Visualisation 3D d'un système de particules issu de capteurs de température

B. LANGE[1] N. RODRIGUEZ[1] W. PUECH[1] H. REY[2] X. VASQUES[2]

[1] LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier)

Université Montpellier 2, LIRMM - UMR 5506 - CC 477
161 rue Ada, 34095 Montpellier Cedex 5 – France
{benoit.lange, nancy.rodriguez, william.puech}@lirmm.fr

[2] IBM, Montpellier
Rue de la vieille Poste - BP 1021
34006 Montpellier
{REYHERVE, xavier.vasques}@fr.ibm.com

Résumé

Cet article traite de la visualisation 3D interactive de données issues de capteurs situés dans le centre de calcul IBM Montpellier (data center). Il est important de visualiser ces données dans le but d'analyser et par la suite réduire la consommation d'énergie totale dans des data centers. Un moteur de visualisation a été développé afin de pouvoir comprendre, en temps réel, le comportement des capteurs. Ce moteur se base sur un système de particules qui remplit et représente l'espace d'une salle de calcul. Pour cela, nous utilisons les diagrammes de Voronoï et la triangulation de Delaunay afin de pondérer l'influence de chacune des particules en fonction de leur distance aux capteurs. La visualisation d'une salle en 3D avec un grand nombre de données issues des capteurs est possible grâce à cette solution. La contrainte principale de notre visualisation 3D est l'interactivité du système qui doit garantir un meilleur suivi de l'évolution des mesures prises par les capteurs.

Mots clefs

Capteurs, diagramme de Voronoï, triangulation de Delaunay, visualisation temps-réel, système de particules.

1 Introduction

Dans cet article, nous présentons une nouvelle méthode que nous avons développée pour modéliser et visualiser en 3D des données issues de capteurs de température d'un data center. De manière plus générale, ces capteurs peuvent être de différents types et prendre des mesures telles que la température, la pression, le bruit, l'humidité et même le poids des personnes. La modélisation et la visualisation de cet ensemble de capteurs doit permettre de mieux appréhender le comportement général du data center. Dans ce contexte, nous avons identifié plusieurs points clés pour notre système : fiabilité, rapidité d'exécution, qualité de rendu et fluidité de l'interaction. Dans leurs travaux, Madougou *et al.* [1] parlent de temps réel pour les applications ayant un nombre d'images par seconde de l'ordre de 30 FPS (Frames Per Second). Mais pour d'autres applications, une définition plus stricte du temps réel est souvent nécessaire. Par exemple, les contrôleurs de jeux ou caméras peuvent avoir un rafraichissement bien plus important. La caméra de la PlayStation 3 produit une acquisition de l'ordre de 60 FPS tandis que la caméra de la wiimote a 100 FPS.

Un des problèmes récurrents de la visualisation de données issues de capteurs est l'interpolation de celles-ci afin de prendre en compte tout l'espace et non seulement le champ d'action du capteur. Dans cet article nous présentons deux solutions que nous avons explorées pour résoudre ce problème.

Nous présentons dans la section 2 des travaux en rapport avec notre problématique. En section 3, nous détaillons notre proposition. En section 4, nous présentons les premiers résultats obtenus à partir de données réelles. Enfin en section 5, nous parlons de visualisation interactive et en section 6, nous présentons nos conclusions et perspectives.

2 État de l'art

Dans cette section, nous présentons différents travaux traitant de la visualisation de données et des solutions mathématiques dont nous nous sommes servis dans notre approche.

