Rapid modeling of complex building façades

D. Finkenzeller and A. Schmitt

Institut für Betriebs- und Dialogsysteme, Universität Karlsruhe (TH), Germany

Abstract

Architectural settings occur in many virtual environments. Modeling such highly detailed structures manually is a time-consuming and tedious task. Therefore we propose a system for the rapid modeling of building façades. The designer just has to provide some coarse information, the building outline, its type and style and the computer takes care of creating the exact geometry. Adjacent architectural structures are adapted automatically, e. g. walls are adapted to exactly fit quoins made of different stone blocks. Also highly detailed façade elements like window frames made of single bricks, cornices, etc. are generated. Even at this point the designer can still interfere and change the building outline, the style, as well as window or door frame styles. Such a tool will relieve the designer of the burden of tedious, recurring and over all time-consuming modeling tasks.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

With advances in computer hardware the size and complexity of virtual worlds increases dramatically, like in massive multiplayer online role-playing games (MMORPG) as for example in World of Warcraft [BE06]. Therefore, the modeling of detailed scenes becomes a more important topic and will consume much time and human resources. To tackle this problem we focus on the modeling of highly detailed façades. One problem is that the manual modeling of detailed façades with tools like Maya [Aut06b] or 3D Studio Max [Aut06a] is a very time-consuming and tedious task. Another problem is that models once created cannot be easily changed.

The main task the user or designer has to accomplish is to model a façade with computer support. In this paper, we propose a system which allows the user to create highly detailed façades with ease. The system also supports the possibility to change the style of the whole façade or parts of it very quickly. This relieves the designer of the burden of difficult modeling tasks and gives him a high level control.

Our approach is based on a clear separation of the modeling task in which both, designer and computer have their distinguished duties. The designer provides the coarse building outline, its type and style. Based on this information the machine generates detailed façade structures for door and window holes, their frames, cornices, etc. Adjacent architectural structures are adapted automatically so that geometry does not overlap. At the moment we work on a texture generator that produces textures which exactly fit the geometry to avoid unwanted effects, like structures not adapted to others. The designer can change façade parameters on a high level and the machine does all the necessary recomputation. As a consequence the designer produces more complex structures in less time. He can try different styles on the same building outline and produce higher quality façades and therefore save human resources and money.

2. Related Work

Many different techniques do contribute to the task of automated building and façade generation. Here we introduce some of the most relevant ones.

For procedural modeling there are mainly two common techniques. The first technique are rewriting systems which operate on strings. Especially Chomsky's work on formal grammars caused a huge interest in rewriting systems. A condensed introduction is given in [Sch92]. The second technique uses L-systems. The biologist Aristid Lindenmayer proposed this mathematical formalism—which also uses rewriting—in 1968 as a foundation for an axiomatic theory of biological development [PL96]. The basic idea is

to use a set of rewriting rules or productions to create complex objects by successively replacing parts of a simple object. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel, replacing simultaneously all letters in a given word.

There are several research papers concerning the procedural modeling of whole cities using grammars and L-system like [PM01], [GPSL03], and [Arn00]. They mainly focus on creating whole cities based on a set of production rules. Parish and Müller [PM01] create the geometry for the buildings with a parametric stochastic L-system. For the façades they use procedurally generated textures.

Wonka et al. [WWSR03] introduce split grammars for the generation of buildings. They represent the façade as a non-terminal shape which can be further split into smaller non-terminal shapes leading in the last step of splitting to terminal shapes like windows, wall elements etc. In [BBJ*01] and [BJDA01] Birch et al. present techniques for the interactive modeling of buildings and architectural structures. They focus on the reduction of the number of parameters to a manageable size and to generate details procedurally. Legakis et al. [LDG01] present a method for cellular texturing for architectural models. Upon the underlying geometry they perform texturing operations considering the interdependencies between cells for vertices (corners), edges and faces. Havemann et al. [HF01] use the combination of polygonal mesh modeling and subdivision surfaces, which they call *Combined BRep*, for modeling architectural structures like ornaments and window frames. Particularly, a multi resolution approach is proposed where a view dependent refinement of a coarse structure is done at runtime.

The last paper mentioned leads us to the important objective of adequate geometry representation. Mäntylä [Män88] introduces different boundary models. Faces, edges and vertices are represented in a tree like structure not only describing the geometry model but also connection information between faces, edges and vertices.

3. Modeling process

To model a building like the one in Figure 7, the user has to draw the coarse outline and choose the appearance from a given set of styles. The information is then stored in a graph structure, representing the building in a symbolic way with necessary geometrical information. Upon this information the entire building is automatically created. At any time the user can modify the information in the graph. Each level, even each window etc. can be assigned an individual style from the set. Completely new styles for window or door frames and cornice profiles are created using a simple Logo-like [MAA84] description language. The effort and amount of time needed to model such a building ranges from a few minutes—if predefined styles are used—to about one or two hours—when completely new styles are created.

