Instant Architecture

Peter Wonka[*,†] Michael Wimmer[†] François Sillion[‡] William Ribarsky[*]

[*]Georgia Institute of Technology [†]Vienna University of Technology [‡]INRIA

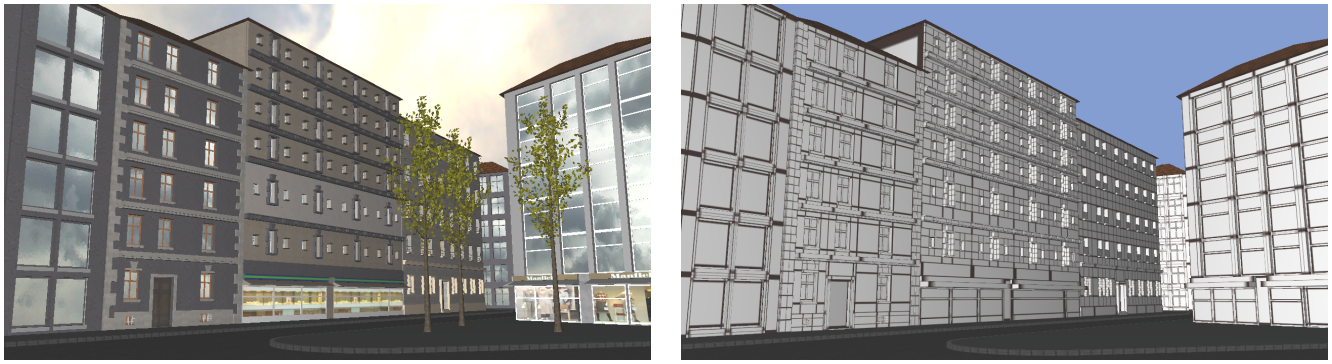


Figure 1: Left: This image shows several buildings generated with split grammars, a modeling tool introduced in this paper. Right: The terminal shapes of the grammar are rendered as little boxes. A scene of this complexity can be automatically generated within a few seconds.

Abstract

This paper presents a new method for the automatic modeling of architecture. Building designs are derived using split grammars, a new type of parametric set grammar based on the concept of shape. The paper also introduces an attribute matching system and a separate control grammar, which offer the flexibility required to model buildings using a large variety of different styles and design ideas. Through the adaptive nature of the design grammar used, the created building designs can either be generic or adhere closely to a specified goal, depending on the amount of data available.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.6.3 [Simulation and Modeling]: Applications

Keywords: modeling, building design, architecture, grammars

1 Introduction

This paper addresses the problem of automatic building modeling, as a tool for urban reconstruction and the design of new construction. The original motivation for our work is an interactive urban planning application where urban planners and residents explore different design choices in the context of a reconstructed environment. For such a system, tools for fast urban reconstruction and construction of single buildings up to large areas of differing motifs and styles are needed. However, current tools for detailed urban construction are quite labor intensive. Besides this urban planning scenario, we envision other applications like traffic and driving simulation, information visualization, military simulation and location-based entertainment.

The basic philosophy of our work is a modeling approach that functions effectively both on partial and detailed specifications. Everything is modeled in high visual quality and detail; but as information about an existing or planned space becomes available, the details of the generated model will shift to be in better agreement with reality (or the specific design imagined by a planner). If only little information is specified, the details will be a plausible estimate of what the model might look like with the system picking and choosing based on what it knows to be true.

The tool we use to achieve this goal is a design-grammar (Figure 1). In the context of design grammars, L-systems have achieved impressive results, especially in the area of plant modeling [Měch and Prusinkiewicz 1996; Prusinkiewicz and Lindenmayer 1991; Prusinkiewicz et al. 1994] and recently in the area of street modeling [Parish and Müller 2001]. Still, L-systems cannot easily be adapted to the modeling of buildings since they simulate growth in open spaces. Buildings on the other hand have stricter spatial constraints and their structure usually does not reflect a growth process. A powerful tool used by architects are shape grammars [Stiny 1975], which can be used to generate different kinds of architecture [Downing and Flemming 1981; Duarte 2002; Flemming 1987; Koning and Eizenberg 1981; Stiny and Mitchell 1978] and art [Koning and Eizenberg 1981].

