Textures volumiques interactives

Alexandre Meyer et Fabrice Neyret
iMAGIS, laboratoire GRAVIR/IMAG-INRIA
`[Alexandre.Meyer|Fabrice.Neyret]@imag.fr`

Résumé : *Ce papier présente une méthode interactive pour effectuer le rendu de scènes complexes répétitive comme par exemple un paysage, de la fourrure, des tissus organique, etc... C'est une adaptation au Z-buffer des textures volumiques avec l'idée d'utiliser les capacités graphiques des nouvelles cartes graphiques. Notre approche consiste à représenter un motif de la texture volumique par une superposition de tranches, oú une tranche est en fait un polygone texturé contenant l'information de couleur (l'éclairage étant donc précalculé). Le motif de texture est plaqué répétitivement sur une surface (une colline, un chat, etc...). Nous montrons des résultats de notre implémentation, une scène de 13 millions de polygones virtuels animée à 5 images/secondes sur une station SGI* O_2.

Mots-clés : Textures volumiques, temps réel

1 Introduction

La complexité visuelle est une partie importante du réalisme d'une scène, spécialement pour une scènes comme un paysage, de la fourrure, des tissus, etc... Si nous représentons cette complexité, souvent répétitive, par des polygones cela prend beaucoup de temps de calcul et engendre des effets d'aliasing. Dans certain cas ces détails peuvent être représenté avec une texture 2D, mais la plus part du temps ils sont réellement 3D, c'est-à-dire produit un effet de parallaxe quand le point de vue bouge (exemple : des arbres sur une colline). L'autre approche consistant à supprimer les plus petits polygones ne donne pas un résultat visuel satisfaisant. La situation est encore pire si l'on veut obtenir une application temps réel, ce qui fait que souvent le temps réel est synonyme de pauvreté visuelle en terme de détails.

Le fait que les détails ne soient pas plats n'implique pas que l'on doivent nécessairement les représenter avec un modèle 3D à base de polygones, complexe et cher. L'impression de 3D est une notion progressive qui fait entrer en jeu beaucoup de facteurs : la forme globale de l'objet, les mouvements dû à la parallaxe, les occultations, les ombres, les réflections diffuses, etc... En fonction de la taille des détails certaines de ces propriétés peuvent suffire pour donner une impression de 3D. Un bon moyen pour donner cette impression de 3D sans aliasing est de ne représenter que le minimum d'information nécessaire à ce que l'on peut voir.

1.1 Travaux antérieurs

Les textures volumiques, introduite par Kajiya et Kay en 1989 [KK89] et étendues par Neyret [Ney95b] [Ney95a] [?], utilisent 3 différents niveaux pour représenter l'information (voir figure 1) :

- les grandes variations comme la surface d'une colline ou le dos d'un animal sont codées en utilisant une triangulation classique de la surface.
- le niveau de détails moyen, comme l'herbe ou les poils, qui sont concentrés au voisinage de la surface, est codé en utilisant un volume de référence stocké une seul fois et plaqué répétitivement, à la manière d'une texture 2D (une instance de ce volume de référence est appelée *texel*.
- le niveau de détails fin, comme les microscopique variations de chaque objet, sont codés par un modèle de réflexion stocké dans chaque voxel. Dans l'extension multiéchelle des textures volumiques ce niveau correspond au niveau du pixel.

Si l'on met en relation ce formalisme avec les critères d'impression de 3D cité en introduction, on peut voir que l'on utilise une description explicite (par des polygones) pour le niveau d'échelle

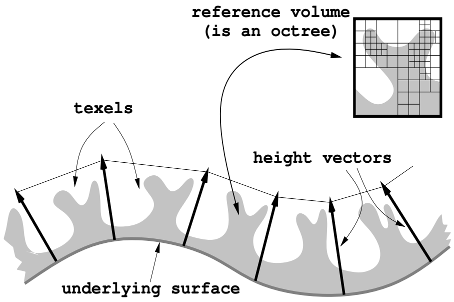

FIG. 1: Spécifications des textures volumiques.

