Semiautomatic Rule Assist Architecture Modeling

Hua Liu, Hongxin Zhang, and Hujun Bao

State Key lab of CAD&CG, Zhejiang University, Hangzhou, China
{sun_day, zhx, bao}@cad.zju.edu.cn

Abstract. This paper presents a novel rule-driven architecture modeling technique. Different from grammar based procedural modeling approaches, our proposed method, called *rule assist architecture modeling (RAAM)*, tends to integrate user interactions with implied modeling rules. Construction rules, configure rules and constrain rules are introduced in our method to minimize user interactions and enhance modeling efficiency. The experimental results demonstrate the efficiency and flexibility of our method to generate villas and skyscrapers.

1 Introduction

Architecture models are widely used in many computer graphics applications. Commonly, people use commercial modeling software [9, 10] to build 3D architecture models with high details. It is a tedious task to model thousands of different buildings, which are often used in virtual cities and urban reconstruction. Recent years, grammar based procedural modeling methods are introduced to model buildings [6, 8] and synthesize virtual cities [7] efficiently. Unfortunately, aforementioned methods are only good at modeling virtual buildings in similar style. To model buildings in different styles, the user has to define rules for each of them, which turns out to be inefficient. Moreover, writing desirable production rules is only possible for professional users. Although the priority can be assigned to each rule in [8], grammar based methods do not allow enough control mechanisms over the architecture modeling process.

In this paper, we present an interactive architecture modeling approach which is assisted by rules, called *rule assist architecture modeling (RAAM)*. Different from existing grammar based modeling approaches, the modeling process of our RAAM is mainly controlled by user interactions. The assist rules maintain construction information, default configuration data and modeling constrains of the target models. In our modeling system, rules are activated and executed automatically to reflect user interaction and commands. Construction rules are utilized to reduce user interactions and help users to complete the nonsignificant part. Additionally, the RAAM also allows users to modify predefined rules. Our proposed approach can help users to control the architecture modeling process more efficiently than previous method.

The rest of the paper is structured as follows. After reviewing procedural modeling methods in section 2, we introduce the assist rules in section 3. Then

the user interface are presented in section 4. In section 5, implementation details are provided and several modeling examples are demonstrated by using our prototype system. At last, we draw conclusions and discuss future work in section 6.

2 Relate Works

Rule and grammar based modeling techniques were developed in the first years of 20th century, such as L-System and parameter L-System, which are mainly used in plant modeling [2–5].

In recent years, several researchers extended the rule based techniques and leveraged them in other modeling domains. In [7], an extended L-System is applied to generate streets of a virtual city. Population density, height map and water map are employed as the input parameters that influenced the generation of the streets. After that, split grammar [6], which is a subset of the shape grammar [1], is proposed to describe the rules of shape splitting. Control grammar is also introduced in that paper, which is used to distribute the attributes of the modeling rules and selecting the next executed rule by matching attributes. The modeling process started from a bounded simple shape, and then split it recursively to form a complex shape with many details progressively. By using the splitting rules, [16] can produce a class of models automatically. More recently, [8] addressed application related details in the context of procedural modeling of buildings, such as the definition of the context sensitive shape rules and the concise notation. It also address the intersection problem of the architecture modeling process.

As formalizing knowledge of architecture construction is the essential part of our method, we would recommend starting with books that emphasize structure of architecture, such as a visual dictionary [11], the Logic of Architecture by Mitchell [12], Space Syntax [13], Design Patterns [14], and studies of symmetry [15].

