Hierarchical representation of virtual cities for progressive transmission over networks

J. Royan
France Telecom R&D
Rennes, France

R. Balter
France Telecom R&D
Rennes, France

C. Bouville
France Telecom R&D
Rennes, France

Abstract

Interactive network-based navigation over large urban environments raises difficult problems due to the size and complexity of these scenes. In this paper, we present a client-server system allowing navigation over 3D cities in real time. Due to a novel progressive and hierarchical representation of 3D models of densely built urban areas, only perceptible details for all the regions visible from a given viewpoint are progressively streamed to visualisation clients. Furthermore, efficient coding methods are used to compress the representation data allowing quick start-up of the interactive visualisation with a highly-detailed model. This is achieved through a set of dedicated algorithms allowing a very large city model to be structured into a multi-resolution representation. The method efficiently exploits the fact that most automated modelling techniques of urban scenes provides $2D\frac{1}{2}$ models (building footprint, height, altitude, ...). So as to efficiently and faithfully model complex buildings, a procedural representation for roofs and facades is proposed. Finally, we present an MPEG4 compatible implementation based on the introduction of new node types with the associated bitstream.

1. Introduction

Many solutions are now available to automatically generate 3D models of huge urban environments. Most of them are based on Geographic Information System. This GIS are built from classical cadatral maps (terrestrial measurement), or by using laser scanners or photogrammetric methods to provide dense point clouds. A segmentation applied on these points clouds or directly on the aerial photographs used for photogrammetry, provides a $2D\frac{1}{2}$ representation (footprints, heights and altitudes of buildings). All these solutions create huge representations that are inapropriate for a network-based visualisation. Other solutions based on procedural methods generate compact city models which are unfortunately far from reality.
To cope with network bandwidth limitations, several techniques need to be used:

- Compression: city models are very bulky (often more than 10Gb. It is therefore absolutely necessary to do geometry as well as texture compression).
- Progressivity: it is essential to allow the user to navigate straight after being connected to the server whatever the coarseness of the model at start-up. As data are received from the server, the city model is progressively refined by adding the appropriate details. To preserve coding efficiency, these model refinements must be incrementally coded so as to avoid redundancies.
- Adaptability: the refinement of the scene must be carried out in depth order i.e. from near to far regions and must be fully reversible so as to avoid error accumulation when the viewpoint changes. Furthermore, this refinement data stream should be adapted to the available network bandwidth.

In the following, we will consider only $2D\frac{1}{2}$ city models because of their widespread use and their low production cost. Besides, these representations are inherently procedural and thus more compact than a pure 3D polygon representation. However, such $2D\frac{1}{2}$ models do not lend themselves very well to progressive transmission and view-dependent simplification. The goal of this paper is to propose a solution to these problems by providing a new modeling and representation technique that fulfills the three aforementioned requirements. It includes the following contributions:

- a hierarchical representation technique (called PbTree) that allows incremental updates of the remote client visualisation model
- a set of dedicated algorithms to build this hierarchical representation from a usual $2D\frac{1}{2}$ city model
- a coding scheme to compress the PBTree updates for adaptative and progressive streaming,
- a procedural coding to model complexe roofs and facades,

- the description of a MPEG-4 based client-server implementation.

The remainder of the paper is organized as follows. After reviewing some background works on the visualization of huge environments, Section 2 presents a reminder on PBTree principles. Section 3 describes the procedural modeling and coding of facades and roofs proposed in MPEG-4 standardization. In Section 4, we describe the visualization client-server system with efficient functionnalities well-suited for our representation. Then, we comment our results in Section 5, to finally conclude.

1.1. Previous Work

In this section we present the background works on large scenes visualization. We focus on the case of urban scenes, analysing how they fulfil the requirements mentioned above.