2.1 Solutions de visualisation

Dans cette partie, nous présentons des solutions qui exploitent des données de grande taille. Les premiers travaux que nous avons étudiés utilisent des données issues de capteurs qui analysent les mouvements du manteau terrestre. Le flux de données est si important qu'il est très complexe

de le visualiser correctement [2, 3]. Ainsi les auteurs souhaitent réaliser une application interactive qui affiche ces données en 3D. Pour la visualisation, ils disposent d'une salle équipée de plusieurs écrans (10 au total). Pour traiter le flux de données, les auteurs ont besoin de deux calculateurs haute performance (High-performance computing : HPC) : un pour les données et un pour le rendu. Le traitement se veut donc en temps réel. L'apport principal de cet article est le traitement des données sur des architectures parallèles. Les auteurs mettent en évidence le fait de traiter des données et de les stocker en mémoire en fonction de leur usage. Par exemple chaque processeur du HPC possède un fichier qui lui est propre, celui-ci lui permet de savoir quelles données il doit traiter. Le résultat n'est toujours pas en temps réel mais des optimisations sont prévues.
Certaines applications utilisent des techniques basées sur un système de particules pour la visualisation [4]. Kapferer *et al.*, expliquent la solution qu'ils ont utilisée afin de visualiser les résultats de simulations astronomiques. Leur contrainte principale est la visualisation en temps réel, mais la qualité du rendu fait aussi partie des points clés. En effet, certaines visualisations ou algorithmes suppriment des caractéristiques importantes qui auraient dû être visualisées. Le problème posé ici peut être résolu avec un calcul par GPU (Graphics Processing Unit). Dans cet article, les auteurs se servent de la puissance de calcul des GPU afin de traiter des données massives au lieu d'utiliser un HPC. Les auteurs font des tests sur différents frameworks qui ne leur donnent pas entière satisfaction d'un point de vue visuel. Afin de dessiner les étoiles, les auteurs se servent des points à la place des sphères habituelles. Ceci permet un affichage plus dense par rapport aux solutions existantes. Tous les calculs sont effectués dans le GPU afin d'approximer les trajectoires de chaque particule. Ensuite, le système de visualisation sur écran large leur donne la possibilité de voir les données de façon plus détaillée que sur un ordinateur classique. Différents algorithmes sont utilisés pour la visualisation comme des algorithmes de simplification par cellule ou par extraction d'iso surface. La solution par GPU semble dans leur cas la plus adaptée pour les simulations d'objets en déplacement.

2.2 Solutions mathématiques

Dans cette section, nous étudions des solutions mathématiques de partitionnement et des solutions d'inclusion de points dans une forme. Dans un premier temps, nous présentons brièvement l'extraction de diagramme de Voronoï et la triangulation de Delaunay [5]. Nous étudierons ensuite l'inclusion de points dans une forme simple.
Les diagrammes de Voronoï et la triangulation de Delaunay font partie des outils retenus pour notre solution. Différentes librairies fournissent ce type d'extraction mathématique dont la librairie QHULL. Barber *et al.* présentent la librairie QHULL, une solution qui sert à produire des enveloppes convexes pour un maillage 3D [6]. Les premiers résultats montrent que le coût de calcul est moins important que lors de l'utilisation d'autres algorithmes. Il est possible d'utiliser les méthodes de la librairie QHULL avec différentes approches mathématiques comme la triangulation de Delaunay ou l'extraction de diagramme de Voronoï. Le coût en temps de calcul est l'un des plus faibles des librairies existantes. Une autre librairie étudiée est décrite par Rycroft [7] extrait des diagrammes de Voronoï sur des objets de grande taille.
Dans cette partie, nous décrivons l'inclusion d'un point dans un maillage. Cette technique permet de connaitre l'emplacement d'un point par rapport à une forme géométrique. L'article de Li *et al.* [8] présente un benchmark entre divers algorithmes de test d'inclusion. Il définit un certain nombre de méthodes tels que grid, octree et bsp-tree. Au final, les résultats présentent l'algorithme de Feito et Torres [9] comme le plus stable et robuste pour la détection de l'inclusion d'un point dans un polygone. Il est inspiré de la triangulation de Delaunay et introduit la notion d'arêtes visibles. Il démontre aussi comment calculer l'inclusion d'un point dans un polygone.
John M. Snyder introduit en 1987 le principe du Ray tracing. Il présente les aspects mathématiques de cette solution dans [10]. Il détermine qu'un modèle peut être vue sous une forme hiérarchique et qu'il est possible de lui appliquer récursivement les rayons de lumières sur chacune de ces facettes. Il introduit diverses équations afin de reproduire les effets de la lumière sur un objet de grande taille. Il présente aussi différentes implémentations à travers une liste ou une grille et présente les résultats. Ces résultats ont largement démontré leur intérêt, par exemple, il montre une forte corrélation entre le rendu du Ray tracing et le nombre de polygones d'un maillage.