A fundamental difference of our approach to previous work is that we set up a large database of grammar rules and model a variety of designs with the same rule database, rather than creating an individual grammar for each object to be modeled. Because of the complexity of such a grammar, there are several possible rules that can be chosen at each step of the derivation. The most common solution when designing with grammars is therefore to manually se-

[*]{wonka|ribarsky}@cc.gatech.edu, Atlanta, GA 30332-0280

[†]wimmer@cg.tuwien.ac.at, 1040 Vienna, Austria

[‡]Francois.Sillion@imag.fr, 38334 Saint Ismier Cedex, France

Figure 2: The left building shows how the random application of rules leads to mildly chaotic results even for small rule databases, however the chaos even increases as more rules are added as shown to the right. For an example using the rules from the left building, but with order established through the control grammar and attribute matching, see Figure 11.

lect suitable rules in the design process. While this is a successful strategy for some applications, we aim for a completely automatic derivation in the grammar after some design goals have been specified. As we use a stochastic process to automatically choose rules for derivation using the grammar, we have to be careful to maintain consistency and to exclude unwanted instances. This problem will be especially apparent as the size of the rule database increases (see Figure 2). One way to make sure that only useful instances are generated is to encode all possible design choices in the grammar itself. However, this is infeasible due to the combinatorial complexity involved. The way we deal with this problem is to factor out important design ideas from the grammar, and use a separate mechanism to distribute them spatially in an orderly manner, allowing different ideas to blend together—this can be observed in real buildings (Figure 3).

We will address these aforementioned problems and present the following contributions:

- We introduce a new type of design grammar which we call "split grammar". The novelty of split grammars lies in the restrictions on the type of allowed rules. These restrictions make split grammars powerful enough for the modeling of buildings, but simple enough to allow a controlled and automatic derivation of the grammar.

- We introduce a parameter matching system that allows the user to specify multiple high-level design goals and controls randomness to guarantee a consistent output.

- We introduce the idea of control grammars, simple context free grammars which handle the spatial distribution of design ideas not randomly, but in an orderly way that corresponds to architectural principles.

The rest of the paper is structured as follows. After reviewing relevant previous work, Section 3 provides an overview of our framework. Sections 4 and 5 present the main contributions of the paper, split grammars and the attribute matching/propagation system respectively. Example building designs created using the framework can be found in Section 6, a discussion of the concepts introduced in the paper in Section 7, and our conclusions and plans for future work in Section 8.

2 Related work

A promising avenue for urban reconstruction is the idea to reconstruct models using photographs [Debevec et al. 1996; Dick et al. 2001; Jepson et al. 1996; REALVIZ 2002], videos and range scanning [Karner et al. 2001; Ribarsky et al. 2002; Teller 2001]. While these methods produce excellent models with high accuracy, they are quite labor intensive and inherently limited to reconstruction tasks.

Figure 3: Photograph of a building which shows vertical coherence (balconies and material) and horizontal coherence (window styles).

Grammars, mainly L-systems, were very successfully applied to the modeling of plants [Měch and Prusinkiewicz 1996; Prusinkiewicz and Lindenmayer 1991; Prusinkiewicz et al. 1994; Shlyakhter et al. 2001] and streets [Parish and Müller 2001]. The general trend in this recent work is to create a simulation part separate from the L-system in order to have a more powerful control over the application of rules. With regard to the application of L-systems to buildings, we have to consider that the structure of a building is fundamentally different from the structure of plants (or streets)–most importantly, a building is not designed with a growth-like process, but a sequence of partitioning steps.

Therefore, other alternatives seem to be more suitable to model architecture. The most common approach is the construction and analysis of architectural design using shape grammars, introduced by Stiny [1975]. These grammars operate directly on shapes and have been shown to be useful in the analysis and construction of many architectural styles [Downing and Flemming 1981; Duarte 2002; Flemming 1987; Koning and Eizenberg 1981; Stiny and Mitchell 1978]. However, the derivation in such a grammar is usually done manually, or in a computer assisted manner, with a human deciding on the rules to apply. This makes shape grammars a very useful tool for some applications, but prevents rapid modeling of larger areas.

