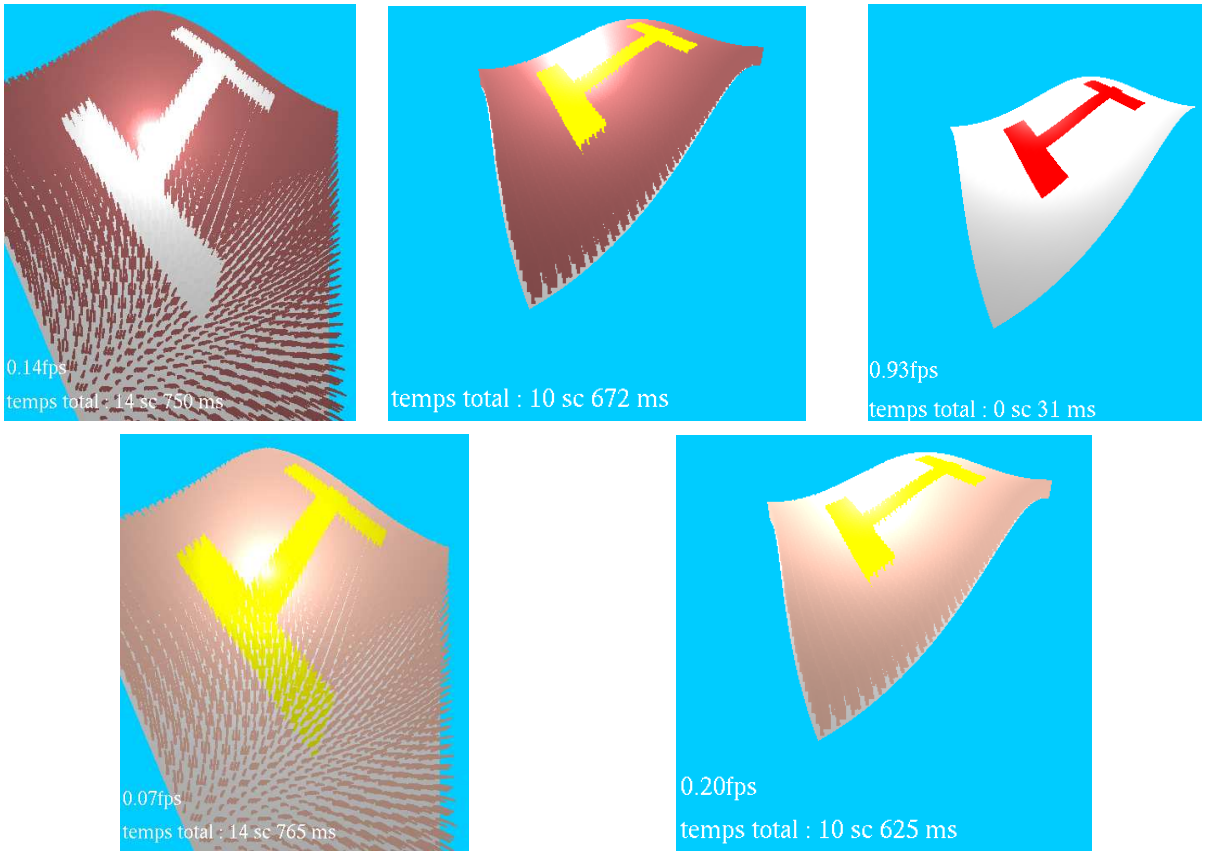
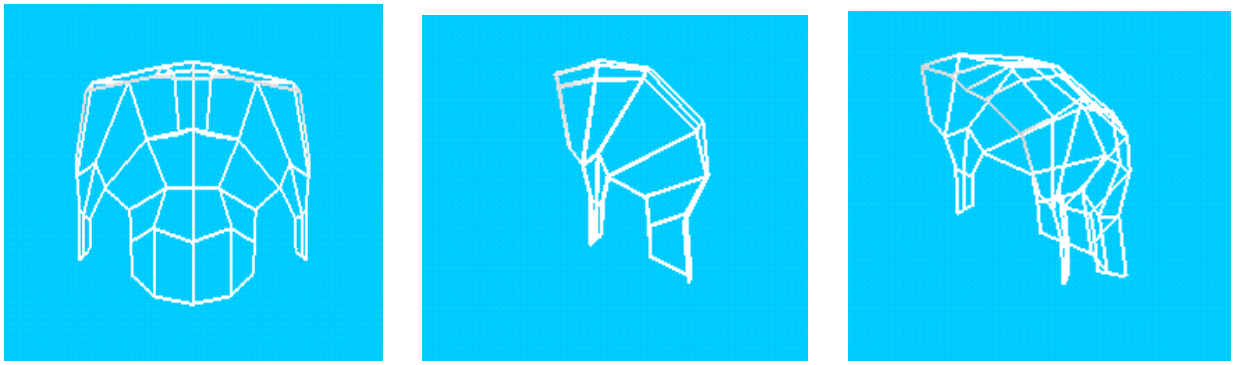


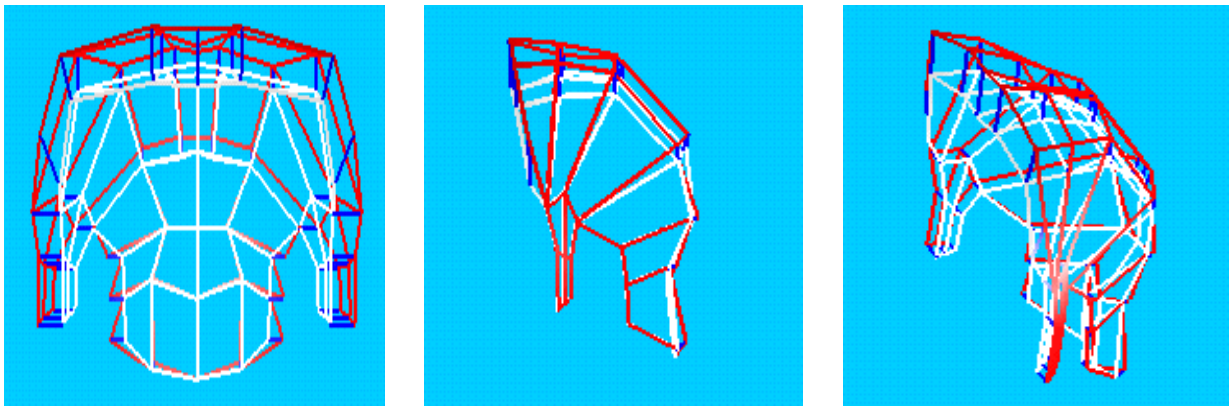
Figure 125 : Ecriture de la lettre H sur la peau; en utilisant les propriétés associées à chaque boite de la peau volumique (couleur et indexe de couleur), au lieu d'utiliser différents texels.

