

MPEG-4 ADAPTIVE STREAMING FOR VIRTUAL CITIES FLYOVER

J. Royan, C. Bouville, P. Gioia

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 stream proposed for MPEG-4 standardization allowing navigation over 3D cities in real time. Thanks to a novel progressive and hierarchical representation called PBTree, only perceptible details for all the regions visible from a given viewpoint are progressively streamed to visualization clients. So as to efficiently transmit city model over network, a coding scheme to compress the PBTree is used.

1. INTRODUCTION

Many solutions are now available to automatically generate 3D models of huge urban environments. Except procedural models, all these solutions create huge representations that are inappropriate for a network-based visualization. To cope with network bandwidth limitations, compression, progressivity and adaptability are essential features.

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 itself very well to progressive transmission and view-dependent simplification.

The remainder of the paper is organized as follows. After presenting a reminder on PBTree principles that fulfills the requirements of a network-based navigation over huge urban environments, Section 3 describes the nodes and the bitstream syntax, while detailing the coding scheme to compress the PBTree updates for adaptive and progressive streaming. Then, we comment our results in Section 4, and finally, we conclude with some future work directions.

2. PB TREE PRINCIPLES

The PBTree is a tree-structured representation of urban environments allowing multi-resolution visualization [1]. This representation is basically $2D\frac{1}{2}$. Each building model consists in a footprint associated to two scalars representation

its height and altitude. This footprint-elevation representation can be easily enhanced so as to include detailed description of roof and facade models. Our multi-resolution representation uses a solution similar to the *vertex tree* of Hoppe [2]. In the PBTree, each node describes a building at a given level of detail. Suitable atomic simplification operations has been defined to reduce gradually the complexity of the scene. These simplification operations are applied according to a specific cost function that estimates the visual error introduced by the simplifications.

To reduce the complexity of the urban scene, simplification must be applied not only to each building separately, but over the entire $2D\frac{1}{2}$ model of the city. Figure 1 shows the three simplification operations that can be applied on the city model: simple vertex remove (S_1), footprint merge along adjacent edge (S_2), and the merge of two neighboring buildings (S_3). The PBTree is built by recursively applying simplification operations. At each simplification step, the simplification (operation and operands) is chosen so as to minimize an error criterion.

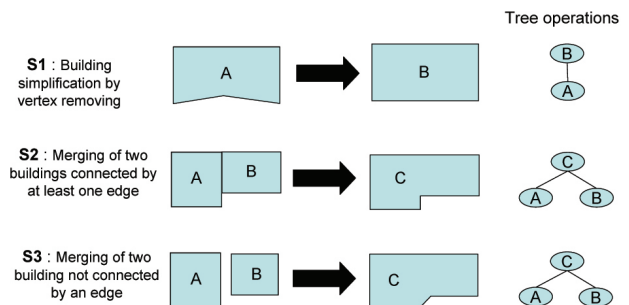


Fig. 1. Simplifications that can be apply on a $2D\frac{1}{2}$ city model.

The scene model is organized in a tree structure: the *Progressive Building Tree* (cf Figure 2). Each node of this tree, called *Building Node*, contains the $2D\frac{1}{2}$ representation of a building set at a given level of detail (pointers to parent and children, footprint description, height, altitude, index to facade texture, . . .). A vertex array is associated to this tree.

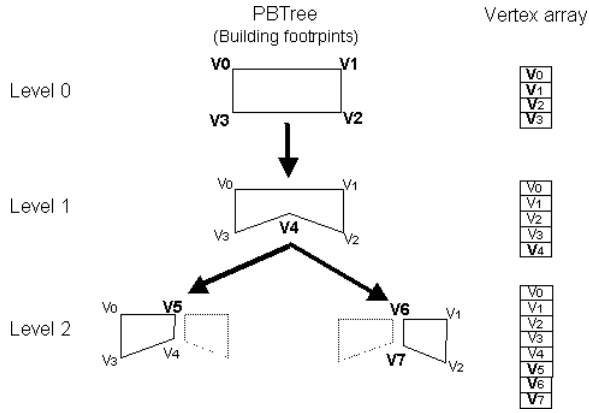


Fig. 2. Progressive and hierarchical representation using a building tree (in bold, the new vertices required for expanding a Building Node).