For a categorization of other grammars and grammar-like formalisms in general, see Stiny [1980b]. A good recent overview of grammars in architecture is given by Mitchell [1990]. In his book, "The Logic of Architecture", he explains fundamental concepts like design algebras and architectural languages. Other notable work that discusses a formal analysis of space and design is the work on space syntax [Hillier 1996], imageability [Lynch 1960], Stadtgestalt [Moser et al. 1985], design patterns [Alexander et al. 1977] and symmetry [Leyton 2001; March and Steadman 1974; Shubnikov and Koptsik 1974; Weyl 1952]. Additionally, Alexander's work on the analysis of city structure [Alexander 1965] applies to many general design problems.

Interestingly enough, automatic modeling techniques are already available for enhancing existing architectural models, for example using cellular textures [Legakis et al. 2001] and texture synthesis [Wei and Levoy 2000].

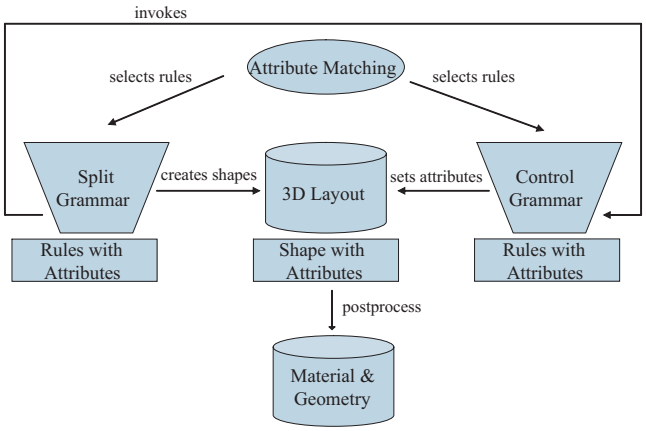


Figure 4: An illustration of how the attribute-matching system and the control grammar interact with a split grammar via attributes to steer the derivation process.

3 Overview

This section gives a short overview of the structure of our automatic building modeling framework and how it interacts with the user.

At the core of the system is a spatial attributed design grammar, the split grammar. This grammar is used to derive a 3D layout of a building consisting of simple, attributed shapes. Starting from an initial shape, the grammar generates the façades of the building, which are then in turn split into their structural elements, down to the level of individual design elements like window sills, cornices etc. The attributes assigned to the shapes in the final design are used in a postprocess to define the geometry and material associated with each shape.

The terms rule, grammar symbol and attribute, which will be explained in more detail in sections 4 and 5, can for now be intuitively understood from the example used later in Figure 8, which shows 6 different rules that can be applied to a shape. The shape itself is associated with a grammar symbol (WIN). Both the grammar symbol and the individual rules are associated with attributes (simple, blind etc. in the figure), the functions of which will be explained shortly.

As illustrated in Figure 4, the derivation process is controlled by an interleaved application of a split grammar and an external control grammar, both of which make use of an attribute matching system in order to select between multiple available rules. The control grammar is used to calculate and distribute attributes to the shapes generated with a split grammar. The attribute system builds on the fact that attributes are not only assigned to shapes, but also to rules. The attribute matching system can therefore compare the attributes assigned to a shape with the attributes assigned to the different available rules in order to select the rule with the best match. One step in the derivation process works as follows:

1. The split grammar searches the rule database for all rules matching the current shape.
2. The attribute matching system is invoked to select a rule based on the attributes of the current shape and the candidate rules.
3. Attributes are copied from the current shape to all shapes generated by the selected rule.
4. In addition, a control grammar is invoked in order to distribute design ideas spatially (e.g., set different attributes for the first floor of a building) in an orderly manner. Derivation in the control grammar is subject to the same attribute matching system as in the split grammar.
5. The split grammar is invoked recursively for the shapes generated in the current rule.

