PBTree – Une nouvelle représentation progressive et hiérarchique pour la visualisation en réseau de vastes environnements urbains

J. Royan[1] C. Bouville[1] P. Gioia[1]

[1] France Telecom R&D

4 rue du clos courtel 35512 Cesson Sévigné - France

{jerome.royan, christian.bouville}@rd.francetelecom.fr

Résumé

Ce papier décrit une nouvelle représentation progressive et hiérarchique pour de vastes environnements urbains. Elle permet le survol de ville sur un système de visualisation en réseau. Grâce à cette méthode, la navigation n'est plus seulement limitée à un parcours au niveau du sol, mais autorise le survol de ville. Ceci a pu être réaliser grâce à l'utilisation d'un ensemble d'algorithmes dédiés permettant la décomposition d'une ville en une représentation multi-niveaux de détail. Cette méthode exploite le fait que les technologies de modélisation automatique de ville sont généralement basées sur le traitement de données $2D$ (vidéos, photographies aériennes, plans du cadastre, ...), pour obtenir une représentation $2D\frac{1}{2}$ (emprises au sol des bâtiments, hauteurs, altitudes) beaucoup plus compressée qu'une représentation $3D$ classique. A partir de cette représentation pré-calculée des bâtiments d'une ville, un serveur peut progressivement envoyer au client des détails perceptibles des régions visibles depuis un point de vue donné.

Mots clefs

Navigation 3D urbaine, réseau, représentation hiérarchique, Multi-niveaux de détail.

1 Notre approche

La grande majorité des techniques de reconstruction automatique de villes se base sur des données de Systèmes d'Information Géographique fournissant une représentation $2D\frac{1}{2}$ des bâtiments (leurs emprises au sol, leurs hauteurs et altitudes). Ces données sont issues de plans cadastraux, ou sont calculées à l'aide de méthodes de photogrammétrie exploitant des images aériennes ou des lasers scanner aériens. Les techniques d'acquisition au niveau du sol (lasers scanner ou photogrammétrie) fournissent des modèles de facades détaillés, mais le coût d'acquisition d'une ville entière et le manque de robustesse des techniques de reconstruction sont beaucoup trop important (problème d'occlusions, limitation du recul des appareils d'acquisition, etc). C'est pourquoi nous nous sommes concentrés sur la transmission de modèles dit $2D\frac{1}{2}$ constituant la grande majorité d'une ville, même si quelques modèles de bâtiments spécifiques, comme les monuments, conserveront une représentation $3D$ et seront transmis comme tel.

Sachant que le taux de compression d'un modèle $2D\frac{1}{2}$ par rapport à un modèle $3D$ est supérieur à 1 :10 (sommets $2D$ et non $3D$, un seul polygone : l'emprise au sol, et non toutes les façades, etc), cette représentation sera utilisée pour la transmission des modèles de bâtiment. Seulement, ces modèles $2D\frac{1}{2}$ ne pouvant être rendus directement par le matériel graphique, une phase de reconstruction $3D$ à la volée est nécessaire du coté client (la complexité de la reconstruction dépend des capacité du terminal).

Afin de respecter certaines contraintes du réseau, nous tacherons de concevoir une représentation scalable, compacte et adaptative, basée sur la génération de niveaux de détail progressifs du modèle $2D\frac{1}{2}$ d'une ville.

2 État de l'art

Actuellement, aucune technique ne permet la visualisation en survol quasi instantanée d'une ville de plus de 30 000 bâtiments en réseau.

Les méthodes de simplification polygonale sont essentiellement adaptées aux objets maillés [**?**][**?**][**?**], mais un modèle $2D\frac{1}{2}$ d'une ville s'apparente plus à une *soupe de polygones* (aucune information topologique entre les bâtiments) . Il existe des techniques de simplification pour un bâtiment donné, respectant certaines caractéristiques spécifiques comme la verticalité des murs. Seulement, ces techniques de simplification ne sont pas assez drastiques, puisqu'elles ne simplifient pas un ensemble de bâtiments (par exemple en fusionnant deux bâtiments proches).

