A Hierarchical Topology-Based Model for Handling Complex Indoor Scenes

D. Fradin, D. Meneveaux and P. Lienhardt

Signal Image Communication laboratory, CNRS-FRE2731, University of Poitiers, France
{fradin,daniel,lienhardt}@sic.univ-poitiers.fr

Abstract
This paper presents a topology-based representation dedicated to complex indoor scenes. It accounts for memory management and performances during modelling, visualization and lighting simulation. We propose to enlarge a topological model (called generalized maps) with multipartition and hierarchy. Multipartition allows the user to group objects together according to semantics. Hierarchy provides a coarse-to-fine description of the environment. The topological model we propose has been used for devising a modeller prototype and generating efficient data structure in the context of visualization, global illumination and 1 GHz wave propagation simulation. We presently handle buildings composed of up to one billion triangles.

Keywords geometric modelling, large buildings, partitions, hierarchical model, visualization.


1. Introduction
We are interested in complex indoor scenes with tens of floors and hundreds of rooms with furniture. Such scenes can easily require several gigabytes of memory; they are composed of a high number of geometric primitives as well as a large variety of information (photometry, textures, radiometry, etc.). Moreover, the scene display and editing becomes hardly practicable due to the amount of information. To face these problems with a general modeller, the user has to elaborate specific modelling strategies depending on the nature of the environment (architecture, automotive, etc.).

In this paper, we propose a scene structure adapted to modelling operations while providing enough information for visualization, lighting simulation or wave propagation. For instance, object surfaces are used with lighting simulation and rendering. Conversely, wave propagation rather relies on walls volumes and their properties.

Indoor scenes (Figure 1) are naturally organized into subparts: floors, wings, rooms, walls, etc. At a lower level, this corresponds to subdivisions of the 3D space into vertices, edges, faces and volumes. Subdivision elements are commonly called cells, linked together through adjacency and incidence relationships defining the model topology.

Our representation corresponds to a structured space subdivision. It describes the building topology with fine details including data for both modelling and rendering. Many types of information can be provided depending on the application: wall volumes, rooms, adjacency and visibility graphs such as those defined in [1–3], etc. These decompositions can be efficiently used for global illumination with radiosity [4,5] or the photon-mapping approach [6] described in the results.

We believe that it is worth to make use of topological information about the scene even though it increases the storage of all the data structure involved: (i) topology provides a coherent representation of space subdivisions with adjacency/incidence relationships; (ii) it facilitates the scene construction (for instance through local operations such as split, extrusion, etc.); (iii) taking topological information into account can drastically reduce the time complexity of visibility computations during the rendering process.
Our topological model extends generalized maps proposed in [7] with two important features: multipartition and hierarchy. The multipartition concept allows to create groups of objects (groups of rooms for example in Figure 2(b)) while hierarchy makes it possible to perform a given processing only on a subpart of the scene (adding furniture in a given room, Figure 2(a)). We also present a building modeller prototype and applications using our representation. Results demonstrate that the model we propose is suitable for very large buildings. This paper is dedicated to the scene representation; we do not provide a complete modeller for architecture nor a new method for optimised rendering or global illumination (see [6]).

Our contributions concern the following points:

- A multipartition and hierarchy representation dedicated to complex indoor scenes;
- a set of topological operations associated with this representation;
- a modeller prototype for large buildings with furniture;
- the use of our model for several applications concerning global illumination and 1 GHz wave propagation.

The remaining of this paper is organized as follows. Section 2 presents the work most closely related to our approach. Section 3 describes our hierarchical model, based on generalised maps. Section 4 provides some details about construction operations and implementation. Section 5 discusses the results we obtained and provides applications examples relying on our representation.

2. Related Work and Choices

We aim at representing large buildings, actually corresponding to 3D topological objects, made up with vertices, edges, faces and volumes which do not necessarily have regular shapes. Many topological models have been proposed in the literature for handling different classes of subdivisions (oriented surfaces, manifold, nonmanifold, etc.) in any dimension. Examples of such structures are adjacency graphs [9], ordered models (as defined in [10]), 2D or 3D edge-based models [11–13] or higher dimensional models [7,10].