There are several ways in which to influence the design of the buildings generated in our approach: First, the designer can directly modify the split grammar and the control grammar. This is the most powerful design tool, and will mostly be used when designing a whole new class of buildings, e.g., following a new building style. Second, the user can modify the attribute values of the starting shape of the split grammar. The attributes of the starting shape control high-level aspects of the building design like the building style or the age of the building, or its use. When creating a large number of buildings, these attributes can be chosen automatically following some given distributions (attributes may also be left unspecified). Third, the user can modify the attribute values attached to the grammar rules of the split and control grammars. This allows fine-grained control over certain aspects of the design for individual buildings without requiring a change to the grammar itself.

4 Split grammars

This section describes the spatial grammar which forms the syntactic basis for our building model. We propose to use a three-dimensional design grammar, which allows treating the design of a whole building in a uniform framework, including its 3D structure. The grammar described in this section determines the spatial layout of a building.

The control process for both selecting designs from the design space defined by the grammar and attaching semantic information like materials and geometry to this design is described in section 5, and the interpretation of these attributes in section 4.4.

4.1 Shape

Our approach draws from the work on shape grammars pioneered by Stiny [1980a]. The appeal of shape as a formal concept is that spatial relations and dependencies can be consistently encoded in the grammar itself. Shape grammars are a powerful tool to express very general designs. However, the automatic application of shape-grammar rules is difficult because of the frequent emergence of new shapes in the derivation process. A formalism that is more amenable to computer implementation is the notion of set grammars [Stiny 1982], which treat shapes as symbolic objects and therefore do not require difficult sub-shape matching procedures. The fundamental primitive manipulated by our grammar is a shape, given in the following definition (as in Stiny [1980a]):

Definition 4.1 *A shape is a limited arrangement of straight lines in three-dimensional Euclidian space.*

A line $l = p_1, p_2$ is uniquely determined by two distinct points p_1 and p_2. Two lines are identical if their endpoints are identical, and consequently, two shapes are identical if they contain the same set of lines. Shapes can be attributed with a set of labeled points. A labeled point $p : A$ is a point p with a symbol A attached to it. A labeled shape is denoted by $\langle s, P \rangle$, where P is the set of its labeled points. Furthermore, shapes can be parameterized by allowing coordinates of the endpoints of the containing lines to be variables. The result of applying an assignment g to the variables in the parameterized shape s is denoted by $g(s)$.

4.2 Grammars

Spatial design techniques require a more general concept of grammars than provided by common string grammars like context-free grammars or L-systems. A general grammar can be defined over an

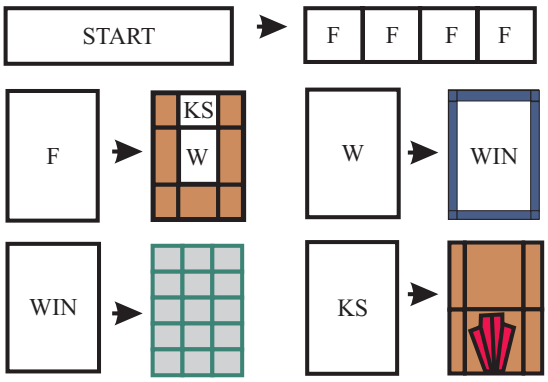


Figure 5: The rules for a simple example split grammar. The white areas (which contain symbols) represent the non-terminal shapes, colored elements are the terminal shapes of the split grammar. The start symbol is split into 4 façade elements, which are further split into a window element, a keystone element and some wall elements etc.

algebra of objects, $\langle U, +, -, F, \leq \rangle$, closed under operations $+$ and $-$ and a set of transformations F, so that if u and v are members of U, then so are $u + f(v)$ and $u - f(v)$, where f is a member of F. The notion that an object u "occurs" in v is generalized via the match relation $\leq$, which should be defined such that u occurs in v iff there is a transformation $f \in F$ with $f(u) \leq v$.