3 Assist Rules

Comparing with the traditional modeling systems, our *RAAM* has assist rules between users and operation commands. For users, they only need to tell the system where and what components they want to create. Then the assist rules will translate their designs into real operation commands. Additionally, assist rules also detect the correction of modeling results and call the corresponding commands automatically, when it is necessary. Obviously, the key part of *RAAM* is the assist rules, which formalize the knowledge of architecture construction. The assist rules work with a configuration of components and operation commands. Here, we introduce them first:

Component: A component consists of a symbol, geometry (geometric attributes) and numeric attributes. Components are identified by their symbols

which is either a *terminal symbol* (e.g., door or window) or a *non-terminal symbol* (e.g., hall or veranda). The corresponding components are called terminal components and non-terminal components. The most important geometric attributes are the center position P and a size vector S. These attributes define an oriented bounding box of component in space. All the result models are constructed by terminal components non-terminal components.

Operation Commands: An operation command is described as $C()$, which can be a transform function, a creating function or a editing function etc. Operation commands exactly define the action on the components.

The assist rules not only describe the construction information, but also include the default configuration and modeling constrains. So we divide the assist rules into three categories: *construction rules*, *configure rules* and *constrain rules*.

3.1 Construction Rules

A class of buildings with similar style are always consist of a set of finite components. The generation of the components are described by the construction rules, which are defined as a set R in our paper. Similar as the production rules presented in [8, 16], for each r, $r \in R$ is represented as Equation. 1,

$$\alpha \rightarrow C_1(\beta_1)C_2(\beta_2)\cdots C_i(\beta_i), \tag{1}$$

where α and β_i are components. C_i are operation commands. Terminal components are created by simple shapes or loaded directly from the component library, such as [17]. For example, the window component, which is loaded from the library, is described by the construction rule: $window(s,p) \rightarrow Load(name,s,p)$, where *name* indicates which component to be loaded and s,p are attributes of the window. Body components are created by blocks which are represented as: $body(s,p) \rightarrow T(CreateBlock(s),p)$, where T is a transform function.

Non-terminal components, such as hall, veranda etc., consist of terminal components or non-terminal components. Here, we will show how a veranda component is constructed. The related rules are shown as follow:

Table 1. rules for generate a veranda

id	construction rules
1.	$veranda(s,p) \rightarrow vB(V_{bottom}(s,p))vL(V_{left}(s,p))vR(V_{right}(s,p))vF(V_{front}(s,p))$
2.	$vB(s,p) \rightarrow T(CreateBlock(s),p)$
3.	$vL(s,p) \rightarrow T(CreateBlock(s),p)$
4.	$vR(s,p) \rightarrow T(CreateBlock(s),p)$
5.	$vF(s,p) \rightarrow T(railing(s),p)$
6.	$railing(s) \rightarrow Repeat(pillar(R_s(s)),R_n(s))railing_{top}(R_{top}(s))$
7.	$pillar(s) \rightarrow CreateCylinder(P_{top}(s))CreateCylinder(P_{bottom}(s))$
8.	$railing_{top}(s) \rightarrow CreateBlock(s)$

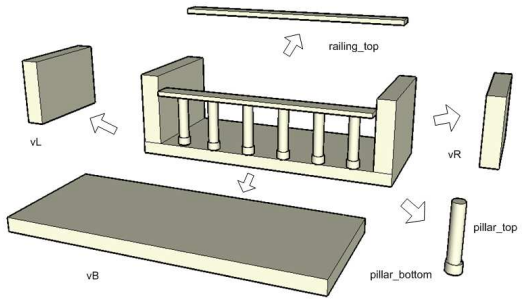


Fig. 1. construction of veranda

Here, $V_{bottom}, V_{left}, V_{right}, V_{front}, R_s, R_n, R_{top}, P_{top}, P_{bottom}$ are configure rules which will be introduced in the next subsection. $CreateBlock$ and $CreateCylinder$ are both operation commands, which will create a parameterized basic shape immediately. $Repeat(s, n)$ is also an operation command, which make a component cloned n times. As shown in Fig. 1, we can see how a veranda is built up. In this process, users only need to specify parameters of a veranda and the whole generation work is done automatically by assist rules.

