- Draft -

Continuous Level-of-Detail Modeling of Buildings in 3D City Models

Jürgen Döllner
University of Potsdam
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
juergen.doellner@hpi.uni-potsdam.de

Henrik Buchholz
University of Potsdam
Hasso-Plattner-Institute
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam, Germany
henrik.buchholz@hpi.uni-potsdam.de

ABSTRACT

This paper introduces a concept for representing and modeling buildings in GIS at continuous levels of quality. Buildings are essential objects of virtual 3D city models, which serve as platforms for integrated, urban geoinformation. Existing concepts for the representation of buildings are restricted to a specific level-of-quality such as block models, roof-including models, architectural models, and indoor virtual reality models. The continuous level-of-quality approach unifies the representation of heterogeneous sets of buildings, which occur in most virtual 3D city models. It also leads to a systematic method for the incremental refinement of buildings – an important requirement of the long-term management of virtual city models. In our concept, a building's geometry is structured on a per-floor basis; each floor refers to a floor prototype, which is defined by a ground plan, walls, and wall segments. To specify the appearance projective textures across floors and textures per wall segment are supported. Application-specific data can be associated similar to appearance information. These few components already allow us to express efficiently most common building features. Furthermore, the approach seamlessly integrates into CityGML, an upcoming standard for virtual city model data.

Categories and Subject Descriptors

I.3.3 [**Computer Graphics**]: Picture/Image Generation - *display algorithms, viewing algorithms.* I.3.6 [**Computer Graphics**]: Methodology and Techniques - *interaction techniques.* I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism - *color, shading, shadowing, and texture.*

General Terms

Management, Design, Standardization.

Keywords

3D GIS, City Models, Buildings, Level-of-Detail, Virtual Reality.

1. INTRODUCTION

Virtual 3D city models and, more general, 3D geovirtual environments serve to present, explore, analyze, and manage georeferenced data. Their typical components include terrain models, building models, vegetation models as well as models of roads and transportation systems. They serve as platforms that integrate 2D and 3D geodata as well as georeferenced thematic data.

After a long time of feasibility studies, prototypic implementations, and geo-specific uses, first domain-specific applications and systems appear that incorporate virtual 3D city models as essential parts such as in facility management applications, real estate portals as well as entertainment and education products. Consequently, we can expect a large number of potential users and uses that require customized and specialized versions of 3D city models and a corresponding support by GIS.

For GIS an important question is how to represent buildings of virtual city models. Existing concepts are restricted to a specific level-of-quality such as block models, roof-including models, architectural models, and indoor virtual reality models, generally driven by the acquisition technology involved. Problems that result include:

- It is difficult to integrate buildings from different sources and varying level-of-quality.
- It is difficult to transform a given building onto a higher level-of-quality.
- There is no standardized way of encoding building-specific semantics in the representation.
- It is difficult to design modeling tools that support the modeling process across different levels-of-quality.

To overcome these problems we introduce a continuous level-of-quality representation for buildings. *CLOQ buildings* allow us to unify the representation of heterogeneous sets of buildings, which occur in most virtual city models used in real-world applications and systems. The concept of CLOQ buildings also leads to a systematic method for the incremental refinement of buildings and facilitates the implementation of intuitive modeling tools. Fig. 1 shows an example of a CLOQ building. It forms part of the Berlin 3D city model.

Requirements of the long-term management of virtual city models are concerned with how these models can be maintained and how the process of authoring and customization can be defined. Using CLOQ buildings, for example, an urban planner can create new buildings and incrementally refine existing buildings within a

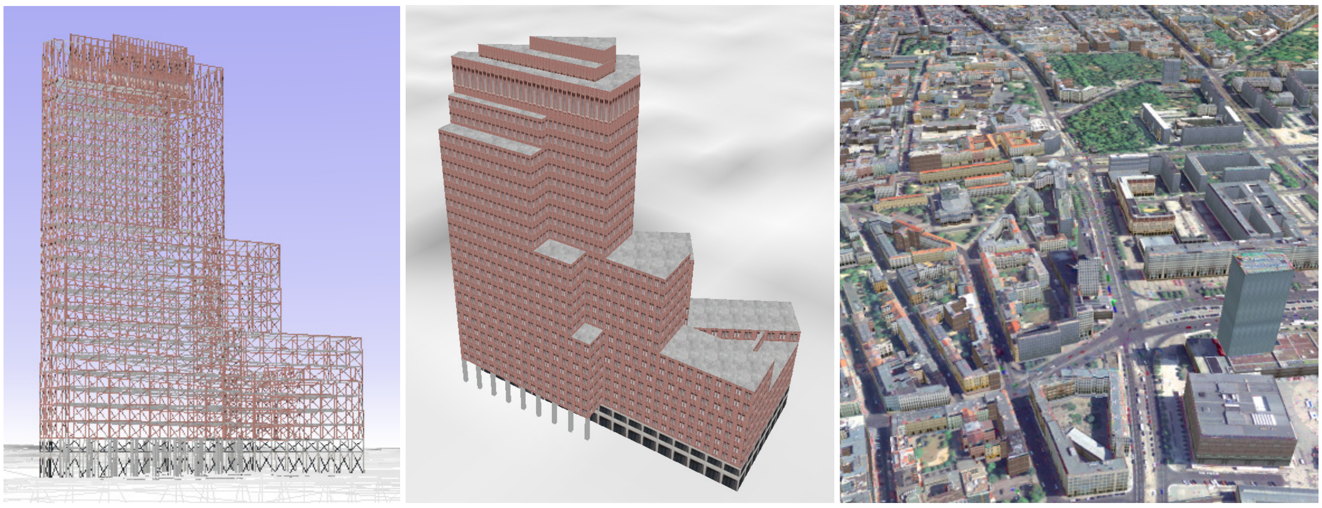


Fig. 1. The Kollhoff building at the Potsdamer Platz represented as CLOQ building for the Berlin 3D city model.

uniform framework and tool set during the whole planning process.