5 Habillage d’un crâne



vue de face **vue de Côté** **vue selon un angle déterminé**

Figure 126 : Modélisation d'un crâne de 38 polygones.



vue de face **vue de côté** **vue selon un angle déterminé**

Figure 127 : Modélisation de leur peau volumique.

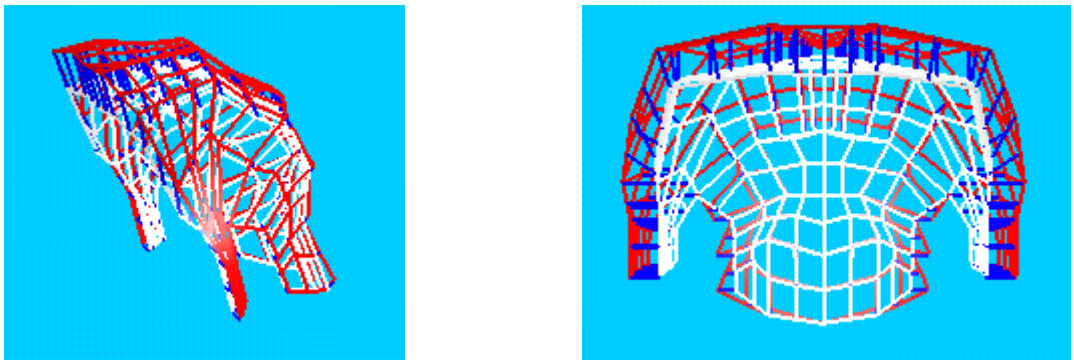


Figure 128 : Modélisation de peau volumique d'un crâne de 304 polygones.

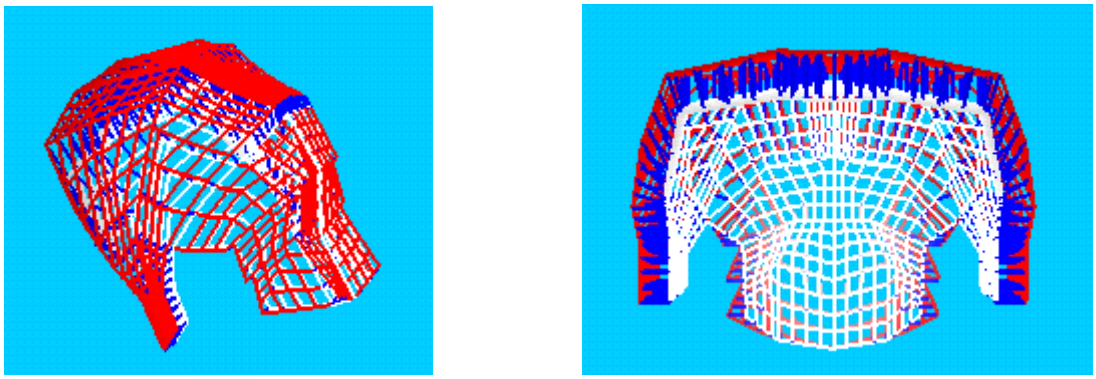


Figure 129 : Modélisation de peau volumique d'un crâne de 608 polygones.

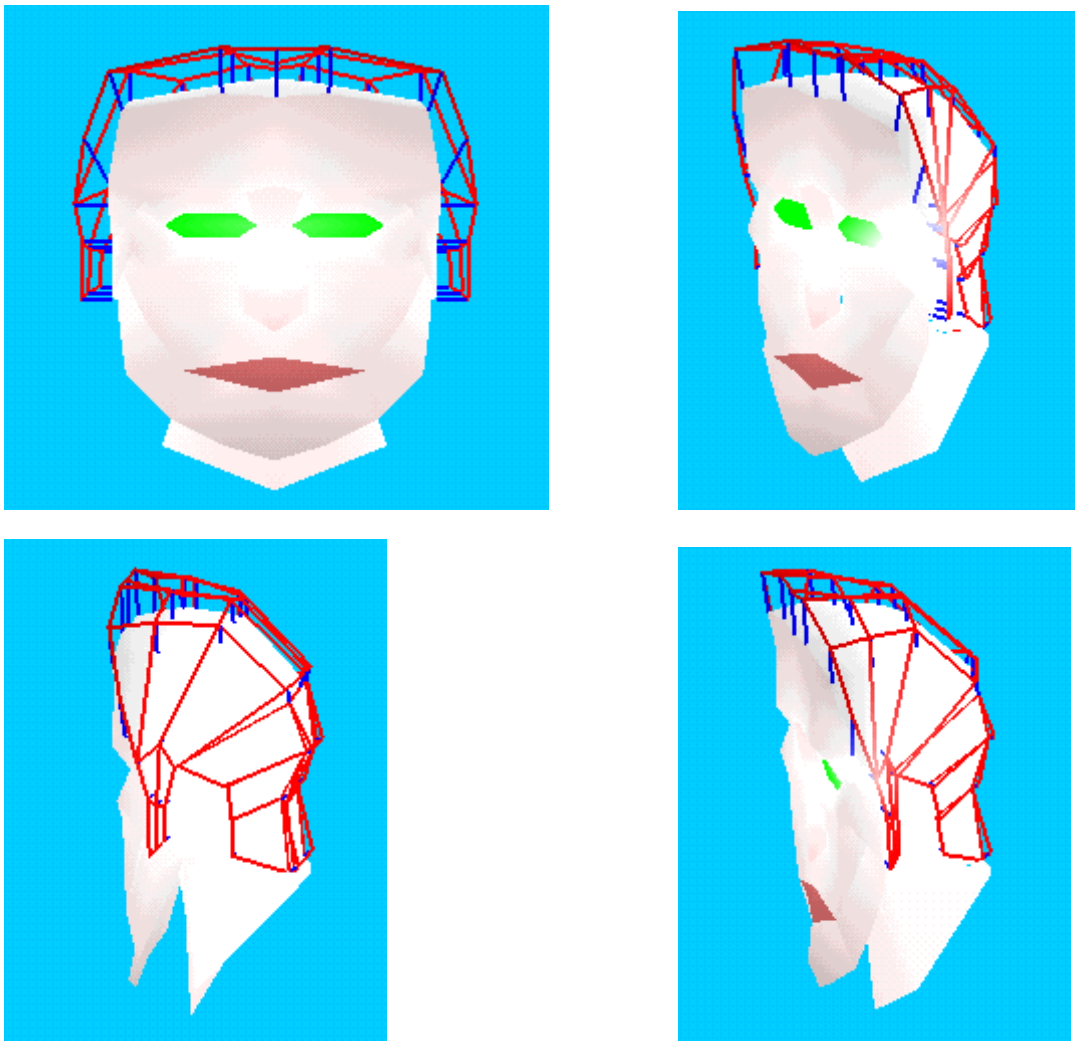


Figure 130 : Différents points de vue d'une tête.

