ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume II-2, 2014
ISPRS Technical Commission II Symposium, 6 – 8 October 2014, Toronto, Canada

PROGRESSIVE TRANSMISSION OF 3D BUILDING MODELS BASED ON STRING GRAMMARS AND PLANAR HALF-SPACES

Martin Kada

Institute for Geoinformatics and Remote Sensing (IGF), University of Osnabrück
Barbarastr. 22b, 49076 Osnabrück, Germany
martin.kada@uni-osnabrueck.de

Technical Commission II

KEY WORDS: Building, Three-dimensional, Modeling, Geometry, Algorithms

ABSTRACT:

As there are numerous applications for 3D city models with a wide range of model requirements regarding geometric accuracy and granularity, there is also a high demand for such models at different levels of detail (LOD). And although their reconstruction and cartographic generalization has been widely studied, particularly with regard to 3D building models, their encoding for a progressive storage and transmission is up to now not profoundly explored and sufficiently solved. Most often building models at different LODs are considered as discrete entities that are not related to each other. In this paper we present a progressive encoding and transmission scheme for 3D building models that is easy to understand and implement for the end user as well as flexible and extensible for the model producer. The progressive scheme is based on string grammars and describes a sequence of successive LODs as a dynamic set of production rules. In order to restrict the effects of LOD changes on a local range of the progressive string representation, we use a solid modelling approach based on planar half-spaces to construct 3D buildings. The generation of such progressive string grammars is shown and examples are given.

1. INTRODUCTION

Since the early developments and first implementations of 3D city models roughly two decades ago, numerous applications like cellular network planning, solar, pollutant, noise, flood, energy and visibility analyses, location-based services, 3D maps and virtual globes, as well as car and pedestrian navigation systems have emerged. Due to their wide range of model requirements regarding geometric accuracy and granularity and their device and system limitations like processing power, memory, and screen size, network bandwidth, and energy consumption, etc., there is nowadays an enormous and still to a large extent unsatisfied demand for 3D city models at different levels of detail (LOD), particularly with regard to man-made objects like buildings. The meanwhile well-established standard CityGML accommodates for this by defining a model and mechanism for describing building objects in five different LODs with respect to their geometry, topology, semantics, and appearance (Gröger and Plümer, 2012). Because CityGML was primarily developed for storing 3D city models in large spatial data bases and for file based exchange, its LOD concept is purely static and describes building LODs as discrete entities.

Real-time visualization applications, however, often implement a dynamic LOD concept in which the different LODs are built upon another and the visualization features a smooth geometric transition or image based blending between them. Web based applications additionally try to optimize the trade-off between available network bandwidth and quick response time by first transmitting an initial coarse representation of the whole scene and then continuously delivering more and more details in small and manageable data chunks until no visual improvement is possible for the current point of view. As the viewer moves around, the position of the viewer is continuously evaluated and the LOD of each object adjusted if necessary, thereby triggering further transmission of data, e.g. if the viewer moves towards an object. In order to avoid transmitting redundant data only the differences between LODs are send in the form of incremental updates. This is usually achieved by progressively encoding the 3D models with increasing details.

The progressive meshes scheme introduced by Hoppe (1996) is probably the most prominent example for progressive encoding, transmission, and displaying of 3D models. It works on triangle meshes of general objects with arbitrary shapes and encodes incremental changes as a sequence of vertex split operations. Hongsheng et al. (2009) adopt this idea for a progressive simplification and transmission of triangulations of 2D building polygons. Sester and Brenner (2009) show the decomposition of cartographic generalization of building ground plans into simple geometric and topologic operations on 2D polygons with the purpose to generate multiscale object representations. Due to the simple operations used, these approaches allow for smooth animations between the fine granular and stepwise changes in the LODs. Ai et al. (2004) present a hierarchical decomposition of 2D polygons into a series of convex hulls and bounding rectangles, their progressive transmission as additive and subtractive change patches, and its applicability to building ground plans. Royan et al. (2006) represent 2.5D virtual cities as a progressive building tree based on parameterized 3D models.