2. RELATED WORK

2.1 Building Data Acquisition

Building models can be systematically derived by a wide range of techniques for acquisition, classification, and analysis of urban data derived from, for example, laser scans, aerial photography, and cadastre information bases. The initial creation is a technically challenging and economically cost-intensive task. For a detailed overview of methods, see Hu et al. [9], Ribarsky et al. [15], Bauer et al. [3], and Förstner [7]. Outputs of these techniques can be directly transformed into CLOQ buildings.

The geometric level-of-quality of buildings derived by automatic techniques is usually relatively low such as in the case of block models, which are encoded by 2D ground plans associated with an height information. Block models are commonly used as initial data basis for virtual 3D city models. CLOQ buildings having only one floor correspond to block models.

For more detailed buildings semi-automatic techniques of photogrammetry [8] are used or they are constructed by CAD or 3D modeling tools. The resulting buildings are represented, in general, using formats of 3D computer graphics (e.g., VRML or 3DS from Studio Max) because these models are supposed to be finalized by 3D graphics design applications. Eventually buildings of highest detail result including indoor and room features. Most notably buildings represented this way do not preserve semantics of building parts unless special conventions are adopted.

Independently, procedural techniques for creating virtual 3D city models have emerged in the scope of computer graphics, intended for research, simulation, and educational purposes. In particular, specific markets such as the movie and game industry have a high demand for a cost and time-efficient creation of realistic, complex urban environments. Their representation, however, is tightly linked to and optimized for a specific visualization system.

Parish and Müller [14] present a system that creates a complete 3D city model using a small set of statistical and geographical input data. The system provides tools for generating roads, allotments, buildings, and procedural textures. Wonka et al. [19] introduce a concept for instant architectural building models. In their approach, building designs are derived using parametric set grammars, an attribute matching system, and a separate control grammar to derive buildings having a large variety of different styles and design ideas. In conclusion, procedural techniques, which are not concentrating on real-world geodata, do not need to address the heterogeneous building models and long-term maintenance.

2.2 City Model Representations

Independent of the way of creation, virtual 3D city models can be exported as 3D scenes in standard 3D scene formats (e.g., VRML, X3D, or 3DS). While scene description languages and scene graph systems offer a broad repertoire of generic graphics functionality, they do not provide specialized means for 3D geodata-based objects. Consequently, it is generally difficult to represent, to preserve, and to take advantage of object semantics.

The CityGML initiative (Kolbe et al. [10]) addresses the need for a domain-specific, semantics-preserving description of virtual 3D city models based on XML. As Altmaier and Kolbe [2] introduce, CityGML supports four different level-of-details:

- Block models derived by extruding ground plans to an average (measured or estimated) height (LOD-1);
- Block models including roof geometry and differentiated heights within a single building (LOD-2);
- Detailed building models with (LOD-3), and
- Architectural building models including indoor and room features such as doors, staircases (LOD-4).

This schema expresses principal quality levels as found in buildings delivered from today's acquisition techniques, but does not allow us to express building models between these discrete levels and does not facilitate the systematic refinement of a building model from quality level to the next.

An integral solution for a continuous modeling across these quality levels would enhance the expressivity, address practically important intermediate quality levels, and enable developing effi-

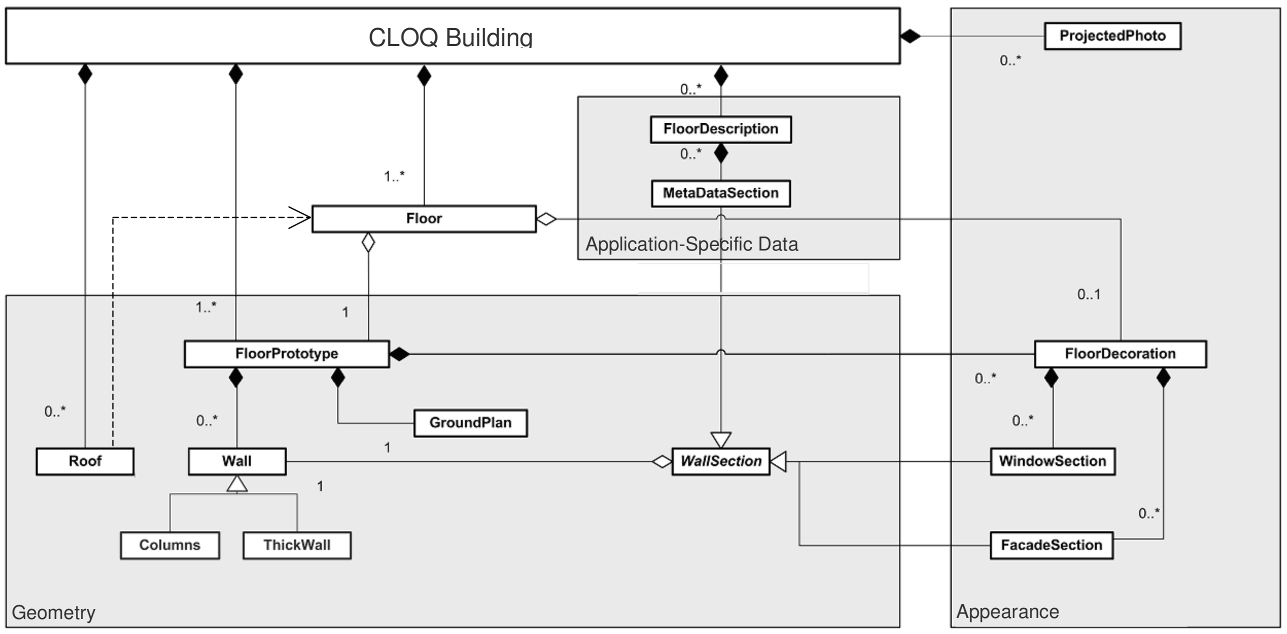


Fig. 2. UML class diagram for CLOQ buildings.

