Building Generation for Multi-Sensor Simulation

Carole Nissoux
Oktal Synthetic Environment
carole.nissoux@oktal-se.fr

Mathieu Larive
Oktal Synthetic Environment
mathieu.larive@oktal-se.fr

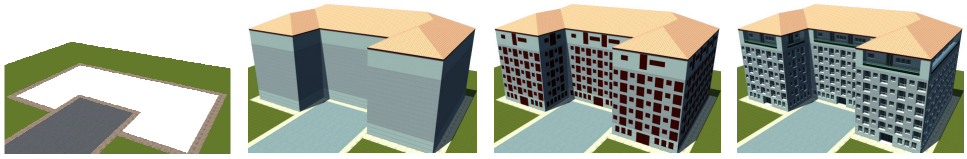

Figure 1 From left to right: Entry data (building footprint), first derivation step, setting up extrusions and final building.

ABSTRACT:

Urban Warfare training can be achieved with infrared/NVG embedded sensors, such sensors can also be used for infrared image recognition and target identification training. In the infrared domain, the 3D representation of a facade is necessary in order to have realistic representation with regards both to thermal computation and thermal effects. In the radar domain, this type of facade modelling is quite necessary with regards to edge and corner reflection effects. During the specifications of infrared enhanced vision systems or infrared embedded security systems, the usage of simulation reduces the cost and time required to validate a product. But to ensure the reliability of such virtual simulation, it is necessary to use a large panel of simulation tests. Creating these tests, and especially the one in urban area could be time consuming. The capacity to quickly generate credible digital city models helps the users to achieve their studies. Detailed modelling of realistic towns is a real challenge for computer graphics. Modelling a virtual city that is detailed enough to be credible is a huge task that requires lots of hours of work. In this context, automatic approaches can bring a real added value. We present a new technique to automatically generate building exteriors. Our technique relies on the definition of building templates that will be applied on building descriptions. Building frontages are generated using a 2.5D wall grammar based on a set of rules that can be simple or detailed enough to fulfil the users wishes. Our method is as easy to use as the texture repetition but provides a higher level of realism and diversity in the resulting buildings. Then little information is necessary to generate a whole building: the walls and roof height, the building 3D footprint and the chosen template. This information can be stored within a Geographic Information System.

INTRODUCTION

Recent improvements of applications in virtual reality, video games and simulation of city expansion have increased the suitability of digital mock-ups for urbanism studies. For example, urbanisation leads to emerging problems such as the influence of electromagnetic radiations, the prediction of the noise propagation to create noise pollution models or the forecast of urban transportation networks. On the other hand, the next generation of Global Navigation Satellite System (GNSS) will take advantage of city 3D models. The capacity to quickly generate credible digital city models helps the users to achieve their studies.

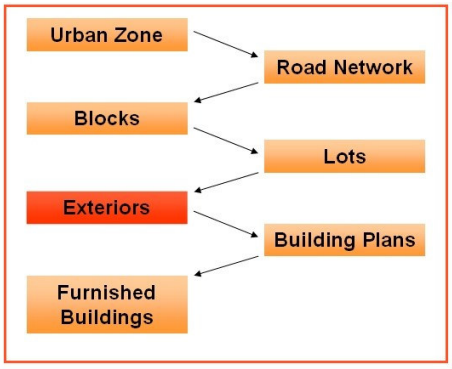

Figure 2: Hierarchical division of city generation

Detailed modelling of realistic towns is a real challenge for computer graphics. Modelling a virtual city that is detailed enough to be credible for a visitor is a huge task that requires thousands of hours of work. In this context, working on automatic approaches can bring a real added value. Such automatic approaches represent a promising research topic that must be developed since the current results do not satisfy the previously defined needs. As shown in Figure 2, the process of city generation can be divided into seven stages.

In this paper we focus on the stage that deals with the generation of building exteriors. We suggest to handle this stage by the use of building templates. A building template consists of three different sets of templates (roofs,

frontages and groundwork). The generation of groundwork can create polygons that connect the 3D building footprint to the building walls. The generation of roofs uses a straight skeleton based method to create various roof types. The generation of the frontages is based on a 2.5D parametric **wall grammar**. The main topic of this paper is the frontages generation method that **OKTAL-SE** has developed and its associated wall grammar.