VRML or X3D browsers are the most common solution to visualize 3D model over networks. But its basic "'download-and-play"' paradigm is inappropriate for huge urban models, even though some forms of on-demand loading are possible. The problem of 3D model simplification has been extensively studied for the particular case of manifolds[1], but the case of large urban environments modelled by sets of disjoint building models are of a completely different nature. The irregular nature of such geometry is closer to so-called *polygon soups* than regular topological meshes. Actually, these scenes are composed of millions of buildings but each of them has very few polygons and little geometric redundancy, which is not suitable to mesh decimation. In the same way, many view-dependent simplification techniques have been proposed for the specific case of walkthrough navigation. Some of them are based on visibility culling (visibility computation)[2][3] however they are inefficient for flyover navigation because of the very low occlusion compexity. Others replace far geometry by computed images (impostors)[4] so as to remove imperceptible geometric details while preserving good visual quality. However, the improvement in geometric complexity is obtained at the expense of transmission efficiency because at equal visual error, impostors are more demanding in bandwidth resource than geometric models [5]. Another technique consists of processing only points without explicit connectivity instead of triangles [6]. However, point-based rendering is well-suited to complex objects that are viewed from a great distance compared to the object size. This is not the case for most building representations where details are represented by textures mapped on a raw geometric model.

Finally, a new multiresolution representation for very large building sets is proposed in [7]. This representation will be briefly described in the following.

2 PBTree Principles

The PBTree is a multi-resolution representation of a city based on a $2D\frac{1}{2}$ model of buildings consisting in a set of footprints associated with the height and altitude of each building. This footprint elevation model can be enhanced with several parameters so as to model a detailed roof and facade. To obtain a multi-resolution representation, a solution similar to Hoppe's *vertex tree* is used. However, several adaptations have to be applied to take into account the specific case of a $2D\frac{1}{2}$ model of buildings. First, each node has to contain a building at a given level of detail. Then, suitable simplification operators have to be defined to reduce the complexity of the scene. Finally, in order to assign priorities to the simplifications that will be applied, a specific cost function has been defined. This cost function assess the visual error due to the simplifications compared to the original model.

2.1 The PBTree

2.1.1 Context

Let S be the set of vertices in $\Re^2$ of the scene represented in $2D\frac{1}{2}$, and let $P \in S^n$ be the set of polygons representable from the set S. We express a building as a tuple $b = (bfp, h, a)$ where bfp specifies its building footprint in P, h is its height, and a its altitude. We define by $\widehat{B}$ in b^k the set of the k initial buildings of the scene. We call B^0 the coarsest set of buildings representing the scene. Thus, the initial set of buildings $\widehat{B} = B^n$ can be simplified into a coarser set of buildings by applying a sequence of n successive simplification transformations:

$$(\widehat{B} = B^k) \rightarrow simpl_{n-1} \ldots \rightarrow simpl_1 B^1 \rightarrow simpl_0 B^0 \quad (1)$$

Because each simplification transformation $simpl$ is invertible, we can represent the set of buildings of the scene as a simple building B^0 together with a sequence of n $dvlp$ records:

$$B^0 \rightarrow dvlp_0 B^1 \rightarrow dvlp_1 \ldots \rightarrow dvlp_{n-1}(B^n = \widehat{B}) \quad (2)$$

The t-uple $(B^0, \{dvlp_0, \ldots, dvlp_{n-1}\})$ forms a progressive and hierarchical representation of $\widehat{B}$. Each development transformation $dvlp$ introduces the addition of vertices as well as the modification of certain building footprints (cf Figure 1). The t-uple $(\{b_i, \ldots, b_{i+m}\}, b_j)$ defines a simplification transformation $simpl_{n-1}$ where $\{b_i, \ldots, b_{i+m}\}$ represent the buildings in B^n, and b_j the corresponding simplified building in B^{n-1}. Respectively, the t-uple $(b_j, \{b_i, \ldots, b_{i+m}\})$ represents an expansion transformation $dvlp_n$.

The resulting sequence of building set $BFP^0, \ldots, BFP^n$

is particularly adapted to a selection of levels of detail depending of the view-point. In addition, geomorphs can be used in order to visually smooth transitions between the different levels of detail.

2.1.2 Node definition

The progressive and hierarchical representation is based on a tree structure holding all mergings and simplifications of buildings: The *Progressive Building Tree* (cf Figure 1). Each node of this tree, called *Building Node*, contains the $2D\frac{1}{2}$ representation of a building at a given level of detail. A vertices array defining the building footprints associated to building nodes is attached to this tree. Each building node has to contain a $2D\frac{1}{2}$ representation allowing the $3D$ reconstruction of the associated building. It is defined by the following elements:

- a tree node structure,
- a building footprint (a polygon indexed to the vertex array),
- a height,
- an altitude,
- every parameter allowing a more complex procedural $3D$ modelisation of the building.