Incidence graphs (such as [9]) do not allow multi-incidence (see [10]). Coherence constraints necessary to represent an orientable 3D manifold with an incidence graph cannot easily be expressed. The definition of construction operators accounting for these constraints and guaranteeing thus topological consistency of modelled objects can be also difficult for dimension 3 and higher (see [14]). This is the reason why ordered models have been introduced. They are mainly defined with a single type of basic elements and links between these elements [7,10].

For the reasons explained above, we chose an ordered topological model. Complexity studies have shown that for 2D and 3D manifold (surface subdivisions or $\mathbb{R}^3$ subdivisions), costs for representing an ordered model and an incidence graph are comparable [15]. This is even more true when taking geometry, photometry and other attributes into account.

The objects we wish to represent are buildings composed of volumes (floors, walls, rooms, etc.) sharing faces. Topologically speaking, it corresponds to 3D orientable manifolds. In [16], it has been shown that models defined to represent 3D manifolds are comparable either to 3D maps (for orientable ones without boundaries) or to 3D generalized maps (for orientable or not orientable ones, with or without boundaries). Even though 3D generalized maps are a bit more costly than 3D maps in terms of memory representation, we chose this model because it allows to represent objects with explicit boundaries, operations are easier to implement and the cost difference is low. Moreover, generalized maps are defined homogeneously in any dimension.

The major drawback is related to the data structure size. For complex environments, several million polygons may have
to be defined; the model needs to be structurally improved. This is why we propose a hierarchical description adapted to large buildings.

Topology-based hierarchical representations have been used in several areas. For instance, in the field of image processing, pyramidal models have been used for long to handle several subdivisions of a given 2D image. Each region is linked with its associated decomposition in the next/previous level [17,18]. In the context of 3D applications, multiresolution meshes offer several descriptions (more or less precise) of the same object. For example, hierarchical descriptions based on simplex-based models (or convex cells) have been proposed in [19–21]. The Multiresolution Simplicial Model [22,23] unifies the previous models. The difference between two successive levels is expressed through transformations applied to topology and geometry (with or without overlaps). The object is then represented as an acyclic graph accounting for the different levels. This model does not explicitly represent each hierarchy level, some operations have to be performed to reconstruct a given description of the object. Since an explicit description of the scene is necessary for our application, we chose to conceive a specific hierarchical model.

Rendering complex scenes is still not straightforward for two main reasons: (i) the memory needed for storing the database and (ii) the number of objects to handle. Space subdivision methods help to manage the memory and reduce the number of stored objects. In this case, a topological representation is required. Generally, the scene subdivision is (semi)automatically reconstructed from a list of polygons [24–26]. The scene subdivision dramatically reduces the inter-region visibility computation cost. There exists a vast literature in this area (regular grids, BSP/kD-trees, etc.) with several approaches dedicated to large buildings [1,3,4]) for walkthrough [27] or lighting simulation [4,5,28]. However, the obtained results do not reach the precision of a manual partitioning. Topological information used during the modelling process, when available, can be most useful to avoid this reconstruction. Our representation makes it possible to extract adjacency or visibility graphs used in [1,4,5], or viewpoint-based visibility (see Section 5). Such extracted information has already been used for rendering, global illumination [6] and 1 GHz wave propagation [29].

3. Hierarchical Model

Our representation makes use of three cells and links between elements. Our model relies on generalized maps (Section 3.1), enriched with multipartitions (Section 3.2) and hierarchy (Section 3.3).

3.1. Generalized Maps

We choose generalized maps defined in [7] for the following reasons:

- They can represent subdivisions of 3D space;
- They are defined in an homogeneous way: a single type of basic element (Figure 3 in 2D and Figure 4 in 3D). This simplifies the formal definition of many operations.
- Operations for merging cells have already been defined [30], providing a theoretical basis for our hierarchical representation
- Any type of attributes (such as geometry, photometry, texture, etc.) can be associated with any n-dimensional cell.