3. MPEG-4 STREAMING

We propose a bitstream syntax extension so to allow hierarchical footprint-based modeling.

3.1. Coding principle

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

3.2. The nodes

We have added a node associated to a stream, that defines a set of object based on footprint-elevation, and called FootPrintSet:

```
FootPrintSet {
  exposed field MFGeometryNode children [ ]
}
```

where the *children* field specifies a list of footprint-based objects (such as buildings) that are rendered according to the current viewpoint. This list is updated at each viewpoint change to adapt the scene complexity to the view.

3.3. The stream

In this part, the bitstream is described using an object-oriented-like syntax.

3.3.1. Decoder Specific info

A Decoder Specific Info is associated to each stream. It contains different parameters to be used to decode messages:

```
class FootPrintsDecoderConfig {
  int   FPOBJECTType
  int   MaxNbFootPrints
  int   FPNbBits
  float Step
  float MinX
  float MaxX
  float MinY
  float MaxY
  if (FPOBJECTType==BUILDING)
  {
    float MinAltitude
    float MaxAltitude
  }
}
```

where **FPOBJECTType** is the type of footprint (0 for classical footprints, 1 for buildings, ...). **MaxNbFootPrints** is the number of footprint-based objects corresponding to the number of nodes in the PBTREE. **FootPrintNbBits** is the number of bits used to decode the footprint object indices. **Step** is the acquisition accuracy of the scene, used as a quantification to code the new vertices coordinates on few bits. Finally **MinX**, **MaxX**, **MinY**, **MaxY** define the 2D bounding box of the scene. For the particular case of buildings, the parameters **MinAltitude** and **MaxAltitude** are added to obtain a 3D bounding box.

3.3.2. Messages

We have two types of messages. It can be first an initialization that represents the coarser representation of the scene. In this case, the root node of the PBTREE is sent to the client. The other case is one of a node refinement, allowing reconstructing all its children.

```
class FootPrintMessage {
  int      index
  bit      type
  FPNewVertices  FPNV
  int      indexNbBits
  if (type==refinement)
  {
    int      offspring
    for (i=0; i<offspring; i++)
    {
      int(indexNbBits)  localIndex
      float              metricError
      FPIndices         FPI
      if (FPOBJECTType==BUILDING)
        FPBuildingParameters params
    }
  }
  else
```

```

{
  float      metricError
  int        nbRings
  for (int i=0; i<NbRings-1; i++)
    int(indexNbBits) firstVertexIndex
  if (FPObjectType==BUILDING)
    FPBuildingParameters params
}
}

```

where **index** is the index of the root node in the case of a initialization message, or the index of the node to refine in the case of a refinement message. **type** is the type of the message (initialization or refinement). **FPNV** gives all the new vertices useful to reconstruct the current buildings (not already present in the parent node footprint), and is detailed next. **indexNbBits** is the number of bits used to code the vertices indices. **metricError** gives a geometric error between the current model and the original model. This error can be compared to a function of the distance from the viewpoint to the object to determinate if a node has to be refined or collapsed. In the case of a refinement, **offspring** is the number of children of the node being refined, **localindex** gives the index of each child, and **FPI** gives the index face set allowing reconstructing each ring of the current footprint (describe just next). For initialization, all new vertices are given by FPNV, and are ordered counter-clockwise. **NbRings** is the number of rings of the current footprint, and **firstVertexIndex** indicates which vertex comes first in a new ring.

```

class FPNewVertices {
  int      coordtype
  int      nbNewVertices
  for (i=0; i<nbNewVertices; i++)
  {
    int(coordtype) deltaX
    int(coordtype) deltaY
  }
}