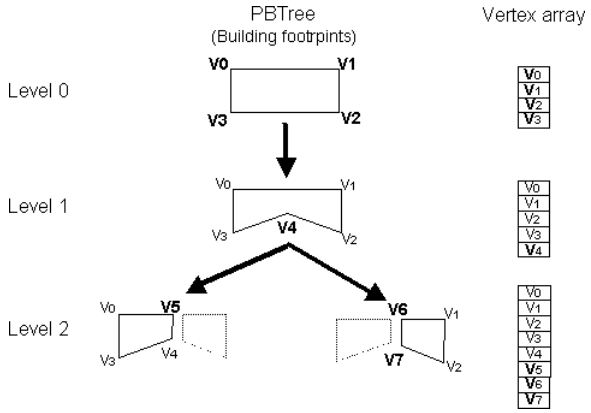

Figure 1: $2D\frac{1}{2}$ progressive and hierarchical representation using a building tree. In bold, the new vertices required for expanding a Building Node.

2.2 Simplifications

To reduce the complexity of the urban scene, simplification must be applied not only to each building, but over the entire $2D\frac{1}{2}$ model of the city. Figure 2.2 shows the three simplifications classes that can be applied on the city model: a building simplification by removing a vertex of its footprint (S_1), a merge of two buildings connected by at least one edge (S_2), and the merge of two buildings not connected by one edge (S_3). For this last simplification, less implicit than the other ones, we have introduced a new connexity relation called T-connexity that will be described next.

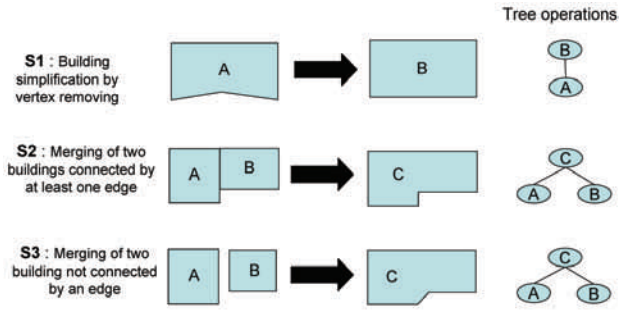

Figure 2: Simplifications that can be apply on a $2D\frac{1}{2}$ city model.

2.2.1 Building simplification

This simplification consists in merging two consecutive facades by removing a point of the footprint. The vertex to be removed has to respect the following three rules:

- Rule 1: every removed vertex must be placed in a concavity. Actually, by recursively removing vertices corresponding to convex angles, we would end up with a point.
- Rule 2: a removed vertex must not be shared by another building, otherwise the simplification would modify two buildings, creating two new fathers nodes for a simplification (strictly forbidden for a progressive tree).
- Rule 3: the chosen vertex must modify the footprint as little as possible. For a given $bfp \in P$, the vertex p_n is to be removed if $\forall p_i \in bfp$ with $i \neq n$, $area(p_{n-1}, p_n, p_{n+1}) < area(p_{i-1}, p_i, p_{i+1})$ (with respect to rules 1 and 2).

2.2.2 Merging two buidlings connected by at least one edge

This simplification consists of merging the two footprints corresponding to the merged buildings. If bfp_1 and bfp_2 are merged to create the building bfp_r, we compute the altitude a_r and the height h_r of the resulting building as follows

$$\begin{array}{l} a_r = MIN(a_1, a_2) \\ h_r = \frac{((h_1+a_1)*area_1)+((h2*+a_2)*area_2)}{area_1+area_2} - a_r \end{array} . \quad (3)$$

2.2.3 Merging two buildings T-connected

The merging of two non-connected buildings is based on the $2D$ free space triangulation of the scene (or the triangulation of the scene constrained by the footprints as shown in Figure 3). In order to merge two non-connected buildings, we merge them with the triangles they share. To specify