élevé ; au niveau moyen une représentation volumique suffit pour les effets de parallaxe et les occultation (ce niveau correspond à quelques pixels) ; le modèle d'illumination stocké dans chaque voxel simule la géométrie au niveau inférieur à celui du pixel.

Le rendu des texels comporte des similitudes avec le rendu volumique qui utilise aussi le ray-tracing. En 1994 Lacroute et Levoy ont introduit une nouvelle approche [LL94] adaptée au nouvelles fonctionnalités des processeurs des cartes graphiques, qui a rendu interactif le volume rendering. Cette approche consiste à regrouper les voxels en tranches qui peuvent ainsi être représentés par une texture contenant une valeur de transparence. Le rendu consiste alors à rendre la superposition de ces tranches. Les cartes graphiques d'aujourd'hui permettent le rendu de faces texturées transparentes sans coût prohibitif supplémentaire.

1.2 Notre approche

Ce papier présente une méthode permettant le rendu interactif de scène complexes et répétitive. L'idée est d'adapter les textures volumiques présentées précédemment au Z-buffer en utilisant la même approche que celle de Lacroute et Levoy. Nous voulons ainsi obtenir le même type de complexité visuelle qu'avec les textures volumiques, mais avec interactivité en plus.

Contrairement au rendu volumique la quantité de donnée volumique que nous traitons est petite. Nous pouvons donc rendre plusieurs volume interactivement. Par exemple un volume de 64 tranches peut être rendu avec 64 faces texturées (i.e. 2×64 triangles) alors que le même modèle représenté avec des polygones en contiendrait plusieurs milliers.

D'un autre cote l'utilisation des processeurs graphiques apporte des limitations : l'éclairage et les ombres sont précalculées, ils ne seront pas remis a jour si la source de lumière se déplace. Alors que le ray-tracing permet le déplacement de la lumière en gardant des ombres exactes.

Le plan de ce papier est le suivant : le paragraphe 2 décrit la représentation de notre modèle ainsi que l'algorithme de rendu correspondant ; celui-ci est amélioré dans le paragraphe 4.3. Le paragraphe 3 indique comment construire les objets à partir de modèles existants. La partie animation du paragraphe 5 est similaire à l'animation en ray-tracing des textures volumiques. Ce papier finit par une présentation des résultats (paragraphe 6) et une conclusion.

2 Représentation de base et rendu

Dans ce paragraphe nous introduisons notre méthode pour représenter des objets complexes constitué d'une surface couverte de détails répétitifs et nous expliquons comment en effectuer le rendu. La création des données fait l'objet du paragraphe suivant.

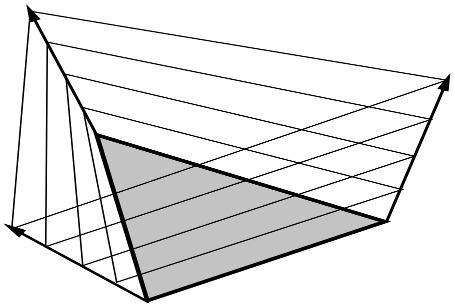

FIG. 2: Un texel est dessiné en rendant une superposition de faces texturées.

2.1 Structure de données

A la manière des textures volumiques utilisés en ray-tracing [Ney95b] 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 (voir la figure 2).

La partie volumétrique de notre modèle est différente de celle utilisé en ray-tracing : chaque voxel de notre 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.

2.2 Rendu

Le rendu utilise une carte graphique standard comportant des fonctionnalités 3D (OpenGL dans notre implémentation), en dessinant les faces texturées au dessus de chaque face de la surface texelisée. Nous pouvons utiliser le mipmap [Wil83] implanté en hardware pour supprimer l'effet d'aliasing quand la surface est lointaine ou bien que l'angle de vue est rasant. Il est à noter que l'utilisation de texture, de la transparence et du mipmap ne coûte pratiquement pas plus cher 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 l'on rend les polygones dans un ordre quelconque. Tant que l'on utilise une transparence qui ne peut prendre comme valeur que 0 ou 255 cela ne pose pas de problème : les pixels dont l'alpha vaut 0 ne sont pas tracés et les pixels dont l'alpha vaut 255 recouvre entièrement ce qu'il y a derrière. Lorsque l'on rend un pixel dont la transparence est semiopaque on prend bien en compte les pixels se trouvant derrière seulement si ils ont été rendu avant. Une solution pour résoudre ce problème est de rendre les tranches de l'arrière vers l'avant, ce qui est trivial au sein d'un unique texel mais qui demande un trie selon les Z des texels lorsque la scène en comporte plusieurs. Pour un unique texel rendre les faces de l'arrière vers l'avant est également utile aussi pour éviter les erreurs du à l'imprécision de stockage des Z.