In this paper, we present a progressive transmission scheme for 3D building models that is based on string grammars. During the last few years grammars have become particularly popular to describe, generate and reconstruct detailed 3D building and city models (Wonka et al., 2003; Müller et al., 2006). Milde et al. (2008) identify complex roof structures in the region adjacency graph of a segmented airborne laser point cloud as words of a formal grammar. Ripperda and Brenner (2009) incorporate a façade grammar as structure information in their reconstruction process. Dehbi and Plümer (2011) automatically learn grammar rules of building parts. Becker (2009) shows the inference of structure from one building façade to other building façades e.g. where reconstruction is not possible. The modelling of indoor architecture by grammars is shown in (Becker et al., 2013).

The description of building models by string grammars is per se hierarchically structured. In order to obtain a coarse model from a given grammar, it seems straightforward to alter the grammar to leave some of the rules out, short-cut or replace them by rules that describe the portion of the model in fewer details. The set of grammar rules, however, must then be organized so that their progressive examination leads to different LODs.

2. PROGRESSIVE TRANSMISSION SCHEME

The main aspect of our contribution is the definition of a simple progressive transmission scheme for 3D building models that is easy to understand and implement for consumers; the side that receives and decodes a model stream in order to retrieve and reconstruct imbedded LODs. Our scheme is expressed in a most general way to allow for uttermost flexibility; meaning that it is not restricted to an explicitly pre-defined set of operations that manipulates the 3D models. Rather the side that produces the LODs describes building models as string representations using Boolean combinations of half-spaces and their manipulation by so called production rules that define a string grammar. The operations are only implicitly given by the way the production rules are stated and how they change over the course of the LOD transmission. In contrast to usual string grammars, however, each set of production rules for a given LOD uniquely defines a specific model.

On the producer's side, the LODs of a 3D building model are generated and encoded as production rules. The former aspect, the way how LODs are generated is, however, not our concern in this paper. A LOD hierarchy could e.g. be a side product of an automatic 3D building reconstruction process, although current approaches do not consider generating different LODs at this time. For a current and in depth overview of 3D building reconstruction approaches, see e.g. (Brenner, 2010) and (Haala and Kada, 2010). Another option is cartographic generalization where e.g. 3D building models are geometrically simplified with regard to commonly recognized building regularities, e.g. the assumption that walls are aligned coplanar, parallel, and rectangular. Two examples for cartographic simplification and aggregation approaches for 3D building models are given in (Li et al., 2013) and (Kada, 2011). Also a manual editing tool for LOD generation is imaginable. The focus in this paper is on the latter aspect, however, the encoding of production rules for a given LOD hierarchy.

3. 3D BUILDING MODELING USING HALF-SPACES

Before we introduce the progressive transmission scheme based on string grammars, we first define the string representation that is used here to describe 3D building models.

In principle, the transmission scheme can be used with any character or binary string representation. E.g. the well-known text (WKT) or well-known binary (WKB) representations that describe 3D models by their boundaries would be a viable choice (Open Geospatial Consortium Inc., 2011). However, boundary representations are rather cumbersome to manipulate and are therefore seldom used for construction purposes. E.g. if we want to add a chimney to a building with a saddleback roof (cp. Figure 1), we need to alter all polygonal faces that it touches. This would mean in our example that in addition to defining four additional faces for the chimney itself, the string representation of our 3D building model would need to be modified at three further positions, namely the three polygons of the roof and the front gable. Or if we want to crop one side of the building, we need to add one face and modify another four faces. Because of the necessity to modify a string representation of a 3D model at different positions even for small alterations makes the boundary representation not well suited for our suggested transmission scheme. Changes to the string model should be rather few and local. Another drawback of the boundary representation is that it is rather difficult to define compact operations with the production rules of string grammars (cp. section 4).