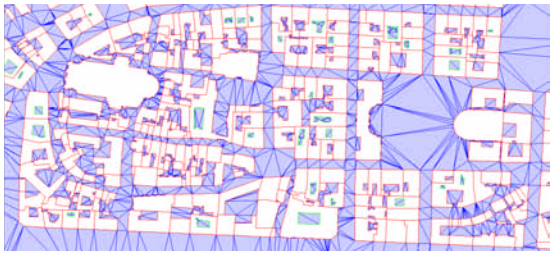

Figure 3: $2D$ free space triangulation of a urban scene.

which shared triangles must be merged, we introduce a new connectivity relation called T-connexity, and defined as follows. Two footprints E_1 and E_2 are T-connected if:

- First case: there exists at least a triangle pair (T_1, T_2) so that

$$\begin{array}{l} 1,\ T_1.A_e.T_2 \\ 2,\ (E_1.A_e.T_1) \wedge (T_1.A_v.E_2) \\ 3,\ (E_2.A_e.T_2) \wedge (T_2.A_v.E_1) \end{array} . \quad (4)$$

- Second case: there exists a triangle T_1 so that

$$(T_1.A_e.E_1) \wedge (T_1.A_e.E_2)(\Rightarrow E_1.A_v.E_2), \quad (5)$$

where T_i are called the T-connexity triangles, A_e (resp. A_v) corresponding to the relation of adjacency by a edge (resp. a vertex). The resulting footprint is the merging of the two original footprints with the corresponding T-connexity triangles (see Figure 4). Its height and altitude are computed using Equation 3.

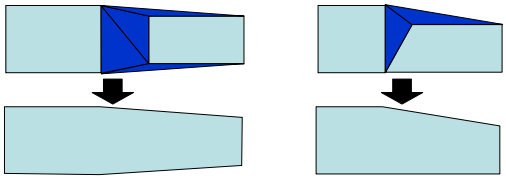

Figure 4: Merging two buildings T-connected.

2.3 The metric Error

To reduce the urban scene complexity, we compute the PBTree by recursively applying the previous simplifications. But to assign priorities to these simplifications, a metric error is associated to each of them. This metric error must take into account various criteria:

- The height difference between the potentially merged buildings ($diff\_height$), so as to merge in last resort a tower next to a little residence.
- The altitude difference between the potentially merged buildings ($diff\_altitude$), so as to merge in last resort a building on the bottom of a cliff with one on the top.
- The difference in volume between the resulting building and the original ones ($diff\_volume$): consider building B resulting of the merge of B_1 and B_2; we have

$$\begin{array}{rl} diff\_volume = & \sum_{i=0}^{i<n} |h - h_i| * Area_i \\ & + \sum_{j=0}^{j<m} |h - h_j| * Area_j \\ & + (\sum_{k=0}^{k<p} TriArea_k) * h \end{array} , \quad (6)$$

  where h is the height of B, h_i and $Area_i$ (resp. h_j and $Area_j$) is the height and the footprint area of the original building from which B_1 stems (resp. B_2), and $TriArea_k$ is the set of T-connexity triangles merged during the many and successive simplifications to obtain B.
- The minimal distance between the potentially merged buildings (min_dist), allowing for the streets to be kept as long as possible during the simplification process.

Finally, the Metric Error is computed as follows

$$\begin{array}{rcl} MetricError & = & diff\_volume \\ & * & \alpha * e^{|diff\_altitude|} \\ & * & \beta * e^{|diff\_height|} \\ & * & \delta * e^{min\_dist}, \end{array} \quad (7)$$

where α, β, and δ are scalar allowing to weight the different criteria.

2.4 PBTree computation algorithm

This simplification algorithm takes as input the set of initial buildings represented in $2D\frac{1}{2}$ and computes the corresponding building tree. The algorithm is initialized by assigning the input buildings to each leaf of the building tree, and by searching for all potential simplifications of the scene. These simplifications are organized in a priority list based on the metric error. The simplification with the smallest metric error is removed from this list and applied to the urban scene. The PBTree is updated as shown in Figure 5. The algorithm ends when the list of potential simplifications is empty.