Pour choisir l'ordre de tracé il suffit d'effectuer un produit scalaire entre le vecteur normale à la surface et la direction de vue, ceci si le texel n'est pas trop déformé par les vecteurs de hauteurs.

Comme nous utilisons un Z-buffer classique pour rendre chaque face, l'intersection de deux texels ne posent pas de problème, ce qui n'était pas le cas en ray-tracing. Cette propriété importante est illustrée sur la figure 9 (droite) dans le paragraphe 6.

3 Création du motif de référence

Dans ce paragraphe nous décrivons comment créer un motif de texture 3D. Nous faisons la conversion d'un modèle existant sous un autre format (i.e. facettes, ...). Bien sur utiliser une

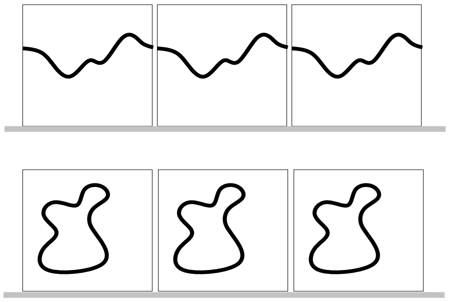

FIG. 3: En haut : topologie cyclique du motif. En bas : motif isolé.

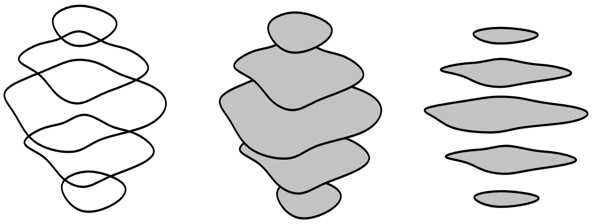

FIG. 4: Gauche : contours. Milieu : Contours remplis. Droite : On voit à travers les tranches.

approche texturale pour répéter les détails apporte des contraintes sur la construction du motif : le motif de texture 3D (c'est-à-dire le volume de référence) va se plaquer répétitivement sur la surface en suivant les coordonnées textures (u,v) qui se trouvent sur chaque sommet. Ceci implique que notre volume de référence soit cyclique, soit que les objets se trouvant dans notre volume ne "touchent" jamais les bords du volume, soit que ces objets se prolongent cycliquement comme l'illustre la figure 3.

Une fois que l'utilisateur a choisi son modèle pour représenter les détails, il reste à construire le motif de référence pour que le modèle reproduise le même effet visuel que la représentation polygonale. nous procédons donc de la façon suivante :

- couper en tranches le modèle 3D,
- évaluer l'éclairement en chaque point,
- remplir l'intérieur de l'objet

Ce dernier point sur la conversion de représentation polygonales est un point clé. La description de l'objet est une surface donc chaque tranche n'est qu'un contour de l'objet (voir figure 4 à gauche). Lorsque que la direction de vue n'est pas orthogonale à la surface on pourra voir l'intérieur de l'objet. Il faut donc que l'intérieur de l'objet soit plein, les pixels à l'intérieur de l'objet doivent avoir une opacité égale à un, en outre la couleur de ces pixels doit correspondre à la couleur du bord de l'objet de manière à avoir une impression visuelle de continuité exacte (voir figure 4 au milieu). Nous utilisons donc un algorithme de remplissage pour chaque couche.