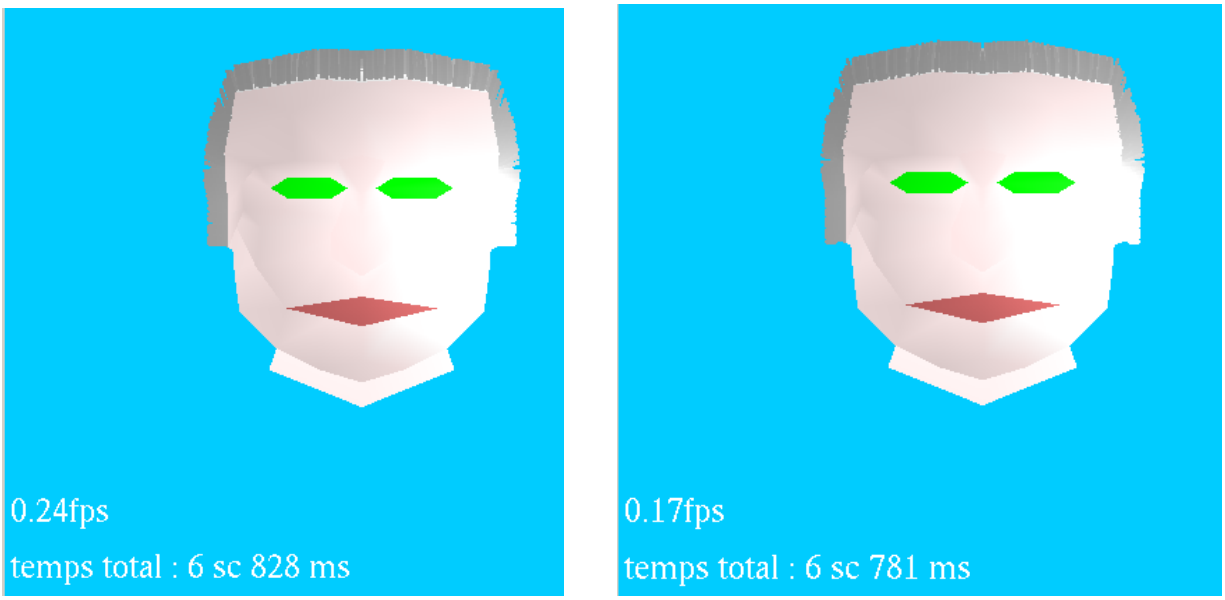


Figure 131 : A gauche Tête avec discontinuité de fourrure ; à droite tête avec continuité de fourrure.

Figure 132 : Habillage d'un crâne de 608 boites .

Figure 133 : Habillage d'un crane, en utilisant la propriété de boites pour ajouter une mèche blanche.

6 Habillage d’une maillage triangulaire

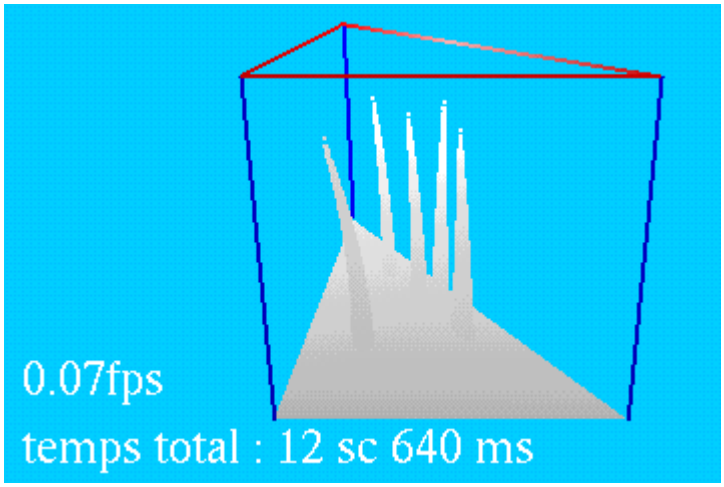


Figure 134 : Habillage d'une boîte de forme triangulaire.

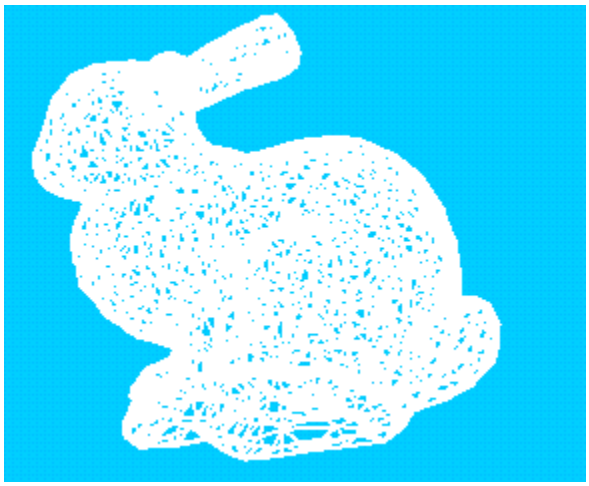


Figure 135 : Lapin de Stanford, formé de 2914 triangles.

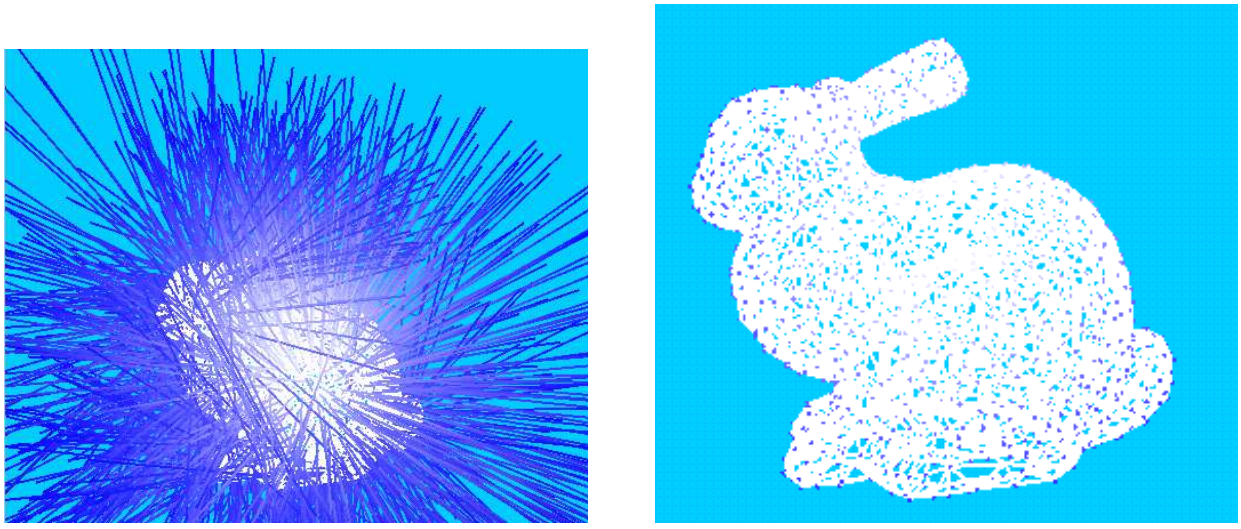


Figure 136 : Calcul de normale, à gauche les normales brutes; à droite les normales coiffées.

