Usage, Usability, and Utility of 3D City Models, 02006 (2012)
DOI: 10.1051/3u3d/201202006

All range and heterogeneous multi-scale 3D city models

S. He[1], G. Besuievsky[2], V. Tourre[1], G. Patow[2] and G. Moreau[1]

[1]*LUNAM Université, École Centrale de Nantes, CERMA, Nantes, France*
[2]*Geometry and Graphics Group, Universitat de Girona, Spain*

Abstract. 3D City Models (3DCM) are key features into decision making of several urban related problems. Therefore 3DCM are needed by several applications, but the required level-of-detail (LoD) of the model depends on the application. Our goal is to propose a multi-scale 3DCM production and use method. Our approach consists of merging, procedural modeling, graph rewriting techniques, and a generalization technique to handle all different kinds of LoD of a 3DCM. In this way, it allows to handle various heterogeneous LoDs of a complete urban city model. We test our proposal with the 3DCM of the City of Nantes for a rendering application. Our results can also be applied to other LoDs criteria to match other 3DCM-based needs.

1. INTRODUCTION

3D city models (3DCM) are fully recognized as useful tools in several urban usages, and these models are used in several fields related to urban planning: decision-making processes, analysis of city characteristics, architectural and urban design, simulations of physical phenomena, information or scientific visualization, official communication, and so on.

Besides the problem of a unified data format to share the 3DCM between these various applications, each application needs specific geometric data in order to perform its own process, and sometimes a single application needs the same city model at various scales. In order to avoid a costly multiplication of several models of the same city for each specific need, the 3DCM can be adapted to each application by using a multi-scale 3DCM.

The production of a multi-scale 3DCM implies a multilevel modeling and a multi-representation of the city model; so different representations of 3D city models at different Levels of Detail (LoDs) are required.

There are two ways of obtaining a 3D model at a given LoD: modeling from scratch or deriving from existing models. Modeling depends on data acquisition techniques or production techniques, whereas derivation is confined by the starting models. Photogrammetry and laser scanning technologies are usually used to capture 3D urban data. These techniques are very efficient to obtain detailed models but there is too much data to remove to obtain low precision model. Procedural modeling is used to produce data according to specific rules. Recent research shows it possible to fit rulesets used for procedural modeling to existing datasets (e.g. images and LIDAR data), so procedural modeling is a technique that is now able to model any kind of urban environment, from modeling synthetic cities to represent accurate existing ones. This method can produce 3DCM with details efficiently but lacks to control the amount of data generated. In general, LoD can be manually set at the rules to have local control density. At the other end of the processing scheme, generalization techniques are used to simplify existing data. They allow to simplify the model but precision depends on input data.

Our goal is to provide a multilevel modeling framework able to simplify an existing model and to add precision when it is needed in order to obtain a multi-scale 3DCM. Once produced, this model can give an application the desired LoD by dynamically changing the scale. Moreover the model precision can be heterogeneous, as some applications need various LoDs in the same model.

Our approach is to combine in the same pipeline a procedural modeling technique and generalization techniques. The framework can start with not-so-precise data (like GIS cadastral data), then add precision with some given ruleset onto one model area, and finally allows to simplify the model onto another area. The framework can collapse different discrete LoD in order to give any kind of precision. As it handles very different LoD, we could argue that we are approaching the concept of all-range LoD into 3DCM.

Section 2 presents previous works on LoD techniques into 3DCM. Section 3 gives an overview of our framework and section 4 explains details about this framework. Section 5 shows our first results with a rendering application. Section 6 brings up a discussion on usages of LoD into 3DCM before conclusion.

2. LOD HANDLING INTO 3DCM

Providing detailed building models is very important for applications involving spatial analysis and realistic visualization. Actually, the more detail provided, the greater the amount of geometrical information available. However, due to computer limited storage and memory resources, building models have to be reduced in complexity in order to have an efficient treatment. Level-of-detail techniques have been largely developed in computer graphics with the aim of reducing geometry since the first work presented by Clark [1], followed by the seminal work by Luebke and Erikson [2]. The interested reader can refer to the book by Luebke et al. [3] for a more exhaustive and complete survey of LoD techniques.

As far as a urban models are concerned, a few LoD proposals come from the use of a semantically well-defined dataset structure, as for example the CityGML schema [4]. CityGML differentiates between five consecutive LoD-levels, where objects become more detailed with increasing LoD regarding both geometry and thematic functionality differentiation. They range from a coarsest level in LOD0, which is essentially a two and a half dimensional digital terrain, to full interior structures like rooms, stairs, and furniture in the top level LOD4. These different LoDs often arise from independent data collection processes, and facilitate efficient visualization and data analysis. In that approach, the different levels of detail are pre-made and no smooth transition is allowed. However, the idea of using defined semantics for building construction may improve general LoD approaches. We include in our method semantics as a base to select the important structures that must be specially preserved. Furthermore, we explore the combination of semantic LoD with standard polygon reduction using different error criteria.

We can categorize the existing LoD production techniques into three groups: controlled-generation methods, detailed-to-coarse methods, and integration of different LoD objects.

2.1 Controlled-generation methods

Contrary to other methods that start from a highly-detailed city model, controlled-generation methods derive higher LoDs from lower ones, for example, extruding 2.5D building blocks from 2D cadastral maps. In fact, the majority of existing 3D city models are started derivation them by extrusion. The requirement of coverage can be fulfilled, but it is still inadequate for the production of full range LoD representation due to the lack of detail. In particular, procedural modeling is a very flexible technique which can be used for modeling from scratch, but can be also used for deriving finer LoDs from coarser ones.