Cette solution n'est pas suffisante lorsque l'angle de vue est très rasant puisqu'alors on peut voir à travers les tranches (voir figure 4 à droite). Ce problème est résolu au paragraphe 4.3.

3.1 Construction d'une tranche avec son éclairage

Dans le cas d'une description standard d'un objet par des polygones (par exemple un fichier OpenInventor), nous utilisons un algorithme standard de rendu pour couper en tranches l'objet et obtenir au passage le calcule de l'éclairage. Pour ceci nous utilisons les deux plans de cliping qui définissent ainsi une tranche de l'objet (comme le montre la figure 5). Chaque image RGBA ainsi calculée nous donne une tranche (la valeur alpha est importante). On déplace ces deux plans de cliping pour obtenir chaque tranche. Il est a noter que l'éclairage est ainsi fixé à la

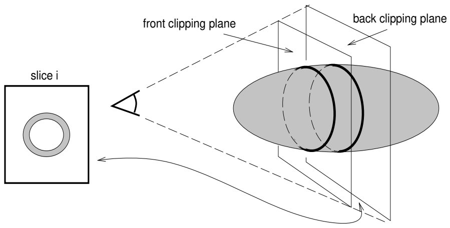

FIG. 5: Construction d'une tranche en utilisant les outils de rendu classique.

construction du volume de référence, ce qui est une limitation par rapport à la méthode utilisée en ray-tracing.

On peut également utiliser un ray-tracer pour rendre chaque tranche et ainsi obtenir un éclairage et des ombres exactes.

3.2 Remplissage de l'intérieur de l'objet

Avec une description polygonale de l'objet, les tranches contiennent juste le contour de l'objet. Nous remplissons donc l'objet en deux passes :

- marquage de l'intérieur de l'objet
- remplissage des pixels marqués avec la "bonne" couleur

3.3 Conversion en texel d'autres représentations.

Surfaces implicites

Pour construire notre volume de référence nous pouvons utiliser les surfaces implicites [BBB+97]. Les surfaces implicites ont l'avantage d'être pleines à l'intérieur de l'objet ce qui nous évite de passer par la phase de remplissage. Pour construire notre volume de référence à partir d'une surface implicite nous évaluons simplement la fonction en chaque voxel de notre volume.

De la même manière nous pouvons convertir les hypertextures [PH89].

Champs de hauteurs

Les champs de hauteurs permettent de spécifier les détails d'une surface. Nous partons d'une image 2D en niveau de gris ([0..255]). Chaque niveau de gris nous donne la hauteur de la colonne se trouvant au pixel correspondant. La construction de notre volume de référence à partir d'une image de hauteurs est ainsi triviale. Nous utilisons une deuxième image contenant l'éclairage précalculé, les normales étant obtenues à partir du gradient de l'image de hauteur et ainsi nous pouvons pré calculer l'éclairage.

Nous avons utilise un bruit de Perlin [Per85] pour produire l'image de hauteur (voir figure 11 au paragraphe 6). Comme ce bruit est analytique nous pouvons obtenir directement les normales sans passer par le gradient. Avec ces normales et l'image d'éclairage on construit ainsi le texel.

Le remplissage est similaire à la procédure décrite au paragraphe 3.2.

4 Problème avec un angle de vue rasant

Quand l'angle de vue est très rasant, par exemple lorsque l'on voit le sommet d'une colline sur lequel des texels sont plaqués, des tranches horizontales ne donnent pas un résultat visuel satisfaisant. En effet on peut ainsi voir à travers le texel, entre les tranches (voir figure 2). Dans

FIG. 6: Critère basé sur la quantité d'information que l'on peut voir de l'intérieur de l'objet.

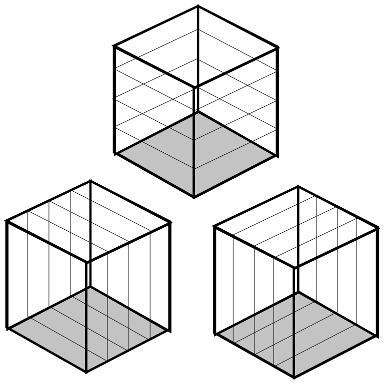

FIG. 7: Trois directions de tranches.

ce paragraphe nous introduisons un critère de qualité qui indique quand l'apparence d'un texel est satisfaisant ou non, et nous ajoutons deux directions de tranches de manière à avoir toujours une apparence correcte quelque soit le point de vue.