cient cross-LOD rendering and interaction techniques. For example, in many applications users want to transform an LOD-2 building into an LOD-4 building during a refinement process.

Conceptually, there is also no sharp distinction between LOD-2 and LOD-3 buildings: Even for LOD-2 buildings, it can be necessary to add significant geometric details such as an entrance hall if these details are perceptually important – the corresponding quality level could be considered to be between 2 and 3. Similarly, in modern architecture the distinction between indoor and outdoor is softened, e.g., if large parts of a facade are made of glass, a minimal indoor model including floors and main walls would be required for a detailed building model – the corresponding quality level would be between 3 and 4.

The concept of CLOQ buildings extends the CityGML modeling schema adopting the semantic schema but introducing an independent schema for modeling geometry and appearance.

2.3 Real-Time Rendering of 3D City Models

Most applications need to display virtual 3D city models in real time to enable the direct interaction and manipulation by the user. The representation of building models, which tend to occur in large numbers even for small-size cities, is transformed into a 3D scene representation; complex buildings need to be decomposed and transformed into suitable rendering primitives, and the appearance data such as façade images need to be bundled and transformed into graphics textures.

Optimization techniques for real-time rendering (Akenine-Möller and Haines [1]) operate on general graphics primitives and include view-frustum culling, occlusion culling, back-face culling and impostors (Schaufler [16]). Out-of-core visualization techniques are applied to cope with massive data sets stored on external memory (Davis et al. [4]; Lindstrom and Pascucci [13]). Techniques have been described for large-scale virtual environments such as described by Willmott et al. [18] and for large-scale texture data. For example, Lefebvre et al. [12] proposed a GPU-based approach for large-scale texture management of arbitrary meshes.

The modeling schema for CLOQ buildings does not assume any 3D scene representation and optimization technique since both are independent subjects best implemented within the graphics system actually used.

3. CLOQ BUILDING REPRESENTATION

The scope of CLOQ buildings encompasses simple block models, models with partially defined or complete roof geometry, and detailed indoor and outdoor building features.

One the one hand, our goal is to find a minimal set of components that achieve a maximum of expressivity with respect to commonly required building features in virtual 3D city models. On the other hand, the components should allow users to intuitively and efficiently create and refine building models. Furthermore, the implementation of efficient real-time rendering algorithms should be facilitated. CLOQ buildings intend to supply such a framework and provide a unified, compact data structure.

In the following, we introduce the object-oriented model of CLOQ buildings shown in Fig. 2. A CLOQ building represents a single building entity of a 3D city model. It is implemented as an object that is constituted by geometry, appearance, and thematic objects.

3.1 Floors

A CLOQBuilding object is composed of one or more Floor objects, each of which defines the ground plan as well as the walls placed on top of it for a single floor of the building.

A Floor object always refers to a FloorPrototype object, which contains the actual floor specification. We introduce this indirection for two reasons. First, it compactly represents similar floors within a multi-storey building, which represent the majority of buildings in a typical dense urban area. Second, this way we preserve the semantics of floors having the same construction. Floor prototypes enable users to perform refinements efficiently and rendering systems to optimize the internal graphics encoding.

Each floor prototype is defined by its GroundPlan object. It consists of one or more polygons that define the area on which walls can be constructed. Each polygon is defined by its outer loop and optionally inner loops that model holes (e.g., courtyards). We allow for multiple polygons since a floor may consist of several components not necessarily directly linked. By polygon we refer to a 2D shape that not only can consist of vertices joint by straight edges but also can contain curved segments such as B-splines.

A ground plan defines the base plate for walls and, therefore, it is a mandatory object for each floor. It additionally defines its height, that is, the thickness of the base plate. The thickness can be zero in the case of an abstract ground plan or can be positive if the floor should be modeled as solid object. For example, solid ground plans allow us to model protrusion elements such as terraces in a visually convincing but still efficient way.

3.2 Walls

On top of a ground plan we can place Wall objects. A wall represents a vertical, planar polygon that is constrained to directly lie on top of its ground plan.

By default, a wall has a thickness of zero, that is, it is represented as a single polygon. A specialized wall object of type ThickWall, however, defines a positive thickness. Here, the wall is geometrically instantiated as a 3D solid object. Thin walls are sufficient if they form a closed surface and can therefore be seen only from outside, while thick walls are appropriate for those walls whose upper or side parts are potentially visible such as in the case of free-standing walls.

Walls are constrained to be non-intersecting within the same floor. If walls intersect, they have to be split into parts. Walls are not constrained with respect to their height, that is, a wall can have less height as the floor itself or can even be higher. For example, low walls can represent the fronts of a balcony, whereas the sides of a chimney starting at the basement floor can be extended through the roof.

Since CLOQ buildings are primarily intended for visual building models, they do not need validation with respect to the statics of a building.