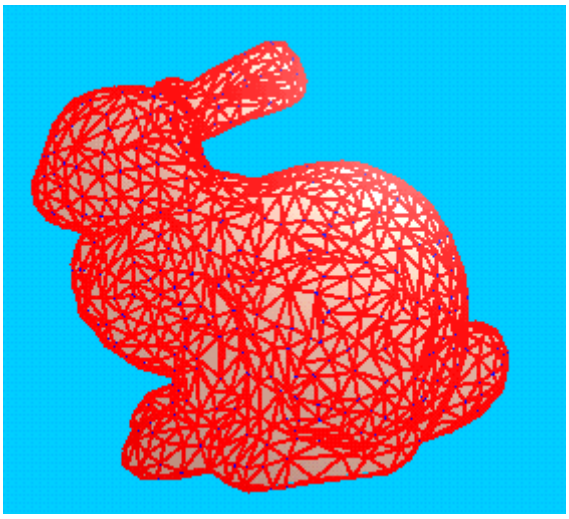


Figure 137 : La peau volumique construite autour du lapin.

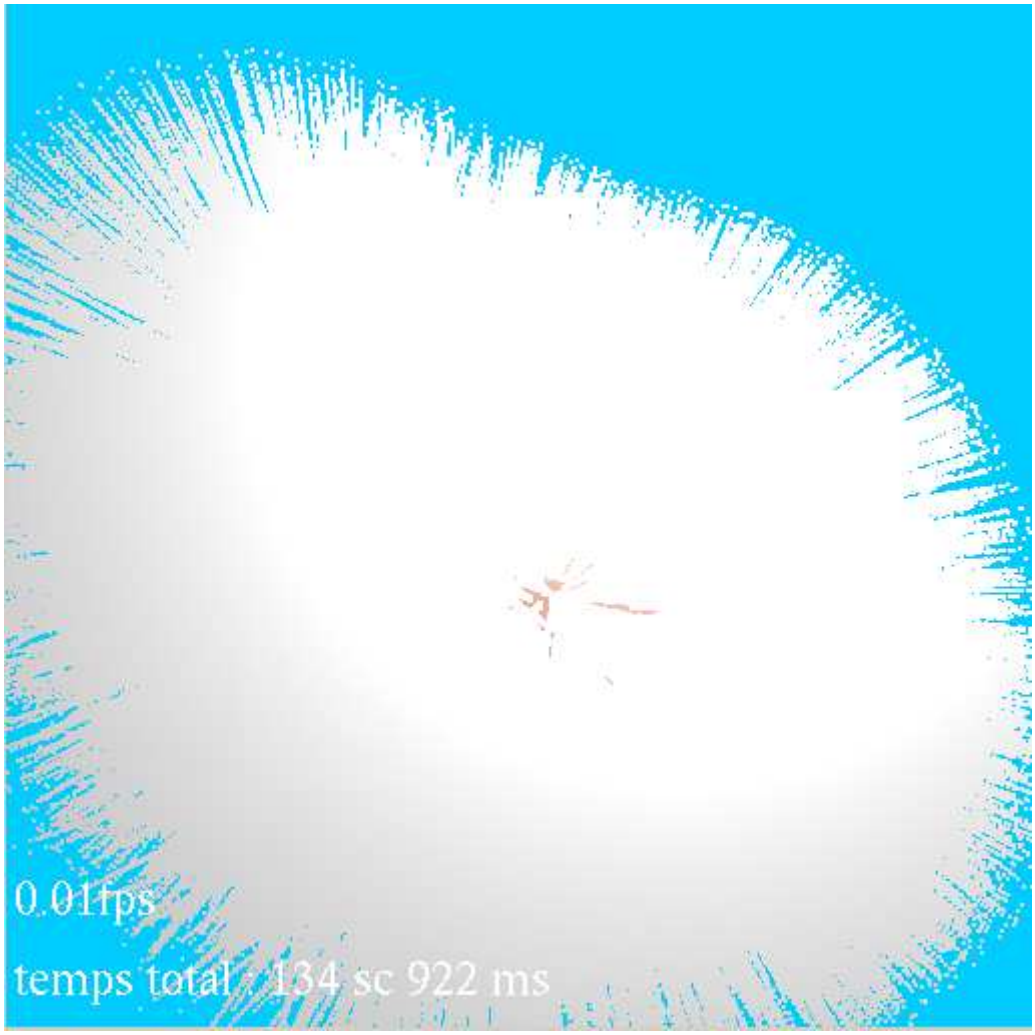


Figure 138 : Habillage de lapin sans coiffage.

Figure 139 : Habillage de lapin coiffé par motif isolé, on aperçoit la peau du lapin.