4.1 Critère de qualité

Quand le direction de vue est loin de la verticale du texel, il est possible que l'on voit à travers les tranches parce que la projection à l'écran des tranches superposées (voir figure 2 à droite) n'est pas continue. Ceci dépend de l'angle a, de la hauteur h entre deux tranches et de la largeur e de l'objet (voir figure 6). Le premier critère de qualité $h.tan(a)/e <= 1$ nous indique donc quand un défaut apparaît. Nous utilisons ce critère pour choisir le nombre de tranches à afficher pour avoir un aspect visuel correcte pour un angle donnée, ou pour basculer vers une autre direction de tranche (voir paragraphe 4.2).

Nous pouvons utiliser ce critère pour limiter la quantité de pixel, que l'on peut voir à l'intérieur de l'objet. Si l'utilisateur ne veux pas que l'on voit plus de d pixels à l'intérieur de l'objet (voir figure 6) le critère précédent nous donne $h.tan(a)/d <= 1$. Une valeur maximale pour d est $e/2$, en effet la couleur de l'intérieur de l'objet ressemble à la couleur du bord donc si l'on dépasse le "milieu" de l'objet on va voir une couleur du coté opposé qui a souvent le contraste inverse. Ce critère va aussi nous donner le nombre de tranche à avoir (au minimum) pour respecter une qualité donnée.

4.2 Trois directions de tranches

La solution pour ne pas voir à travers le texel lorsque l'angle de vue est très rasant est de stocker deux directions de tranches supplémentaires (voir figure 7).

Au moment du rendu le produit scalaire c_i entre les trois directions et la direction de vue indique laquelle des trois directions de tranches à utiliser. Comme on le suggérait dans le paragraphe précédent, on peut aussi choisir la direction de tranche où le critère de qualité est le plus avantageux. La tangente de l'angle de vue est $\sqrt{1-c_i^2}/c_i$, donc la direction à utiliser est le i pour lequel $(1/c_i^2 - 1) \times h_i^2$ est minimum (où h_i est la longueur L_i du texel dans la direction

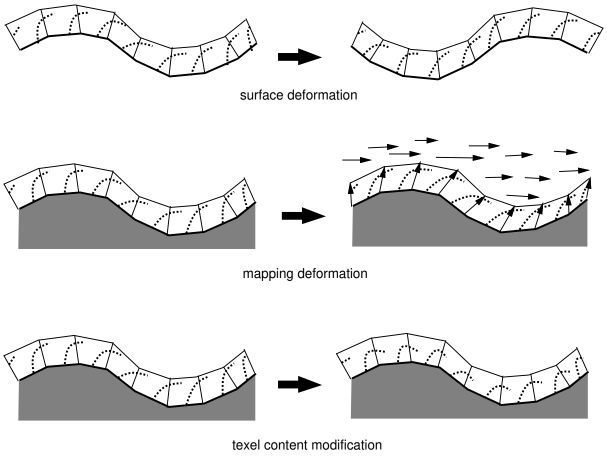

FIG. 8: Trois directions de tranches.

i divisé par le nombre N_i de tranches). Un volume peut alors être visualisé correctement de n'importe quel point de vue.

4.3 Optimisation

Rendre une scène complexe avec ce modèle consiste en fait à rendre des milliers de polygones texturées, ce qui au bout du compte peut être assez coûteux.

On peut utiliser les critères vus au paragraphe précédent pour limiter au maximum le nombre de faces texturées à rendre. Pour une qualité visuelle donnée les critères nous donnent le nombre de polygones à rendre, ce nombre est la plupart du temps inférieur au nombre de tranches réellement existantes. Si l'on supprime simplement certaines tranches on va perdre de l'information, on procède donc à une sorte de MIP-mapping [Wil83] dans la 3e dimension de la texture. On combine donc deux images pour n'en faire qu'une qui contiendra la somme des informations des images de départ. Ceci augmente un peu l'occupation mémoire mais améliore aussi les performances en terme de nombre d'images par seconde.