Definition 1 - generalized maps [7]: Let \( n \geq 0 \); an \( n \)-dimensional generalized map \((n-G\text{-map})\) is defined by an \((n+2)\) - tuple \( G = (D, \alpha_0, \ldots, \alpha_n) \), where:

- \( D \) is a finite set of darts;
- \( \alpha_i \) is an involution on \( D \) for \( 0 \leq i \leq n \); (a bijection \( f \) is an involution \( \iff f^2 = 1d \));
- \( \alpha_i \circ \alpha_j \) is an involution for \( i \geq 0 \) and \( i + 2 \leq j \leq n \).

Intuitively, a dart can be seen as a cell-tuple [10], i.e. a sequence of incident cells of increasing dimensions. For instance in Figure 3, dart 1 in (b) corresponds to vertex \( v \) of edge \( e \) of face \( F \) in (a).

Orbits: An orbit is described by a dart and a set of involutions. It provides the set of all darts that can be reached by any composition of the given involutions (graph traversal, cf. Figure 3).

Cells: An \( i \)-dimensional cell (or \( i \)-cell) incident to a dart is an orbit composed of any composition of all involutions except \( \alpha_i \).
Attributes: Each cell can be provided with different types of data such as point coordinates for vertices (here, we only use a linear representation), photometry for faces or any semantical information. Data are associated with an orbit (often defining a cell).

In practice, attributes associated with a cell are stored on a single dart. This implies to scan all darts of the cell to retrieve the information but reduces the required memory and facilitates modifications. If attributes are often needed, it is possible to propagate the information onto each dart to accelerate queries. More details about implementation can be found in [31].

We are mainly interested in grouping volumes as rooms, walls, etc. For instance, we can define groups corresponding to a gallery for guided walkthrough in a museum, groups of rooms corresponding to the same floor or same wing, etc. It can also be useful to group several elements of the scene according to their use or appearance.

Moreover, a hierarchical representation is well suited for dealing with levels of details and memory management. For instance, visibility computations between rooms do not require furniture; when adding a table or a chair, only the selected room should be displayed, not the whole scene.

We propose to separate the representations of multi-partition and hierarchy, because they can be handled by different sets of operations. For the sake of efficiency, we also need the model to require a small size of memory. We chose to represent partitions with the help of Boolean marks associated with involutions. Hierarchy explicitly defines details associated with cells avoiding redundancies.

To sum up, the main features of our topological model are the following:

- representation of groups of cells (e.g. set of rooms for a single floor);
- hierarchical building representation: facade, floors, rooms, furniture, etc.;
- able to handle a high number of polygons (our largest scene is composed of 1 billion polygons);
- with efficient data access (darts, geometry, other attributes) for modelling and rendering.

3.2. Multipartition

From a technical point of view, a G-map can be considered as a graph where nodes correspond to darts and edges link two nodes when the corresponding darts are linked by an involution α. A connected component for a G-map fits the usual notion of connected component for a graph. For representing connected subgraphs, one has to distinguish between the subgraph edges and the other edges. A Boolean mark can easily represent this distinction.

Formally, we get the following definition:

**Definition 2 - group involutions and multi-partitions:** Let \( G = (D, \alpha_0, \ldots, \alpha_n) \) be an \( n \)-G-map. A group involution \( \alpha^g \) is an involution on \( D \) such that:

- \( (D, \alpha_0, \ldots, \alpha_{n-1}, \alpha^g) \) is a G-map;
- \( \forall d, d' \in D \) such that \( \alpha^g(d) = d' \), either \( \alpha^g(d) = d \) and \( \alpha^g(d') = d' \).

A multipartition is represented by a set of group involutions, one for each partition (see Figure 5).

This definition provides a mean to group volumes in a 3-G-Map. It can be easily extended for grouping cells in any dimension.

Note that the definition of \( \alpha^g \) links implies new orbits called group orbits (coverage of groups). New attributes called group attributes can thus be defined. In practice, they are used for storing semantical or geometrical information. As explained above, \( \alpha^g \) have essentially been defined for grouping volumes such as rooms or walls. However, we also use \( \alpha^g \) for grouping aligned edges or \( \alpha^g \) for grouping coplanar faces, so that attributes can be factorized and some geometric computations are avoided.