One of the first procedural techniques describing 3D city generation was introduced by Parish and Müller [5], based on the idea of L-systems[6]. Automatic LoD-generation is obtained by starting from the building envelope as an axiom. The output of each rule iteration represents a refining step in the building generation. A similar approach, but using a template-based street network generation method was presented by Sun et al. [7]. Procedural methods have also been extended to the synthetic generation of buildings [8–10] and to the creation of buildings and facades imitating real-world structures [11]. This means that, currently, procedural modeling techniques have the same expressive power and accuracy as any other traditional technique, with the additional practicality for manipulation. Chen et al. [12] proposed using tensor fields to guide the generation of street graphs. However, these approaches require the explicit specification of a set of rewriting rules and usually the model is generated at once, which limits editing. Later, Lipp et al. [13] presented an interactive editing environment for procedural buildings, but their system presented some usability issues related to the fact that they continued using a text-oriented rule-based paradigm. On the other hand, the **skyline**Engine system [14] is based upon an interactive flow-oriented metaphor that is both simpler to use and to understand. This idea was later extended to buildings [15] using a visual-editing metaphor plus a graph-based approach, which proved to be effective for design and automatic processing. In these approaches LoDs are generated by executing a predefined, fixed number of rules which defined the current level of detail. The Unreal Development Kit [16] provides a solution intended for artistic development. It uses a graph-based visual paradigm and has a specific command that uses a render-to-texture technique that allows the user to specify a level from where the geometry is not generated but rendered with a properly combination of textures in order to get a realistic view. Cullen and O'Sullivan [17] propose a method for procedural city generation based on an object oriented shape grammar and a parallel geometry cache strategy to maintain a high frame rate. The object oriented building design allows to use LoD specification, but as happens in CityEngine [18], each level is specified manually. For more information, we recommend the interested reader the survey from Kelly and McCabe [19] or the more recent ones by Watson et al. [20] or Vanegas et al. [21].

2.2 Generalization

Starting from finer LoDs, generalization is the main technique used to derive coarser LoDs by removing unimportant details. In 3D computer graphics, a great number of simplification algorithms have been developed [22, 23] and can be used to generate LoD models. These algorithms have shown great capabilities of reducing geometrical complexity of generic 3D objects, but they can hardly be adapted for city objects, like buildings which tend to be a large collection of independent low polygon-count objects. In geographic information science, along with simplification, generalization encompasses many other concepts, such as selection, aggregation, typification, and symbolization, thereby being more capable of handling relationships and semantics. However, comparing with 2D generalization, which has a long history in cartography [24], 3D generalization is still in its infancy. As far as 3D cities are concerned, the generation of different LoD models is not only aiming at a lower computational cost, but also at proper abstraction of underlying information adapting to different scales, so that users can get a better perception favoring their tasks.

Among all city objects, 3D generalization has been mainly applied to terrain and buildings. Terrain geometry can be easily generalized for smooth, view-dependent rendering, as connected triangular irregular networks (TIN) and grid structures can be simplified using many techniques [25]. On the other hand, buildings, as salient city objects, draw the most attention of researchers [26, 27].

Many researchers addressed single building generalization. Thiemann and Sester [28] proposed to segment a building into meaningful parts, and to decompose the whole generalization process into segmentation, interpretation and generalization phases. Mayer [29] and Forberg [30] developed scale-space techniques for simplifying buildings partly based on the opening and closing morphological operators. Kada [31] proposed to define parts of simplified buildings as intersections of half-planes and

to divide buildings into cells and to detect features by primitive instancing. The work of Rau et al. [32] focused on collapsing faces from known constructive structures as walls and roofs. These algorithms are based on pure geometric models, mainly dedicated to feature detection and segmentation. By taking semantic information into account, Fan et al. [33] proposed a method for generalization of 3D buildings modeled by CityGML, showing that good visualization properties could be obtained by only using the exterior shell of the building model.

Generalization of building groups is less studied. Anders [34] proposed an approach for the aggregation of linearly arranged building groups, without considering semantic information. Guercke et al. [35] studied the aggregation of LoD1 building models in the form of Mixed Integer Programming (MIP) problems. He et al. [36] proposed a generalization of LoD2 building groups by translating the 3D generalization task into 2D footprint scope issues with the support of semantic attributes. They proposed to partition a building model into meaningful units suitable for generalization, and then to divide such unit into *Top* + *Body* to facilitate roof generalization.

Generalization at city scale starts to get attention. Glander and Döllner [37] proposed cell-base generalization by maintaining a hierarchy of landmarks, which are preserved. In the work of Mao et al. [38], buildings are divided into clusters by the road network and grouped with their close neighbors in each cluster. Chang et al. [39] presented a large scale simplification approach based on "urban legibility" intended for better preserving understandability for complex urban spaces at all levels of simplification. Guercke and Brenner [40] proposed a framework for the generalization of 3D city models, within which custom feature types and generalization algorithms can be defined. Glander et al. [41] discussed the concepts for automatic generalization of virtual 3D landscape models.

2.3 Integration

Integration of different LoD objects is a complementary method to the above-mentioned ones for the production of multiple representations of 3D city models. A 3D city model may be composed of individual objects at different LoDs, thereby yielding different representations of the city. For instance, some landmark buildings require to be highlighted with more details, while block models would be enough for the others. On the other hand, 3D city objects are usually modeled separately. Integration of these models is inevitable, what raises the issue of interoperability and consistency due to the fact that existing models are usually made by different organizations for different purposes with different accuracy, and in different formats. Thus, the degree of complexity of integration highly depends on the available source data.

3. FRAMEWORK OVERVIEW

In order to obtain multi-scale 3D city models at different LoDs, we propose a framework using a combination of available techniques: extrusion, generalization, integration, and procedural modeling.

From now on, when talking about LoDn, we refer to the LoD denotations from CityGML standard [42]. LoD0 is essentially a 2.5D Digital Terrain Model. LoD1 is the well-known blocks model comprising prismatic buildings with flat roofs. A building in LoD2 has differentiated roof structures and thematically differentiated surfaces; LoD3 denotes detailed architectural models. LoD4 completes a LoD3 model with interior structures. In addition, another three LoDs (e.g. LoD0A) will be introduced to describe more inter-level cases. We indicate the complexity of an area by the number of polygons.

Thanks to the availability of sophisticated cadastral database, LoD1 buildings with extensive coverage can be derived from their footprints and associated heights. Although geometrical complexity of each LoD1 building is already very low, information density will be remarkably raised up when they come in a large quantity all over the city. Therefore, LoD1 buildings still need to be generalized, so as to give a better abstraction adapting to the scale.