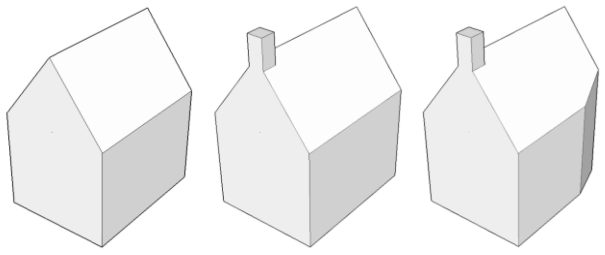

Figure 1. 3D building model with a chimney and a cropped side as two added details

We therefore suggest using planar half-spaces that are combined with Boolean set operations. For detailed descriptions of half-space modeling, see e.g. (Mäntylä, 1988) or (Foley et al., 1990). In summary, a solid is defined as a combination of simple point sets, each described by a characteristic function h. A point belongs to a point set if it satisfies this function. We use linear inequality functions that define half-spaces that consist of points on or behind a plane:

$$h = Ax + By + Cz + D \leq 0 \qquad (1)$$

As we often need the complement of a half-space, but one that shares the boundary, we define the complement to be:

$$h^C = Ax + By + Cz + D \geq 0 \qquad (2)$$

By applying the Boolean set operation intersection to a set of planar half-spaces, a convex polyhedron is constructed. Some care has to be taken to generate finite and valid solids, the latter is ensured by using regularized Boolean set operations (see e.g. (Foley et al., 1990)) that do not produce dangling faces, lines and points. Concave shapes are then constructed by the union of convex polyhedrons. A convex example building m_1 formed from seven half-spaces (h_1 to h_7) is given by (cp. Figure 2):

$$m_1 = h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_5 \cap h_6 \cap h_7 \qquad (3)$$

To gain the cropped side building model m_2, only half-space h_8 needs to be added to the existing intersections of half-spaces:

$$m_2 = m_1 \cap h_8 \qquad (4)$$

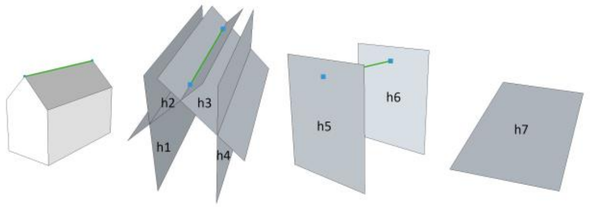

Figure 2. Saddleback roof defined by seven planar half-spaces

Here, we use symbol m_1 both to shorten the equation and for better illustration. Generally, our string representations of the models are written out:

$$m_2 = h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_5 \cap h_6 \cap h_7 \cap h_8 \quad (5)$$

Now, adding a chimney to the saddleback building component to gain model m_3 is e.g. accomplished by uniting the existing building model m_2 with the intersections of the six half-spaces (h_5, h_7, and h_9 to h_{12}) that form the chimney:

$$m_3 = m_2 \cup (h_5 \cap h_7 \cap h_9 \cap h_{10} \cap h_{11} \cap h_{12}) \quad (6)$$

Note that we use round brackets to group intersections together to form convex components. This is not really necessary if the union operation is defined to precede the intersection, but the brackets help to maintain readability. In the above example the two half-spaces h_5 and h_7 are conveniently reused in m_3 to cut down the number of half-spaces that need to be defined. We could actually even reorganize the formula and factor out the intersection of those two half-spaces:

$$m_4 = h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_6 \quad (7)$$

$$m_5 = h_9 \cap h_{10} \cap h_{11} \cap h_{12} \quad (8)$$

Models m_4 and m_5 are the components m_1 and m_2 without the half-spaces h_5 and h_7, which we factor out to gain:

$$m_6 = (m_4 \cup m_5) \cap h_5 \cap h_7 \quad (9)$$

We use a simplistic representation of 3D models and restrict the half-space models to always be a union of intersections. This does not restrict the shape of the resulting models, but leads to non-optimal representations considering both the number of half-spaces and the number of Boolean operations when constructing complex models. However, the derivation of production rules from such a normalized form is easier than from a hierarchically structured model. Also we do not use the Boolean difference operation. This is not a constraint on the transmission scheme as the scheme itself does not require any specific form. It is just a matter of how we generate the production rules at the moment. In future work, we might extend our work to hierarchical half-space models and also make use of the difference operation as this might positively influence the string length of the model definition.