Walls of one floor can be placed independently of walls from the adjacent floors. If we had to provide an automatic validation with respect to the statics of a building, the wall editing would have to be constrained appropriately. Since CLOQ buildings are primarily intended as visual building models we do not provide this kind of functionality at the moment.

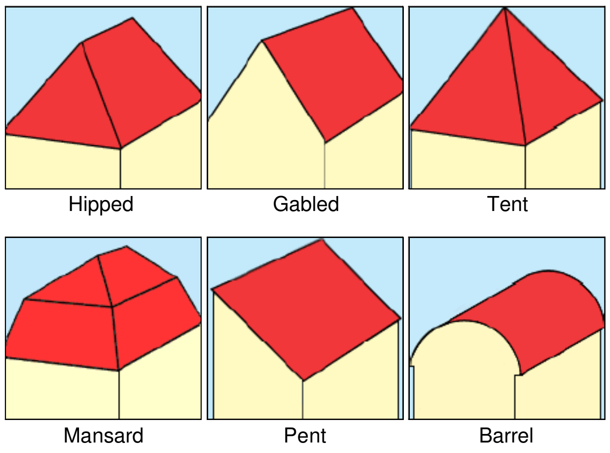


Fig. 3: Examples of roof types.

3.3 Roofs

The simplest roof type is the flat roof, a planar roof geometry situated on the top of the walls of the topmost floor. Many cadastral databases classify roofs according to their principal shape (Fig. 3) as gable roof, hip roof, gamble roof, mansard roof, tent roof, etc.

In CLOQ buildings, the *roof floor* represents a top-most floor used to specify roof geometry. The ground plan of the roof floor is independent of the ground plan of the underlying floor. A characteristic element of most real-world buildings is a roof overhang, that is, the roof floor exceeds the area of its underlying floor.

To model these non-planar roof types, we define for the roof floor a *roof skeleton*, that is, a network of characteristic edges of the roof geometry (Fig. 4). The skeleton can be considered to be a two-dimensional network, specifying at each of its vertices a height. The resulting 2.5-dimensional surface represents the 3D roof geometry.

The skeleton approach has two major advantages: First, it pro-

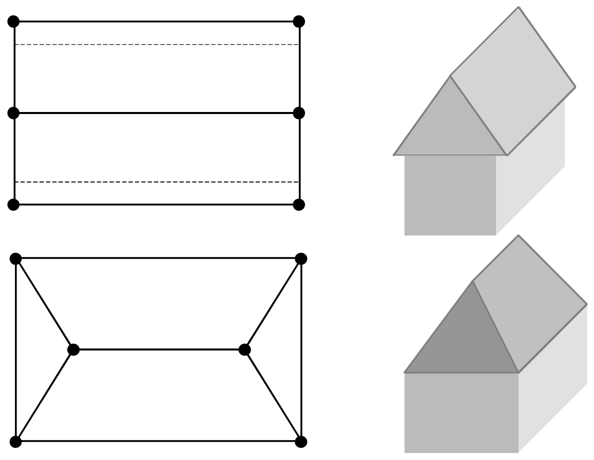


Fig. 4. Example of skeleton for a gabled roof with overhang (a). Example of a skeleton for a hipped roof (b).

vides a systematic and still intuitive approach for editing most common roof types. Second, there are several techniques in computation geometry that allow us to automatically compute a roof skeleton for a given roof floor (e.g., straight skeletons discussed by Felkel and Obdrmalek [6] and extended by Laycock and Day [11] for different roof types).

With the introduced building blocks, floors, walls, and roofs, we are able to express the principal geometry of most buildings as required by virtual city models. This compact object model keeps the implementation as well as the usage of CLOQ buildings simple.

3.4 Floor Decoration

FloorDecoration objects are responsible for specifying the appearance of CLOQ buildings; they are parts of a floor prototype. The strategy of the floor decoration is to assign appearance information to horizontally arranged sections of walls, identified by WallSection objects. These sections can refer to a whole wall or part of it.

The appearance of a wall section can be defined by two types of WallSection objects: WindowSections and FacadeSections. A FacadeSection describes the overall appearance of a wall. One way to define a facade section is the assignment of a facade pattern texture containing windows as well as the surrounding surface material as a single image. The second, more flexible way, is to model the windows explicitly. In this case, the FacadeSection describes only the wall material, and an additional WindowSection defines the positions and appearance of all visible windows separately.

3.5 Projective Textures

Projective textures represent an alternative method for specifying appearance of CLOQ buildings. They are intended for image data captured from real-world buildings and generally are relevant for several walls and wall segments. Projective textures are useful in practice for the rapid photorealistic modeling of the appearance based on digital photography.

A ProjectedPhoto object refers to a 2D texture, e.g., a digital photo taken from the building's facades. This texture is projected into the 3D space, using an auxiliary *projector-wall* that is independent of floors, ground plans, walls, and wall sections. A projector-wall is exclusively used to project a given texture onto those building parts that are geometrically hit by an orthogonal projection originating from the projector-wall towards the building. and explicitly enabled for it. The number of projector-walls and their position and orientation is independent of the building's geometry.

The more procedural approach of defining the appearance based on texturing wall segments is useful if facades show a repetitive pattern and if the appearance is designed such as in the case of planned buildings. Both approaches can be combined within a single CLOQ building since appearance objects can be explicitly activated and deactivated, respectively, for individual parts of the building geometry.

3.6 Application-Specific Building Data

An essential requirement for applications and systems using virtual 3D city models is that application-specific data can be associated with any parts of buildings. For this reason, application-specific data is represented by "first-class objects" similar to appearance data.