The objective of the work described in this article is to create a system that imports building footprints defined by 3D polygons and building heights. Then the system uses these data to create 3D building models as specified by the user requirements. Typical usage will start with a 3D ground model obtained from a Geographic Information System (GIS). The building outlines and heights can be obtained automatically or defined by the user inside a GIS (that was the case for the example shown in Figure 14). Then, building templates have to be assigned to building outlines, this can be done manually, randomly or using any socio-statistical data available into the GIS. Once the setup has been performed, our system will create the building models taking into account all these previously defined information.

Figure 3: Details of the building described in Figure 1

We have designed a system able to generate any geometric details found on usual building frontages. Our system easily handles the vertical and horizontal alignments and repetitions used in almost every human construction. One important requirement was to focus on specific patterns, so our system also allows to benefit from building properties that can be identified inside a given building frontage. That is why we have created a wall grammar based system designed only for frontages in order to simplify its usage and take advantage of frontage specificities. Another purpose of our works is to facilitate on the one hand the resort of repetition schemes on every part of our frontages and on the other hand the usage of previously generated 3D objects (such as balconies or cornices).

1.1 OVERVIEW

This paper is organised as follows: Section 0 describes the wall grammar used to generate frontages, then Section 0 deals with the description of groundwork and roof templates. Our results can be found in Section 0, discussion about advantages and drawbacks of our tool in Section 0, and conclusions and future work in Section 0.

FRONTAGES TEMPLATES

This section constitutes the main contribution to our research work. A frontage template contains a list of keywords, a primary wall and potentially a default material. A frontage template can only be chosen if it is geometrically compatible with the frontage (i.e. the minimal and maximal width and height of the frontage template are compatible with the width and height of the frontage). Among the templates that can fit the frontage, the choice depends on the keywords stored by the user in the footprint data and the template. For example, the user can define the **blind** keyword for a frontage without any aperture, or the **entry door** keyword for the main segment of the building footprint. The frontage templates containing the **blind** or **entry door** keyword will then be preferred for those frontages. The default material is used by the walls that do not have a background material so as to ensure graphical coherence for a complete frontage.

1.2 WALL GRAMMAR

We have chosen a formal grammar representation to describe our frontages. We wanted to propose a system efficient and simple to understand and use. We decided to work on a **wall grammar** that can be seen as a **split grammar** that uses walls as its elements (instead of shapes).

We have reached a minimal set of five rules, a terminal one (see 1.2.2), two position rules (see 1.2.3 and 1.2.4) and two repetition rules (see 1.2.5and 1.2.6). This choice allows us to easily deal with instantiation and repetition of given walls.

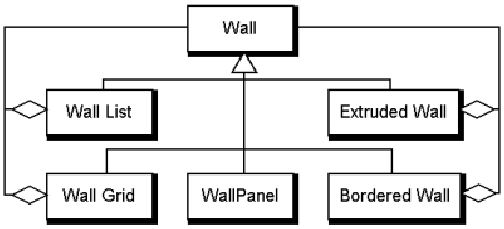

Figure 4: The relations between our classes/derivation rules

Formal grammar: it consists of a finite set of **terminal symbols** (the letters of the words in the formal language), a finite set of **non-terminal symbols**, a finite set of **production rules** with a left- and a right-hand side consisting of a word composed with these symbols, and a **start symbol**.

Each rule (wall) is able to determine its required space (min/max dimension in vertical and horizontal directions) by a simple computation on itself and its children. Then, among the geometrically possible primary rules (frontage template), one is chosen for the current frontage using the keywords defined in the templates and in the building description.

1.2.1 ABSTRACT WALL (*W*)

The abstract class **Wall** stores the information that is shared by the different rules (see Figure 4 for class diagram).

It deals with the background material as well as the final dimensions of the wall (horizontal and vertical). The minimal and maximal dimensions are computed depending on the wall's internal element size. Moreover, the user can define preferred dimensions that will automatically be used if they are geometrically valid.

In the following descriptions, W can be any instance of abstract wall, i.e. Bordered Wall, Extruded Wall, etc.

1.2.2 WALL PANEL (*WP*)

A **wall panel** is the simplest element of our system. It is our unique **terminal symbol**.

Besides the data that each wall shares (dimensions and background material), a wall panel can have an external reference to a 3D object or a decoration material (for example a texture of a window). In this case, the 3D object will be instantiated in the centre of the wall panel surface. When a 3D object is instantiated, the user can choose it to create the background faces (or not).