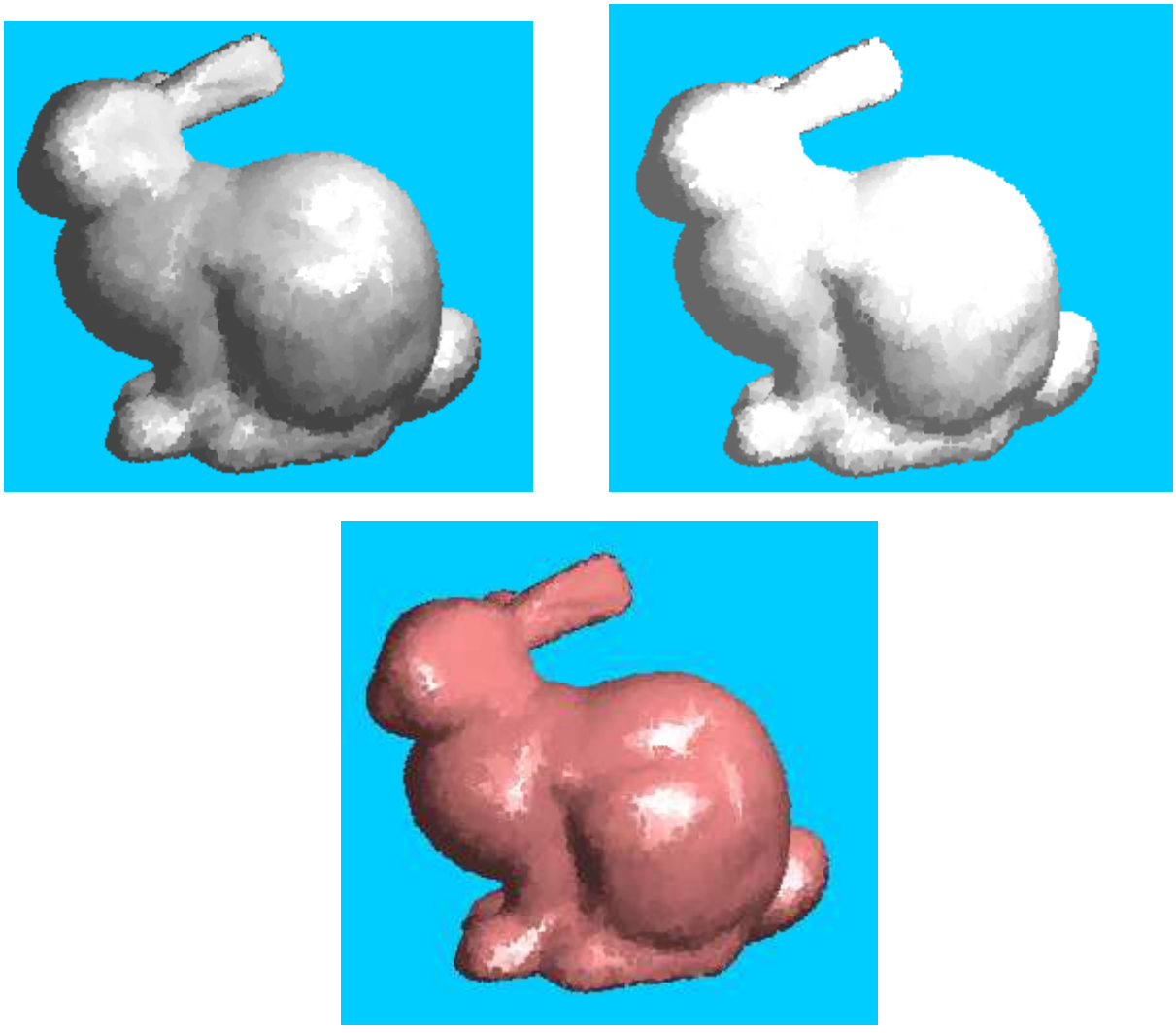


Figure 140 : Habillage de lapin par un motif gris, blanc et magenta. (temps de rendu sur un P4 de 3G est inférieur à 15 sc)

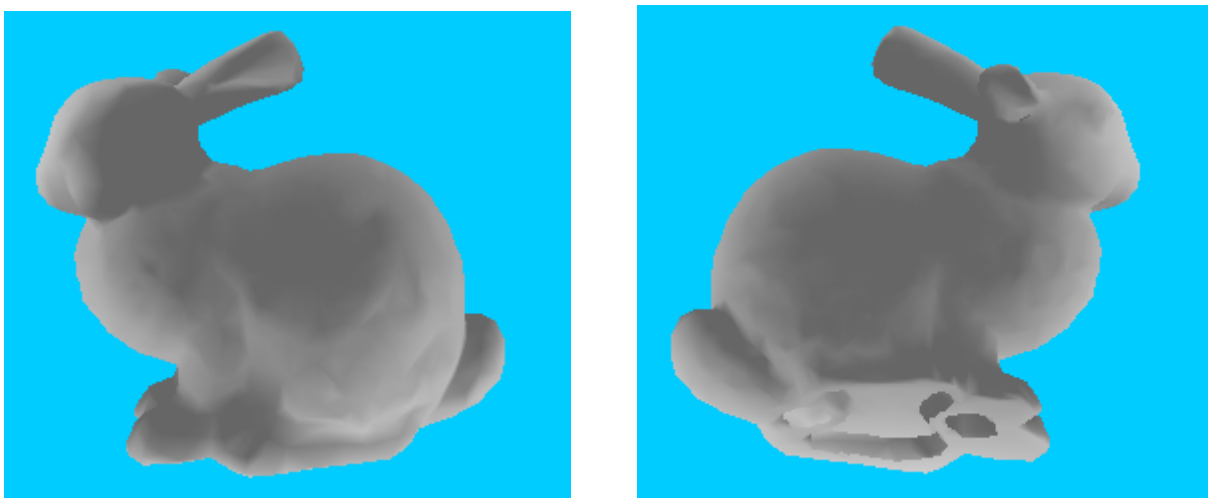


Figure 141 : Utilisation de normale de la surface sous jacente (triangle) avec la normale du surfel. (temps de rendu sur P4 de 3G est inférieur à 15 sc)

7 Conclusion

L'objectif de ce chapitre a été réservé exclusivement au développement et à l'exposition de quelques images générées par notre méthode afin d'habiller des surfaces tridimensionnelles. Ces images prouvent la flexibilité de l'algorithme de rendu, qui se résume en :

- L'indépendance entre le volume de référence et l'algorithme de rendu, ce qui nous permet d'atteindre les améliorations suivantes :
  - Réduire le stockage du volume de référence tout en ignorant la composante couleur.
  - Possibilité de mélanger la normale de surface avec celle du surfel (Figure 141).
- La possibilité d'utiliser les propriétés des boites formant la peau volumique (Figure 125).

Ce que nous pouvons retenir tout au long du parcours qu'a accomplie la science en matière d'imagerie. Il y a une nette et appréciable amélioration tant sur le plan machine que sur le plan méthodes et techniques adaptées dans cette optique

CONCLUSION GENERALE

Aujourd'hui, la synthèse d'image connaît un essor notable. Dès son apparition, l'homme a réussi à représenter son univers avec un degré d'affinité, de réalisme et sans contraintes de l'espace observé et la position de l'observateur, ce qui est impossible dans les techniques traditionnelles (peinture, orthographie, vidéo, .. .etc.).

Notre étude a consacré une large part à la mise en œuvre d'une méthode multi-échelles pour un affichage efficace, une meilleure visualisation des scènes complexes, exhibant une grande répétitivité. Celle-ci a pour but l'élimination d'élément géométrique de répétitivité dans ces scènes afin de le relayer par un volume contenant un nuage de points de cette géométrie à plusieurs niveau de résolution. L'habillage de l'objet s'obtient avec un rendu à base de points, durant lequel une simple optimisation de projection « warping » est suffisante pour minimiser la distorsion. De plus un modèle d'éclairement local est utiliser.