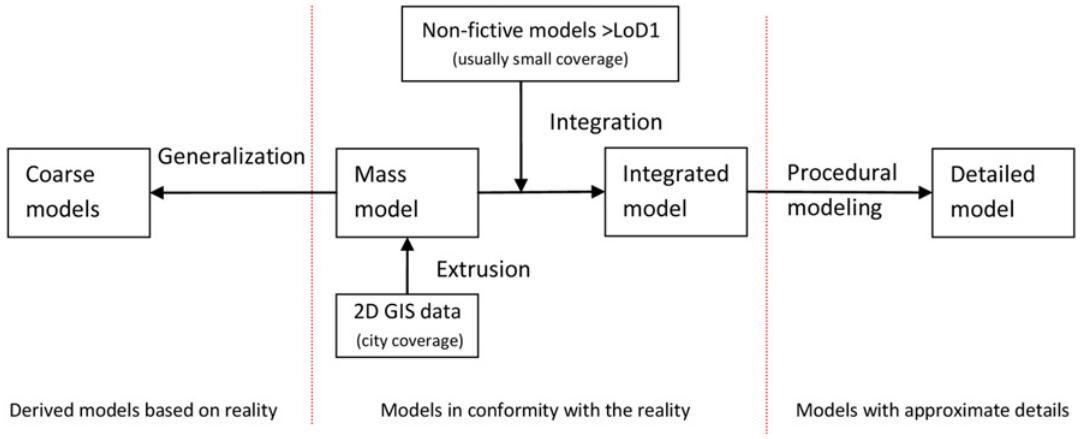


Figure 1. The main framework for producing full range representations of 3D city models.

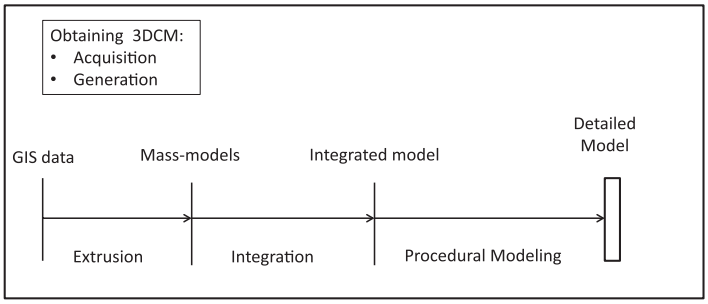


Figure 2. Generating the city model from data to full geometry model.

On the other hand, LoD1 buildings are insufficient for the tasks on small scale, like street view navigation. A feasible solution is to integrate more detailed models from other sources. Setting aside the issue of interoperability and consistency, integration still cannot promise detailed models covering the whole city. Procedural modeling is a promising technique for geometry detailing, especially in terms of the level of automation. High level of automation in city scale detailing however can not ensure that the added details are authentic. But having only detailed structures in fiction, a 3D city model is still capable of providing plenty of GIS services, additionally with enhanced visual impacts.

The proposed framework for producing full range representations of 3D city models is depicted in Figure 1. Four different techniques are applied to handle different initial models and to derive different LoDs. Extrusion and integration have no difficulty in using data conforming to the reality, and generalization is performed on such models. Procedural modeling may introduce fictional details that are similar to the real objects.

4. FRAMEWORK DETAILS

The two main phases that compose our framework are the city model generation, from data to detailed 3D geometry (Figure 2), and the obtaining of the multi-scale representation (Figure 4). We describe both of them in this section.

4.1 Obtaining 3DCM

4.1.1 GIS processing and extrusion

Our technique starts from an input GIS model (e.g. cadastral data provided by the local city council) and extrudes it to obtain the 3D mass-models. Quite often, this GIS information is very good, reliable and well formatted. However, there are situations where we find that a piece of cadastral GIS data is plagued with errors and inconsistencies, being much unreliable for any sort of 3D reconstruction. This is the reason why important companies like Google Inc. and Microsoft ® use aerial and street-side imagery as sources. However, we realize that, in many cases, cadastral GIS data is the only existing, cheap, and available source of information. To address this problem we have integrated in our system the cleaning and structuring algorithm by Pueyo et al. [43], which guarantees that the GIS data is not only usable, but ready for 3D extrusion.

4.1.2 Integration

It is not difficult to generate LoD1 building models covering the whole city with a sophisticated cadastral map, which can also ensure that the generated city model is consistent with the actual city in terms of topology and measurements. Integrating with other available city objects in 2D, like roads, water bodies and vegetation, a city model can be enriched. Integrating with other 3D building models at higher LoDs, the level of detail of the underlying city model can be increased. The authenticity of the integrated city model can be assured by using the 3D models produced from survey data (e.g. acquired by photogrammetry and laser scanning), or based on architectural drawings. However, such detailed models are usually only available for disconnected small areas. It is difficult to obtain detailed models covering the whole city using integration alone.

4.1.3 Procedural Modeling for detailing

From the previous processing and extrusion step we obtain the city model as a collection of mass-model buildings. For detailing the model with all architectural element structures as windows, balconies or doors, we based our representation on a procedural modeling of buildings approach [9]. Please, observe that we preserve the roofs obtained in previous stages, as our procedural techniques are applied to the facades of the buildings only. Procedural urban modeling follows the basic principles of a shape grammar, which main concept is based on a rulebase: starting from an initial axiom shape (e.g. a building outline), rules are iteratively applied, replacing shapes with other shapes. The whole production process can be seen as a graph where each node represents an operation applied to its incoming geometry stream and the leaf nodes are the geometry assets. This representation results in a Directed Acyclic Graph (DAG), where nodes denote rules and the links the stream of geometry [44]. Figure 3 shows a building example and its rules using a visual graph-based rule model. Observe that in our current implementation we use procedural modeling to add detail to our buildings, but it would be possible as well to extract these rules directly from measured data using the techniques by Musialski et al. [10] or Müeller et al. [11], accurately recreating a real urban environment.

4.2 Obtaining multi-scale models

4.2.1 Geometry simplification

We design a simplification method based on two considerations for model reduction: the possibility to simplify any shape in the hierarchical ruleset automatically according to view-dependent criteria, and the possibility to select specific interesting shapes that may be important and where a more detailed representation would be required.