FloorDescription objects contain application-specific data assigned

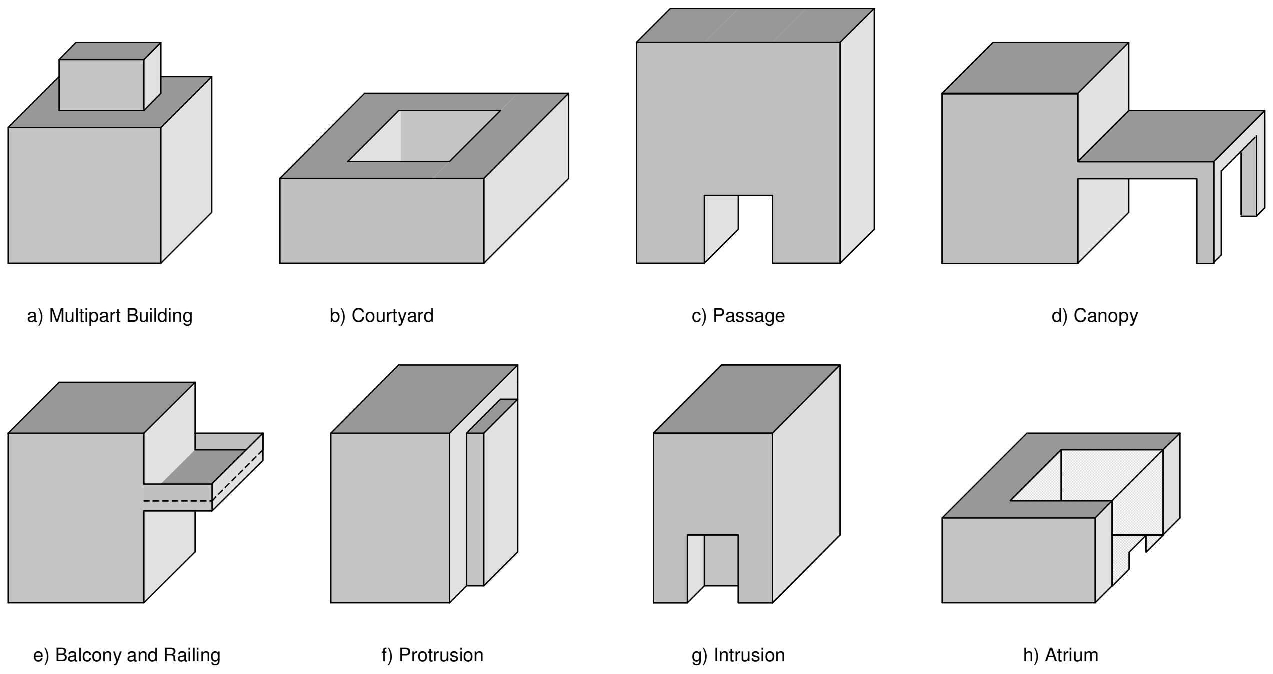


Fig. 5. Examples of supported CLOQ building features.

to individual parts of CLOQ buildings. Each floor description specifies a table that stores values for an application-specific set of key-value pairs. They are related to the floor as a whole.

To assign application-specific data to individual components of floors, a floor description can store MetaDataSections. Similar to floor descriptions, a meta-data section contains a table of key-value pairs. Since meta-data section are specialized wall sections, they can relate the meta-data to specific parts of the walls. For instance, a floor could host multiple shops for which detailed information is stored in meta-data sections.

4. CLOQ BUILDING FEATURES

Although the number of component types is small, CLOQ building are able to express a large collection of common building features. In our approach, we concentrate on those features required by typical applications and systems using virtual city models. In general, these requirements are concerned with the abstract structure of a building and related visual aspects.

Examples of supported common building features (Fig. 5) include:

- *Multipart Buildings*. For each part a separate ground plan is defined with an individual floor structure and height.
- *Courtyards*. They are described by inner polygon loops of the corresponding ground plan. The ground for courtyards can be any floor.
- *Passages*. A passage is specified by cutting the walls of a floor appropriately. If the passage covers more than one floor, the intersected floors have to split their ground plans.
- *Canopy*. A canopy is modeled by extending the ground plan of the upper floor and connecting the roof by columns specified as part of the bottom floor.
- *Balconies and Terraces.* A balcony (or a terrace) is modeled by solid extension of the ground plan together with thick walls along the boundary of the balcony.
- *Railings*. To model a railing, we construct a wall whose appearance is defined by a transparent texture of a railing. This way, only opaque parts of the texture become visible. Of course, this technique applies to all “flat features” of a building but is limited because from close views the simplified geometry becomes apparent.
- *Protrusions and Intrusions*. These features can be modeled simply by varying the ground plan of a floor. For example, entrances are typical intrusion features in buildings.
- *Indoor Structures*. Rudimentary indoor structures such as interior walls, doors, atriums, and stairways can be modeled since ground plans support complex polygons and are not restricted to outside walls.

Creative users might be able to derive more features. A number of features need to be explicitly supported, e.g., staircases. Nevertheless, features represented in CLOQ buildings are still intended as abstractions. Architectural models may need to provide more details.

5. EDITING CLOQ BUILDINGS

The concept of CLOQ buildings concentrates on principal parts of a building including roofs, floors, and facades. This allows us to capture a large bandwidth of building types and facilitates the design of intuitive authoring tools for creating, manipulating, and refining CLOQ buildings.