Nous préconisons une nouvelle méthode de représentation et de visualisation de texture volumique. Celle-ci offre une solution aux problèmes de la complexité géométrique des scènes par la combinaison des textures volumiques classiques et le paradigme de rendu à base de points, ainsi que l'acquisition d'une méthode souple. Les principaux avantage sont :

- Réduction d'espace mémoire : la texture volumique permet de réduire le stockage par l'utilisation des éléments de références ; le rendu à base de points permet de compressé ce dernier en utilisant le nuage de points au lieu de l'information de réflectance.
- Minimisation de temps de calcul (rendu) : le plaquage de texel est réalisé au sein d'une plate forme de rendu à base de points au lieu du lancer de rayons, ce qui nous a permis un gain en temps de calcul ; particulièrement lorsque le volume de référence est un objet complexe (elle ne dépend pas de la complexité d'objet, mais du nombre de points représentant ce dernier).
- Minimisation de l'aliassage et de la distorsion.
- La méthode est exploitable sur un PC.
- La possibilité d'affranchir l'animation de texture volumique puisque la déformation est déjà prisse en compte.
- La qualité d'image ne dépend plus du niveau de résolution de l'octree mais de la qualité d'échantillonnage utilisée lors de la création des LDIs.
- Indépendance du modèle de matériau et d'illumination.
- Facilité d'intégration avec OpenGL est visualisation des texels isolés.

Néanmoins, cette méthode soufre de certains inconvénients, à savoir :

- Le LDI n'offre pas une représentation de structure mince.
- Le z-buffer hiérarchique ne permet pas l'affichage des objets transparents ou semi-transparents.
- L'échantillonnage par lancer de rayons peut durée longtemps.

HORIZONS

L'idée essentielle de la texture volumique multi- résolutions utilisée pour l'habillage des surfaces 3D est très simple. Cependant, un grand nombre de détailles entrent en jeu à la mise en œuvre de la méthode de construction du volume de référence et de l'habillage par le texel créé. Nous n'avons pas pu explorer les différentes options qui se présentaient à chaque étape ; c'est pourquoi nous soulignons ici de nombreuses perspectives envisageables à notre système.

Pour ne citer ; les différentes parties à enrichir ou à améliorer dans notre système :

- La génération des surfaces 3D à habiller : la modélisation d'autres types de surfaces tel que, les surfaces implicites, tout en tenant en considération les cas où ces surfaces présentent de fortes déformations.

- La construction de volume de référence : nos ambitions pour la construction du volume de référence se résument aux points suivants :
  - L'utilisation des points différentiels offre la possibilité de rendre les objets transparents lors de la reconstruction d'image.
  - La compression du volume de référence : consiste à conserver seulement la position du point et la normale. Ainsi l'indexe de matériau est associé à chaque boite de la surface volumique.
  - Introduction d'une nouvelle hiérarchie qui combine la représentation à base de points avec le rendu à base d'image (voxel coloring) : au niveau haute résolution on trouve les voxels coloring et en basse résolution on trouve les surfels.
- L'habillage de la surface (le rendu) :
  - L'utilisation de carte graphique pour accélérer le rendu.
  - L'utilisation d'autres méthodes de reconstruction d'image, afin de rendre les objets transparents (tel que le A-buffer).
  - Exploiter la notion texel dans le texel.
  - Utilisation des méthodes plus puissantes pour calculer l'ombre tels que le pixel-shading, vertexe-shading ou les cartes d'ombrage.

Bibliographie

[Alexa01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin et Claudio T. Silva « Point Set Surfaces ». In *IEEE Visualization 2001*. 21–28. ISBN 0-7803-7200-x.

[Alexa03] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin et Claudio T. Silva « Computing and Rendering Point Set Surfaces ». In *IEEE Transactions on Visualization and Computer Graphics 9, 1*. January 2003.

[Antoine01] Miné Antoine, « Texture procédurales en temps réel avec OpenGL ». *Rapport de stage* - laboratoire iMAGIS .16 juillet 2001.

[Assarsson99] Ulf Assarsson and Tomas Möller « Optimized View Frustum Culling Algorithms ». Chalmers University of Technology. Department of Computer Engineering. *Technical Report 99-3*; March 1999.

[Assarsson00] Ulf Assarsson and Tomas Möller « Optimized View Frustum Culling Algorithms for Bounding Boxes ». Department of Computer Engineering. Chalmers University of Technology, Sweden. Accepted for publication in *journals of graphics tools.* February 2000.

[Baar01] J. van Baar, M. Zwicker, H. Pfister et M. Gross « Surface splatting ». *In Computer Graphics*. In SIGGRAPH '01 Proceedings. Août 2001.

[Blais98] Martin Blais « Excursions en rendu par champ de lumière: champ de visibilité et ré-illumination » , *Mémoire de Maîtreès sciences (M.Sc.)* . Université de Montréal, Octobre 1998.

[Botsch02] M. Botsch, A. Wiratanaya, L Kobbelt « Efficient High Quality Rendering of Point Sampled Geometry » In *Rendering Techniques 2002* (Proceedings of the Eurographics Workshop on Rendering 02). Springer Verlag, 2002.

[Chang99] Chun-Fa Chang, Gary Bishop et Anselmo Lastra « LDI Tree: A Hierarchical Representation for Image-Based Rendering » University of North Carolina at Chapel Hill. In *Computer Graphics*, SIGGRAPH '99 Proceedings, pages 291–298. Los Angeles, CA, August 1999.

[Chen01] B. Chen and M. X. Nguyen. « POP: A Hybrid Point and Polygon Rendering System for Large Data ». In *IEEE Visualization 2001*, pages 45–52, October 2001. ISBN 0-7803-7200-x.

[Cohen01] J. D. Cohen, D. G. Aliaga, and W. Zhang « Hybrid simplification: combining multi-resolution polygon and point rendering ». In *IEEE Visualization 2001*, pages 37–44, October 2001. ISBN 0-7803-7200-x.

[Debunne04] Gilles Debunne, « Rendu à base d'images ». *DEA IVR* . 2004.

[Décoret02] Xavier Décoret, Frédo Durand, François X. Sillion et Julie Dorsey « Billboard Clouds ». *Rapport d'activité INRIA*, juin 2002.

[Duguet03] Florent Duguet et George Drettakis « Flexible Point-Based Rendering on Mobile Devices ». *INRIA*. Rapport de recherche n° 4833; Mai 2003.

[Eberly02] David Eberly «Intersection of a Sphere and a Cone». *Magic Software*, Inc. http://www.magic-software.com. March 8, 2002.

[Fleishman03] S. Fleishman, M. Alexa, D. Cohen-Or et Claudio T. Silva « Progressive Point Set Surfaces ». *ACM Transactions on Graphics*, Vol. 22, No. 4. October 2003.

[Gortler96] Stephen J. Gortler, Radek Grzeszczuk, Richard Szeliski, Michael F. Cohen « The Lumigraph ». Proc. SIGGRAPH '96. *In Computer Graphics Proceedings*, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 43-54.

[Greene93] Ned Greene Michael Kass Gavin Miller «Hierarchical Z-Buffer Visibility ». *In Computer Graphics Proceedings.* June 1993.

[Gross02] M. Gross, H. Pfister, M. Alexa, M. Pauly, M. Stamminger, and M. Zwicker. « Point-based computer graphics ». *Eurographics '02 Tutorial T6*, 2002.