3.3. Hierarchy

An explicit representation is desired for buildings since the scene should not be recomputed every time a new process
is applied. Generally, construction operations are performed with the addition of new details; for instance when adding walls in a floor, furniture in a room and so on. In practice, a cell is detailed by a set of cells. For adding details to a set of cells, it is possible to use our group representation.

For a given level and a given cell in this level (e.g. a room in a floor), the cell is duplicated and modified with successive topological operations such as split of i-cells, extrusions, etc. A link \( \eta \) is defined between the current level and its lower level: \( \eta \) associates each dart to its copy (such as in Figure 6).

**Definition 3 - hierarchy:** A hierarchy is a sequence of G-maps \( \langle (D^\alpha, \alpha_0', \ldots, \alpha_n') \rangle_{n-in,m}, m \) being the number of levels. Levels are linked together with hierarchical links \( \eta \). Note that \( \eta_j \) is a bijection from a subset of \( D^\alpha \) with a subset of \( D^{\alpha+1} \). The model is kept consistent through our construction operations.

4. Operations and Modeler

Our modeller describes a building as a 3D space subdivision organized according to our multipartition and hierarchy model. The topology representation is kept hidden to the user unless he selects the option ‘visualize topology’ (topology is shown on Figure 16(a)).

The building modeller is based on a topological kernel implementing G-maps in C++ [31]. A 3-G-map is defined by a set of darts; a dart is a data structure containing four pointers towards darts: \( \alpha_0, \alpha_1, \alpha_2 \) and \( \alpha_3 \). All the darts corresponding to an i-cell can be recovered from a dart using a traversal as in a usual graph, where nodes are darts and edges are \( \alpha_j, j \neq i \). The dart data structure also contains Boolean marks used for traversals and modification operations of the G-map. A list of attributes is stored in the dart class. An attribute is a triple composed of a type, the information to store and a byte that identifies the orbit (vertex orbit for coordinates, face orbit for photometry, etc.).

Together with the data structure, the kernel defines elementary operations for manipulating darts: inserting/removing darts, linking darts with \( \alpha_j \), managing marks, adding/removing attributes. Other operations provide: cells insertion/removal, traversals, control of attributes coherence, etc. Thus, materials for volumes, photometry for faces, coordinates for vertices, etc. are automatically handled. Note that attributes associated with a given cell can be either carried by only one dart for avoiding redundancy or referenced on all the cell darts for fast retrieval.

Higher-level operations have already been defined upon generalized maps. For instance co-refinement and Boolean operations in the context of geology [32], mesh compression in arbitrary dimensions [33] or chamfering [34].

For our application, several software layers have been added: a first layer defines partitions; the second layer is used for hierarchy (note that each hierarchy level has its own multi-partition); the third layer sets up all the operations in charge of the building construction.

4.1. Multipartition Layer

For implementing multipartitions, additional marks have been associated with involutions \( \alpha_i \) inside the dart class. Each Boolean indicates whether two darts are grouped. Every \( \alpha_i \) pointer is associated with 8 Booleans (stored in one byte). A unique Boolean is sufficient to describe a set of groups corresponding to one partition. Thus, with a byte, eight different partitions can be defined for each involution. A group is then identified by a dimension \( i \) and a partition number \( p \) (between 0 and 7). Figure 7 shows an example of a partition: all the corridors of a floor are grouped.

We have also added in the G-map class (containing a set of darts) the operations allowing to manipulate partitions: grouping two cells of same dimension, managing group attributes, etc.

The list of all the darts in a given group \( (p \) and \( i \)) is obtained through a traversal of all the cells grouped with \( \alpha_i, j \neq i \). The traversal is processed as in a graph with edges defined as \( \alpha_j, j \neq i \) and \( \alpha_i \) when its \( p^\text{th} \) Boolean is set to true. We have also defined a method for testing if two darts \( d_1 \) and \( d_2 \) belong to the same group in a given partition (associated with a given dimension \( i \)). This method applies a traversal from \( d_1 \) in the group for finding \( d_2 \). In the worst case, all the darts in the group have to be checked. However, this method is only used for attributes management.