4. Building outline

Our main goal is to have arbitrary floor plan outlines for every level in a building. For this purpose we follow the aspect of a vector oriented approach. Additionally we have to take two aspects into account. First we discovered that it is necessary to have basic architectural information, e. g. the location of projections on walls, balconies, etc., on an early stage of modeling. The second is that we need spatial information of adjacent architectural structures both on a single level and between two adjacent levels.

In the next two subsection we describe the methods which meet our demands.

4.1. Floor plan outline

To achieve arbitrary floor plan outlines with necessary information about basic architectural structures, we compose convex 2D polygons to a floor plan outline. We refer to a convex polygon as a *floor plan module* or *fpm*. Each fpm represents an architectural structure in the façade with necessary information, e. g. type, material, basic geometrical information, etc. To receive a single floor plan outline several fpms are combined. The connection between two fpms has the following information:

- the two connected fpms: fpm_a and fpm_b, one of them to be *dominant*;
- their connecting edges: e_c of fpm_a and e_d of fpm_b;
- start and end point of the connection on the dominant fpm's edge: $p_S, p_E \in [0,1]$, $p_S < p_E$;

Figure 1(a) shows an example of two connected fpms: fpm_2 connected to fpm_1 via their edges e_5 and k_1. The floor plan in Figure 1(b) is composed of six fpms and shows the first level floor plan from Figure 7. The colored lines represent the outline of the floor plan. Each line strip of the same color depicts a distinguished façade element. The yellow, green and blue line strips represent façade projections.

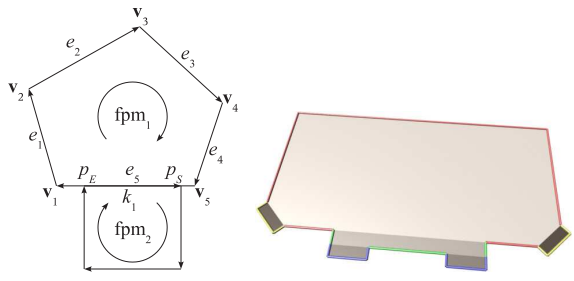


Figure 1: *Floor plan modules.*

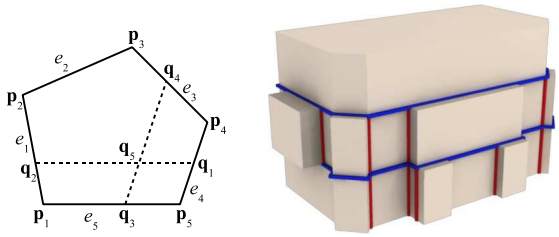
4.2. Multiple levels

When extending a single floor plan to buildings with multiple levels, we start upon the fpms on the first level. The

subsequent fpms arise from their underlying fpms. We destinguish three methods. An underlying fpm can be:

- omitted, so no fpm will be created for the next level,
- fully present on the next level or
- subdivided as shown in Figure 2(a), each convex combination of the subdivisions can form a new fpm for the next level or
- extend, a new fpm is added, e. g. create an oriel.

During the operation, necessary connections will be retained. To support structures like oriels or balconies, etc. the newly created fpms receive connections to fpms representing these structures, like the oriel on the left side on the second level shown in Figure 2(b). Also depicted in Figure 2(b) are the seams of adjacent structures—red for intra level and blue for inter level—which are automatically generated.



(a) (b)

Figure 2: *Subdivided fpm and seams of adjacent structures.*

5. Façade representation

Now that we have a representation for the coarse outline of a building, the next step is to generate walls and corners with detailed structures.

5.1. Corners and walls

Basic walls and *corners* arise from the coarse outline of every level's floor plan. Basic walls are further subdivided into *walls* and optionally *wall spacers*. Wall spacers and corners can be refined to stacks of arbitrary blocks. In Figure 7 the quoins consist of single blocks as well are the elements between the walls.

5.2. Cornices

Additionally corners, walls, and wall spacers can be supplied with cornices. The contour of a cornice is defined via a Logo-like [MAA84] description language, reduced to two simple commands *angle* and *arc*. Angle draws a straight line (initially starting at the origin) of a given length and direction whereas arc draws an arc with a given radius and sweep angle. Figure 7 shows several different cornice types on every level of the building.

5.3. Doors and windows

For doors and windows the first step is to define coarse rectangular holes relatively to a given wall. In the next step the four edges of the hole can be refined. Each edge can be replaced by a polygon defined with the description language we use for the cornices (the dotted line in Figure 3). Additional parameters for offsets to the left d_l, right d_r, and top d_t can be set. In Figure 3 the original edge $(\mathbf{f}_l, \mathbf{f}_r)$ is refined to the dotted line between the edge $(\mathbf{l}, \mathbf{r})$.