Les techniques basées points ont peu d'intérêt pour des objets à la géométrie simple et très texturés [**?**](nécessité d'avoir la même résolution pour les textures que pour les points qui contiennent beaucoup plus d'informations non nécessaires). Ces techniques basées point seraient intéressantes pour la transmission d'objets récurrents et complexes d'une ville (végétation, mobilier urbain) afin d'optimiser les temps de rendu.

Les techniques basées images sont assez spécifiques aux

systèmes de visualisation locaux [**?**]. En effet, la durée de validité d'un imposteur est assez faible durant une navigation, même si certaines techniques d'imposteur multi-couches prolonge cette durée de validité[**?**][**?**]. Dans le cas d'un système de visualisation en réseau, il est alors nécessaire de transmettre régulièrement de nouvelles images qui surchargent grandement la bande passante du réseau.

Quant aux techniques de visibilité conservative [**?**][**?**], elles ne sont applicables que pour une navigation au niveau du sol (occlusions restreintes en survol).

3 Le PBTree

3.1 Présentation du PBTree

Le PBTree est une représentation progressive de modèles urbains $2D\frac{1}{2}$, permettant la visualisation de vastes villes en réseau[**?**]. le PBTree consiste en une arborescence, ou chaque noeud représente un bâtiment ou un ensemble de bâtiments en $2D\frac{1}{2}$ (voir fig **??**). Ce PBtree permet la transmission progressive de la géométrie d'une ville à travers le réseau. Ainsi, lorsque le client s'approche d'un bâtiment, il envoit une requête au serveur qui lui renvoit l'information permettant de developper un noeud du PBTree, et ainsi d'obtenir une représentation plus fine du bâtiment.

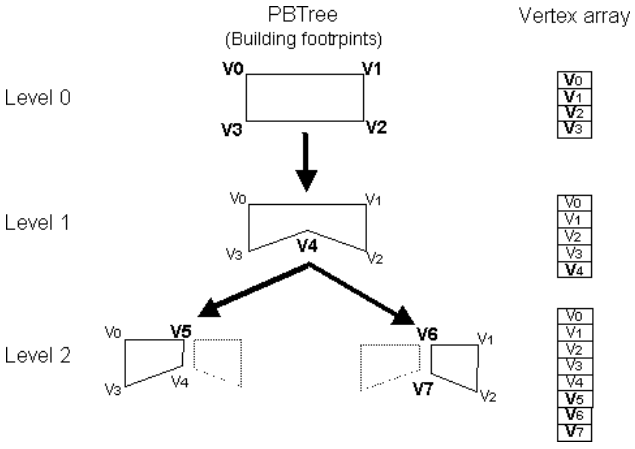

Figure 1 – *Le Progressive Building Tree, représentation $2D\frac{1}{2}$ hiérarchique d'une ville.*

3.2 Construction du PBTree

Reconstruction de la topologie de la ville. Étant donné que les bases de données de villes sont constituées d'une soupe d'emprises au sol de bâtiments, on effectue une triangulation contrainte de l'espace vide entre les bâtiments sans ajout de points supplémentaires (pas de points de Steiner permettant d'obtenir une triangulation contrainte de Delaunay, voir fig **??**). Nous obtenons ainsi un maillage $2D\frac{1}{2}$ de la scène nous permettant de connaître la topologie exacte de celle-ci.

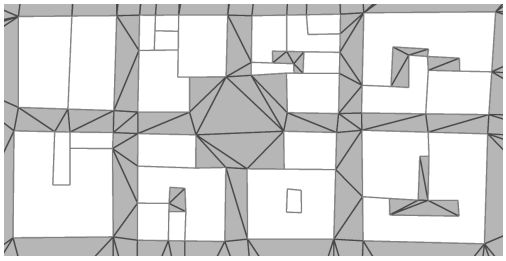

Figure 2 – *Triangulation contrainte par les emprises au sol des bâtiments permettant de connaître la topologie exacte de la scène.*

Simplifications applicables. Afin de réduire la complexité géométrique d'une ville, l'algorithme peut appliquer trois simplifications différentes :