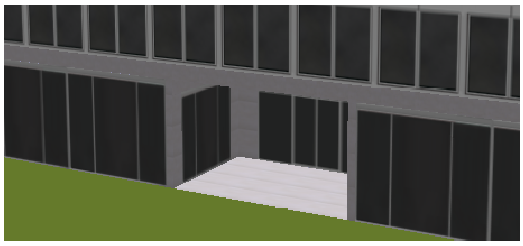

Figure 5: An example of background face suppression in a Wall Panel

Besides a lot of practical tricky cases have to be handled. For example, this is useful for 3D objects that model entry halls, because in this case a background face must be suppressed (in order to not hide the 3D objects placed inside the building), see Figure 5.

1.2.3 BORDERED WALL (*BW*)

A **bordered wall** (see Figure 6-left) is a wall with four margins (left, right, top and bottom) and a central element that references a wall.

Each margin has a size and a resize policy that can be **minimum**, **maximum** or **fixed**. The default value for the resize policy is minimum, which means that the border cannot be smaller than the defined size.

$$BW \rightarrow W$$

1.2.4 EXTRUDED WALL (EW)

An **extruded wall** (see Figure 6-right and Figure 8) is a wall that extrudes its child wall according to a given depth.

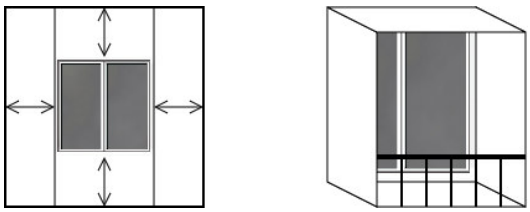

Figure 6: Left: bordered wall. Right: extruded wall with a negative depth

This depth can be positive (respectively negative): the child wall and its border faces are in front of the frontage (respectively the child wall goes inside the frontage). Figure 7 illustrates the usage of the depth.

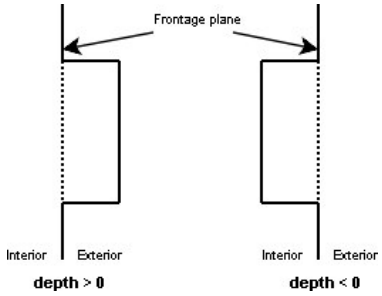

Figure 7: Positive and negative value of the depth of an extruded wall

Four Booleans are used to define if the depth faces have to be generated or not. Moreover, an extruded wall can contain a 3D model that will be instantiated on its surface (i.e. depth = 0). This 3D model is used for decoration purpose as the guardrail in Figure 6.

$$EW \rightarrow W$$

Figure 8: An example of an Extruded Wall with an external reference (a 3D model

This wall is important in our system since it allows to create relief on the frontage, which make them much more realistic (as illustrated by Figure 9).

Figure 9: Example of building with extrusion at different times of day

1.2.5 WALL LIST (*WL*)

A **wall list** contains several walls and an orientation that can be either horizontal or vertical.

The different walls will be created from left to right (for a horizontal wall list) or from bottom to top (for a vertical one) in order to cover the frontage zone assigned to the wall list.

$$WL \rightarrow W_1 W_2 \ldots W_n$$

1.2.6 WALL GRID (*WG*)

A **wall grid** contains a unique wall that can be repeated in one or two directions (vertically and/or horizontally).

For each of these directions, the number of repetitions is controlled by the minimum and maximum cardinalities (h times in the horizontal direction and v times in the vertical direction).

$$WG \rightarrow W^{(1-h)(1-v)}$$

GROUNDWORK AND ROOF TEMPLATES

1.3 GROUNDWORK TEMPLATE

A building footprint can be a non-planar polygon. Nevertheless, the floor of the building is assumed to be flat and horizontal. Groundwork is used to adjust each building with the ground. A groundwork template contains a groundwork type and a groundwork material texture if needed.

The different groundwork types currently available are:

- **Z min:** The frontage starts from the minimum footprint altitude, no additional faces are generated, but part of the frontages can be buried (some openings can be covered by the ground). This groundwork type is mainly used when the ground altitude variations are small or when the template is meant for that case (i.e. chalet template).
- **Z max**: The frontage starts from the maximum footprint altitude, no additional faces are generated. Typically, the building is floating above the ground. This groundwork type is interesting when the user wants to take care of the groundwork himself.
- **Extruded**: The frontage starts from the maximum altitude of the footprint and then groundwork faces are used to link the frontages to the ground. For each segment of the building footprints, some vertical faces are created between this segment and the original footprint beneath it.