3 Proposition

Dans cette section, nous développons les différentes propositions que nous avons faites pour la visualisation de données issues de capteurs de température.
Notre problématique est de visualiser l'évolution thermique dans une salle de calcul. Pour ce faire nous avons utilisé une modélisation de l'espace de notre salle de calcul avec les différents capteurs. Après nous avons utilisé des méthodes mathématiques afin d'interpoler les données issues de ces capteurs.

3.1 Modélisation

Ainsi, afin de modéliser une salle, nous avons utilisé la géométrie du data center. Elle consiste dans un pavé de taille définie par ses trois dimensions : Longueur ($L \in \mathbb{N}$), Largeur ($l \in \mathbb{N}$) et Hauteur ($H \in \mathbb{N}$) illustré en figure 1. Le data center étudié est composé de diverses couches de capteurs ($C \in \mathbb{N}$). Dans notre modèle étudié, les couches sont aux nombre de trois. Enfin sur ces couches, on dépose un nombre M de capteurs. Ceux-ci sont placés dans l'espace grâce au triplet $\{X_i, Y_i, C_i\}$, où $i \in \mathbb{N}$ est le numéro

de capteur, leur localisation dépend de leurs coordonnées réelles dans la pièce. Enfin pour finir la modélisation, on utilise un sytème à particules afin de modéliser l'espace de la pièce. Dans notre approche une particules est un objet déposé régulièrement $N \in \mathbb{N}$ représente le nombre de particules du système. Ces particules sont espacées de $X \in \mathbb{R}$. Leur nombre peut être calculé de la manière suivante :

$$N = \frac{((L+1)*(l+1)*(H+1))}{X^3}. \quad (1)$$

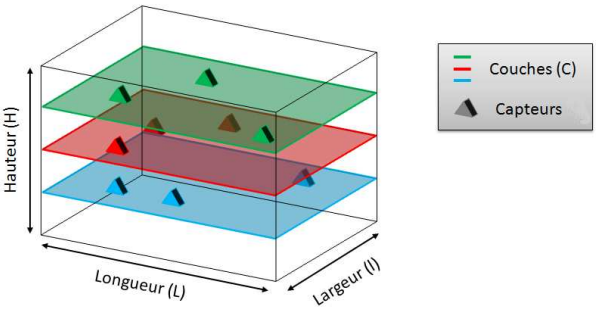


Figure 1 – *Modélisation de la salle de calcul et représentation des différentes couches de capteurs.*

3.2 Partitionement

Nos particules ne sont pas localisées par rapport aux capteurs. Pour cela, on utilise différentes méthodes afin de leur attribuer le coefficient d'influence des capteurs qui les entourent. On utilise donc deux méthodes de partitionnement : la triangulation de Delaunay et l'extraction de cellule de Voronoï.

La triangulation de Delaunay est le dual du partitionnement de Voronoï. Celle-ci va nous permettre de mieux calculer l'influence des particules en localisant entre quels capteurs elles se situent. Cette méthode est utilisée afin de pouvoir localiser les particules dans le maillage des capteurs. Le fonctionnement de Delaunay est simple, la fonction recherche les sommets les plus proches autour d'un point. La structure obtenue est un maillage triangulé. En 2D, la triangulation de Delaunay produit une série de triangles. Dans un monde 3D, il ne s'agit plus de triangles mais de tétraèdres comme le montre la figure 2.
Suite à la triangulation de Delaunay, il est nécessaire de faire une analyse des particules par rapport aux tétraèdres. Cette étape a pour but d'identifier à quel tétraèdre appartient une particule, c'est à dire si elle est située à l'intérieur des différents plans du tétraèdre. Cette méthode est issue de la solution introduite par le lancer de rayons [7]. Nous proposons de tester pour chaque face du tétraèdre si une particule est incluse dans le tétraèdre. Pour cela, la solution calcule les normales de chaque face, puis elle les projette à partir de la particule. Si au moins 3 rayons coupent 3 faces alors le point appartient au tétraèdre. Les particules