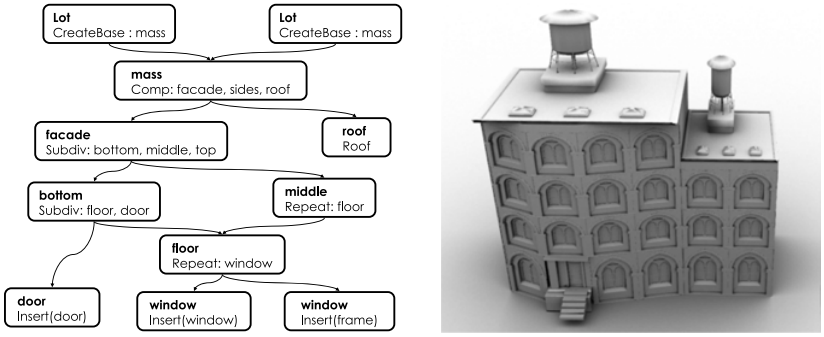


Figure 3. A graph-based model of rules (right) is used to obtain a building model (left). Rule parameters are here omitted.

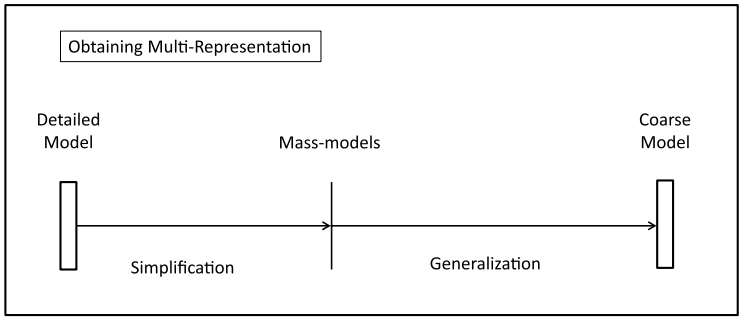


Figure 4. Obtaining multi-representation.

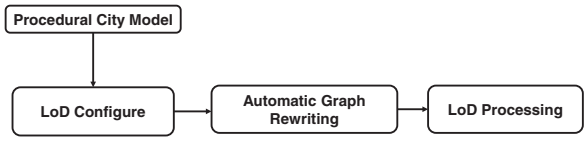


Figure 5. LoD processing for city detailed models.

From a building model designed procedurally using rules as shown in Figure 3, the LoD reduction criteria is configured through a script according to the user needs that may include view-dependent requirements or even semantic labels based on the relevance of the geometry that needs to be kept with more detail. From this configuration, the graph model is automatically rewritten into a new one. Finally, the geometry-reduced model is obtained by processing the resulting graph (see Figure 5). We consider two levels of simplification, an asset level for detailed geometry, and a higher level where a whole facade can be simplified.

Asset level

In order to process the corresponding graph transformation for a given LoD configuration we introduce new commands to the standard CGA procedural modeling: a *ConditionalLoD* rule, that evaluates the reduction that must be applied to the products from a given quality criteria, and the *InsertLoD* command that is responsible for the insertion of geometry assets at the corresponding level-of-detail.

The *ConditionalLoD* command introduces a flexible way to evaluate how much the shapes need to be simplified. From quality criteria, it decides whether a shape must be maintained at full resolution,

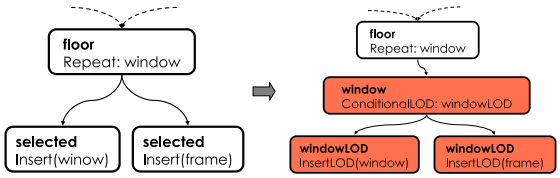


Figure 6. Asset-level graph transforming. The original graph (left) is transformed by grouping *Insert* rules that shares the same predecessors and creating a *ConditionalLoD* rule for each group properly connected to each *InsertLoD* of the group (right).

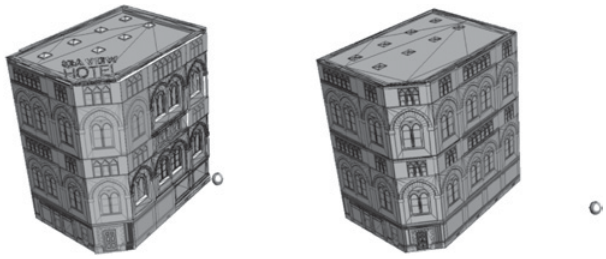


Figure 7. Higher-level geometry replacements example with textures set at facade levels of a building. From a closer viewpoint (left) some assets are not reduced (see darker windows). From a distant viewer (right) all geometry facades are replaced, each one by a single textured polygons.

reduced, or represented as a texture. In the case of reduction, it also computes the percentage of polygons that must be kept. The evaluation is performed over the shape product to be replaced, and then applied on the geometry of the corresponding asset using the *InsertLoD* command. We develop different simplification criteria based on the distance to a virtual viewpoint and the angle orientation of the products. The reduction criteria is configured manually through a script according to the user needs. The criteria evaluation returns an integer factor, ranging 0 to 100, that represents the percentage of polygons that must be preserved by the polygon reduction engine. The *InsertLoD* command is responsible for the replacement of geometry assets at different reduction levels. It acts within a switch-case scheme according to the three different kinds of replacements that can be obtained: full, reduced or texture.

Our simplification technique it is based on a graph-transformation technique of the original ruleset graph-model. The main transformation consists on replacing each *Insert* rule of the original graph by a pair of sequential *ConditionalLoD* - *InsertLoD* rules (see Figure 6).

Facade level

Our system also works on a higher-level replacement of larger blocks of elements, like for example replacing a whole facade of a building with a single texture polygon. If all assets of a given rule are being simplified completely to texture, then the whole rule is considered as a single polygon (see Figure 7).

4.2.2 City scale generalization

A modern city usually consists of hundreds of thousands of buildings, which will lead to a high level of information density and computational complexity as well, even though they are modeled at LoD1. Therefore, further generalization is needed especially at city scale. Since a major problem of 3D city models is the huge amount of composing objects, we believe the divide-and-conquer strategy can be adopted, and road network can be regarded as a natural partition of a city [37, 38].