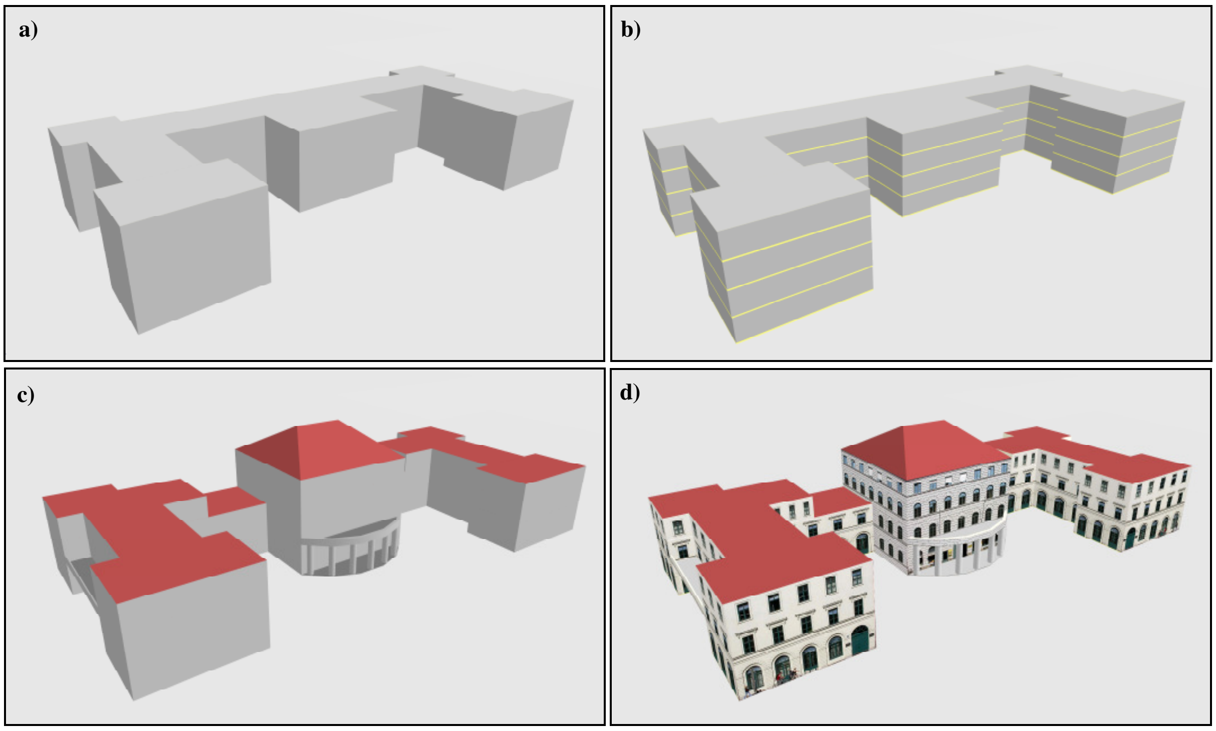


Fig. 6. Transforming a block model into a refined CLOQ building. a) Block model. b) CLOQ building split into floors. c) Geometry refinement. d) Appearance refinement.

From Block Models to CLOQ Buildings

To illustrate a typical use case, assume that an initial block model should be refined. First, it is transformed into a simple CLOQ building by splitting the model into a number of floors having the same ground plans and outside walls. Fig. 6 shows the refinement process of a block building into a CLOQ building:

1. The initial block model results from extruding 2D ground plans (Fig. 6a).
2. The model is subdivided into floors according to a list of floor heights (Fig. 6b).
3. The top floors are modified to distinguish different building parts, roof geometry is added to the top floor of the center building part, the basement floor is enhanced by columns, and a balcony is added to the left part (Fig. 6c).
4. Façade textures are added, which can be specified either by composing different texture patterns or as projective textures, e.g., using digital photography (Fig. 6d).

The example shows a frequent requirement in applications based on 3D city models: The existing 3D city model needs to be partially and incrementally developed according to current project goals or management decisions.

5.1 CLOQ Building Editor

For the design of the CLOQ building editor we assume that non-expert users (e.g., non-architects) should be able to construct and refine CLOQ buildings. Since floors are the dominant conceptual elements, the 2D floor editor is the core component.

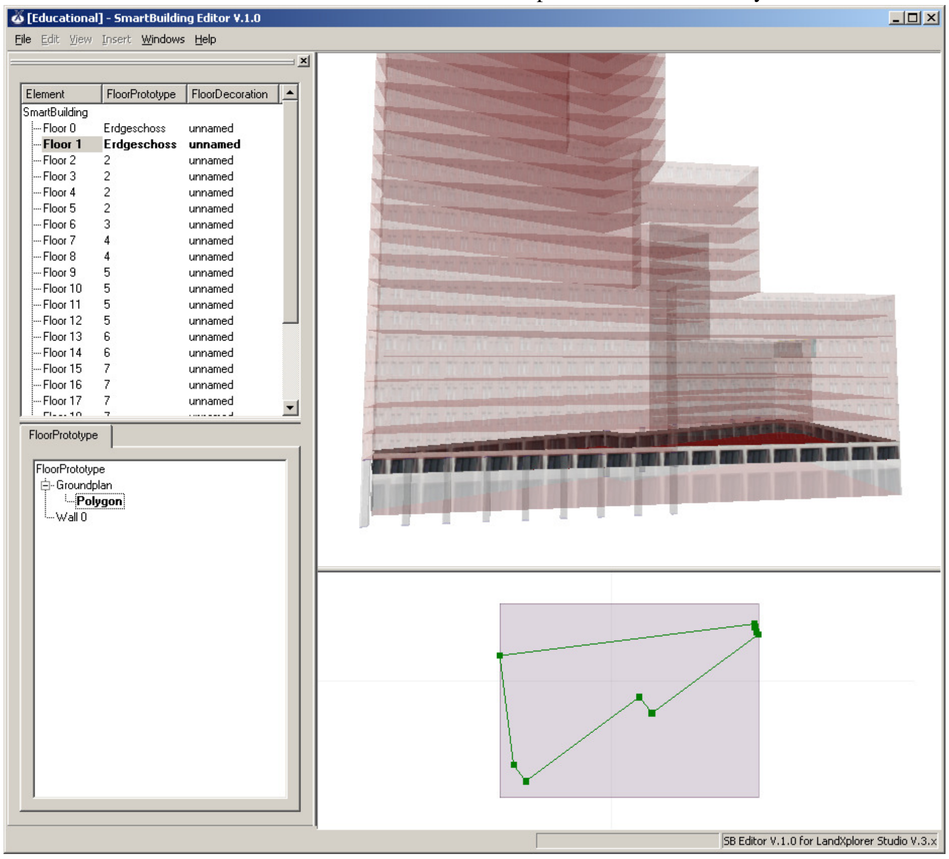
Fig. 7 shows a snapshot of the CLOQ building editor of our implementation. The object tree (Fig. 7 top-left) lists the Floor objects and indicates the corresponding floor prototypes. The selected floor prototype can be edited by the 2D editor widget (Fig. 3 bottom-right). The user can directly manipulate ground plan polygons, walls, and wall sections. Changes are immediately reflected by the 3D view of the CLOQ building.