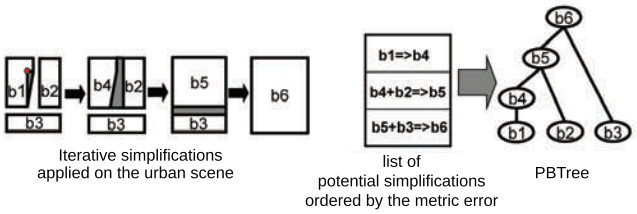

Figure 5: Principle of the PBTree construction algorithm.

3 Building procedural modeling

The $2D\frac{1}{2}$ representation is reduced to a prismatic building representation with flat textures for roofs and facades. This representation is well-suited for flyover navigation but is inapropriate for street level view. In order to faithfully model buildings, solutions based on procedural modeling are presented next. The goal of this procedural modeling is to accurately describe a great majority of buildings with very few parameters. The number of parameters used to model a building is to be directly tied to the genericity and the precision with which the building is reconstructed. A few parameters are not enough to describe characteristic buildings with complex facades and roofs (churches, monuments, ...), and too many parameters can increase the stream size by adding redundant data. We propose a procedural model as compact and generic, knowing that in the particular case where our representation cannot fulfil specific building modeling, it can be swapped for a compressed mesh.

3.1 Roofs

Roof structures can be very complex. However, in most cases, due to arhictecural constraints, the roof framework can be computed by determining the straight skeleton of the building footprint. There exists many different roof types however, a complex roof can be modelled by superimposing simpler roofs, or by breaking up a building in different prisms with specific roofs for each of them. Four simple roof types have been chosen to describe a roof : a flat roof, a hip roof, a gable roof, a salt box roof.
The 3D reconstruction of the roof is computed on the fly on the client side for its visualization. To accurately describe a roof, we specify the following parameters :

- the slope of the roof, only one if it is the same for each roof panel, or several if panels have different slopes.
- an eave projection,
- a height, to be able to crop the roof
- the support wall, in the case of a salt box corresponding to a roof with a single panel.

In the case of a hip roof that has the same slope for each roof panel, the 3D reconstruction of the roof is done by computing the straight skeleton of the building footprint. The computation of a gable roof with the same slopes for each panel is based on the straight skeleton too, except that a projection of the extreme vertices on the corresponding footprint edges is computed. We call extreme vertices, those that are directly connected by two successive footprint vertices. Finally, if the roof has different slopes, we reconstruct it by computing the intersection of the different planes defining the roof panels.

Moreover, a roof can be cropped at a defined height, by computing the intersection of the roof 3D model with a horizontal plane. The polygons resulting in this intersection are used as footprints to reconstruct the superimposed roofs (see figure 6).

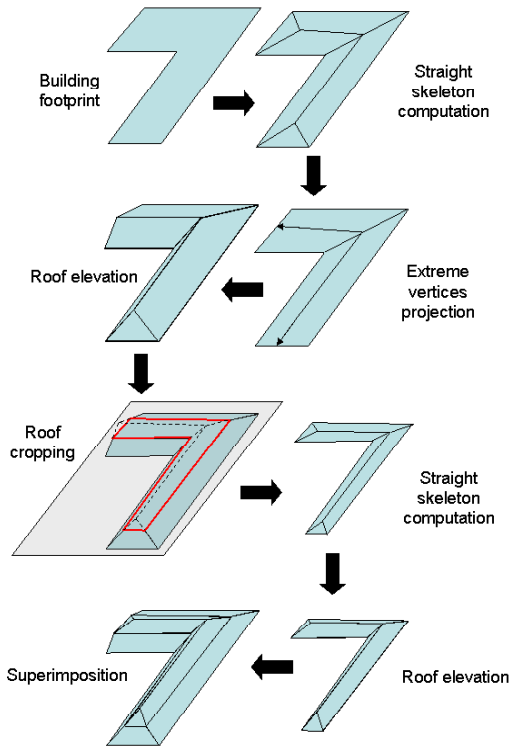

Figure 6: The different steps for roof reconstruction: straight skeleton computation, projection of extreme vertices in the case of a gable roof, roof elevation, crop operation, reiterate these previous steps for each superimposed roof.