- **Fusion de deux facades consécutives d'un bâtiment :** En supprimant le sommet de l'emprise au sol qui modifie le moins la géométrie du bâtiment (ou l'air de l'emprise au sol).
- **Fusion de deux bâtiments connexes :** En fusionnant les emprises au sol des deux bâtiments. L'altitude du bâtiment résultant est l'altitude minimale des deux bâtiments fusionnés, et son hauteur est une moyenne des deux hauteurs des bâtiments fusionnés pondérée par la surface de leurs emprises au sol.
- **Fusion de deux bâtiments non connexes :** Cette simplification moins implicite que les précédentes, utilise la triangulation contrainte par les emprises au sol des bâtiments pour déterminer l'emprise au sol du bâtiment résultant de cette fusion (voir fig **??**-c). La hauteur et altitude du bâtiment issu de cette fusion suit les mêmes règles que dans le cas de bâtiments connexes.

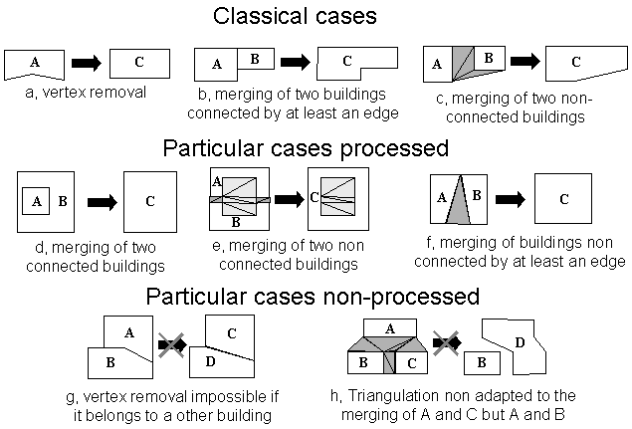

Figure 3 – *Exemples de simplifications.*

L'algorithme de simplification. L'algorithme de simplification prend en entrée le maillage $2D\frac{1}{2}$ issu de la triangulation de la scène contrainte par les emprises au sol des bâtiments. L'initialisation de l'algorithme consiste à attribuer à chaque feuille du PBTree les bâtiments initiaux,

et à rechercher toutes les simplifications potentielles pouvant être appliquées au maillage $2D\frac{1}{2}$. Seulement, pour déterminer quelle simplification doit être appliquée en priorité, une fonction de coût est attribuée à chaque simplification potentielle, permettant de les stocker dans une liste triée (Potential Simplification List). Cette fonction prend les paramètres suivants :

– La différence de hauteur entre les bâtiments pouvant être fusionnés ($diff\_height$),
– La différence d'altitude entre les bâtiments pouvant être fusionnés ($diff\_alt$)
– Le volume additionnel engendré par la simplification ($area$). Plus ce volume sera important, plus la transition lors d'une fusion de bâtiments durant la navigation sera visible.
– La distance minimale entre deux bâtiments pouvant être fusionnés ($dist\_min$), permettant une conservation des rues lors de la simplification d'une ville.

Cette fonction de coût est minimal lorsque $diff\_height$, $diff\_alt$, $area$, et $dist\_min$ sont minimaux. Après le test de plusieurs fonctions de coût, la suivante s'est révélée particulièrement bien adaptée au cas des bâtiments :

$$cost(diff\_height, diff\_altitude, dist\_min, area) = area.\exp(\alpha.diff\_height + \beta.diff\_alt + \gamma.dist\_min) \quad (1)$$

où α, β, γ sont des facteurs de poids.

La figure **??** décrit le déroulement de l'algorithme de simplification permettant de générer le PBTree, utilisé pour la transmission progressive de la géométrie d'une ville à travers le réseau. A noter qu'une mise à jour locale de la triangulation de l'espace vide est nécessaire après chaque application d'une simplification. La figure **??** montre l'état des données durant le processus de simplification. L'algorithme finit lorsque la liste des simplifications potentielles est vide. Le PBTree contient alors toute la représentation progressive et hiérarchique de la ville.