Most frequent operations include adding and modifying floors, walls, and wall sections. For instance, it is easily possible to extend a CLOQ building by a penthouse by replicating its top floor and reshaping the ground plan of the new floor. A selected floor may also be enhanced by adding geometric details such as columns. To integrate application data, e.g., user can specify a shop window-texture as a section of a wall. To edit a wall section, the user specifies the start and end points of the section on the ground plan polygon.

5.2 Editing Ground Plans and Walls

In the simplest case, the base plate of a floor is completely hidden by surrounding walls. If a building contains a terrace or if two subsequent floors have different ground plans, the base platform becomes partially visible (such as in Fig. 6c and 6d).

Walls are specified by polylines and height values. Singular line segments of a polyline can be replaced by arcs to model curved shapes. Most walls are only visible from outside and, therefore, do


[Educational] - SmartBuilding Editor V.1.0
File
Edit
View
Insert
Windows
Help
FloorPrototype
SB Editor V.1.0 for LandXplorer Studio V.3.x

Element	FloorPrototype	FloorDecoration
SmartBuilding		
Floor 0	Erdgeschoss	unnamed
Floor 1	**Erdgeschoss**	**unnamed**
Floor 2	2	unnamed
Floor 3	2	unnamed
Floor 4	2	unnamed
Floor 5	2	unnamed
Floor 6	3	unnamed
Floor 7	4	unnamed
Floor 8	4	unnamed
Floor 9	5	unnamed
Floor 10	5	unnamed
Floor 11	5	unnamed
Floor 12	5	unnamed
Floor 13	6	unnamed
Floor 14	6	unnamed
Floor 15	7	unnamed
Floor 16	7	unnamed
Floor 17	7	unnamed

```
FloorPrototype
 Groundplan
   Polygon
 Wall 0
```

Fig. 7. The CLOQ-building editor of the LandXplorer 3D city model system.

not need to be solid. Freestanding walls, however, such as the boundaries of a terrace, can be seen also from above or from an indoor perspective. In this case, thick walls are used. The special case of ColumnWalls allows for the alignment of a row of columns along the wall's polyline. The height of walls can be automatically determined by the distance to the next floor or be specified explicitly.

Walls together with appropriate textures can be instrumentalized to model railing, cutouts, or interior decorations such as paintings. In general, the textures applied to such a wall will be mostly transparent. Although walls are planar objects, it provides a straightforward method to incorporate those elements for visualization and illustration purposes.

5.3 Editing Floor Decorations

CLOQ buildings support two techniques of façade texturing:

- Projective textures: These textures are orthogonally projected onto a building regardless of the geometry structure of the building. Conceptually, for each projective texture we define an invisible ghost-wall, which serves as projector wall. Projective textures are effortless to assign for complex building shapes and, therefore, can be used for the rapid modeling of existing buildings.
- Composition of texture patterns: For individual sections of a building façade, we can specify a material texture and a window texture using wall sections. Each wall section belongs to a certain wall object and specifies a range on this wall. Using catalogues of standard materials and window types, wall sections for a façade can be edited instantaneously.

The advantage of projective textures is that they can directly map facade data captured by digital photography. In contrast, the composition of texture patterns frequently models typical, but non-authentic facades. It does not involve the problem of occluded facade parts, e.g., by trees in front of the facade, and allows for ad-hoc modeling of buildings that are only planned or proposed.

5.4 Assigning Application-Specific Data

Floor descriptions provide means to integrate application-specific data into CLOQ buildings. To represent the data, CLOQ buildings use generic two-column, multi-row tables, called attribute tables. An attribute table stores key-value pairs. Both, keys and values, are formatted as strings, which can contain textual, categorical, and numerical contents.

For a single CLOQ building we can specify general attribute tables such as address, owner, usage, etc. for the whole building. In addition, attribute tables can be specified for individual floors and for individual wall sections.

Meta-data sections are a special form of wall sections that specify attribute tables (instead of appearance attributes). In contrast to wall sections for textures, floor attribute tables and wall sections are not defined for floor prototypes but for each floor separately. For instance, in an office building, each floor could be rent by a different company although the appearance of all floors would be equal.