3.2 Configure Rules

In our modeling process, the user do not need to specify all modeling parameters. Most of them can be calculated by the corresponding components. For example, a window or a door component always has a same dimension value of p with the wall, which it belongs to. The bottom of a roof component always has the same size with the top of the corresponding body component. The others can be determined by domain knowledge, e.g., the width and height of each step of a stair component always has a fixed scale.

At some time, if the default value generated by configure rules conflicts with users' purpose, we also allow them to edit the default configure rules or utilize the interaction tools to set the value directly. If most of the configure rules have satisfied users' purpose, it could be great helpful to improve the modeling efficiency.

A configure rule consists of a symbol (string) and a function $F(X)$. The symbol indicates which component the configure rule is related to and X is the input parameter. The function F, which is commonly a linear function or a constant, indicates the default value of the component's attribute. In the previous subsection, we mentioned some configure rules in the modeling of a veranda component. For instance, the configure rule V_{bottom} is written as:

$$V_{bottom}(s,p) : \begin{cases} scale_{bottom} = 0.2 \\ s_{bottom} = (s_u, scale_{bottom} * s_v, s_w) \\ p_{bottom} = (p_x, p_y - 05 * s_v(1 - scale_{bottom}), p_z) \end{cases} \quad (2)$$

where $scale_{bottom}$ is a constant, which may be changed by users. As shown in Equation. 2, $V_{bottom}(s, p)$ returns the size vector s_{bottom} and center position p_{bottom} of the component vB(veranda bottom). The configure rule will be activated when the related component is created, which is always invoked by construction rules. Here, we should notice that the input of users have the higher priority than the configure rules. If the user has specified the attributes, the corresponding configure rule will not be used anymore.

3.3 Constrain Rules

As the modeling process is semiautomatic and is controlled by users, we cannot know the sequence beforehand. The new added component probably influence the existing ones and cause bad results and vice versa. Traditionally, the correction is confirmed by user themselves. But in our method, it is automatically done by the constrain rules.

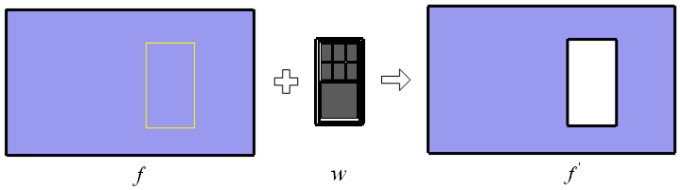


Fig. 2. constrain rules

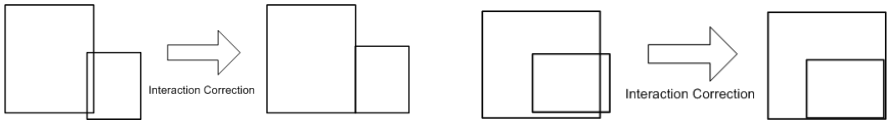
As shown in Fig. 2, a new added window w will intersect an existing wall f, if we directly load the window component and put it in the scope, which is indicated by the yellow rectangle. In general, users will add a hole in the wall before they put a window on it. This task is not the purpose (creating a window on the wall) of the user, But it is essential. We call this type of tasks as *additional tasks* and make them done automatically by constrain rules as many as possible. The constrain rules are defined as:

$$T(s, o) : con \rightarrow C(X) \tag{3}$$

where T is a function which detect the constrains between current scene(s) and the new added object(o). The sign con indicates the condition value. When the return value of $T()$ is equal to con, the operation command $C(X)$ is executed:

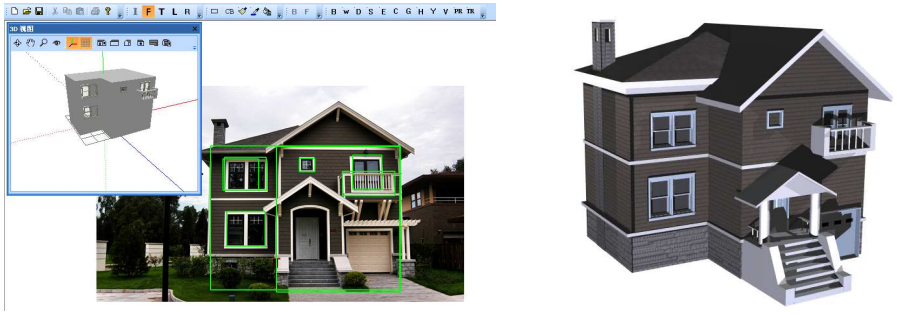
$$Intersect(f, w) : true \rightarrow hole(f, rec) \tag{4}$$

For example: as shown in equation 4, this constrain rule is activated when a new window is added into the scene.It detects intersections between walls and window and add a hole on the wall when the detection result is matched. The result is shown in Fig.2.



(a) (b)

Fig. 3. automatic correction



(a) 2D drawing (b) The result model

Fig. 4. Modeling of a villa

4 User Interface

In our prototype system, we use simple 2D user interface. Users may paint from front, back, up, left and right, five directions totally. The only painting tool used here, is a rectangle drawing tool. A rectangle from one view side determines two dimensions of the position p and the size vector s. And the left dimension value will be determines by a rectangle from the other view sides or directly from the configure rules.

Automatical Correction: As we choose 2D painting as the interaction, rectangles maybe not express the user's input exactly. As shown in Fig. 3, two rectangles drawn by the user in the left side are intersected. It will be error if the coordinates of input rectangle are parameterized directly. To solve this problem, an automatical correction step is performed before the parameterized step. The new added component is detected with the related components which exist already. If the dimension error is in a given threshold, the new added component will be resized and aligned to the exist corresponding one, as shown in Fig. 3(a) and Fig.3(b).

5 Implementation and Results

We implement the rule assist approach for modeling villas and high buildings. Fifteen components and ten operation commands are defined in our implementation. As all user interaction are in 2D, referenced photos may be used to help

(a) Referenced Photo (b) Virtual building

Fig. 5. Building Reconstruction

users to determine the position of components and also it is helpful to get material and texture images directly from photos, especially in the application of urban reconstruction. The interface of our application is illustrated in Fig. 4(a). And Fig.4(b) shows the result villa model, which is modeled referring to the left photograph.

5.1 Material and Textures

As the result model is combined by predefined components and all the components are well oriented, we can easily assign material and calculate texture coordinates automatically. When selecting a piece of the referenced photo and the corresponding component, image quilting [18] is applied to synthesize texture for filling the faces of component.

5.2 Results

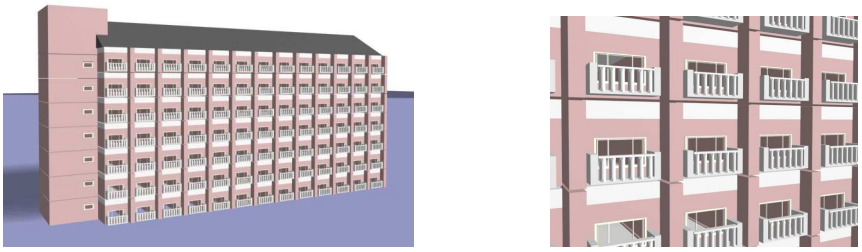
In this paper, we implemented the rules assist modeling approach to model villas and skyscrapers. Each model is created in less than one minute including the generation of textures. The generated models (Fig. 5, 6 and 7) are exported from our application and rendered by 3DSMax. Fig.5(a) is the referenced photo of the virtual building (see Fig.5(b)). From this example, we can see that the rules assist method not only suits for the virtual architecture modeling but also can be applied to reconstruct real buildings.