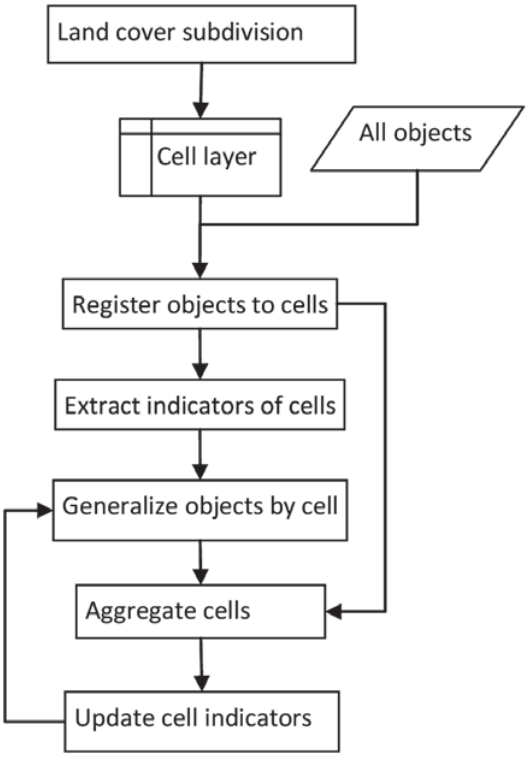


Figure 8. The framework for city scale generalization.

Generalization framework

Road network is firstly used to subdivide the whole city into smaller and meaningful units. Moreover, subdivision can also facilitate model integration, updating and scene graph management, especially when multiple object layers and multiple LoDs are involved.

After subdivision, a 2D land cover layer called cell layer is created, composing of basic units for generalization. All objects of different layers can be then related by registering to the cell layer. All generalization operations are supposed to be performed by the unit of cell. Of course, layers can also be generalized without imposing this constraint by not registering to the cell layer. The major generalization operation employed on the land cover cell layer is aggregation. In each cell, indicators can be extracted from the registered objects and serve as the constraints for object generalization and cell aggregation. In this paper, indicators are extracted for the purpose of generalization and visualization, which may not comply with the concepts established in urban management and urban planning. Indicator extraction is a human-assisted task, thereby varying according to the application and input data. Generalization of city objects can be performed within cells at different levels of aggregation, and thereby multiple levels of abstraction can be obtained. The generalization algorithm is depicted as in Figure 8.

Implementation

Using the generalization framework above, different implementations can be developed according to specific applications. Here we give a possible implementation.

1) Land cover subdivision and registration

First of all, a 2D ground plane is generated by computing the convex hull of all input layers. Then road layer is used to partition the ground plane into cells, thereby obtaining cell layer in the form of a land cover map. Generally speaking, the complexity of road network would result in a lot of fragments in cell layer. But instead of being eliminated, these fragments should be aggregated with other cells. It is important to make sure the cell layer covers all the land, so that all city objects can be registered. After obtaining the cell layer, all city objects can be registered to its composing cells. A road section can be registered to one cell as its border road or inner road. 2D objects can get direct registration, but the complexity depends on the degree of data consistency with road layer, determined by a number of factors from data acquisition to modeling process. In order to avoid handling graphic conflicts, it's better to use the data from the same supplier at the same level of detail. For the registration of 3D objects, 2D footprints need to be generated from 3D models at first.

2) Land use indicator

Since all city objects have been registered to the cell layer, land use can be roughly estimated for the purpose of generalization. In this implementation, two types of land use are defined: *Building* and *Other*. Their definitions are given as below:

Given cell $\boldsymbol{C}$ *with the area* $\boldsymbol{A}$, $\boldsymbol{A_{bldg}}$ *as the building area within* $\boldsymbol{C}$, $\boldsymbol{A_{other}}$ *as the total area of all other objects within* $\boldsymbol{C}$, *the land use of cell* $\boldsymbol{C}$ *is determined as in (1), where* $\boldsymbol{Th}_1$ *is a predefined variable as the threshold.*

$$Landuse_C = \begin{cases} Building & \text{if } A_{bldg}/(A - A_{other}) > Th_1 \\ Other & \text{else} \end{cases} \tag{1}$$

In most cases, A_{bldg} is larger than $(A - A_{other})$, because there are always other unknown or not modeled entities occupying the interspaces between buildings, like extended road surfaces, entry ways, parking lots, and courtyards. In each cell (enclosed by road segments), the proportion of $A_{bldg}/(A - A_{other})$ can indicate building density in this built-up area, and thereby it can be used as constraint for the typification of such area. If building density exceeds a user defined threshold, the land use of this cell is identified as *Building*. Otherwise, the land use of this cell is set as *Other*.

3) Cell-based generalization

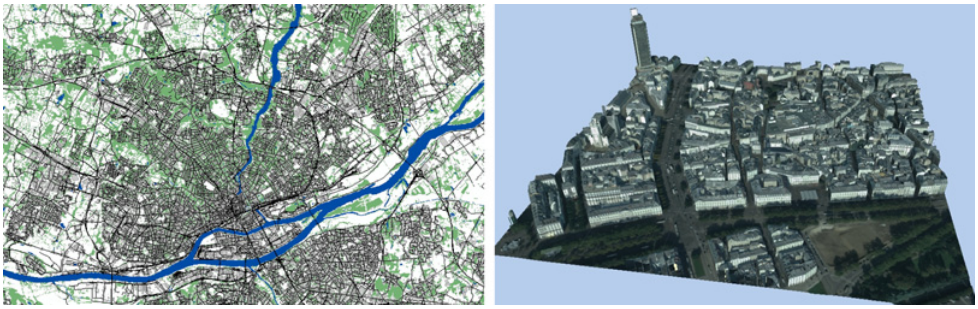
After having generated cell layer and having estimated land use indicator of each cell, two different types of generalization can be performed: (a) generalization of city objets within the same cell; (b) aggregation of cells, which enables further generalization of type(a). Starting from LoD1 city model, another three LoDs are introduced to describe inter-level cases. They are defined as below:

- LoD1: LoD1 building models with optional models of other city objects.
- LoD0C: Adjacent LoD1 building models are aggregated.
- LoD0B: Buildings in the same street block are typified by extruded street block.
- LoD0A: Extruded street blocks are aggregated.