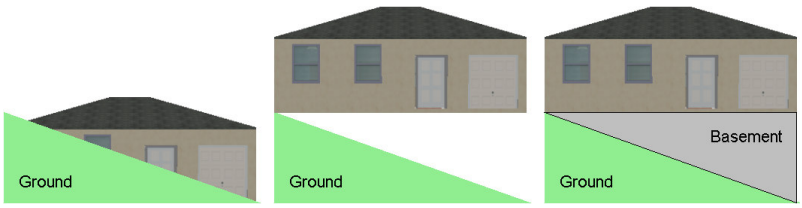

Figure 10: The three different kinds of groundwork templates. From left to right, zMin, zMax and Extruded

Figure 10 shows these groundwork types. The groundwork material will only be used in the case of an extruded groundwork. If no material is given by the template, the groundwork faces use the default material of the building template.

1.4 ROOFS

One of the objectives of this work is to be able to accept any non-intersecting polygon as entry data for our templates. It must even accept polygons with holes. This leads us to use the **Straight Skeleton** method [Eppstein and Erickson1998] on the polygons describing the building footprint. As this algorithm is valid for such polygons, we are able to offer several roof types that can always be applied on our buildings, whatever the configuration.

The available roof types can be classified into four categories: the **flat roofs**, the **one-sloped roof**, the **two-sloped roofs** and the **four-sloped roofs**. The first two categories contain only one roof. The flat roofs can have a border on their side, whose height is the same as the roof height. The one-sloped roof is obtained by putting the longest footprint segment at the roof height and its furthest segment at the wall height. As the building outlines are not restricted to quadrilaterals, the main difference between the last two categories is the presence of gables on the generated roofs: the four sloped roof templates do not create gables whereas the two-sloped roofs do. We will now detail our roof templates with two or four slopes. These templates were mainly inspired by the works on the two-sloped roof that can be found in [Laycock and Day 2003]. In order to define more precisely our different roof types, we introduce the notion of **flap** as a part of a roof slope divided into flat parts.

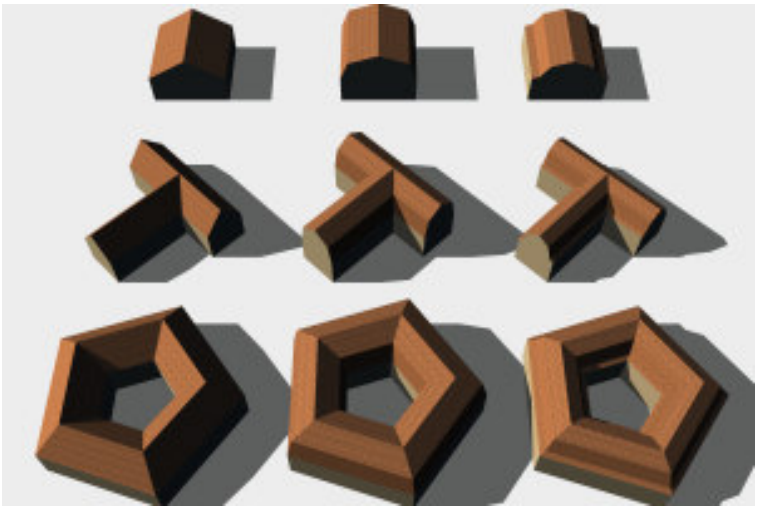

Figure 11: Examples of roofs for various two sloped roof templates. From left to right: standard two slopes, gambrel, shed gambrel

First, we are going to give details on the **two-sloped roofs** that can be seen in Figure 11:

- **standard two slopes:** This roof type creates vertical gable faces on the smallest segments of the building outline. These faces often use the default material defined in the building template in order to give a coherent aspect to the building. Furthermore, the gable faces are created by displacing skeleton points to the building outline segment (in the direction of the skeleton segment). The **regular faces** (roof faces that are not gables) are created with only one flap.
- **gambrel:** This roof type creates two slope roofs with gables. The regular faces is made of two flaps, the lowest one has a steeper slope than the upper ones.
- **shed gambrel:** This roof type creates three flaps for each regular roof face. It can be described as the addition of a quasi-horizontal flap at the bottom of a gambrel roof.