5 Animations

Trois manières d'animer les textures volumiques sont possibles [Ney95a] (voir figure 8) :

- déformer la surface sur laquelle se trouve les texels.
- déformer l'orientation des vecteurs de hauteur se trouvant sur chaque sommet de la surface (par exemple pour simuler du vent dans l'herbe).
- animer le volume de référence à la manière d'un dessin animé.

Ces méthodes, prévues à l'origine pour le ray-tracing, sont utilisables dans notre modèle interactif. La modification des vecteurs de hauteurs, qui peut se faire par exemple en suivant un modèle physique nécessite de recalculer les coordonnées des tranches texturées à chaque pas de l'animation. Stocker une animation du contenu du volume de référence nécessite plus de mémoire texture mais il est à noter que sur SGI O_2 il n'y pas de différence entre la mémoire vive et la mémoire texture donc ceci n'est pas vraiment une limitation dans notre implémentation, mais cela peux en être une sur d'autres plates-formes.

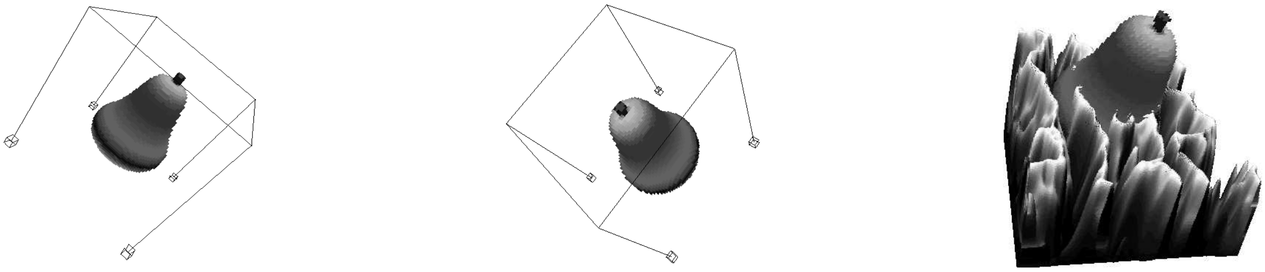

FIG. 9: Gauche, milieu : poire 64^3. Droite : Superposition de deux texels.

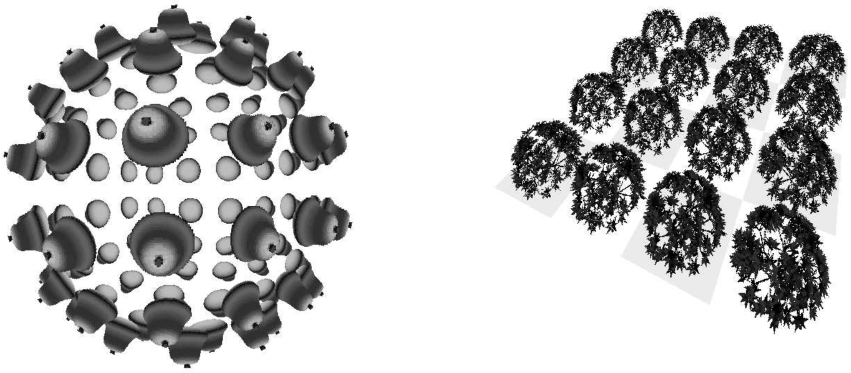

FIG. 10: Gauche, milieu : poire 64^3. Droite : Superposition de deux texels.

6 Résultats

Le premier exemple est un objet Inventor représentant une poire ayant environ 1000 faces. La figure 9 montre un unique texel à la résolution de $64 \times 64 \times 64$ depuis différents points de vue (utilisant différentes directions de tranches). A droite la figure illustre que l'on peut superposer deux texels sans que cela ne pose problème.