Definition 4.2 *A grammar $G = (N, T, R, I)$ consists of the nonterminal vocabulary $N \subseteq U$, the terminal vocabulary $T \subseteq U$, a (set of) initial object(s) $I \subseteq N$ and a set of rewriting rules (productions) $R \subseteq UxU$.*

A rule $a \rightarrow b$ in a grammar with parameterized objects is applicable to u whenever there is a transform $f \in F$ and a variable assignment g such that $f(g(a)) \leq u$, in which case, under rule application, the object v is produced, given by the expression

$$v = (u - f(g(a))) + f(g(b)).$$

Note that U is usually the closure of the sets N and T under the operations $+$ and $f \in F$. To illustrate what this definition means in terms of traditional string grammars, U would be the set of strings in a given vocabulary $N \cup T$, $u + f(v)$ and $u - f(v)$ would mean insertion and deletion of a substring at a position designated by f, g would be the identity operation and $\leq$ would be the substring relation.

Of particular interest are *set grammars*, defined in the following:

Definition 4.3 *A* set grammar *is a grammar over the vocabulary B, where the algebra U is the power set of the defining set B, the operations $+$ and $-$ are set union and set difference respectively, the matching relation $\leq$ is the subset relation, and $f(S)$ is the set $\{f(s)|s \in S\}$ for a subset $S \subseteq B$.*

4.3 Split grammar

Split grammars as introduced in this paper are a specialized type of set grammar operating on shapes. Its suitability for the automatic modeling of buildings stems from the fact that restrictions have been carefully chosen so as to strike a balance between the expressiveness of the grammar, i.e., the number of different designs it permits, and its suitability for automatic rule selection. The objects manipulated by the grammar are certain attributed, parameterized, labeled shapes, which we call *basic shapes*.

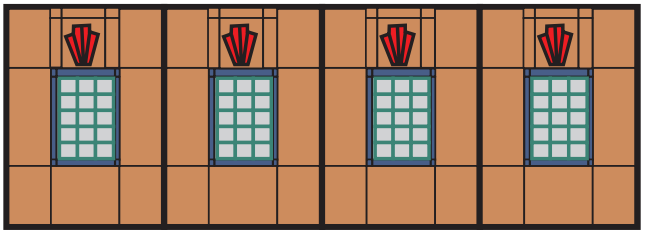


Figure 6: This figure shows the result of the derivation of the grammar in Figure 5.

.

Definition 4.4 *A basic shape b is given by $b = \langle s, P, V \rangle$, where s is a simple shape centered on the origin, P is a set of three labeled points defined by the intersection of the positive coordinate axes with the (imaginary) faces of the shape, and V is a symbol from a vocabulary N' [1] with arbitrary attributes attached to it, where a subset $T' \subseteq N'$ forms terminal symbols. A simple shape in this definition is a shape that contains all lines defined by the edges of a closed, convex 3D geometric object.*

Basic shapes are simple building blocks of the grammar, e.g., cuboids, cylinders (made up of polygonal segments) or prisms. Since basic shapes are defined in terms of simple geometric objects, the volume of a basic shape and its boundary faces are understood to refer to the volume and boundary faces of the defining geometric object.

The vocabulary of a split grammar is the set $B = \{f(b)|b \text{ is a basic shape}, f \in F\}$ (note that $f(b) = \langle f(s), f(P), V \rangle$). The set of allowable transformations F will be the affine transformations.

Definition 4.5 *A split is the decomposition of a basic shape into shapes from the vocabulary B.*

For example, a split replaces a cuboid with $n \times m \times k$ cuboids, arranged in a grid with parameterized splitting planes.

Definition 4.6 *A split grammar is a set grammar over the vocabulary B (defined by the basic shapes) where the following types of rules are allowed:*

- *A (possibly context sensitive) split rule, i.e., a rule $a \rightarrow b$ where a is a connected subset of B and b contains the same elements as a except for one element, to which a split is applied.*
- *A (possibly context sensitive) conversion rule $a \rightarrow b$, which transforms one basic shape into another. a is a connected subset of B which contains one basic shape, and b contains the same elements as a, except that the basic shape has been replaced by another. The only restriction is that the basic shape in b has to be contained in the volume of the basic shape in a.*

A set of shapes $B' \subseteq B$ derived in a split grammar is called a building design if it contains only terminal shapes, i.e. elements $b \in B$ with an associated symbol $V \in T'$.

Figure 5 gives examples for productions in a split grammar, with the final result in Figure 6. For illustration, a more complex example of a derivation can be found in Figure 11.