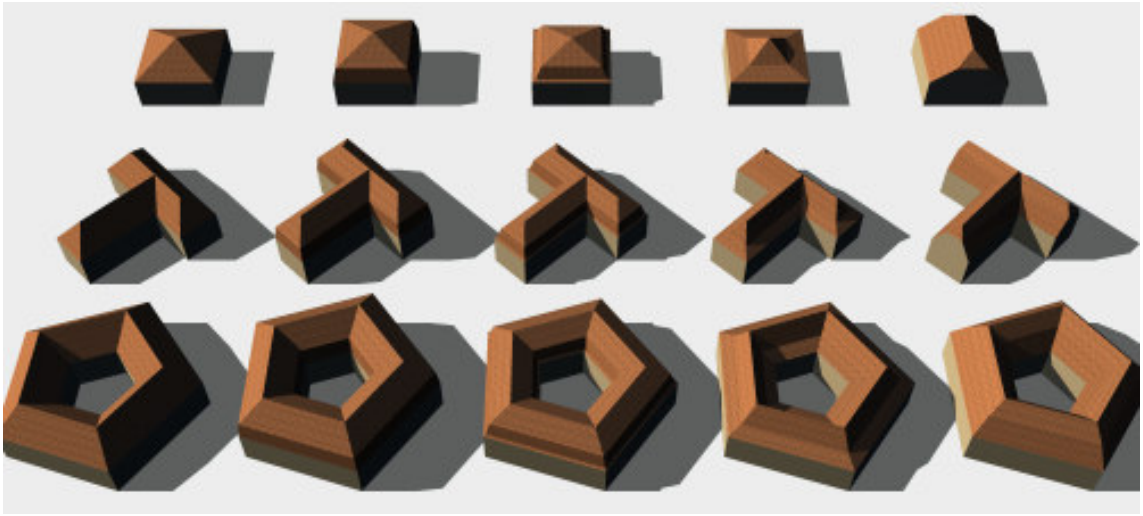

Figure 12: Examples of roofs for various four sloped roof templates. From left to right: four slopes, mansard, pagoda, porch, Dutch hip

Then, we are going to give details on the **four-sloped roofs** that can be seen in Figure 12:

- **standard four slopes:** This roof type creates only regular faces, with a constant slope (i.e. one flap per slope).
- **mansard:** This roof type can be seen as an extension of the gambrel one with only regular faces. Each slope contains two flaps, the lowest one has a steeper slope than the upper one.
- **pagoda:** This roof type aims at generating pagoda roofs. It can be seen as an extension of the shed gambrel ones with only regular faces that contain three flaps each.
- **porch:** This roof type creates only regular faces. Each slope contains two flaps, the highest one has a steeper slope than lesser one.
- **Dutch hip:** The roofs created by this roof type are hybrid roofs with an upper part (standard four sloped roof) placed on a standard two-sloped roof (with gables).

A roof template is defined by a roof type, an overhang type, an overhang size, a slab size and some materials (for the roof, gables, overhang and slabs). When the user wants to create roofs bigger than the building outline, he can use overhang. Then he can decide if they are closed, opened or have slabs. Slabs are vertical faces built along the outline of the roof. Overhangs and slabs can be defined for all our roof types.

RESULTS

Figure 13: Three different buildings generated from the same building footprint. Left: a Hausman building with a mansard roof (94 faces). Middle: a glass building, with 2 external objects for the entry hall and a flat roof (350 faces). Right: a highly extruded building with a four-sloped roof (5600 faces).

1.5 PERFORMANCE AND COMPLEXITY

One of our test scenes includes 17362 buildings. The building footprints were defined from an IGN planimetry file (IGN is the French National Geography Institute), each building footprint and height has been manually defined by a graphic designer. Then, this user has created his own building templates and has applied them on the buildings using the SE-AGETIM tool (cf. 1.6). A view of the final scene can be seen in Figure 14, the building generation took 7 minutes and 55 seconds, and 920,182 faces were generated.

Figure 14: Urban area made of 17 362 buildings, the building generation took 7mn and 55sec, 920 182 faces were generated.

In order to test the complexity of our building templates, we have used them on this scene. Using the simplest of our building templates (comparable to the one on the left in Figure 13), we have generated this scene in 6 minutes and 51 seconds. The final scene contains 618,050 faces. Using the most complex of our building templates (comparable to the one on the right in Figure 13), we generated this scene in 3 hours and 27 minutes. The final scene contains 25,449,776 faces.