For grouping two i-cells \( C \) and \( C' \) in the partition number \( p \), the operation assumes that \( C \) is already linked to \( C' \) by \( \alpha_i \), along a \( (i-1) \)-cell \( C'' \). In every dart of \( C'' \), the \( p^\text{th} \) Boolean mark corresponding to \( \alpha_i \) is set to true. For this operation,
attributes have to be checked. If some attributes of the same type are defined in $C$ and $C'$, only one has to be retained. In practice, we keep arbitrarily the one defined in $C$. When ungrouping cells, all the corresponding marks are reset on $C''$. The corresponding group can either remain connected or be disconnected into two subgroups. In the latter case, attributes have to be duplicated in each subgroup.

4.2. Hierarchy Layer

Each hierarchy level is represented by one G-map. Therefore, one level can be loaded into the memory and managed independently of the others for local operations such as wall creation or furnishing. The hierarchical link $\eta$ is represented by two pointers in the dart class: a pointer toward the parent dart and one toward the child dart (respectively called $\eta_p$ and $\eta_c$). The G-map class contains also two pointers indicating the parent and child G-maps.

The main operation for hierarchy construction consists of creating the detail of a given cell. First, a child G-map has to be created if the corresponding hierarchy level does not exist. Second, the cell is reproduced in the child G-map. New darts are linked to the original ones with $\eta$ links. Later on, this new cell can be modified using usual G-map modelling operations with cells insertion/removal. Note that these operations may affect several lower and upper hierarchy levels. When a modification operates at a given hierarchy level, it can be necessary to propagate the changes to the parent cells. In practice, modifications are propagated by the way of high-level operations defined in the building-dedicated layer (see next Section).

On the one hand, child pointers are defined for every dart of a parent volume. On the other hand, only a small number of darts in the child G-map are directly linked to their parent darts. Therefore, when operations are propagated to the upper hierarchy level, the parent cell of a given dart has to be efficiently found. The traversal is performed according to cells of increasing dimension (from the vertex associated with the given dart to the corresponding volume). This traversal stops as soon as a dart contains an instantiated $\eta_p$ link.

Figure 8 illustrates the possible configurations for a given dart $d$. If $d$ is directly connected by $\eta_p$ (dart number 1 on the figure), its parent dart is returned. When $\eta_p$ is not defined (darts 2, 3 and 4), we first cover the edge carried by the dart. If this edge contains a dart directly linked to the parent cell (e.g. dart 2), the function returns the parent dart. Otherwise (darts 3 and 4), the dart containing the $\eta_p$ link is searched within the volume boundary faces (darts not linked by $\alpha_3$). If it is found on the boundary (dart 3), $d$ is located in a face detail. Otherwise (dart 4), $d$ is inside a parent volume and we thus return any dart of the parent volume (all the detail is traversed until a parent dart is found). For each case, the dart returned is given with the orbit indicating the parent cell (vertex, edge, face or volume). To sum up, the parent of dart 1 is given by $\eta_p$, dart 2 has the same parent as dart 1 (with a different orbit), the parent of dart 3 can be any dart in the parent face of dart 1 and dart 4 can be associated with any dart of the parent volume. In the last two cases, our algorithm always returns the closest dart in terms of number of composition of $\alpha$ involutions.

Hierarchical operations sometimes need to mark darts in several hierarchy levels. This is why hierarchical marks have
been defined. These marks are also used for selecting cells (rooms, objects, walls, and so on) and for defining local working areas.

4.3. Building-Dedicated Operations

The two layers described above have been enlarged with higher level operations concerning building creation such as extrusion, opening insertion and furnishing. Buildings are described with five levels of hierarchy: building facade (Figure 11(c)), outer walls (Figure 11(d)), floors with inner walls (Figure 11(f)), rooms and furniture (Figure 11(g) and 11(h)).

We have made a distinction between wallpapers and walls (Figure 9). Wall volumes define the building structure while wallpaper faces bound room volumes. Typically, with this decomposition electromagnetic attributes are associated with walls for radio wave propagation while photometry is associated with wallpapers for lighting simulation and rendering. Thus, portals are defined as empty volumes with two transparent faces.