6. CONCLUSIONS

CLOQ buildings provide a continuous level-of-quality and aim at efficient solutions for the incremental development of buildings in virtual city models. Due to their per-floor construction, they can be perfectly handled by direct-manipulation interfaces, providing intuitive tools for building refinement. CLOQ buildings address the main use case in virtual city model applications, the project-driven and event-driven customization and reengineering of city model components.

The CLOQ building concept has been implemented as a part of the LandXplorer system, an authoring and presentation tool for 3D city models. We have observed in a variety of use cases that with CLOQ buildings we can approximate complex building models in a time efficient way. They are also suited for large-scale 3D city models, and they can be mapped to an internal graphics representation that allows for real-time photorealistic and non-photorealistic rendering (Döllner et al. [5]). Of course, CLOQ buildings are not intended to substitute architectural modeling approaches because they do not consider issues of statics and detailed building layouts but serve as GIS representation schema.

One application example is a decision-support system in urban planning. Using CLOQ buildings, proposed changes can be interactively performed within a given virtual 3D city model, so that the effect of the modification can be evaluated and discussed immediately. Another application is concerned with managing interactive 3D location plans. Using CLOQ buildings, building models that exhibit characteristic exterior and interior features are created and maintained by the CLOQ-building editor.

As next steps, we are working on analyzing and mapping arbitrary CityGML-based building models to CLOQ buildings. We also would like to investigate high-level operations for CLOQ buildings (e.g., adding penthouses, constructing roofs; drilling courtyards, designing entrances, etc.) and using constraints to assist the construction and refinement process. We also expect that CLOQ building could be a powerful intermediate representation for authoring tools based on CityGML.

Acknowledgements

We would like to thank our colleagues of the LandXplorer system (www.3dgeo.de), which has been used as implementation platform. We would like to thank the German environmental foundation Deutsche Bundesstiftung Umwelt for supporting our work within the Lenné-3D research project (www.lenne3d.de).

7. REFERENCES

[1] Akenine-Möller, T., Haines, E. 2002. *Real-Time Rendering*. A K Peters Ltd, 2nd Ed.

[2] Altmaier A., Kolbe, T.H. 2003. Applications and Solutions for Interoperable 3D Geo-Visualization. *Proceedings of the Photogrammetric Week 2003*, Wichmann Verlag.

[3] Bauer J., Klaus A., Karner K., Schindler K., Zach C. 2002. MetropoGIS: A Feature based City Modeling System, *Proceedings of Photogrammetric Computer Vision 2002 (PCV02) - ISPRS Comission III Symposium*, Graz, Austria.

[4] Davis, D., Ribarsky, W., Jiang, T.Y., Faust, N., Ho, S. 1999. Real-Time Visualization of Scalably Large Collections of

Heterogeneous Objects. *Proceedings of IEEE Visualization 1999*, 437-440.

[5] Döllner, J., Buchholz, H., Nienhaus, M., Kirsch, K. 2005: Illustrative Visualization of 3D City Models, *Proceedings of SPIE - Visualization and Data Analysis 2005* (VDA), San Jose, CA, USA, 42-51.

[6] Felkel, P., Obdrmalek, S. 1998. Straight Skeleton Implementation. *14th Spring Conference on Computer Graphics*, 210-218.

[7] Förstner, W. 1999. 3D City Models: Automatic and Semiautomatic Acquisition Methods. *Proceedings Photogrammetric Week,* University of Stuttgart, 291-303.

[8] Haala, N., Brenner, C. Laser Data for Virtual Landscape Generation.

[9] Hu, J., You, S., Neumann, U. 2003. Approaches to Large-Scale Urban Modeling. *IEEE Computer Graphics and Applications*, 23(6):62-69.

[10] Kolbe, T. H., Gröger, G., Plümer, L. 2005. CityGML – Interoperable Access to 3D City Models. *First International Symposium on Geo-Information for Disaster Management GI4DM*, 2005.

[11] Laycock, R.G., Day, A.M. 2003. Automatically Generating Roof Models from Building Footprints. *Proceedings of WSCG*, Poster Presentation, 2003.

[12] Lefebvre, S., Darbon, J., Neyret, F. 2004. Unified texture management for arbitrary meshes, INRIA Research Report No. 5210.

[13] Lindstrom, P., Pascucci, V. 2002. Terrain Simplification Simplified: A General Frameworkfor View-Dependent Out-of-Core Visualization, *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239-254.

[14] Parish, Y., Müller, P. 2001. Procedural Modeling of Cities. *Computer Graphics (Proceedings of ACM SIGGRAPH 2001)*, 301-308.

[15] Ribarsky, W., Wasilewski, T., Faust N. 2002. From Urban Terrain Models to Visible Cities. *IEEE Computer Graphics and Applications*, 22(4):10-15.

[16] Schaufler, G. 1998. *Rendering Complex Virtual Environments*. Dissertation, University of Linz.

[17] Tanner, C.C., Midgal, C.J., Jones, M.T. 1998. The Clipmap: a Virtual Mipmap. *Computer Graphics (Proceedings of the ACM SIGGRAPH 1998)*, 151-158.

[18] Willmott, J., Wright, L.I., Arnold, D.B., Day, A.M. 2001. Rendering of Large and Complex Urban Environments for Real-Time Heritage Reconstructions. *Proceedings VAST 2001: The International Symposium on VR, Archaeology, and Cultural Heritage*, 111-120.

[19] Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W. 2003. Instant Architecture. *Computer Graphics (Proceedings of ACM SIGGRAPH 2003)*, 669-677.