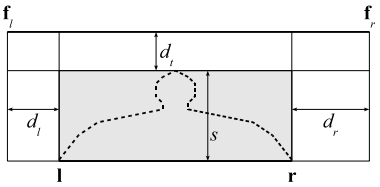


Figure 3: *Example refinement for a window edge.*

The inner part of the hole is now filled with a frame for each refined edge. Figure 7 shows different frame types that can be used: simple, cornice and bricks.

The frames for the window pane are kept very simple. They run along the inner edges determined by the above defined frames. The have a central cross and are of type cornice. All windows in Figure 7 have such window pane frames.

6. Roof generation

To control the appearance of the entire roof we use the individual floor plan modules. Every top level fpm has its own roof type information. Figure 4 shows three different roof types. When the entire roof is generated, the roof information of adjacent fpms are combined to a single roof. The step from single fpms roofs to an entire roof is shown in Figure 5.

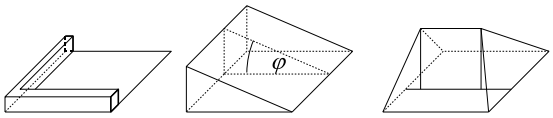


Figure 4: *Three roof types: flat, pent, and gable roof.*

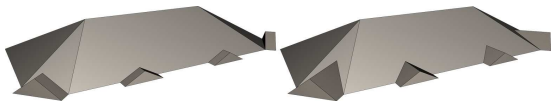


Figure 5: *Step from single roofs to an entire roof.*

7. Geometry engine

The information about the façade, including fpms, walls, windows, etc., is represented in an hierarchical structure as shown in Figure 6. The geometry engine considers its data and produces the detailed geometry. An example of an entire façade is given in Figure 7.

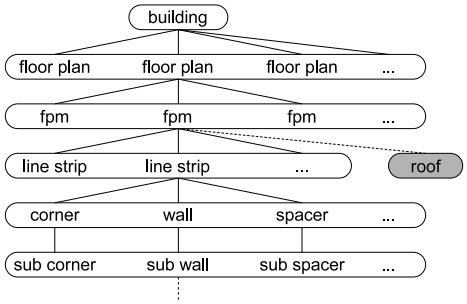


Figure 6: *Façade hierarchical structure.*

Figure 7: *Façade created with the actual system.*

8. Conclusion and future work

In this paper we proposed a system for the rapid modeling of façades, in which a hierarchical structure is used. A major limitation of our approach is that it can only describe classical building styles, e. g. organic structures are not possible. In future the rules that describe different façade styles, its proportions, etc. will be represented in a knowledge base. With a graphical user interface the designer draws the coarse building outline and the knowledge base is queried to produce the detailed façade structures. To have a more realistic appearance for the walls, we are developing a texture generator to create adapted brickworks.

References

[Arn00] ARNOLD D.: Economic reconstructions of populated environments - progress with the charismatic project, 2000.

[Aut06a] AUTODESK: *3D Studio Max*, 2006. http://www.autodesk.com/3dsmax.

[Aut06b] AUTODESK: *Maya*, 2006. http://www.autodesk.com/maya.

[BBJ*01] BIRCH P. J., BROWNE S. P., JENNINGS V. J., DAY A. M., ARNOLD D. B.: Rapid procedural-modelling of architectural structures. In *Proc. of the 2001 conference on Virtual reality, archeology, and cultural heritage* (2001), ACM Press, pp. 187–196.

[BE06] BLIZZARD-ENTERTAINMENT: *World of Warcraft*, 2006. http://www.worldofwarcraft.com.

[BJDA01] BIRCH P., JENNINGS V., DAY A. M., ARNOLD D. B.: Procedural modelling of vernacular housing for virtual heritage environments, 2001.

[GPSL03] GREUTER S., PARKER J., STEWART N., LEACH G.: Undiscovered worlds - towards a framework for real-time procedural world generation. In *Proc. of the Fifth Intern. Digital Arts and Culture Conference* (2003).

[HF01] HAVEMANN S., FELLNER D.: A versatile 3d model representation for cultural reconstruction. In *Proc. of the 2001 conference on Virtual reality, archeology, and cultural heritage* (2001), ACM Press, pp. 205–212.

[LDG01] LEGAKIS J., DORSEY J., GORTLER S.: Feature-based cellular texturing for architectural models. In *Proc. of the 28th conf. on Computer graphics and interactive techniques* (2001), ACM Press, pp. 309–316.

[MAA84] MACDOUGALL A., ADMAS T., ADAMS P.: *Learning Logo on the Apple II*. Simon & Schuster, 1984.

[Män88] MÄNTYLÄ M.: *An Introduction to Solid Modeling*. Computer Science Press, Maryland, 1988.

[PL96] PRUSINKIEWICZ P., LINDENMAYER A.: *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, USA, 1996.

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *Proc. of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 301–308.

[Sch92] SCHÖNING U.: *Theoretische Informatik kurz gefasst*. BI Wissenschaftsverlag, 1992.

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. *ACM Trans. Graph. 22*, 3 (2003), 669–677.