The building is constructed with the help of coarse-to-fine operations. First, from a 2D shape designed by the user, the building structure is created by the way of extrusion operations. The three first hierarchy levels are automatically defined: facade, retaining walls and floors (Figure 11(a) to 11(d)). The facade is the root hierarchy volume; retaining walls with floors and ceilings are defined in the child volume; each floor is duplicated in the third level in order to be modified with the following operations.

When the user clicks for selecting a cell, a traversal identifies the closest dart within the current hierarchy level. This dart is then used for applying the following operations.

Inner walls insertion is used to describe rooms. Frequently, new walls are held by existing ones and require to be linked. Thus, new faces have to be placed for receiving the inserted wall and linked with $\alpha_3$. In most cases, the wall insertion subdivides a room into two new rooms.

Openings (doors or windows) can be placed into walls. An opening is defined as an empty volume placed inside a wall with two transparent faces leading to two rooms. Two examples of opening insertion are provided in Figure 10. Openings are always inserted in the third hierarchy level. The insertion always affects the child level (the room level) and sometimes the whole hierarchy. For instance, when a window is placed in a room (Figure 10(c)), the retaining walls and the facade have to be updated. First, the window is inserted as a transparent face in the room for both the floor level and the room level. Second the parent hierarchy level is modified so that the corresponding retaining walls contain an empty volume with transparent faces. Third a transparent face leading to the outside is placed in the facade. During this construction, transparent faces are linked together with $\eta$-links. Finding the faces concerned by modifications is performed using the operation described above for retrieving the parent cells of a dart.
Figure 11: Construction steps of a simple building.
Figure 12: Screenshots of three different buildings constructed with our modeller (with and without automatic furnishing).

Note that the algorithm checks opening validity before its insertion: (i) the opening must be smaller than the wall, (ii) its position must be completely inside the wall and (iii) it does not cross another portal. The position is automatically corrected by the program when one of these points is not verified. The user can move the mouse to precisely place the opening on the selected wall.

At any time, a floor can be copied into another one so that rooms, walls and openings do not have to be defined again by the user. This function removes the already existing floor, duplicates the entire subtree and updates the hierarchy for windows.

Once the structure of rooms has been defined, it is possible to add furniture. In the room level, furniture is represented as a bounding box, with a filename, a translation vector and a rotation matrix. The same file can be used several times; this corresponds to clones of the same object. The bounding box insertion consists of a volume inclusion operation defined in the G-map kernel operation. Point and surface light sources are also considered as furniture with radiometry. For simplifying room furnishing, we provide several operations such as copy/paste of furniture, furnishing scripts and random object placement.

Multipartitions are used for grouping rooms sharing at least one portal or connected walls. When the user selects rooms for grouping, the modeller automatically adds the corresponding openings in the group. Semantical information or any type of attributes can be associated with the whole group.

In this software layer, we also define the attributes dedicated to buildings. Faces and volumes can be associated with radiometry and photometry (face reflectances, volume materials, and so on). This information is used during lighting simulation and wave propagation. The scenes can be exported into various file formats such as faces for visualization, volumes for wave propagation or adjacency graphs for lighting simulation (as explained in Section 5.2). Exporting the scene consists of a top-down scene hierarchy traversal. For adjacency graphs, cells connectivity are deduced from both topological and hierarchical links (α and η).

### Table 1: Four furnished buildings; disk space is given with gzip compression.

<table>
<thead>
<tr>
<th>Building</th>
<th>No. of Polygons</th>
<th>No. of rooms</th>
<th>Disk (compressed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw model</td>
<td></td>
<td></td>
<td>Raw model Hierarchy</td>
</tr>
<tr>
<td>L-Building</td>
<td>336.5K</td>
<td>27</td>
<td>3.55 MB</td>
</tr>
<tr>
<td>Z-Building</td>
<td>1.074M</td>
<td>22</td>
<td>10.4 MB</td>
</tr>
<tr>
<td>Octagon</td>
<td>5 250K</td>
<td>232</td>
<td>54.8 MB</td>
</tr>
<tr>
<td>Tower</td>
<td>1.074 billion</td>
<td>17 800</td>
<td>8.5 GB</td>
</tr>
</tbody>
</table>

4.4. Construction Steps

As explained above, the scene produced by the modeller contains a lot of information, necessary for modelling operations. The program only displays the current hierarchy level. An option allows the user to display the next hierarchy levels except furniture geometry (replaced by bounding boxes) as shown in Figure 11(g). The modeller guides the user according to the following steps.