In this implementation, city model at LoD0C is generated by aggregating adjacent LoD1 buildings within the same cell. City model at LoD0B is derived by typification. If the land use of a cell is *Building*, all its containing buildings can be typified based on the cell polygon. After cutting out all other objects from the cell polygon, the remainder can be extruded to LoD0B blocks by assigning the average height of all containing buildings. If the land use of a cell is *Other*, no operations will be performed. As land use indicator of each cell has been extracted, further aggregation of cells can be enabled so that further abstraction can be achieved. We aggregate the cells whose area is no more than Th_2 and with land use estimated as *Building*, where Th_2 is a predefined threshold. After obtaining new cells by aggregation,

Table 1. Specification of input data.

Theme	Format	Dim	Scale/LoD	Coverage	Source
Road Railway	Shapefile (polyline)	2D	1:5000	whole city	IGN BDTopo
Building Vegetation Water body	Shapefile (polygon)				
Building	CityGML	3D	LoD2	a parcel	IGN Bati3D



(a) 2D cadastral maps (©IGN BDTopo) (b) 3D CityGML model (©IGN Bati3D)

Figure 9. Input data.

city objects within the same cell can be generalized. Applying the same typification method for deriving LoD0B city model to the new cell layer, LoD0A city model can be obtained.

5. RESULTS

The proposed framework was implemented for visualizing 3D Nantes, France. Five 2D layers and one 3D layer are given as the source data, as shown in Figure 9. The specification is given in Table 1. We apply these data to evaluate all the procedures previously described.

5.1 Model generation

Provided with the input data, LoD1 city model can be obtained by extruding 2D cadastral buildings and integrating other 2D objects, as shown in Figure 12(a). Integrated with a parcel of LoD2 buildings, the LoD of the city model can be increased. However, we do not denote such as LoD2 city model until all its containing building models reach to LoD2. In order to distinguish such kind of city models, we use a new way of denotation - LoD1/2 - to imply that only a part of buildings in the whole city have reached to LoD2. In practice, we can also indicate the proportion of LoD2 buildings. In Nantes city case, LoD1/2 (2.7% LoD2) city model is obtained (the top image in Figure 10). The LoD1 building models in the overlay area are replaced by the LoD2 buildings.

Staring from this LoD1/2 city model, geometric details of buildings can be added by using procedural modeling technique. In this case, we applied it to the integrated LoD2 buildings, as shown in Figure 10. First, rules are designed manually according to the local architectural structures observed. Then, assets to insert final detailed geometry (windows, door and balconies) are selected and set from a 3D repository.

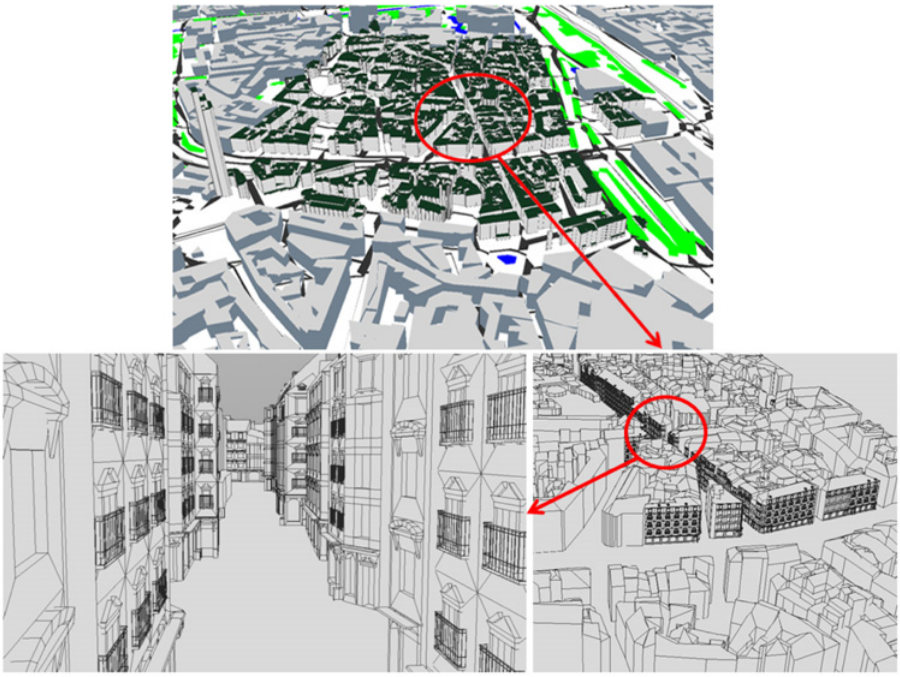


Figure 10. Procedural modeling for detailing: from the LoD1/2 city model (top), a parcel of LoD2 building models are procedurally added with geometry details.

5.2 Multi-scale representation

5.2.1 Geometry details simplification

We apply the simplification method described in Section 4.2.1 to the procedural street model shown in Figure 10. For this test, we configured a simple view-dependent criteria where the distance of an aerial flying reference point is used as the parameter for deciding the level of detail that must be used according to the proximity of the building. Figure 11 shows wire-frame rendering resulting models at different positions of the flying point. Note the geometry density variation in the street. Close assets are maintained at full resolution, while far away buildings are completely reduced to textures (shown as white planes to emphasize differences). Table 2 shows polygon reduction factors of the model using the algorithms described in Section 4.2.1. The geometry is reduced to about one fifth for each frame. Note also that as we are focusing on only one procedural street there is not so much reduction difference between the low-asset level and the low-facade level (each facade represented as one polygon).

5.2.2 City scale generalization

Taking LoD1 city model as input, further generalization can be performed at city scale, so as to reduce information density and to achieve better abstraction of the city. In our case, LoD1 city model, as shown in Figure12(a), consists of five thematic layers, among which building layer gives the most impact on the overall appearance and road layer is essential for the following generalization approach. By implementing our method of city scale generalization proposed in Section 4.2.2, another three LoDs can be obtained: LoD0C, LoD0B, and LoD0A, as shown in Figure 12.

Figure 11. A reference point (red ball) is used to guide dynamic LoD simplification along a procedurally modeled street. Note the density geometry detail changes according to the proximity to the reference point.

Table 2. Factor reductions of the city model considering one frame (Figure 11, bottom), complete asset-level reduction and complete facade-level reduction.