As should have become apparent, the reason we use half-space modeling is that the changes to their string representation are in most cases local. In order to construct the different LODs m_1, m_2, and m_3, we only added one string to the end of each previous LOD; whereas the string of a boundary representation would need to be altered at more than one position to have the same effect. In summary, we modify convex components by adding more half-spaces to their intersections and add further components by appending them with a union operation.

4. TRANSMISSION STRING GRAMMAR

As stated above, our progressive transmission scheme is based on production rules that form a string grammar. Each LOD of the 3D building model is described by a set of production rules. It should be stressed that the producer of such a grammar must ensure that the set of production rules unambiguously and completely describe a building model. Otherwise the receiver of the grammar is unable to decode the model. It is, however, easy for the receiver to verify that the grammar generates a correct string, e.g. by detecting rules that produce recursive loops.

A string grammar G is often defined as the tuple (N, T, P, S), where N is the set of non-terminals, T the set of terminals, P the production rules, and S the start symbol. For an introduction on formal grammars, see e.g. (Sipser, 2012). Starting with the start symbol S, each production rule replaces each substring that fits the left side of the rule with the string of symbols that stand on its right side. A non-terminal symbol must always stand on the left side of a production rule.

The set of terminal symbols consist of symbols that are needed to describe a 3D building model in half-space modeling. So for most grammars, we have $T = \{m, =, h_1 \ldots h_n, \cup, \cap, \setminus, ^C, \emptyset, U, (,)\}$ or a subset thereof. Here, m denotes the 3D building model itself, h_1 to h_n stand for the n half-spaces the model is constructed from, the Boolean set operations $\cup$, $\cap$, and $\setminus$, the complement ^{C} of a set, the empty set $\emptyset$, the universe U, and round brackets. The half-space parameters are not defined within the grammar as this would pose an unnecessary flexibility that is disproportional to the effort. The four plane parameters A, B, C, and D of each planar half-space are set by a standardized string statement (outside the grammar).

The progressive transmission starts with an initial set of rules that describes the coarsest 3D building model. It must at least contain a production rule with the start symbol on its left side so that a model is actually generated. Such a rule usually describes the Boolean combination of the initial half-spaces. Every rule might also contain non-terminal symbols. Then there must be further production rules that define how to replace these non-terminal symbols. In addition to the set of production rules, the plane parameters A, B, C, and D of all half-spaces that are used are defined (cp. Figure 3). The grammar for the above saddleback example building could be given as:

$$S \rightarrow m_1 = (h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_5 \cap h_6 \cap h_7\ R_1) \quad (10)$$
$$R_1 \rightarrow \varepsilon \quad (11)$$

The non-terminal R_1 in equation (10) can be thought of as a handle or anchor. It identifies the position where the string is later altered in succeeding LODs. As it is not used in this LOD, it solely maps to the empty string (11). To add half-space h_8 to the model in the next LOD, the production rule (11) is overwritten by transmitting rule (12) that also has R_1 on the left side. In contrast to general grammars, we do not allow production rules to have the same left side. Newer rules always overwrite older rules.

$$R_1 \rightarrow \cap\ h_8 \quad (12)$$

The application of production rule (12) on the resulting string of (10) and (11) produces the model m_2 of equation (4). Half-space h_8 can later be removed again by re-sending production rule (11) if necessary; thus replacing rule (12) again.