1. **Building shape**

The user first defines the building contour and provides the number of floors as well as the wall thickness and height. The modeller automatically creates the building structure (see Figure 11(a–d) and Movie 01 in website [35]).
2. Editing floors

The user can select and edit each floor independently. Inner walls are defined using a 2D view, the user draws a polygonal line that is extruded for generating the wall (Figure 11(e)). The program automatically deduces all room volumes.

Several types of openings such as doors and windows are proposed to the user (Figure 11(f)). They can be placed either on inner walls or outer walls. Movie 02 and Movie 03 in [35] illustrate the result of this operation.

While defining the building structure, the user can load a plan of a building (or a horizontal section). The image is shown in the background. This option is much useful for reproducing existing buildings (see Figure 16(b)). The user also has the possibility to add his own attributes to faces and volumes such as materials (concrete, plaster, etc.) or semantic information (offices, libraries, classrooms, etc.).

3. Room furnishing

Each room can be enriched with furniture (Movie 04 in [35]). Practically, due to the high number of polygons, object geometry and topology cannot fit in the memory. Thus, only bounding boxes are displayed; furniture is stored on the disk unless the user decides to visualize the detail of a particular room.

Note that inserted objects are either defined by a G-map with its geometry or described by a list of polygons. In this latter case, object topology needs to be reconstructed. The object is first revised so as to suppress degenerated triangles and triangulate noncoplanar polygons; or involutions are automatically retrieved from the resulting list of polygons when possible.

Table 2: Computing time with and without preprocessing for 400 × 300 pixel images of two buildings (L-shaped with 300K polygons and octagon with 5 millions triangles) without taking lighting into account. PC corresponds to the pre-computation using the topological structure. PI represents the time needed by POV-ray for reading the scene and constructing the accelerated data structure while PRT is the time needed for computing one image. T is the total computing time in both cases.

<table>
<thead>
<tr>
<th>L-shaped building</th>
<th>Brute force</th>
</tr>
</thead>
<tbody>
<tr>
<td>With topology</td>
<td>Brute force</td>
</tr>
<tr>
<td>PC</td>
<td>PI</td>
</tr>
<tr>
<td>(1) 9'' 1'' 1''</td>
<td>1'18'' 6'' 1'24''</td>
</tr>
<tr>
<td>(2) 9'' 2'' 2''</td>
<td>13'' 31'' 7'' 1'38''</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Octagon building</th>
<th>Brute force</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3) 10'' 1'' 2''</td>
<td>13'' Impossible to load</td>
</tr>
<tr>
<td>(4) 12'' 4'' 2''</td>
<td>18'' with 512 MB RAM</td>
</tr>
</tbody>
</table>
4. Final building

When each room has been described, the user can see the entire building with all the hierarchy levels (see bottom of Figures 12 and 16, Movie 05 in [35]), except furniture for saving memory.

5. Results

The following results correspond to scenes that have been constructed with our modeller and used with various rendering systems.

5.1. Buildings

Using the modeller described in this paper, we have built several scenes made up with from 300K up to 1 billion triangles. Table 1 provides the disk space required for each of them. Note the difference between raw and hierarchical models; the raw description includes the complete list of triangles with photometric attributes while the hierarchical representation uses cloned pieces of furniture.

Figure 12 presents images of our buildings. Our modeller interface with our biggest manually modelled scene is illustrated in Figure 1. Finally, we have created a building made up with 1 billion triangles spread out in 17,800 rooms and 101 floors (see Figure 12 bottom). Furniture has been automatically placed.