[Grossman98a] J. P. Grossman «Point Sample Rendering». *Master's thesis*, Department of Electrical Engineering and Computer Science, MIT. August 1998.

[Grossman98b] J. P. Grossman et W. Dally « Point Sample Rendering ». In *Rendering Techniques '98*, pages 181–192. Springer, Wien, Vienna, Austria, July 1998.

[Guennbaud02] Gaël Guennebaud « Représentations alternatives et visualisation des objets complexes ». *thèse de DEA Informatique de l'Image et du Langage.* Université TOULOUSE III Paul Sabatier 2001/2002.

[Guillou00] Erwan Guillou « Simulation d'environnements complexes non lambertiens à partir d'images : Application à la réalité augmentée », *Thèse de doctorat,* Université de Rennes 1 - Décembre 2000.

[Hasenfratz97] Jean-Marc Hasenfratz - *Travaux de thèse* - Fin 1997. http://w3imagis.imag.fr/~Jean-Marc.Hasenfratz/RECHERCHE/THESE/renduRealiste.html.

[Heckbert89] P. Heckbert «Fundamentals of Texture Mapping and Image Warping ». *Master's thesis,* University of California at Berkeley, Department of Electrical Engineering and Computer Science, June 17 1989.

[Herman92] G. T. Herman «Discrete Multidimensional Jordan Surfaces». *CVGIP: Graphical Modeling and Image Processing*, 54(6):507–515, November 1992.

[Hinsinger01] Damien Hinsinger, Fabrice Neyret, Marie-Paule Cani « Interactive Animation of Ocean Waves ». *Rapport d'activité – iMAGIS* . 2001.

[Kalaiah01] Aravind Kalaiah and Amitabh Varshney « Differential point rendering ». In *Proceedings of 12th Eurographics Workshop on Rendering*, pages 139–150, June 2001.

[Kalaiah03] Aravind Kalaiah et Amitabh Varshney « Modeling and Rendering of Points with Local Geometry ». *IEEE transaction on visualization and computer graphics*, Vol.9, N°. 1, January-March 2003.

[Kenneth97] Kenneth E. Hoff III « Reviewer of a Cellular Texture Basis Function ». *Computer Graphics Research : Procedural Texturing* . 11 Avril 1997.

[Kobbelt04] Leif Kobbelt Mario Botsch « A Survey of Point-Based Techniques in Computer Graphics » *Computer Graphics Group,* RWTH Achen University. 12 July 2004.

[Krivanek03] Jaroslav Krivanek « Representing and Rendering Surfaces with Points ». Czech Technical University Department of Computer Science and Engineering. *Postgraduate Study Report* DC-PSR-2003-03 ; February 2003.

[Krus96] Mike Krus, Françoise Guisnel, Patrick Bourdot et Guillaume Thibault « Niveaux de Détails et Simplification Polygonaux : un tour d'horizon.». AFIG'96, *Quatrièmes Journées de l'Association Française d'Informatique Graphique* - 1996.

[Krus97] Mike Krus « Maillages Polygonaux et Niveaux de Détails étude bibliographique » *Rapport Technique*, Electricité de France, Direction des Etudes et Recherches . Mai 1997.

[Kumar96] Subodh Kumar, Dinesh Manocha, Bill Garrett, Ming Liny « Hierarchical Back-Face Computation » Department of Computer Science. University of North Carolina. Chapel Hill NC 27599 USA. *An Extended abstract appeared in Proc. of Eurographics Workshop on Rendering*, June 1996.

[Leblanc00] Luc Leblanc « Construction et utilisation des bloqueurs pour l'accélération des requêtes de visibilité ». *Mémoire de Maître ès sciences* , Université de Montréal. Août 2000.

[Lefebvre01] Sylvain Lefebvre « la simulation visuellement réaliste de surfaces à base de déchirures ». S*tage de DEA* - 2001.

[Levoy85] Marc Levoy et Turner Whitted «The Use of Points as a Display Primitive » *Technical Report 85-022*, Computer Science Department, University of North Carolina at Chapel Hill, January, 1985.

[Lischinski98] D. Lischinski and A. Rappoport. « Image-Based Rendering for Non-Diffuse Synthetic Scenes. » In *Rendering Techniques '98*, pages 301–314. Springer, Wien, Vienna, Austria, June 1998.

[Luebke01] David P. Luebke « A Developer's Survey of Polygonal Simplification Algorithms ». Université de Virginia - May/Juin 2001.

[Mark97] William R. Mark, Leonard McMillan, Gary Bishop, "Post-Rendering 3D Warping", Proc. 1997 Symposium on Interactive 3D Graphics, pp. 7-16

[Matusik02a] W. Matusik, H. Pfister, R. Ziegler, A. Ngan, and L. McMillan. « Acquisition and rendering of transparent and refractive objects ». In *Eurographics '02 Proceedings*, 2002.

[Matusik02b] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. « Image-based 3D photography using opacity hulls ». In *Siggraph '02 Proceedings*, 2002.

[Max95] N. Max and K. Ohsaki. « Rendering trees from precomputed z-buffer views ». In *Proceedings of the 6th Eurographics Workshop on Rendering*, pages 45–54, June 1995.

[Meyer00] A. Meyer et F. Neyret. « Multiscale shaders for the efficient realistic rendering of pine-trees ». In *Graphics Interface*, May 2000.

[Meyer01a] A. Meyer, F. Neyret, et P. Poulin « Interactive rendering of trees with shading and shadows ». In *Eurographics Workshop on Rendering*, Juillet 2001.

[Meyer01b] Alexandre MEYER, « Représentations d'arbres réalistes et efficaces pour la synthèse d'images de paysages ». *Thèse de Phd* - Université Joseph Fourier , 10 décembre 2001.

[Neyret96] F. Neyret, « Textures volumiques pour la synthèse ». *Thèse de doctorat es sciences*, Université Paris XI Orsay - juin 1996.

[Neyret97] Fabrice Neyret « Qualitative Simulation of Convective Cloud Formation and Evolution ». Septembre 1997.

[Neyret98a] F. Neyret « Modeling, Animating and Rendering Complex Scenes Using Volumetric Textures ». In *IEEE Transaction On Visualisation and Computer graphics* - vol.4, NO 1 – Janvier/Mars 1998.

[Neyret98b] Fabrice Neyret, Alexandre Meyer, « Textures volumiques interactifs ». *Rapport d'activité* , IMAGIS - 1998.

[Neyret00] Fabrice Neyret, « A phenomenological shader for the rendering of cumulus clouds ». *Rapport technique*, INRIA - May 2000.

[Neyret01] Fabrice Neyret, « Complexité Naturelle et Synthèse d'Images ». *Mémoire d'Habilitation à Diriger des Recherches,* 22 octobre 2001.