These results have been obtained on a 2GHz AMD Athlon 64. The screenshots have been generated using the **OKTAL-SE** multi-sensor ray casting engine **SE-RAY** (Figure 1, Figure 11, Figure 12 and Figure 13) and the **OKTAL-SE** real time viewer **SE-GPF** (Figure 3, Figure 14 and Figure 16).

1.6 INTEGRATION

Our system is implemented in C++, while our building templates are described and stored within a XML framework (**OKTAL-SE** XIO format). This project was developed so as to create abstract geometry, thus we do not depend on a given software or 3D library. Currently, only one implementation is available.

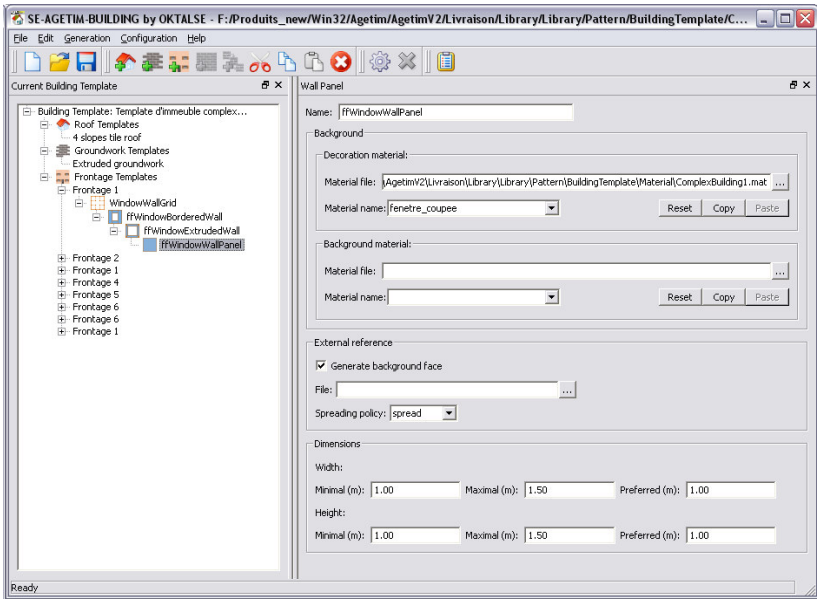

Figure 15: Current state of our frontage template editor.

Our building template description and generation system has been integrated to the **OKTAL-SE SE-WORKBENCH/CHORALE tool suite** to form the **SE-AGETIM-BUILDING** module of the **OKTAL-SE** terrain generator **SE-AGETIM**. This integration to the SE-AGETIM tool allows the user to import various GIS data, such as VMAP, DTED, DFAD SEDRIS or DXF data and then create a set of building footprints, choose the desired building templates and generate the building as well as the surrounding terrain. This implementation is meant to generate large complete zones (currently up to hundreds of thousand km²).

The SE-AGETIM-BUILDING building template editor allows the user to create and edit his own building templates and to calibrate them by testing the frontage and roof templates one by one against any kind of footprint. Figure 15 shows the GUI of this building template editor.

The left panel shows the current state of the building template. The right panel allows the user to edit the current building template element (groundwork template, roof template, frontage template, walls).

A library of very generic building templates has been developed by **OKTAL-SE** and is delivered with the SE-AGETIM tool (in the version including the SE-AGETIM-BUILDING module). Those generic building templates are currently used during large terrain mock-ups generation to populate them with buildings.

OKTAL-SE also used this system to create very realistic mock-ups that can be corrected and regenerated in a minimum of time. Figure 16 shows an example of a mock-up of Toulouse down town.