One notable difference between the split and the conversion rule is that in a split rule, the elements in the sets a and b fill the same volume, whereas in a conversion rule, this is not necessarily the

[1] N' is usually the set of strings

case. For example, a cuboid could be replaced by a three-sided prism in order to model a roof, leaving some space empty.

The derivation in a split grammar is deterministic in the sense that the order in which the subshapes created by a split rule are visited is fixed in advance.

4.4 Geometric interpretation

Although split grammars are inherently three dimensional, a building design derived in a split grammar only sets the syntactic and spatial framework for the final building. The geometry that is actually associated with each shape in the building design is determined by the attributes associated with the symbols in the shapes. To better control the derivation process, each rule can have a set of attributes as well. How these attributes are created, transformed and used to guide the rule selection process is described in in section 5.

Note that when cuboids are the only basic shape available, the building designs derived by the split grammar will be attributed hierarchical grids with arbitrary splitting planes at each subdivision level.

5 Attribute propagation and rule selection

The previous section describes a mechanism to derive spatial layouts for buildings. However, nothing has been said so far about how appearance information like material and texture is handled by the system, nor about how a rule is chosen among several matching rules.

Both questions are tightly related to the role of the attributes attached to grammar symbols. Attributes serve a dual purpose in our approach:

1. Attributes are used to to encode and propagate material information at different granularities. An attribute might encode low-level information such as a specific color value for walls, or a high-level design goal such as a preference about the style of the building, which in turn would lead to a more specific attribute specification later in the derivation.

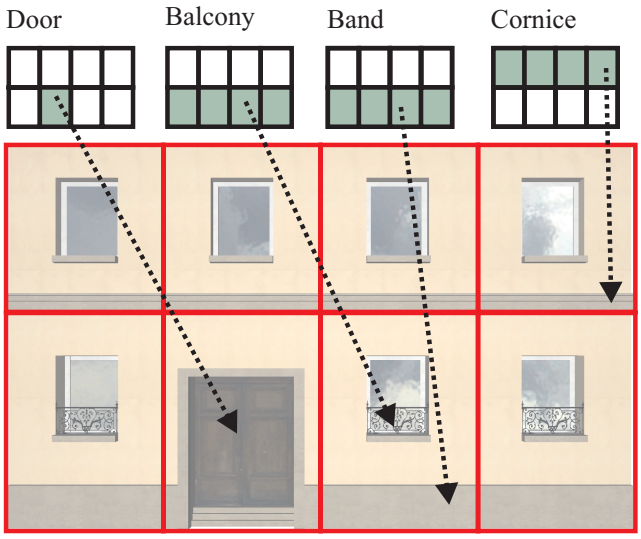
2. Attributes are used to steer the derivation process by selecting a specific rule among a set of matching rules through an intricate attribute-matching process. This means that attributes have an influence on the spatial design derived by the split grammar, as well as on the control grammar (which will be be discussed shortly).

We will first deal with the question of how attributes are propagated and distributed spatially in the grammar.

5.1 Control grammar for attribute propagation

There are three ways that an attribute can be set in our framework. The first two are straightforward, namely setting the attributes of the start symbol in the split grammar by hand (those attributes will influence higher level design decisions), and simply copying attributes from the parent shape to all the generated shapes in a split rule.

However, we also need a mechanism to distribute design decisions spatially in a way that corresponds to architectural principles. For examples, the attributes of different floors of a building should vary slightly (different window elements) or significantly (shops in the first floor), yet should be consistent within one floor. A similar type of coherence can be found for the columns of a façade (see Figure 3). Further, it is necessary to infer lower-level design decisions (like an exact color or a texture) from higher-level ones (like a building style).




```
FACADE_CONTROL ::- DOOR_PATTERN, RANDOM_PATTERN,
                   RANDOM_PATTERN, RANDOM_PATTERN
RANDOM_PATTERN ::- CORNICE | BAND | BALCONY
                   | QUOIN | PILASTER
DOOR_PATTERN ::- DOOR | GARAGE
DOOR ::- <[0,1], door, 1> | <[0,2], door, 1>
         | <[0,MAX], door, 1>
```

Figure 7: This figure shows an example using different scopes for attribute modification, and below, some rules of the control grammar used in the example. Note that the conflict between the attributes door and balcony is resolved through attribute matching.