[Parent02] Rick Parent, « Computer Animation Algorithms and Techniques ». *Edition MORGAN KAUFMANN PUBLISHERS* - 2002.

[Monks99] Michael Monks «Cellular Textures and 3D Painting: 6.838 L15 ». *Cours de synthèse d'image*. http://graphics.lcs.edu/~mcm

[Pauly01] M. Pauly and M. Gross « Spectral processing of point-sampled geometry ». In *SIGGRAPH Computer Graphics Proceedings,* Août 2001.

[Péroche90] B. Péroche et al. « La synthèse d'image ». *Edition Hermès* - 1990.

[Péroche98] B. Péroche et al. « Informatique graphique : Méthodes et Modèles » - 2ème Edition revue et augmentée, *Edition Hermès* - 1998.

[Pfister00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross « Surfels : Surface elements as Rendering primitives ». In *SIGGRAPH 2000, Computer Graphics Proceedings*, 2000.

[Pierre99] Dan Stora Pierre, Olivier Agliati Marie-Paule Cani, Fabrice Neyret, Jean-Dominique Gascuel « Animating Lava Flows ». *Rapport d'activité* - IMAGIS – Juin 1999.

[Porquet04] Damien Porquet « Rendu en temps réel de scène complexe ». www.msi.unilim.fr_porquet\memoire\memoire-html.html - 29 Novembre 2004.

[Ratib97] Karim Ratib « Texture volumique multi-échelle pour l'affichage de scènes complexe ». *Mémoire du grade de Maître és sciences en informatique*, Université de Montréal, Décombre 1997.

[Ren02] L. Ren H. Pfister et M. Zwicker « Object Space EWA Surface Splatting: A Hardware Accelerated Approach to High Quality Point Rendering ». *In Proceedings of EUROGRAPHICS 2002*, Volume 21 (2002), Number 3.

[Reunanen04] Markku Reunanen « Point-Based Modeling ». Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory. 7th April 2004.

[Rushmeier01] Holly Rushmeier, Laurent Balmelli, Fausto Bernardini « Horizon Map Capture » *Rapport d'activité.* IBM T. J. Watson Research Centery - *EUROGRAPHICS 2001*.

[Rusinkiewicz00] S. Rusinkiewicz et M. Levoy « QSplat : A multiresolution point rendering system for large meshes ». In *SIGGRAPH 2000, Computer Graphics Proceedings*, pages343–352. 2000.

[Shade98] J. Shade, S. J. Gortler, L. He, and R. Szeliski « Layered Depth Images. » In *Computer Graphics, SIGGRAPH '98 Proceedings*, pages 231–242. Orlando, FL, July 1998.

[Schaufler98] Gernot Schaufler « Per-Object Image Warping with Layered Impostors ». *Rapport d'activité*. Université de Linz - AUSTRIA (Autruche).1998.

[Schwarze03] Tino Schwarze « Kommunikations mechanismen für paralleles, adaptives Level-of-Detail in VR-Simulationen » - *Thèse de doctorat.*- Université technique de Chemnitz - 5 Mars 2003.

[Stamminger01] M. Stamminger et G. Drettakis « Interactive sampling and rendering for complex and procedural geometry ». In *Rendering Techniques 2001* (Proceedings of the Eurographics Workshop on Rendering 01) - Eurographics, 2001.

[Theubl99] Thomas Theubl «Sampling and Reconstruction in volume visualisation ». *thèse de doctorat*, Université Technique de Vienne, décembre 1999.

[Vincent02] Furminieux Vincent et Matthieu Audoin «LOD et Progressive Meshes », novembre 2002 - *document PowerPoint.*

[Wand00] Michael Wand, Matthias Fischer, Friedhelm Meyer auf der Heide « Randomized Point Sampling for Output-Sensitive Rendering of Complex Dynamic Scenes ». Heinz Nixdorf Institute Paderborn University ; Department of Mathematic and Computer Science D-33095 Paderborn, Germany ; November 2000.

[Wand01] M. Wand, M. Fischer, I. Peter, F. M. Heide and W. Straßer « The Randomized Z-Buffer Algorithm: Interactive Rendering of Highly Complex Scenes », ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA.

[Zwicker01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross « EWA volume splatting » In *Proceedings of IEEE Visualization*, 2001.

[Zwicker02a] M. Zwicker « Point-Based Rendering » *Computer Graphics Lab ETH Zürich*, July 2002.

[Zwicker02b] M. Zwicker, M. Pauly, O. Knoll, M. Gross « Pointshop 3D: an interactive system for point-based surface editing ». *Proceedings of SIGGRAPH 2002, to appear*, San Antonio, TX, July 2002.

[Zwicker02c] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. « EWA splatting ». *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

Rendu de textures volumiques par une représentation à base de points

Résumé

Introduites par KAJIYA et KAY en 1989, Les textures volumiques sont un modèle multi-échelle, qui permet de représenter des géométries complexes. Il utilise trois niveaux différents d'information pour traiter la complexité :

- Les grandes variations : la surface d'une colline ou le dos d'un animal, sont codés par une description géométrique classique (mailles des polygones, carreaux de Béziers, surface NURBS, …).
- Le niveau de détail moyen, exemple l'herbe ou le poils, concentrés au voisinage de la surface, sont codés en utilisant un *volume de référence* stocké une seule fois et *plaqué répétitivement*, à la manière d'une texture 2D. Une instance de ce volume de référence est appelée *texel*.
- Le niveau de détail fin : les variations microscopiques de chaque objet, sont codées par un *modèle de réflexion* stockées dans chaque *voxel*. Ce niveau correspond au niveau du pixel.

Les méthodes classiquement proposées présentent quelques inconvénients, à savoir :

- Le stockage intégral du volume de référence, engendre un surcoût en matière de mémoire.
- Un longue temps de calcul.
- Les effets visuelles d'aliassage et de distorsion.
- Le coût inhérent à l'animation du texel.
- Inexploitable sur un PC.

Le but de l'étude est de remédier à ces inconvénients par l'introduction du rendu à base de points, plus avantageux et connaît un grand essor en ce moment. Afin de réaliser ceci, nous venons de développer une méthode hybride qui marie les textures volumiques et le rendu à base de points. Cette dernière se englobe deux parties :

- En premier lieu : nous développons une représentation compacte et muti-échelle de volume de référence. Celle ci est représentée par un LDC tree normalisé où chaque voxel est un bloc de LDC contenant ainsi une partie d'échantillon de points.
- En second lieu : nous adaptons l'algorithme de rendu d'échantillon de points de *Grossman* au rendu de textures volumiques. En traitant ainsi les problèmes du test de visibilité, de déformation de texture, de plaquage du texel dans la peau volumique, l'antialiassage et la reconstruction d'image.

Mots clés : Synthèse d'image, complexité, rendu à base de points, LDC tree, multi-résolution, textures volumique, réalisme, visibilité, ombrage.