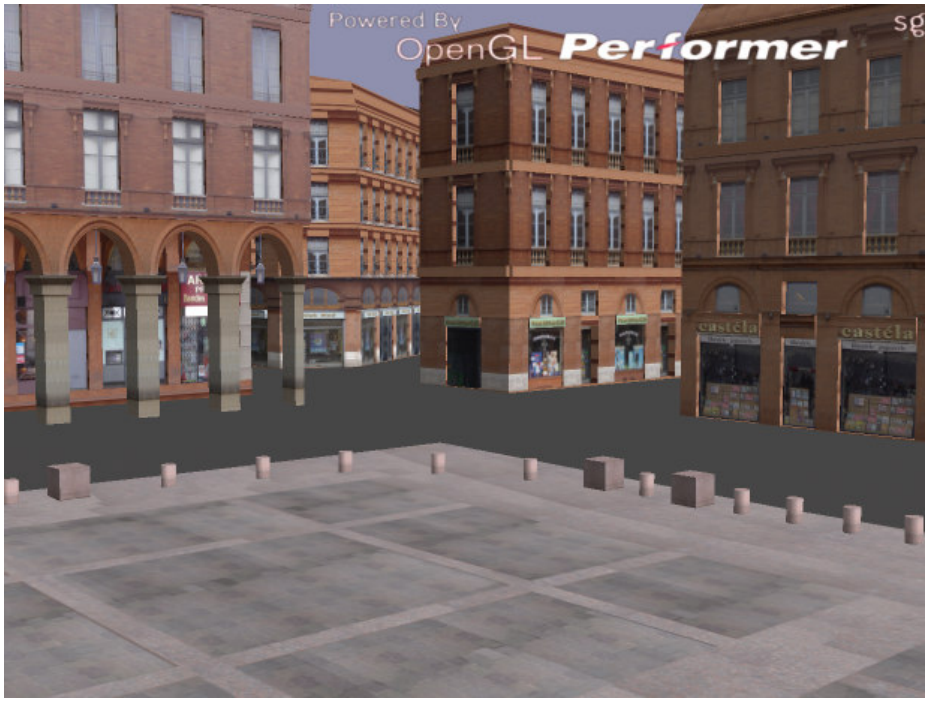

Figure 16: View of the Toulouse mock-up: automatically generated building

The geo-typical SE-AGETIM-BUILDING approach applied to such very geo-specific modelling cases shows the power of the method, compared to a purely manual modelling : The realism is comparable to a manual modelling process but every building can be modified and regenerated automatically, allowing the user to easily and quickly play with geometrical frontage details, especially the ones repeated a lot of times such as windows or balconies. Besides, the building templates can be reused in other mock-ups.

DISCUSSIONS

Number of rules: As the user can define the complexity of his templates, our system can be used to generate credible digital city models as well as cities used for aircraft simulation (less detailed) with our five rules. Even though it is theoretically possible to create a unique template including all of our rules, this system has been preferentially designed so as to enable the user to describe one distinct kind of building. A distinct approach, based on the use of a larger rule database, allows the user to describe buildings using only high-level attributes. Future works about our rule selection mechanism will allow us to create such kind of complex building templates even if it was not our primary goal.

Usability of the system: Several of the templates that have been presented in this paper were created by a graphic designer. Once the user has understood a given frontage template, he can begin to create his own templates by modifying the materials. Then, he can experiment by changing the wall template parameters (dimensions, cardinality for a wall grid, border size and policy for a bordered wall etc.). After these two learning phases are performed, the user knows the five different kinds of rules, and their parameters. From this moment, he is able to create new templates by himself from scratch.

Geometric complexity: As shown in Figure 13, our buildings can contain a limited number of faces. We are able to control the complexity of the generated buildings, using levels of details, textures, geometry simplification (merging coplanar faces) and appropriate templates. It is also possible to control the complexity of buildings by instantiating more often the biggest frontage part (i.e. the ones with less polygons per surface unit).

Multi-sensor simulation: As our buildings are generated using the **OKTAL-SE** multi-domains materials system, they are "naturally" designed for multi-sensor simulation: as long as the user enhanced the materials with the necessary physical properties, they can be used for simulation in any domain (infrared, SAR, etc.).

CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced a new pragmatic approach for generating buildings from their footprints. The main advantages of our work are:

- Generation for any building footprint, convex or not, even with holes.
- Creation of simple or complex frontages, according to the user requirements.
- Availability of various kinds of roofs independently of the footprint complexity.
- It has already been integrated to (and used in) the **OKTAL-SE** terrain modeller, SE-AGETIM.
- Creation of frontages (from photos) of existing buildings.

A current use of our system is the generation of 3D models of cities for the calibration of SAR (Synthetic Aperture Radar) simulation such as the ones described in **[Soergel et al. 2005]**. Our system is able to create cities of a given complexity that helps to validate the results of this kind of simulation. We have to enhance our roof templates in order to enable the user to instantiate 3D models on the roofs, because roof details are difficult to deal with in SAR simulation.