To handle this, we introduce a third way of assigning attributes: the control grammar. The control grammar is a very simple attributed context-free grammar that is used to refine design decisions and to distribute them spatially. The nonterminals of this grammar are descriptive symbols (or strings, like "WOOD"). The terminals of the grammar are attribute modification commands in the form of tuples $\langle c, a, v \rangle$, where c is a spatial locator (or scope) within a split, a specifies an attribute name, and v specifies a value to assign this attribute. The spatial locator c depends somewhat on the type of split rule employed, but can usually be specified by a row-, column- and layer number (for 3D splits), including special markers for all rows, all columns, first row, last row etc. (see Figure 7, where $[i, j]$ means row i and column j).

Where and when the control grammar is invoked is also determined by attributes. There is a special attribute for each shape that can hold an arbitrary start symbol for the control grammar. When a split is about to be carried out, this start symbol is used to derive a list of tuples $\langle c, a, v \rangle$ in the control grammar. These are interpreted as commands to fill in the attribute values for the newly created shapes, such that the attribute a is set to the value v for all shapes matching the scope of the command, i.e. the spatial locator c. Some rules from a control grammar are shown in Figure 7. The start symbol may also be empty, in which case the attributes are just copied directly to the new shapes as explained above.

Note that the control grammar itself shares many of the attributes with the split grammar, and the rule selection process in the control grammar is exactly the same as in the split grammar, i.e., via the attribute matching method described hereafter. Also, the start symbol for the control grammar in a shape can be (and frequently is) set by a control grammar invoked in the parent shape.

5.2 Attribute matching for rule selection

To model a wide variety of buildings, it is necessary that there is a choice among several matching rules at most steps in the derivation both in the split grammar and in the control grammar. For example, the shape with the associated symbol WIN could match the rules 1-6 in Figure 8.

During automatic derivation in a grammar, selecting a rule among several matching ones has to be done in such a way that a) the derivation produces coherent, plausible results, b) the derivation provides sufficient variation in the modeled buildings, and c) certain design criteria specified by the user are met. While a grammar with a small set of rules (e.g., an L-system for one particular type of tree [Prusinkiewicz and Lindenmayer 1991]) can be set up so that all random selections of rules produce nice results, in general, the more rules that are added to a grammar, the more chaotic the result will be, as depicted in Figure 2.

The approach chosen in our framework is to augment not only grammar symbols with attributes, but also the grammar rules. Whenever several matching rules are found at a particular step in the derivation, the attributes specified in the rules are compared to the attributes associated with the current grammar symbol, and the rule with the best match is selected. The attributes associated with grammar rules are specified during the design of the grammar. We describe an approach which allows influencing the rule-selection process in several ways. It is possible to exclude or include rules, prioritize rules, or give preferences in the form of distributions etc.

We assume an attributed grammar symbol S associated with attributes $a_{1,S},....,a_{n,S}$ and a set R of attributed rules, where each rule $r \in R$ is associated with the same number of attributes $a_{1,r},....,a_{n,r}$. The aim is to find the best match between the attributes of the symbol S and the rules $r \in R$. The attributes themselves consist of several components: For rule attributes $a_r = \langle I_r, c_r \rangle$, this is an interval $I_r = [l,u]$, and a boolen containment flag c_r. Attributes of grammar symbols have an additional prioritization factor $f \geq 0$ and a specification of a statistical distribution SD, so that $a_S = \langle I_S, c_S, f, SD \rangle$.

Matching proceeds in two stages. First, a deterministic matching function M_{DV} is used to create a set of candidate rules. This function basically checks whether the intervals specified in the rule attributes and in the grammar symbol attributes overlap, and also allows to prioritize rules. If the resulting set contains more than one rule, then a stochastic selection function selects a rule according to the given statistical distributions.