3.2 Facades

To represent a facade, some charateristics have to be taken into account. First, facade primitives such as windows or doors are generally aligned along the X-axis (horizontal) and Y-axis (Vertical). Secondly, a facade is structured by a set stories. Finally, most of facade primitives have similarities. The goal of this representation is not to model a facade with maximum geometric accuracy, but rather to keep the main characteristics of the facade (color , building material, number of stories, number of windows per story, facade primitive style, etc) by coding it with a compact representation.
For these reasons, a facade is partitionned in cells aligned on the X and Y-axis. More precisely, a facade is partitionned in stories with different heights, and each story is partitionned in cells with different widths. Each cells can be iteratively partitionned in stories and cells. Moreover, each cell can hold a facade primitive such as a 2D texture or a 3D model. To reduce the number of facade primitive that have to be sent to the client, primitives are shared betweeen buildings. In this way, a representative library of facade primitives is sent to the client at the beginning of the session.
In addition, the following parameters assigned to each facade primitive are sent to the client so to allow repeating, scaling and positioning it in its cell:

- Repeat: Allow to repeat a facade primitive (texture or 3D model) in the cell.
- XScale and YScale: Specify the number of facade primitive occurences on X and Y-axis in the case of a repeat mode.
- XPosition and YPosition: Allow to translate the facade primitive in the cell.
- XRepeatInterval and YRepeatInterval : Allow to specify an interval between each facade primitive in the cell in the case of a repeat mode.

Figure 7 visually presents the function of each previous parameters. Depending of the mode used for facade primitive mapping, just a part of these parameters are sent to the client. 3D models used as facade primitives must be refenrenced in a coordinate system where the plane $z = 0$ corresponds to the building wall (X-Axis is oriented to the left side of the facade, and Y-axis to its top). To avoid overlaps between different facade primitives, such as the parent cell representing the whole facade wall, and a children cell representing a window, holes corresponding to facade primitive outline in children cells have to be created in parent cells.
The example shown in figure 8 presents the facade structure and its reconstruction process. The first cell, representing the entire facade, is mapped with a stone wall texture. This cell is partitionned in two stories, and the ground floor is

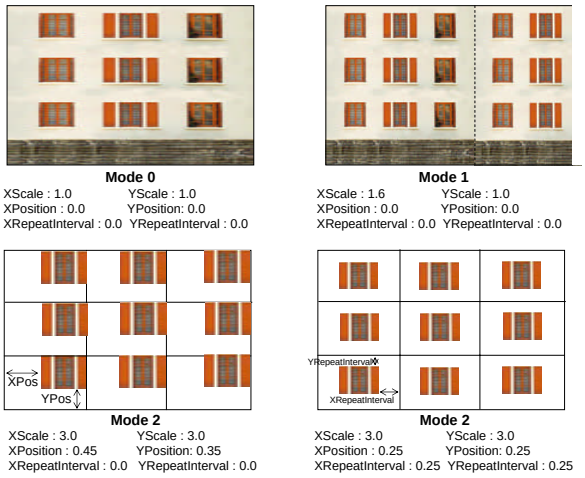

Figure 7: Repeat, scaling and positionning parameters function for facade primitives.

partitionned in three cells. The first and third cells have the same window texture repeated two times along the X-axis, whereas the second cell is partitonned into two stories. The first one is partitionned in three cells, with a 3D door model for the second one, whereas the other cells have a door frame texture.

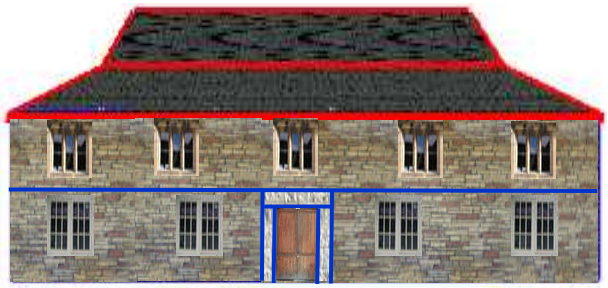

Figure 8: Example of a procedural facade.