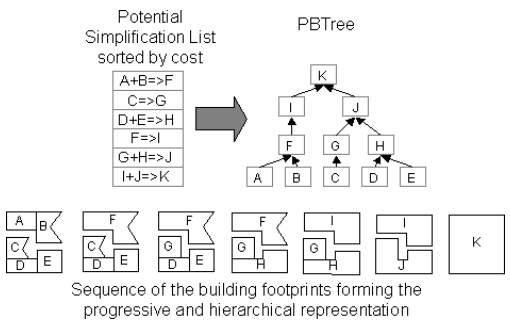

Figure 4 – *Algorithme de génération du PBTree.*

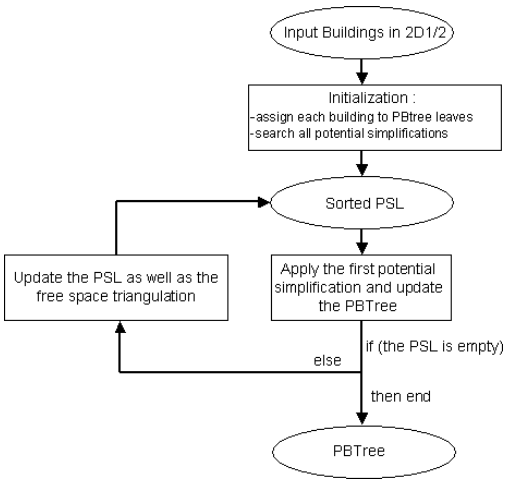

Figure 5 – *Evolution des données durant le processus de construction du PBTree.*

4 Résultats

La figure **??** montre une partie d'une ville à différentes résolutions (A noter la très bonne conservation des rues). La table **??** présente les temps de calcul pour 3 villes différentes. A noter que le temps de calcul important du PBTree n'est pas pénalisant, puisqu'il est effectué une seule fois hors ligne.

Figure 6 – *Exemples de simplifications.*

	# bâtiments	# façades	temps de calcul	fps
Marseille	1360	8362	28s	32
Nice	15603	97742	7mn 52s	27
Rennes	35203	280267	2h18mn27s	23

Tableau 1 – *Nombre de bâtiments, de façades, le temps de pré-calcul du PBTree, et la fréquence d'affichage pour trois villes différentes.*

La table **??** montre l'intérêt de cette représentation en terme de compression, essentielle pour une transmission rapide de cette représentation à travers le réseau.

5 Conclusion

Nous avons présenter une nouvelle représentation progressive et hiérarchique de ville, permettant une navigation fluide au niveau du sol et en survol sur un système client-serveur. Le principe consiste tout d'abord à transmettre une représentation $2D\frac{1}{2}$ des villes, et de générer

	No compression				
	$3D : VRML$		$2D\frac{1}{2}$		
	Normal	Hierar.	Shape	Hierar.	Taux
Marseille	1,86	7,28	0,635	0,283	0,039
Nice	21,7	110	5,64	4,08	0,037
Rennes	57,8	351	12,8	12,8	0,036

	Gzip compression				
	$3D : VRML$		$2D\frac{1}{2}$		
	Normal	Hierar.	Shape	Hierar.	Taux
Marseille	0,205	0,626	0,143	0,089	0,142
Nice	2,34	8,78	0,89	1,28	0,146
Rennes	6,28	26,34	2,17	3,02	0,115

Tableau 2 – *Taille en Mo pour une ville donnée utilisant différentes représentation : VRML, VRML avec des Niveaux de détail statiques, Shape, et notre représentation progressive et hiérarchique. La dernière colonne présente les taux de compression entre les deux représentation hiérarchiques.*

hors ligne une représentation progressive et hiérarchique à l'aide d'une structure arborescente appelée PBTree. Les résultats expérimentaux démontrent son importante efficacité, et les avantages de cette approche :

– Le survol des villes est possible, avec une importante profondeur de champ de vision,
– La transmission d'une ville est progressive et hiérarchique, permettant une complète scalabilité, et une reconstruction dépendante du point de vue (voir Figure **??**). De plus, la navigation peut commencer juste après la connection au serveur (contrairement aux navigateurs $3D$ actuels du type "download and play").
– Les données transmises sont fortement compressées, en utilisant un système de référence local pour les coordonnées des sommets.
– La reconstruction $3D$ d'un bâtiment n'est pas limitée à un prisme, mais peut être agrémentée de détails issus d'une reconstruction procédurale (pour les toit et les façades).