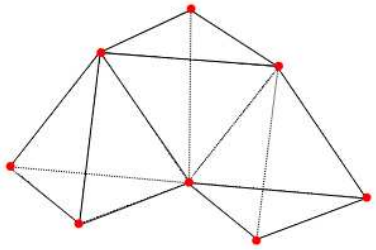


Figure 2 – *Maillage 3D.*

n'appartenant à aucun tétraèdre utiliseront le diagramme de Voronoï pour être prises en compte.
La complexité de cette algorithme est également très élevée, de l'ordre de O (N*M*4), avec N$\in \mathbb{N}$ le nombre de particules et M$\in \mathbb{N}$ le nombre de capteurs.

L'autre méthode utilisée pour les particules externes au maillage de capteurs est l'extraction des cellules de Voronoï. Cette méthode expliqué dans [5] est un système de partitionnement de l'espace. Ce diagramme est surtout utilisé en 2D, mais des solutions existent aussi δD ($\delta \in [0, +\infty[$. Dans le cadre d'un système à particules, cette technique peut être assimilée à une collision entre une sphère et un point. De plus la méthode est discrète contrairement aux méthodes de l'état de l'art. L'algorithme 1 présente cette solution. Afin de pondérer les particules,

Algorithm 1 Extraction de cellule de Voronoï

```
for all particules do
  for all capteurs do
    calculer distance particule-capteur
    if capteur dans sphère d'influence then
      Ajouter capteur à particule
    end if
  end for
end for
```

nous avons utilisé la formule suivante :

$$Mesure\,(p) = \frac{\sum_{i=1}^{M} Mesure\,(i)}{M}, \quad (2)$$

où p représente une particule.
Malheureusement, la complexité de cet algorithme reste très élevé, de l'ordre de O(N*M). La figure 3 illustre les artefactes qui apparaissent lors de l'utilisation de cette solution. Le partionement reste très cubique.

4 Résultats

Dans cette section, nous illustrons la visualisation des données issues d'un centre de calcul IBM. D'abord, il est indispensable de réaliser le modèle géométrique de la salle

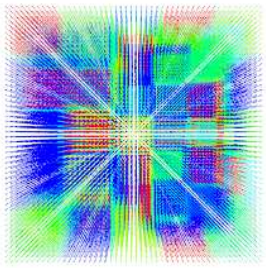


Figure 3 – *Visualisation des cellules de Voronoï en 3D.*

pour la visualiser. La première intuition est de réaliser un maillage. Le problème avec cette solution est qu'elle ne permet pas de modéliser convenablement l'intérieur. Les parties vides ne pourront jamais être modélisées. Deux autres solutions d'affichage de modèle 3D existent : les voxels et les nuages de points. Ces deux manières de modéliser un objet sont très proches. La figure 4 représente le nuage de points que nous avons utilisé pour modéliser l'intérieur d'une salle. Afin de modéliser les différences de températures, une échelle de couleur a été mise en place. Les couleurs chaudes sont représentées en rouge et les couleurs froides en bleu. Un dégradé, passant par le vert et le jaune, est appliqué aux températures médianes. Les pa-

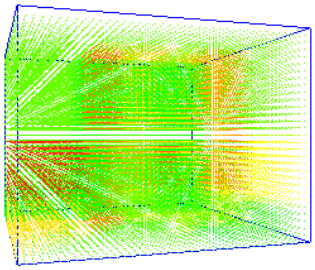
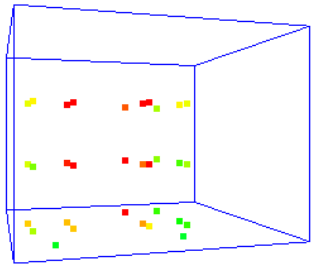