4 MPEG-4 player

The PBTree method as well as the procedural modeling for roofs and facades have been proposed in MPEG-4 standardization (AFX tools). For this reason, the network based visualization system has been implemented on an MPEG-4 player. Thanks to this solution, all the multimedia functionnalities proposed by the MPEG-4 standard, such as video coding, progressive transmission of images by Region Of Interest, 3D sound, 3D avatar animation, can be used to offer new services based on 3D geo-visualization.

4.1 Client-server description

This system is composed of a database server and a visualization client (cf. Figure 9). The visualization client-server runs as an On-Demand Graphic system. First, the server sends the root node to the client to initialize it (initialization message). Then, the system runs iteratively as follows. On receiving this message, the client decodes it, then reconstructs in $3D$ the corresponding building, and decides upon the LOD selection module choice to add it or not to the MPEG scene graph. Each building has four states: coded, decoded, 3D reconstructed or active (or displayed). To each state corresponds a memory cache. Thanks to a memory cache manager a building representation can be removed from the different caches. Now that the node is rendered, the LOD selection module tests if it must be refined or collapse according to the viewpoint. If the node must be refined, a request containing only its index is sent to the server. The server treats it by sending to the client the refinement message encoded using the MPEG4 standard (refinement message).

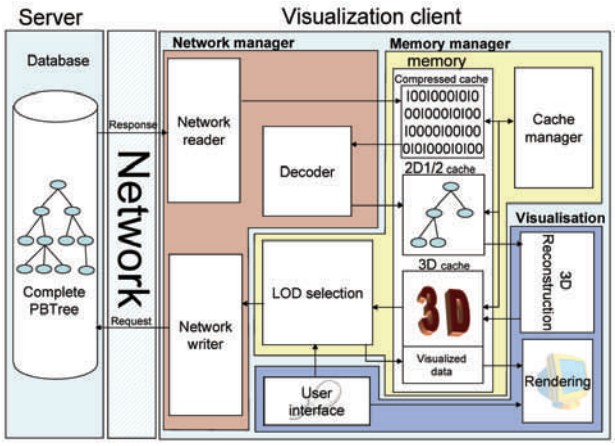

Figure 9: Visualization client-server architecture.

4.2 PBTree coding principle

To avoid transmission of redundant data, some vertices that defined the footprint of a BuildingNode reuse some vertices of its father node. Only new vertices have to be sent to the client, and their coordinates are defined in a local coordinate system centred on the centroid of the father node footprint. As the acquisition of the urban scene is done with a given accuracy, the coordinates of new vertices can be coded with very few bits.

4.3 LOD Selection

The LOD selection consists of determining which nodes have to be expanded or collapsed. The LOD selection is based on a comparison of a metric error with a function of the object-viewpoint distance. A building node $Node$ at a distance $dist$ from the viewpoint is expanded or collapsed along this criterion:

$$\begin{aligned} &\text{expansion test:} && \frac{dist^{\beta}}{\alpha} < MetricError(Node) \\ &\text{collapse test:} && \frac{dist^{\beta}}{\alpha} > MetricError(Node) \end{aligned} \quad (8)$$

where $MetricError$ is defined by equation 7, α and β are scalars allowing a parametrization of the LOD selection function. In the case of a navigation with a limited rotation speed, buildings located outside the view pyramid can be collapsed.

5 Results

The PBTrees have been computed for two city models. The input $2D\frac{1}{2}$ representation format is the *shape* format from ESRI. This format allows associating various parameters (height, altitude, facade texture index, etc) to different types of primitives such as polygons (the building footprints). Table 5 gives the sizes of data files produced with indexed face sets (compressed with MPEG4 Bifs format but without multiresolution) compared to the ones obtained with the proposed PBTree MPEG4 coding method. Indexed face set is not the best representation as regards compression, but it is the only available lossless representation in our case, as building representations cannot take advantage of existing compression methods applied on meshes. These results show the great efficiency of this representation in compression, which makes it the most likely to be used for network based real time navigation inhuge urban environment. The

City	Nb Buildings	Indexed Face Set (Bifs Compression, no multiresolution)	PBTree encoded with MPEG 4	Rate
Rennes	35 000	31 246 Kb	4 669 Kb	**0,15**
Paris	50 000	45 376 Kb	6 456 Kb	**0,14**

Table 1: Comparison in term of compression between the PBTree representation (MPEG 4 coding, progressive, view-dependent) and a classical indexed face set representation (Bifs coding, no multiresolution).

average decoding time for a building rarely exceeds 20 microseconds and the 3D reconstruction time ranges from 0.5 to 5 milliseconds depending of the complexity of the building model.