In the event that a half-space should be removed in a LOD that has not been introduced by a specific rule as h_8 in (12), e.g. h_7 from m_1 as defined by (10) and (11), a production rule can be formulated as:

$$\cap\ h_7\ R_1 \rightarrow \varepsilon \tag{13}$$

This has the effect, that the substring $\cap\ h_7\ R_1$ on the left side of the production rule is replaced in the half-space representation of the model with the empty string.

To add a convex component to a model, the same procedure as shown above is applied, only with the difference, that the non-terminal is now placed outside the brackets.

$$S \rightarrow m_1 = (h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_5 \cap h_6 \cap h_7)\ R_1 \tag{14}$$
$$R_1 \rightarrow \varepsilon \tag{15}$$

The production rule (16) generates the new component, whereas sending (15) at a subsequent LOD would remove it again:

$$R_1 \rightarrow \cup\ (h_5 \cap h_7 \cap h_9 \cap h_{10} \cap h_{11} \cap h_{12}) \tag{16}$$

The ordering of how half-spaces are defined in a component is not significant to its definition and results in the same model. Also there are several ways to express the same production rule just by altering the position of the non-terminal symbol. The removal of half-space h_7 as shown in production rule (13) can be expressed in the following additional ways:

$$\cap\ R_1\ h_7 \rightarrow \varepsilon \tag{17}$$

$$R_1 \cap h_7 \rightarrow \varepsilon \tag{18}$$

However, the ordering of half-spaces and the position of the non-terminal symbols are both important when it comes to formulating string grammars with as few production rules as necessary.

As the individual LODs need to be differentiable, we simply use pairs of tags in angle brackets like <LODn> and </LODn> that are well known from common markup languages. The complete example building from above with both the chopped side and the chimney is given by the text file in Figure 3.

```
<LOD0>
S → m = (h_1 ∩ h_2 ∩ h_3 ∩ h_4 ∩ h_5 ∩ h_6 ∩ h_7 R_1) R_2
R_1 → ε
R_2 → ε
h_1 = A_1, B_1, C_1, D_1
…
h_7 = A_7, B_7, C_7, D_7
<\LOD0>

<LOD1>
R_1 → ∩ h_8
h_8 = A_8, B_8, C_8, D_8
<\LOD1>

<LOD2>
R_2 → ∪ (h_5 ∩ h_7 ∩ h_9 ∩ h_10 ∩ h_11 ∩ h_12)
h_9 = A_9, B_9, C_9, D_9
…
h_12 = A_12, B_12, C_12, D_12
<\LOD2>
```

Figure 3. Three LODs of the example 3D building model in the proposed progressive transmission scheme

So far, we have shown how to remove half-spaces on the level of the grammar by defining production rules that generate string representations without these half-spaces, e.g. h_7 in production rule (13). But half-spaces can also be suppressed on the level of the Boolean model evaluation. To neutralize a half-space, its referencing non-terminal symbol (here H_8) can point to the empty set (∅) or the universe (U). E.g. half-space h_8 that is introduced by production rules (19) and (20) can be disabled by replacing it with the universe (U) in production rule (21):

$$S \rightarrow m_1 = (h_1 \cap h_2 \cap h_3 \cap h_4 \cap h_5 \cap h_6 \cap h_7 \cap H_8) \tag{19}$$
$$H_8 \rightarrow h_8 \tag{20}$$
$$H_8 \rightarrow U \tag{21}$$

This is due to the fact that the intersection with the universe does not alter the resulting model. Now, to disable the whole component of m_1 (and in this example the whole model), h_8 can e.g. be replaced by the empty set:

$$H_8 \rightarrow \emptyset \tag{22}$$

The intersection with the empty set results again in the empty set and the component has no solid representation anymore. This does not influence other components as they are combined with the union operation where the empty set does not alter the result. By utilizing the empty set or the universe in this way, a string grammar can be significantly simplified as no anchor is needed to replace a half-space and its Boolean symbol by the empty string as in production rules (13), (17), and (18).