	Full Model	One frame	Low-asset level	Low-facade level
# Polygons	97666	20867	17311	16619
Remaining polygons		21.3%	17.7%	17.0%

6. DISCUSSION

We propose a framework to produce a multi-scale 3DCM allowing a multi-representation of the model. This framework starts from GIS data and uses both a procedural modeling technique and a generalization technique to create a 3DCM with various LoDs. The LoD of a specific part of this model can be dynamically modified to take into account various criteria as distance of camera or user interest. So this framework embeds two possibilities, adding or removing details, statically during model creation, and dynamically during the model use.

Results show the capability of our approach for dealing with both coarse and detailed urban models. An advantage of our system is that only the resulted relevant geometry is generated, avoiding loading huge amount of geometry.

Having more control, both at local and global level, is a challenge in procedural modeling [20]. Here we can extend traditional approaches by including locator tools that allow to increase progressively the density of the objects close to the important scenario structures, and decrease the density of the farthest objects. They could also be used as an efficient tool for aggressive simplification, eliminating part of buildings that are never seen/used in an application.

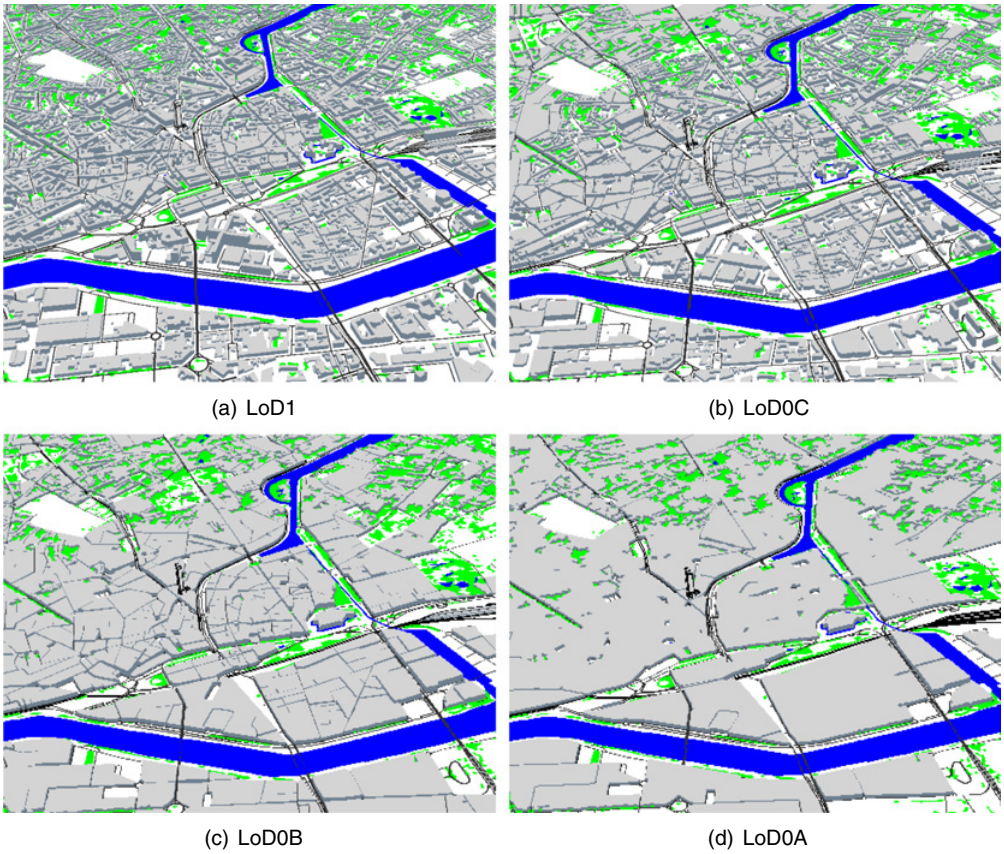


Figure 12. LoD city models derived by generalization. A landmark building (Tour Bretagne) is marked and excluded from generalization.

Comparing our approach for geometry simplification to available procedural LoD solutions like CityEngine [18], the main significant difference is that our method is completely automatic, working procedurally and independently over all products. Therefore, final assets produced by the same rule may have different geometry resolutions according to the simplification criteria, whereas in the mentioned system all reduced products have the same resolution for the whole rule application. Moreover, in their case the simplified geometry must be set manually for each specific asset.

7. CONCLUSION AND FUTURE WORK

A 3DCM with various and heterogenous LoD is useful in several applications as rendering, as shown in this paper, but also simulation, information and scientific visualization, CAD and others. Our framework integrates the constraints related to rendering, but integrating other kind of constraints to define the LoD of the model in a particular area can be done by defining the adapted ruleset. Our next goal is to show others applications that use our framework. The interoperability will be a challenging issue and can be enhanced by using a data format allowing to embed all the LoD data into one file. To go further into the LoD range, it could be interesting to include 2D maps into our model. We can also work onto the mesh size variations as the mesh precision is a critical issue in simulation processes.

This framework enlarges the scope of usability of the 3DCM by giving more flexibility in the data production and the data use processes. It has now to be more tightly connected to research works about semantics to be fully usable.

This work is funded by the LiPaD project (PICS 5786) from CNRS, France and the TIN2010-20590-C02-02 project from Ministerio de Ciencia e Innovación, Spain.