5.2. Rendering

For complex scenes, it remains difficult to handle millions of polygons. The scenes produced with our modeller have been used for several rendering systems, including OpenGL-based visualization, ray tracing and photon mapping. The topological information we propose is used for

- reducing the number of geometric primitives displayed during the modelling process;
- estimating straightforward view-dependent visibility information for POV-ray rendering [36] (Figure 13);
- generating cells and portals data structure [4,5] and computing out-of-core global illumination for very large buildings [6];
- providing the representation necessary for lower frequency electro-magnetic wave propagation simulation [29].

During the modelling process, the building is displayed with the help of OpenGL library. The building containing 5 million polygons only requires at worst 85 MB of memory (corresponding to the graphic user interface plus uncompressed data structure without furniture).

For generating one image with POV-ray, a precomputation process provides the list of rooms potentially visible from the viewpoint. Therefore, our ray-casting procedure uses only room volumes and portals without furniture and walls. Table 2 provides computing time for two scenes. These results have been obtained with a 1 GHz PC with 512 MB RAM. As seen in this table some of these viewpoints could not be rendered brute force with POV-ray because of the high number of polygons.

For lighting simulation, the building is first saved as a set of rooms enriched with an adjacency graph. Portals are defined by transparent polygons connecting the corresponding rooms. In [6], we propose a lighting simulation method that can handle our 1 billion triangles building (Figure 12, bottom) using memory-coherent photon tracing. With our method, only 500 MB of memory have been necessary for computing lighting simulation, though the whole data base requires 110 GB on the disk (8.5 GB compressed). As a result, 1.6 billion photons have been propagated in the scene and more than 400 GB are necessary for storing photons on the disk. After lighting simulation (computing time is given in Table 3), an image can be generated in a few minutes (Figure 15). In [6], a comparison of various acceleration schemes for rendering and lighting simulation proves the efficiency of topological information.

Finally, our modeller also provides adequate data structure for a wave propagation simulation algorithm [29] without furniture (Figure 14). In this case, wall volumes and materials are used rather than room volumes.

6. Conclusion and Future Work

This paper presents a topology-based model dedicated to large buildings. Our data structure allows both modelling and rendering complex indoor scenes. It extends the G-map topological model with multipartitions and hierarchies taking into account memory and time issues. Model robustness has been shown through several applications. First, we provide a set of operations dedicated to large buildings, included in

<table>
<thead>
<tr>
<th>Building</th>
<th>No. of Photons (millions)</th>
<th>Phot-prop. time</th>
<th>Phot-Map time</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-Shape</td>
<td>4.6</td>
<td>1’19”</td>
<td>21”</td>
</tr>
<tr>
<td>Z-Building</td>
<td>3</td>
<td>33”</td>
<td>1”</td>
</tr>
<tr>
<td>Octagon</td>
<td>30</td>
<td>4’04”</td>
<td>2’14”</td>
</tr>
<tr>
<td>Tower</td>
<td>1655</td>
<td>10h1’</td>
<td>3h56</td>
</tr>
</tbody>
</table>
a modeller prototype. Second, the scenes produced are used for ray tracing, photon mapping and wave propagation simulations at 1 GHz. Results are encouraging and show that topological information provided by the modelling process can be advantageously used by rendering systems. Presently, the topological information used during the rendering process concerns essentially high-level information such as volumes adjacency, rooms and portals. In the future, we aim at using lower level descriptions for taking advantage of incidence and adjacency information with faces and edges in the rendering process. Some work concerning the hierarchy reconstruction from existing models is currently in progress, as well as new high-level operations for the modelling process.

References


Figure 16: Images of our modeller: (a) View of the Octagon building with hierarchy and topology. (b) A plan (Chartres cathedral) can be used as background image.


35. Large Building Modeler, SIC Laboratory: http://www.sic.sp2mi.univ-poitiers.fr/mr-archi/cgf05.html.

36. PovRay: Persistence of vision raytracer pty. ltd. the terms “pov-ray” and “persistence of vision ray-tracer” are trademarks of the persistence of vision development team: http://www.povray.org.