5. GRAMMAR GENERATION

There are numerous ways to define a grammar that generates a specific 3D building model. But the difficulty here lies in defining a sequence of production rules that transforms the 3D model step by step through the LODs and where complete grammars are given at certain points in the sequence that forms a valid model. This is accomplished by a dynamic grammar, meaning that each rule can be overwritten by follow-up rules of a successive LOD.

We reduce the complexity of the problem by generating grammars that only produce strings that are in the above mentioned normalized form, i.e. components and their half-spaces are organized as the union of intersections. We also limit our production rules to implement the four basic operations of insertion and removal of convex components to or from a model and the insertion and removal of half-spaces to or from a convex component. These rules are not explicitly stated, but are rather implicitly given by the production rules. Other useful operations, e.g. splitting a convex component into two or more parts and redistributing its half-spaces, are not regarded at this point.

We start by defining a complete grammar for each LOD in the following schema:

$$S \rightarrow m = C_1 \cup \ldots \cup C_n \tag{23}$$
$$C_i \rightarrow (H_1 \cap \ldots \cap H_k) \tag{24}$$
$$H_j \rightarrow h_j \tag{25}$$

There are three types of production rules that define a model as a union of components (23), each component as an intersection of half-spaces (24), and each half-space by its terminal symbol (25). One has to make sure that the components and half-spaces are given the same symbol over all LODs they are defined in. As the order of components and half-spaces are irrelevant for the resulting model, but not reducing the number of generated production rules, we sort all grammars so that the LODs have the same ordering and that components and half-spaces are appended at the end of the right sides of the production rules.

The grammar is then generated by reducing the normalized production rules to new rules that represent the differences between LODs and by aggregating production rules of the same LOD by string substitution.

With all normalized sets of production rules available, the rules of type S → and C → are pairwise compared in back to front order. If there is a difference in a production rule between LOD$_i$ and LOD$_{i+1}$, then a new non-terminal symbol is introduced to hold this difference. Let LOD$_j$ ($j \leq i$) denote the lowest LOD so that the respective production rule can be found in all LODs in the range LOD$_j$ to LOD$_i$. Then a production rule is added to LOD$_{i+1}$ that maps the non-terminal symbol to the difference of the right side of the production rule and another production rule is added to LOD$_j$ that maps the symbol to the empty string. The original production rule is deleted in all LODs ranging from LOD$_{j+1}$ to LOD$_{i+1}$. The non-terminal symbol is appended to the production rule of LOD$_j$.

In the following example, only the production rules S → are illustrated. There are four LODs and LOD$_0$ and LOD$_1$ have the same number of components. The remaining production rules are omitted for simplicity (see Figure 4).

```
<LOD0>S → m = C_1 …                    <\LOD0>
<LOD1>S → m = C_1 …                    <\LOD1>
<LOD2>S → m = C_1 ∪ C_2 …              <\LOD2>
<LOD3>S → m = C_1 ∪ C_2 ∪ C_3 …        <\LOD3>
```

Figure 4. Original production rules of S.

As the production rule S of LOD$_2$ and LOD$_3$ are different, the non-terminal symbol R$_2$ is introduced and two production rules added that maps R$_2$ to the empty string in LOD$_2$ and to the union with C$_3$ in LOD$_3$. The production rule S is removed from LOD$_3$ and R$_2$ appended to the same rule in LOD$_2$ (Figure 5).

```
<LOD0>  S → m = C_1 …                  <\LOD0>
<LOD1>  S → m = C_1 …                  <\LOD1>
<LOD2>  S → m = C_1 ∪ C_2 R_2
        R_2 → ε…                       <\LOD2>
<LOD3>  R_2 → ∪ C_3 …                  <\LOD3>
```

Figure 5. Production rules after first iteration.

Now the production rule S of LOD$_1$ and LOD$_2$ are different and the non-terminal symbol R$_1$ is introduced. Because rule S is the same in LOD$_0$ and LOD$_1$, the symbol R$_1$ is mapped to the empty string in LOD$_0$ (and not in LOD$_1$) and to the difference in LOD$_2$. Production rule S is removed from LOD$_2$ and LOD$_1$ and symbol R$_1$ appended to the rule in LOD$_0$ (see Figure 6).