6 Conclusion and Future Work

In this paper, we present a rule assist architecture modeling method. Different from the existing grammar based ones, the assist rules are activated and executed according to the user interactions. Our proposed assist rule set consists of three types of rules, including construction rules, configure rules and constrains rules. Construction rules are used to model the non-terminal components automatically, which are similar to the existing production rules [8]. Configure rules

(a) Villa Model A (b) Villa Model B

Fig. 6. Villa models



(a) Whole view (b) Local view

Fig. 7. Dorm Building Model

represent the default value and relationship of components' attributes. Constrain rules are activated when the corresponding components are added. Then additional commands are executed automatically to help modeling. All these assist rules are leveraged to generate modeling commands automatically from user interactions. It is worth mentioning that our RAAM also allows users to modify the predefined rules which can provides additional flexibility for modeling different styles of buildings.

In near future, we plan to integrate 2D matching algorithms in our prototype system. Then the selection of terminal-components can be done automatically instead of being chosen by users. We will also explore the possibilities for modeling and quantitating architecture styles. For different purpose, the requirement of geometry resolution is not same. For example, in game applications, for rendering fast the geometry resolution of 3D models is always in a low resolution and details are represented by textures. But in the application of computer aided architecture design(CAAD), users require more geometry details. In the future, we can define a set of construction rules in different resolutions for one component and choosing them according to users' requirement.

References

1. G.Stiny: Introduction to shape and shape grammars. Environment and Planning B 7 (1980), Pion Ltd, London, England. pp. 349- 351.
2. Przemyslaw Prusinkiewicz, Aristid Lindenmayer:The algorithmic beauty of plants,1990, Springer-Verlag New York, Inc. New York, NY, USA.
3. Przemyslaw Prusinkiewicz, Mark James and Radom: Synthetic topiary. In Proceedings of ACM SIGGRAPH 1994, ACM Press, pp. 351-358
4. Radomír Měch, Przemyslaw Prusinkiewicz: Visual models of plants interacting with their environment. SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques,1996, ACM Press, pp. 397-410
5. Przemyslaw.P, Lars Mndermann, Radoslaw Karwowski, and Brendan Lane: The use of positional information in the modeling of plants. In Proceedings of ACM SIGGRAPH 2001, ACM Press, pp. 289-300.
6. Peter Wonka, Michael Wimmer, Francois Sillion, and William Ribarsky: Instant Architecture. ACM Transactions on Graphics,July 2003, ACM Press, volume 22. number 3. pp. 669-677.
7. Yoav I.H.Parish, Pascal Müller: Procedural modeling of cities. In Proceedings of ACM SIGGRAPH 2001, ACM Press, pp. 301-308.
8. Pascal Müller, Peter Wonka, Simon Haegler, Andreas Ulmer and Luc Van Gool: Procedural Modeling of Buildings. In Proceedings of ACM SIGGRAPH 2006, ACM Press.
9. Autodesk 3ds Max software: 3D studio Max, Autodesk Inc. http://usa.autodesk.com .
10. Sketchup Software: Sketchup. Google Inc.http://www.sketchup.com.
11. CHING, F.D. K: A Visual Dictionary of Architecture. Wiley,1996.
12. Mitchell, W. J: The Logic of Architecture: Design, Computation, and Cognition. MIT Press, 1990.
13. Hillier, B.: Space Is The Machine: A Configurational Theory Of Architecture. Cambridge University Press, 1996.
14. Alexander,C., Ishikawa, S., and Silverstein, M.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York, 1977.
15. Shubnikov, A. V. and Koptsik, V. A.: Symmetry in Science and Art. Plenum Press, New York, 1974.
16. Hua Liu, Wei Hua, Dong Zhou and Hujun Bao: Building Chinese Ancient Architectures in Seconds. International Conference of Computational Science 2005, LNCS, May 2005, Springer Berlin/Heidelberg, pp.248-255
17. Google 3D Warehouse, Google Inc. http://sketchup.google.com/3dwarehouse/.
18. Alexei A. Efros and William T. Freeman: Image Quilting for Texture Synthesis and Transfer. Proceedings of SIGGRAPH 2001, August 2001, ACM Press, pp.341–346.