We intend on one hand to improve the source data quality, typically with regard to footprints, using GIS level correction procedures included in the SE-AGETIM tool and on the other hand to improve the realism of our frontages and roofs, as mentioned above, by handling for example complex apertures, chimneys, antennas, zinc work, gutters, etc.

Figure 17: Preliminary results of our road network generation system. The buildings are created using the system described in this article

Finally we also work on the unification with the **SE-AGETIM-INDOOR** module of the SE-AGETIM tool, which allows the user to generate single building with indoor content (room dividers, furniture, etc).
The final goal of our work is to propose a complete city creation system. As shown in Figure 2, the generation of city models can be divided into seven stages. As we are already capable of producing furnished buildings (see **[Larive et al. 2004]** and **[Xu et al. 2002]**), our next research topic will deal with the generation of road networks, lots and building footprints (see Figure 17 for our current results). Once these stages will be completed, we will propose a complete city creation system.

Related Work

[Eppstein and Erickson1998]
Eppstein, D. & Erickson, J.G. 1998. *Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions*. In *Proc. 14th Symp. Computational Geometry*. ACM, 58-67.

[Geoconcept]
Geoconcept, a Geographic Information System developed by the Geoconcept SA company. http://www.geoconcept.com.

[Gerth et al. 2005]
Gerth, B.; Berndt, R.; Havemann, S. & Fellner, D.W. 2005. *3D Modeling for Non-Expert Users with the Castle Construction Kit v0.5*. In *Proceedings of ACM/EG VAST 2005*.

[Haveman 2005]
Haveman, S. 2005. *Generative Mesh Modeling*, PhD Thesis, TU Braunshweig, Germany.

[Joly et al. 2006]
Joly, A.; Le Goff, A; Latger, J.; Cathala, T.; Larive, M. *Real Time optronic simulation using automatic 3D generated terrain, and validation process. ITBMS 2006.*

[Larive et al.2005]
Larive, M.; Dupuy, Y. & Gaildrat, V. 2005 *Automatic Generation of Urban Zones*. *WSCG'2005*, 9-12.

[Larive et al. 2004]
Larive, M.; Le Roux, O. & Gaildrat, V. *Using Meta-Heuristics for Constraint-Based 3D Objects Layout. 3IA'04.*

[Latger et al. 2006]
Latger, J.; Le Goff, A.; Cathala, T.; Larive, M. *Automatic 3D virtual scenes modeling for multi sensors simulation. SPIE 2006.*

[Laycock and Day 2003]
Laycock, R.G. & Day, A.M. 2003. *Automatically Generating Roof Models from Building Footprints*. In *WSCG*.

[Mitchell 1990]
Mitchell, W.J. 1990. *The Logic of Architecture: Design, Computation, and Cognition*. MIT Press.

[Müller et al. 2006]
Muller, P.; Wonka, P.; Simon, H.; Ulmer, A. & Van Gool, L. 2006. *Procedural Modeling of Buildings. In SIGGRAPH.*

[Parish and Müller 2001]
Parish, Y. & Muller, P. 2001. Procedural Modeling of Cities. *ACM SIGGRAPH.*

[Prusinkiewicz and Lindenmayer 1991]
Prusinkiewicz, P. & Lindenmayer, A. 1991. *The Algorithmic Beauty of Plants*. Springer Verlag.

[RauChaplin et al. 1996]
RauChaplin, A.; MacKayLyons, B. & Spierenburg, P. 1996. *The LaHave House Project: Towards an Automated Architectural Design Service*. *Cadex'96, pp. 24-31.*

[Soergel et al. 2005]
Soergel, U.; Michaelsen, E.; Thoennessen, U. & Stilla, U. *Potential of High-Resolution SAR Images For Urban Analysis*. In *URBAN.*

[Stiny 1975]
Stiny, G. 1975. *Pictorial and formal aspects of shape and shape grammars: on computer generation of aesthetic objects*. Ed. *Birkhauser.*

[Wonka et al. 2006]
Wonka, P.; Müller, P.; Hanson, E.; Watson, B. & Neil, E. 2006. *Course Procedural Modeling of Urban Environments*. Tech. Rep., SIGGRAPH.

[Xu et al. 2002]
Xu, K.; Stewart, J. & Fiume, E. Constraint-Based *Automatic Placement for Scene Composition.* Graphics Interface'02.