Once the differences have been encoded in the production rules, the non-terminal symbols C$_i$ and H$_j$ are replaced by the right side of their respective production rules if they are not altered in production rules of subsequent LODs. Recurring production rules in consecutive production rule sets can also be removed from all LODs with the exception of its first occurrence.

```
<LOD0>  S → m = C_1 R_1
        R_1 → ε …                      <\LOD0>
<LOD1>  …                              <\LOD1>
<LOD2>  R_1 → ∪ C_2 R_2
        R_2 → ε…                       <\LOD2>
<LOD3>  R_2 → ∪ C_3 …                  <\LOD3>
```

Figure 6. Final production rules after second iteration

If components or half-spaces are removed from one LOD to the next LOD, we employ the above mentioned trick by mapping components to the empty set and half-spaces to the universe.

6. RESULTS

The described transmission scheme has been implemented on a small example data set with three levels of detail (see Figure 7 to Figure 9). In the first LOD only block buildings with flat roofs are encoded (see Figure 7). All buildings are modeled as one convex component with the exception of the L-shaped building where two convex components are needed.

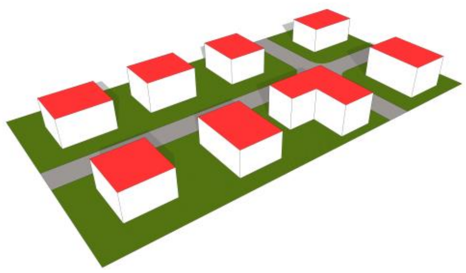
Figure 7. 3D example city model at LOD0

In the intermediate LOD, further half-spaces are inserted to each convex component to form the saddleback and hip roof structures. Also garages, large dormers, and the cross-gable are included as additional components (see Figure 8).

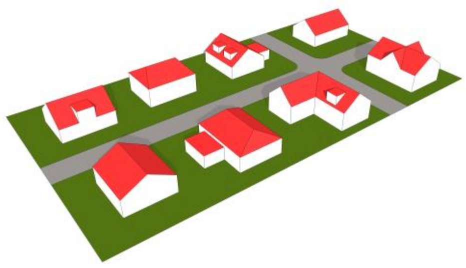
Figure 8. 3D example city model at LOD1

In the last LOD, the garages and the half-hip roof are shaped with additional half-spaces. Small dormers and chimneys are again included as convex components.

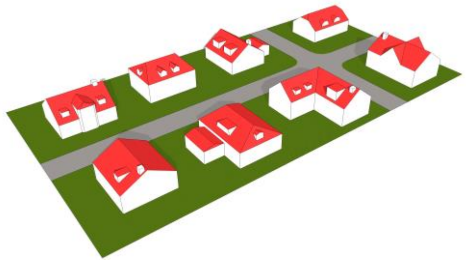
Figure 9. 3D example city model at LOD2

In order to transmit the 3D city model, a further hierarchy level is introduced that defines the city model to be a union of 3D building models. Such a technique could also be used e.g. to implement a cartographic aggregation of buildings.

7. CONCLUSION

We have presented a progressive encoding and transmission scheme for 3D building models that is based on string grammars that generate string representations of planar half-space models. Although the given examples are rather simple, they show the great potential of the approach with regard to its flexibility and extensibility. The production rules seem rather bloated in their chosen character string representation, but they can be more efficiently encoded in binary form. String transformations are also rather difficult to work with directly, but it is possible to formulate abstract operations and map them to production rules. Examples of insertion and removal operations have been given, but also more complex operations are possible. E.g. the splitting of a convex component into several components that together form a non-convex shape would be very helpful in formulating a progressive 3D building model.