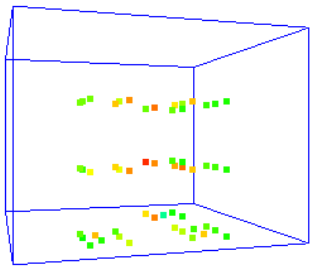
Figure 4 – *Visualisation des particules dans une salle.*

ramètres de notre modèle géométrique sont la longueur, la hauteur et la largeur. Les formes complexes que peut prendre une pièce ne sont pas encore modélisables. De plus l'intérieur de la pièce est considéré comme vide, sans mobilier. Dans notre prototype, nous avons modélisé deux salles de calcul de taille : 3 mètres * 4 mètres * 3 mètres illustrées figure 5.a et 5.b. Les capteurs sont placés dans la pièce sur trois couches. Les premiers sont installés au ras du sol, puis la seconde couche est à un mètre et enfin la dernière à deux mètres du sol. Les particules sont déposés tous les 10 centimètres de manière régulière. Ainsi 36000 particules par salle sont placées. Chaque salle possède son implantation unique des capteurs. Le seul paramètre stable est le fait d'avoir trois couches.

Pour la deuxième solution, les particules sont dans un



(a) Salle 1



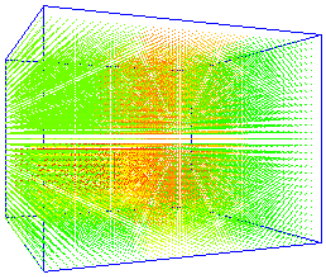
(b) Salle 2

Figure 5 – *Disposition des capteurs.*

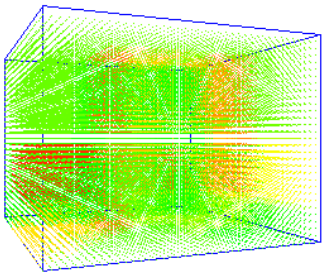
premier temps localisées et pondérées grâce à une triangulation de Delaunay. Pour produire cette triangulation, nous avons utilisé la méthode de QHULL : Qdelauney [6]. L'intersection des particules par rapport aux quatre plans de chaque tétraèdre produit est testée. Si la particule coupe au moins 3 plans, alors celle-ci est considérée comme intérieure au tétraèdre. Pour les particules restantes, nous utilisons le diagramme de Voronoï comme décrit dans la section 3.3. Les figures 6.a, 6.b et 6.c illustrent cette solution.

Afin de segmenter l'espace par Voronoï, nous avons dans un premier temps utilisé les librairies : QHULL [6] et Voro ++ [7]. Les premiers tests n'ont pas été satisfaisants, le système de particules s'adapte mal aux deux solutions présentées. Un coût supplémentaire qui consiste à identifier chaque particule dans les polygones issus de l'extraction est nécessaire.

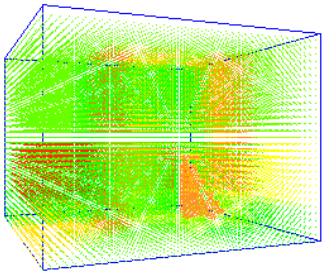
Nous avons donc décidé d'implémenter une nouvelle solution de construction de diagramme de Voronoï qui permet de travailler directement sur les particules. Cette solution est coûteuse en temps CPU. Toutes les particules sont testées et sont comparées aux capteurs. Cette méthode est insuffisante dans le cadre d'une simulation réaliste de température car il n'est pas trivial de pondérer efficacement les particules en fonction des capteurs. Seules les particules centrales à une jointure entre plusieurs zones se trouvent avec une pondération efficace. Les figures 7.a, 7.b et 7.c montrent la pondération des particules avec cette méthode. Le rendu du partitionnement devient alors très intéressant.