La figure 10 présente le plaquage de 96 texels sur une sphère. Si l'écran a la résolution d'une vidéo (768×576) la scène tourne à 3 images/secondes sur une SGI O_2. La scène coûte donc 64×2 polygones texturés alors la même scène rendu avec uniquement des polygones aurait coûtée 96000 polygones. Il est à noter que la poire est un modèle simple, le gain est plus important pour un modèle complexe.

Le second exemple est basé sur un buisson généré par AMAP [dREF$^+$88] qui contient 3500 triangles (le remplissage n'est pas nécessaire ici). Le texel est de taille $64 \times 256 \times 256$. Le rendu de 16 instances de ce buissons sur la figure 10 (droite) tourne à 7 images/seconde.

Le troisième exemple utilise un champ de hauteurs crée par un bruit de Perlin cyclique. Le texel est de taille $64 \times 256 \times 256$ (i.e. avec 64 tranches) (voir figure 12). Un texel de ce type serait représenté par $256 \times 256 \times 2 = 131072$ alors qu'avec notre modèle il est représenté par 2×64 polygones texturées pour une complexité visuelle équivalente (ici le gain en polygones est d'environ 1000). L'image 13 représente le mapping de 96 texels sur une sphère triangulée avec 196 triangles. Le rafraîchissement est d'environ 2 images/seconde toujours sur une O_2. Ce taux de rafraîchissement peut être largement amélioré en utilisant le critère de qualité vu au paragraphe 4.1. De plus la plupart des texels dans cette scène ne sont pas visibles puisqu'ils sont de l'autre coté de la sphère.

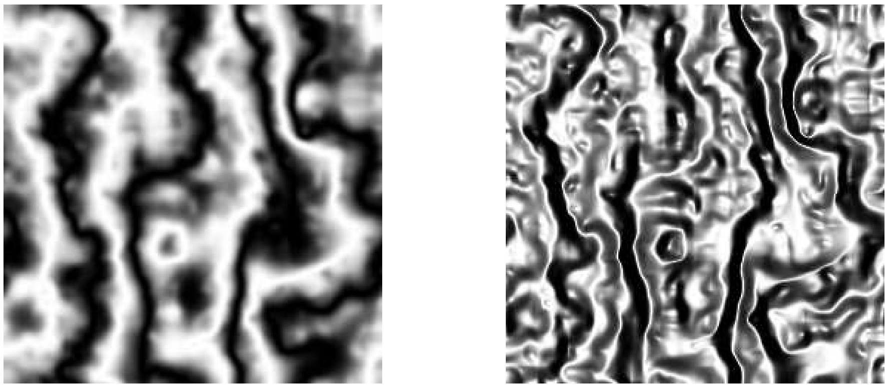

FIG. 11: Texture de Perlin cyclique. Gauche : champ de hauteur. Droite : shading.

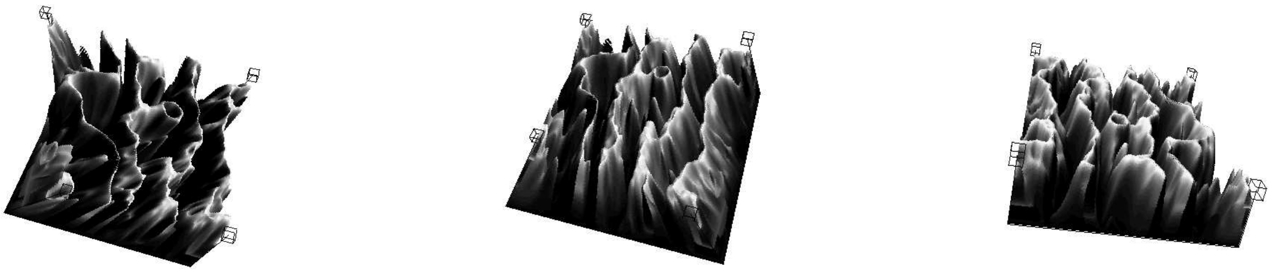

FIG. 12: Des texels déformés.