References

[1] J.H. Clark, Commun. ACM **19**, 547 (1976)
[2] D. Luebke, C. Erikson, *View-dependent simplification of arbitrary polygonal environments*, in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1997), SIGGRAPH '97, pp. 199–208, ISBN 0-89791-896-7, http://dx.doi.org/10.1145/258734.258847
[3] D. Luebke, B. Watson, J.D. Cohen, M. Reddy, A. Varshney, *Level of Detail for 3D Graphics* (Elsevier Science Inc., New York, NY, USA, 2002), ISBN 1558608389
[4] T.H. Kolbe, *Representing and Exchanging 3D City Models with CityGML*, in *Proceedings of the 3rd International Workshop on 3D Geo-Information, Lecture Notes in Geoinformation and Cartography*, edited by J. Lee, S. Zlatanova (Springer Verlag, Seoul, Korea, 2009), p. 20
[5] Y.I.H. Parish, P. Müller, *Procedural modeling of cities*, in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 301–308, ISBN 1-58113-374-X
[6] P. Prusinkiewicz, A. Lindenmayer, *The algorithmic beauty of plants* (Springer-Verlag, New York, Inc., New York, NY, USA, 1996), ISBN 0-387-94676-4
[7] J. Sun, X. Yu, G. Baciu, M. Green, *Template-based generation of road networks for virtual city modeling*, in *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology* (ACM, New York, NY, USA, 2002), pp. 33–40, ISBN 1-58113-530-0
[8] P. Wonka, M. Wimmer, F. Sillion, W. Ribarsky, ACM Transactions on Graphics **22**(4), 669 (2003)
[9] P. Müller, P. Wonka, S. Haegler, A. Ulmer, L. Van Gool, ACM Transactions on Graphics **25**(3), 614 (2006)
[10] P. Musialski, M. Wimmer, P. Wonka, Computer Graphics Forum (Proceedings of EUROGRAPHICS 2012) **31**(2), 661 (2012)
[11] P. Müller, G. Zeng, P. Wonka, L.V. Gool, ACM Transactions on Graphics **26**(3) (2007)
[12] G. Chen, G. Esch, P. Wonka, P. Müller, E. Zhang, ACM Transactions on Graphics **27**(3) (2008)
[13] M. Lipp, P. Wonka, M. Wimmer, ACM Transactions on Graphics **27**(3), 102:1 (2008)
[14] R. Ridorsa, G. Patow, *The skylineEngine System*, in *XX Congreso Español de Iinformàtica Gràfica, CEIG2010* (2010), pp. 207–216
[15] G. Patow, IEEE Computer Graphics and Applications **32**, 66 (2012)
[16] EpicGames, *Unreal engine* (2011), http://udn.epicgames.com/Three/ProceduralBuildings.html
[17] B. Cullen, C. O'Sullivan, Journal of WSCG **19**(3), 119 (2011)
[18] ESRI, *Cityengine* (2011), http://www.procedural.com
[19] G. Kelly, H. McCabe, ITB Journal **14**, 87 (2006)
[20] B. Watson, P. Mueller, O. Veryovka, A. Fuller, P. Wonka, C. Sexton, IEEE Computer Graphics and Applications **28**, 18 (2008)
[21] C. Vanegas, D. Aliaga, P. Mueller, P. Waddell, B. Watson, P. Wonka, Proceedings of EUROGRAPHICS, State of the Art Reports (also Computer Graphics Forum, to appear) pp. 1–18 (2009)

[22] J.D. Foley, A. van Dam, S.K. Feiner, J. Hughes, *Computer Graphics: Principles and Practice* (Addison Wesley, 1995)
[23] M. Guthe, R. Klein, Journal of WSCG **11**(1) (2003)
[24] R.B. McMaster, K.S. Shea, *Generalization in Digital Cartography* (Assoc. of American Geographers, Washington, D.C., 1992)
[25] R. Pajarola, E. Gobbetti, The Visual Computer **23**(8), 583 (2007)
[26] L. Meng, A. Forberg, *3D Building Generalisation* (Elsevier, 2007)
[27] M. Sester, *3D Visualization and Generalization*, in *Photogrammetric Week '07* (2007)
[28] F. Thiemann, M. Sester, *Segmentation of Buildings for 3D-Generalisation*, in *Proceedings of 7th ICA Workshop on Generalisation and Multiple Representation* (2004)
[29] H. Mayer, International Journal of Geographical Information Science **19**(8-9), 975 (2005)
[30] A. Forberg, ISPRS Journal of Photogrammetry and Re-mote Sensing **62**, 104 (2007)
[31] M. Kada, *3D Building Generalization based on Half-Space Modeling*, in *Proceedings of the ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data* (2006)
[32] J.Y. Rau, L.C. Chen, F. Tsai, K.H. Hsiao, W.C. Hsu, in *Advances in Image and Video Technology*, edited by L.W. Chang, W.N. Lie (Springer Berlin / Heidelberg, 2006), Vol. 4319 of *Lecture Notes in Computer Science*, pp. 44–53, http://dx.doi.org/10.1007/119495345
[33] H. Fan, L. Meng, M. Jahnke, *Generalization of 3D Buildings Modelled by CityGML* (Springer, 2009), pp. 387–405
[34] K.H. Anders, *Level of Detail Generation of 3D Building Groups by Aggregation and Typification*, in *In: Proceedings of the XXII International Cartographic Conference, La Coruna* (2005)
[35] R. Guercke, T. Götzelmann, C. Brenner, M. Sester, ISPRS Journal of Photogrammetry and Remote Sensing **66**(2), 209 (2011)
[36] S. He, G. Moreau, J.Y. Martin, *Footprint-Based 3D Generalization of Building Groups for Virtual City Visualization*, in *Proceedings of GEOProcessing 2012* (2012), pp. 177–182
[37] T. Glander, J. Döllner, Computers, Environment and Urban Systems **33**, 375 (2009)
[38] B. Mao, Y. Ban, L. Harrie, ISPRS Journal of Photogrammetry and Remote Sensing **66**(2), 198 (2011)
[39] R. Chang, T. Butkiewicz, C. Ziemkiewicz, Z. Wartell, N. Pollard, W. Ribarsky, IEEE Comput. Graph. Appl. **28**, 27 (2008)
[40] R. Guercke, C. Brenner, *A Framework for the Generalization of 3D City Models*, in *Proceedings of 12th AGILE Conference on GIScience* (2009)
[41] T. Glander, M. Trapp, J. Döllner, *Concepts for Automatic Generalization of Virtual 3D Landscape Models*, in *Proceedings of the Annual Conference of Digital Landscape Architecture* (2011), pp. 127–135
[42] Open Geospatial Consortium Inc., *Opengis citygeography markup language (citygml) encoding standard* (2008)
[43] O. Pueyo, G. Patow, Tech. Rep. IMA12-01-RR, Departament IMA, Universitat de Girona (2012)
[44] S. Haegler, P. Wonka, S. Müller, L. Van Gool, P. Müller, Computer Graphics Forum **29**, 1479 (2010)