Références

[1] H. Hoppe. Efficient implementation of progressive meshes. Dans *Computer and Graphics*, volume 22, pages 27–36, 1998.

[2] D. Luebke et C. Erikson. View-dependent simplification of arbitrary polygonal environnements. Dans *Computer Graphics SIGGRAPH 97*, volume 31, pages 199–208, New-York, 1997.

[3] M. Garland et P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. Dans *IEEE Visualization 98 Conference Proceedings*, 1998.

[4] H. Pfister, M. Zwicker, J. Van Baar, et M. Gross. Surfels : Surface elements as rendering prinitives. Dans *Computer Graphics, SIGGRAPH 2000 Proceeding*, pages 343–352, Los Angeles, 2000. ACM Press.

[5] Paulo W. C. Maciel et Peter Shirley. Visual navigation of large environments using textured clusters. Dans *Symposium on Interactive 3D Graphics*, pages 95–102. ACM Press, 1995.

[6] X. Decoret, G. Schaufler, F. Sillon, et J. Dorsey. Multi-layered impostors for accelerated rendering. Dans *Eurographics'99*, volume 18, 1999.

[7] M. Wimmer, P. Wonka, et François Sillion. Point-based impostors for real-time visualization. Dans *EuroGraphics Workshop on Rendering*, 2001.

[8] Seth J. Teller et Carlo H. Sequin. Visibility preprocessing for interactive walkthroughs. Dans *Computer Graphcis (proceedings of SIGGRAPH 91)*, volume 25(4), pages 61–69. ACM Press, 1991.

[9] P. Wonka, M. Wimmer, Michael, et F.X. Sillion. Instant visibility. Dans *Eurographics 2001*, volume 20, 2001.

[10] J. Royan, C. Bouville, et P. Gioia. Pbtree - a new progressive and hierarchical representation for network-based navigation in urban environments. Dans *VMV 2003*, 2003.

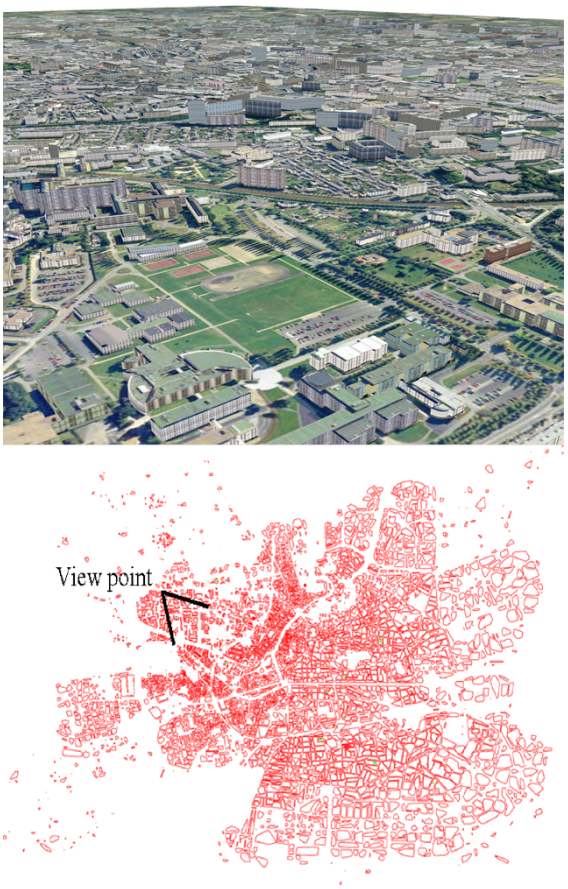

Figure 7 – *Exemple de visualisation en survol de la ville de Rennes.*