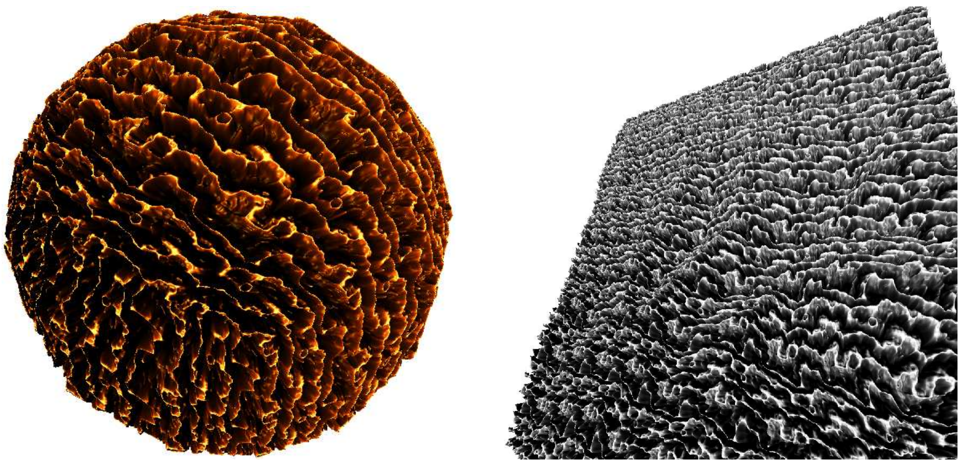

FIG. 13: Gauche : Texels sur une sphère. Droite : Texels sur un plan.

7 Conclusion

Nous avons présenté un nouveau modèle permettant d'augmenter fortement la complexité visuelle d'une scène dans un contexte de rendu interactif, en adaptant au Z-buffer, le modèle de textures volumiques basé sur le ray-tracing [Ney95b] [Ney95a] et ainsi utiliser les fonctionnalités des cartes graphiques. Chaque texel plaqué sur la surface est constitué d'une superposition de faces texturées transparentes. Ces faces texturées sont contrôlées par des vecteurs de hauteurs sur chaque sommet de la surface. Nous avons aussi proposé des méthodes pour construire ces textures 3D à partir de diverses représentations (polygones, surfaces implicites, champs de hauteurs).

Comparé à la version utilisant le ray-tracing la qualité de rendu est bien sûr inférieur (l'éclairage et les ombres sont précalculées). Mais comparé à la pauvreté visuelle des scènes existantes en réalité virtuelle nous avons grandement augmenté la qualité des détails. Parmi les applications possibles de cette technique nous pensons à une implémentation dans un simulateur d'opération chirurgicale : lors d'une simulation la surface de l'organe va se déformer ; enrichir cette surface par une texture 3D permet d'accroître le réalisme.

Parmi les travaux qui sont en cours de réalisation nous pensons enrichir le modèle en permettant la modification de l'éclairage de manière interactive. Pour ceci nous explorons une approche inspirée du bump-mapping.

8 Bibliographie

Références

[BBB+97] J. Bloomenthal, C. Bajaj, J. Blinn, M.P. Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to Implicit Surfaces.* Morgan Kaufmann Publishers, 1997.

[dREF+88] Phillippe de Reffye, Claude Edelin, Jean Françon, Marc Jaeger, and Claude Puech. Plant models faithful to botanical structure and development. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22(4), pages 151–158, August 1988.

[KK89] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 271–280, July 1989.

[LL94] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94*, Computer Graphics Proceedings, pages 451–458, July 1994.

[Ney95a] Fabrice Neyret. Animated texels. In *Eurographics Workshop on Animation and Simulation'95*, pages 97–103, September 1995.

[Ney95b] Fabrice Neyret. A general and multiscale method for volumetric textures. In *Graphics Interface'95 Proceedings*, pages 83–91, May 1995.

[Per85] Ken Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19(3), pages 287–296, July 1985.

[PH89] Ken Perlin and Eric M. Hoffert. Hypertexture. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 253–262, July 1989.

[Wil83] Lance Williams. Pyramidal parametrics. In *Computer Graphics (SIGGRAPH '83 Proceedings)*, volume 17(3), pages 1–11, July 1983.