Figure 10 shows the evolution of the city model refinement, highlighting the good behaviour as regards the street structure. Figure 11 shows various screenshots of the application

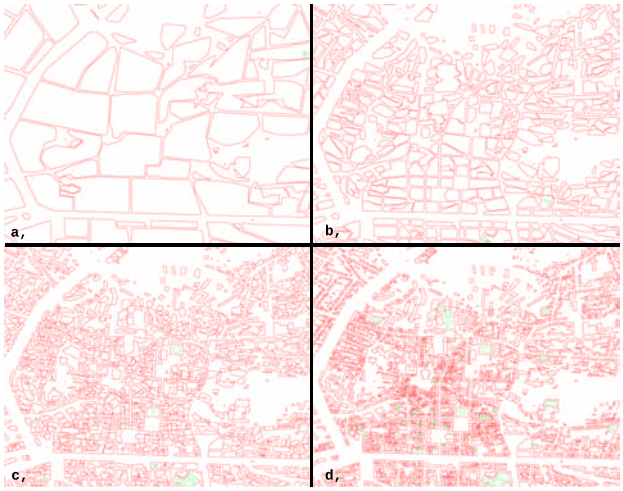

Figure 10: Refinement model evolution showing the progressive transmission of the city

taken during the navigation. Note on the vectorized views that the density of buildings showing the level of the details is much more important near the viewpoint (due to the view dependency of the PBTree).

Figure 11: Screenshots of the visualization system (city of Rennes, 35000 buildings), with th $3D$ and $2D$ respective view of the building footprints showing the view-dependent refinement.

6 Conclusion

We have proposed a new progressive and hierarchical building representation, allowing smooth city flying-over navigation on a client-server system. It is based on a progressive a tree structure called PBTree that allows view-dependent reconstruction of the city model. Furthermore, a procedural representation of roof and facade models has been presented so as to more faithfully model the buidlings compared to simple footprint extrusions. This representation has been proposed for standardization in MPEG-4 AFX tools, and implemented on an MPEG-4 player. Experimental results obtained with this representation demonstrate its great efficiency, and shows the advantages featured by this approach:

- flying over cities is allowed, with an important depth of field,
- the geometry transmission is progressive and hierarchical, allowing a full scalability and view-dependent reconstruction,
- the transmitted data can be easily compressed, using a local reference system for the vertex coordinates,
- the complexity of the $3D$ reconstruction processed by the client is not reduced to a block, but can be enhanced using procedural parameters to generate roofs and facades.

References

[1] H. Hoppe Efficient Implementation of Progressive Meshes In *Computer and Graphics*, volume 22, pages 27-36, 1998

[2] P. Wonka, M. Wimmer, Michael and F.X. Sillion Instant Visibility In *Eurographics 2001*, volume 20, 2001

[3] Seth J. Teller and Carlo H. Sequin Visibility Preprocessing For Interactive Walkthroughs In *Computer Graphcis (proceedings of SIGGRAPH 91)*, 25(4):61-69, 1991, ACM Press

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

[5] W. Pasman and F. W. Jansen Comparing Simplification and Image-Based Techniques for 3D Client-Server Rendering Systems, In *IEEE Transactions on Visualization and Computer Graphics*, P(2):226-240, 2003

[6] H. Pfister, M. Zwicker, J. Van Baar and M. Gross Surfels: Surface Elements as Rendering Prinitives In *Computer Graphics, SIGGRAPH 2000 Proceeding*, pages 343-352, Los Angeles, 2000, ACM Press

[7] J. Royan, C. Bouville, P. Gioia A New Progressive and Hierarchical Representation for Network-Based Navigation in Urban Environments In *Vision Modeling and Visualization*, pages 299-307, Munich, Germany, 2003