As future work, we plan to introduce an implicit definition of animations to allow for smooth geometric transitions between LODs just by changing and transmitting half-space parameters, allow the encoding of textured 3D models, and also regard the optimization of the string representation in order to minimize the number of Boolean operations to describe hierarchical 3D models and to reduce the number of generated production rules.

8. REFERENCES

Ai, T., Li, Z., Liu Y., 2004. Progressive transmission of vector data based on changes accumulation model. In: *The 11th International Symposium on Spatial Data Handling*, Leicester, UK, pp. 85-96.

Becker, S., 2009. Generation and Application of Rules for Quality Dependent Façade Reconstruction. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 64 (6), pp. 640-653.

Becker, S., Peter, M., Fritsch, D., Philipp, D., Baier, P., Dibak, C., 2013. Combined Grammar for the Modeling of Building Interiors. In: *The International Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. II-4/W1, pp. 1-6.

Brenner, C., 2010. Building Extraction. In: George Vosselman, Hans-Gerd Maas (eds.) *Airborne and Terrestrial Laser Scanning*, Whittles Publishing, pp. 169-212.

Dehbi, Y., Plümer, L., 2011. Learning Grammar Rules of Building Parts from Precise Models and Noisy Observations. In: *ISPRS Journal of Photogrammetry & Remote Sensing*, Vol. 66 (2), pp. 166-176.

Foley, J., van Dam, A., Feiner, S., Hughes, J., 1990. *Computer Graphics: Principles and Practice (2nd ed.)*, Addison-Wesley, Reading, Massachusetts.

Gröger, G., Plümer, L., 2012. CityGML - Interoperable semantic 3D City Models. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 71, pp. 12-33.

Hongsheng, L.,Yingjie, W., Qingsheng, G., Jiafu, H., 2009. Progressive Simplification and Transmission of Building Polygons Based on Triangle Meshes. In: Guo, Huadong, Wang, Changlin (eds.) *Proceddings SPIE 7840, Sixth International Symposium on Digital Earth: Models, Algorithms, and Virtual Reality*, Beijing, China.

Hoppe, H., 1996. Progressive Meshes. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96), ACM Press, New York, NY, USA, pp. 99-108.

Haala, N., Kada, M., 2010. An Update on Automatic 3D Building Reconstruction. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 65 (6), pp. 570-580.

Kada, M., 2011. Aggregation of 3D Buildings using a Hybrid Data Approach. In: *Cartography and Geographic Information Science*, Vol. 38 (2), pp. 154-161.

Li, Q., Sun, X., Yang, B., Jiang, S., 2013. Geometric Structure Simplification of 3D Building Models. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 84, pp. 100-113

Mäntylä, M., 1988. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Maryland

Milde, J., Zhang, Y., Brenner, C. Plümer, L., Sester, M., 2008. Building Reconstruction Using a Structural Description Based on a Formal Grammar. In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVII, Beijing, China.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L., 2006. Procedural Modeling of Buildings. In: *ACM Transactions on Graphics, (SIGGRAPH '06)*, Vol. 25 (3), ACM Press, New York, NY, USA, pp. 614-623.

Open Geospatial Consortium Inc., 2011. John R. Herring (Ed.), *OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.* OGC 06-103r4, Version 1.2.1.

Ripperda, N., Brenner, C., 2009. Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. In: *Proceedings of 12th AGILE Conference on GIScience*, Hannover, Germany.

Royan, J., Balter, R., Bouville, C., 2006. Hierarchical Representation of Virtual Cities for Progressive Transmission over Networks. In: *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 432-439.

Sester, M., Brenner, C., 2009. A vocabulary for a multiscale process description for fast transmission and continuous visualization of spatial data. In: *Computers & Geosciences*, Vol. 35 (11), pp. 2177-2184.

Sipser, M., 2012. *Introduction to the Theory of Computation (3rd ed.)*, Cengage Learning, Boston.

Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W., 2003. Instant architecture. In: ACM Transactions on Graphics (SIGGRAPH '03), Vol. 22 (3), ACM Press, New York, NY, USA pp. 669-677.