(a) temps = 0

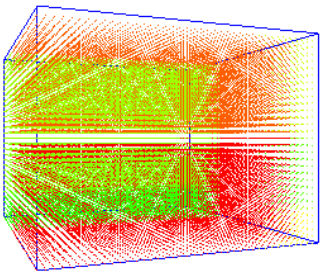


(b) temps = 1

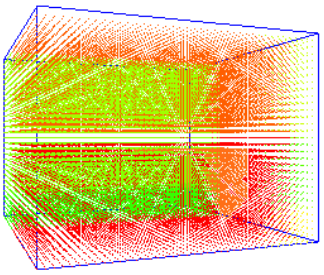


(c) temps = 100

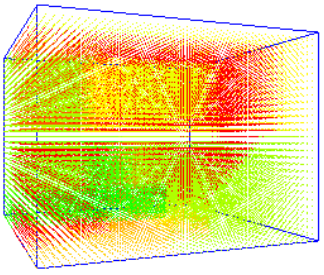
Figure 6 – *Visualisation avec des cellules de Voronoï.*



(a) temps = 0



(b) temps = 1



(c) temps = 100

Figure 7 – *Triangulation de Delaunay et cellule de Voronoï.*

On se retrouve avec des espaces plus fins et des particules pondérables de manière plus simple qu'avec la solution basée que sur les cellules de Voronoï. Les quatre sommets auxquels la particule appartient sont connus. Un simple calcul de norme totale (distance de la particule avec chaque sommet) est nécessaire pour déterminer le coefficient de pondération. Enfin, on applique l'inverse de la distance à chaque coefficient.

La qualité visuelle est améliorée par cette nouvelle approche. Cependant le coût en terme de calcul au lancement de l'application reste important. Cette proposition devra être améliorée. De plus de nombreuses autres possibilités sont à mettre en œuvre comme de véritables modèles physiques venant des études thermiques de bâtiments.

Enfin, au cours de notre implémentation, une mise en évidence de certaines zones de températures s'est avérée importante. Ainsi, un seuil réglable a été mis en place afin de visualiser des intervalles de températures par exemple les températures les plus faibles ou les plus élevées comme illustré figure 8.

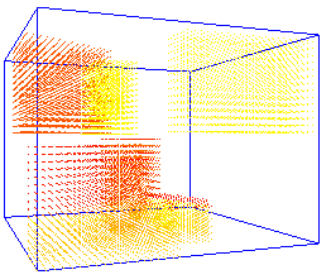


Figure 8 – *Visualisation des températures les plus élevées de la salle de calcul.*

Cette technique apporte une nouvelle approche en matière

de simulation de données issues de capteurs. De nombreuses approches étaient basées sur des systèmes de cartographie de couleur afin de simuler les variations de température. Ici, des modèles mathématiques sont utilisés basés sur la triangulation de Delaunay et les cellules de Voronoï.
Cette solution permet d'avoir un meilleur suivi visuel des incidents qui peuvent intervenir dans une salle de calculs. Les températures critiques peuvent ainsi être appréhendées plus facilement au travers de cette visualisation.

5 Visualisation Interactive

Dans le cadre d'une visualisation interactive des températures d'une salle de calcul, nous avons comme contrainte un travail sur le temps réel. Avec l'approche présentée dans cet article, le rafraichissement de l'affichage a un coût trop important et ne peut se faire rafraîchir qu'à un taux de 11 FPS. Ce coût est dû au trop grand nombre de particules présentes dans la scène. Des optimisations futures devraient permettre de gagner du temps de calcul.
La base de données que nous utilisons ne permet pas non plus de fournir des données en temps réel (de l'ordre de 40 ms). Les capteurs actuels disposés dans les salles de calcul IBM, nous renvoient seulement une information toutes les 5 minutes. En travaillant sur des données simulées, le taux de rafraîchissement minimum est de l'ordre de 300 ms. La complexité pour générer les nouvelles températures est trop élevée. Deux problèmes peuvent se poser : si le flux de données est trop important pour un rafraîchissement en temps réel ou si le flux de données est trop faible afin de donner un rendu intéressant. Le premier problème a été soulevé lors de simulations et l'autre lors de l'utilisation de données issues de capteurs.