```

where **coordtype** is the number of bits used to decode each new coordinates. **nbNewVertices** gives the number of new Vertices. **deltaX** and **deltaY** allow determinating the coordinates ($newX, newY$) of new vertices as follows:

$$\begin{aligned}
& \text{if } (type==initialization) \\
& \quad \begin{cases} newX = \delta X \\ newY = \delta Y \end{cases} \\
& \text{else} \\
& \quad \begin{cases} newX = \frac{\delta X}{step} + gc_x \\ newY = \frac{\delta Y}{step} + gc_y \end{cases}
\end{aligned} \tag{1}$$

where (gc_x, gc_y) are the gravity centre coordinates of node being refined. All the new vertices are stored in an array *NewVertices*.

Footprints are modelled using *IndexedLineSet2D*, and the corresponding indices are coded as follows:

```

class FPIndices {
  int      nbVertexIndices
  for (int i=0; i<nbVertexIndices; i++)
    int(IndexNbBits) index
}

```

where **nbVertexIndices** is the number of indices allowing the reconstruction of the corresponding footprint, and **index** gives the current index. Determinating which vertex v correspond to a index is done as follows:

```

if (index==0)
  new ring
else
  if (type==initialization)
    v=NewVertices[index-1]
  else
    if (index<=NbParentVertices)
      v=ParentVertices[index-1]
    else
      v=NewVertices[index-NbParentVertices-1]

```

where *ParentVertices* is the vertices array describing the footprint of the node being refined, and *NbParentVertices* is the size of this array. All rings are separated by index 0.

Finally, the coding can easily be extended to modelled buildings, by adding specific parameters required to reconstruct in 3D the building:

```

class FPBuildingParameters {
  float      altitude
  float      height
  URL        facadeTextureURL
}

```

where **altitude** is the altitude of the building, **height** its height, and **facadeTextureURL** the URL of the texture that will be mapped on facades. To model more complex buildings with detailed roofs and facades, a new description of building using procedural reconstruction should be studied for standardization.

4. RESULTS

The PBTree have been compute for two cities. Table 4 presents the size of generated files, ones using the classical indexed face set (compressed with Bifs, without multiresolution), the other ones using the PBTree encoded with the proposed MPEG-4 standard (with progressivity and view-dependence). 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, that makes it prone to be used for network-based navigation in huge virtual urban environments. Most buildings can be decode in less than



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

| 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 | 350 000 | 224 907 Kb | 29 230 Kb | 0,13 |

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).

45 microseconds (the decoding complexity is linear), and their 3D reconstruction time take less than 2 milliseconds. Figure 3 shows various screenshots of the application taken during the navigation. Note on the vectorized views that the city reconstruction is view-dependent.

5. CONCLUSION AND FUTURE WORKS

We have proposed a new adaptive stream for network-based navigation over huge urban environments (proposed for stan-

dardization in MPEG-4 AFX tools). This stream is based on a $2D\frac{1}{2}$ multiresolution representation called PBTree, having the advantage of being progressive, compact, view-dependent and scalable. Experimental results obtained with this representation shows its great efficiency.

Procedural reconstruction methods for roofs and facades have to be studied in standardization, to be able to reconstruct well-detailed building models. To reduce popping effects after refinements during the navigation, geomorphing techniques have to be implemented. Finally, this client-server visualization system is being ported to mobile phones to offer a large panel of services.

6. REFERENCES

- [1] P. Gioia J. Royan, C. Bouville, "A new progressive and hierarchical representation for network-based navigation in urban environments," in *Vision Modeling and Visualization*, Munich, Germany, 2003, pp. 299–307.
- [2] H. Hoppe, "Efficient implementation of progressive meshes," in *Computer and Graphics*, 1998, vol. 22, pp. 27–36.