6 Conclusions

Le système de particules présente dans cet article permet de modéliser cette approche une salle de calcul et d'interpoler les mesures de capteurs afin d'obtenir une visualisation 3D des températures. Cette approche permet une interpolation des températures de manière fidèle. L'utilisation de partitionnement mathématique fournit un premier aperçu des possibilités du système. L'extraction du diagramme de Voronoï n'est pas suffisamment adaptée à un rendu correct. Seules les particules entre les cellules ont une pondération efficace. Extraire le maillage des capteurs permet de produire une série de tétraèdres, il suffit d'extraire la position de la particule par rapport aux quatre capteurs les plus proches. Ensuite, pour les particules qui ne sont pas retenues grâce à cette solution, un diagramme de Voronoï est utilisé pour les prendre en compte.
Différentes améliorations sont à prévoir pour notre système. L'utilisation d'un véritable moteur physique afin de définir la température des particules est indispensable. Actuellement les particules n'ont pas d'interpolation dans le temps. Et les données sur lesquelles nous travaillons nous donnent des rafraîchissements toutes les cinq minutes. Ensuite, traiter les données de différents types est indispensable. Actuellement, les données sont totalement homogènes alors que dans le futur les données fournies seront de différents types. Une autre amélioration est de rendre plus fluide l'affichage. Du fait du grand nombre de données, nous avons 11 FPS. Ceci est loin des 24 FPS prévues pour obtenir une visualisation interactive et fluide. Enfin, le manque d'interaction ne permet pas de manipuler facilement l'environnement. Le clavier souris limite grandement celle-ci. Des contrôleurs comme une Wiimote sont à prévoir pour faciliter l'usage de notre visualisateur.

7 Remerciements

Nous tenons à remercier toute l'équipe d'IBM Montpellier, et surtout le PSSC (Products and Solutions Support Center) qui nous a fourni les données et le matériel pour les tests présents et futurs.

Références

[1] S. Madougou, V. Gouranton, et E. Melin. Vers une gestion efficace de donnÃ©es gÃ©oscientifiques complexes pour la visualisation sur grappe de pc. *AFRV*, 2006.

[2] M. Damon, M. Kameyama, M. Knox, D. Porter, D. Yuen, et E. Sevre. Interactive visualization of 3d mantle convection. *Visual Geosciences*, 2008.

[3] Kirk E. Jordan, David A. Yuen, David M. Reuteler, Shuxia Zhang, et Robert Haimes. Parallel interactive visualization of 3d mantle convection. *IEEE Comput. Sci. Eng.*, 3(4) :29–37, 1996.

[4] W Kapferer et T Riser. Visualization needs and techniques for astrophysical simulations. *New Journal of Physics*, 10(12) :125008 (15pp), 2008.

[5] A. Montanvert et JM. Chassery. *GÃ©omÃ©trie discrÃ¨te en analyse d'images*. HermÃ¨s, 1991. ISBN 978-2-7462-1643-3.

[6] C. Bradford Barber, David P. Dobkin, et Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4) :469–483, 1996.

[7] C. H. Rycroft. Voro++ : a three-dimensional voronoi cell library in c++. *Chaos 19*, 2009. Lawrence Berkeley National Laboratory.

[8] Weishi Li, Eng Teo Ong, Shuhong Xu, et Terence Hung. A point inclusion test algorithm for simple polygons. 3480 :769–775, 2005.

[9] F. R. Feito et J. C. Torres. Inclusion test for general polyhedra. *Computers & Graphics*, 21(1) :23–30, 1997.

[10] John M. Snyder et Alan H. Barr. Ray tracing complex models containing surface tessellations. *SIGGRAPH Comput. Graph.*, 21(4) :119–128, 1987.