More specifically, the *deterministic matching function* accumulates the overlap tests from the individual attributes:

$$M_{DV}(S,r) = \sum_{i=1}^{n} m_{DV}(a_{i,S}, a_{i,r})$$

where $m_{DV}(a_{i,S}, a_{i,r})$ is the matching function for one specific attribute. It is based on the evaluation of a logical predicate about the intervals of the attributes, using the following evaluation function for a predicate P:

$$E(P) = \begin{cases} 0 & \text{if } P \text{ is true} \\ -\infty & \text{if } P \text{ is false} \end{cases}$$

Now, for attributes a_S and a_r, the deterministic matching function is calculated using the following formula:

$$m_{DV}(a_S, a_r) = f + E((I_S \cap I_r \neq \emptyset) \wedge (c_S \rightarrow I_r \subseteq I_S) \wedge (c_r \rightarrow I_S \subseteq I_r))$$

The net result of the deterministic matching function $M_{DV}(S,r)$ is that a rule is only selected if for all attributes, the rule and symbol attribute intervals overlap or—if the respective containment flag c_r or c_S is set—fulfill the containment condition (see Figure 8). In this case, the accumulated prioritization flags are returned as a result.

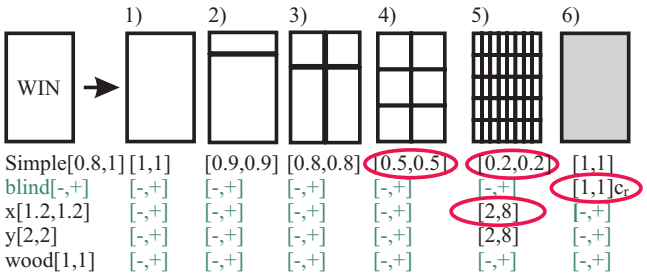


Figure 8: This figure shows a shape with the symbol WIN and six rules which can be applied. Through the attribute matching function, the rules 4-6 are excluded. Attributes causing exclusion are marked in red, default attributes in green (i.e., $[-,+]$ is for $[-\infty,\infty]$).

Usually, the interval I_S for the symbol S is a non-degenerate interval, allowing a range of possible values for this attribute, while the interval I_r for the rule is collapsed to a single value, specifying specific value of the attribute that this rule represents. The prioritization flag f is usually set to 0, and most of the attributes for symbols are initialized with the "don't care" interval $[-\infty,\infty]$.

If more than one rule have the same maximum value $M_{DV}(S,r)$, the system selects from those the rule with the highest *stochastic selection function*, which accumulates stochastic values from the different attributes:

$$M_{SV}(S,r) = p_r \prod_{i=1}^{n} m_{SV}(a_{i,S}, a_{i,r})$$

The stochastic value for each attribute pair is calculated as

$$m_{SV}(a_S, a_r) = f_{SD}(\frac{u_r - l_r}{2}),$$

i.e., the density function f_{SD} of the distribution given by the symbol S, evaluated at the midpoint of the interval I_r given by the rule r. However, the stochastic selection function also has to preserve coherence during a derivation in the following sense: If, for one symbol S the selection returns a rule $r \in R$, we require that for any other symbol S_x during the derivation, the same rule is selected for any subset of R that contains r if the relevant attributes are identical. This could be achieved by maintaining a history of rule selections, but a much simpler way is to precalculate one random value p_r per rule r. This precalculation is done once per building, and the random value is simply multiplied with the product of the individual density values.

We illustrate some effects than can be achieved with these matching functions using the simple example in Figure 8 (where attributes causing exclusion are marked with a circle):

Exclusion interval/value The interval of the attribute "simple" for the symbol does not overlap with rule 4 (which specifies a single value), therefore $m_{DV}(\text{simple}_{\text{WIN}}, \text{simple}_{r_5}) = -\infty$ and the rule is excluded (as is rule 5—both of these windows are too complex).

Exclusion value/interval Another case where a rule is excluded if the rule specifies an interval and the symbol a value. This is typically used for geometric attributes such as width, height and depth of a shape, which are automatically calculated for the associated symbols. Rule 5 is excluded because it requires a width x larger than the shape associated with the symbol WIN provides.

Default values In the example, only the default interval $[-\infty,\infty]$ is specified for the attribute "wood" for all rules, therefore the attribute does not change the selection.

Exclusion from default Some rules should only be applied if specifically requested, such as the blind window (a fake window which is used only as an ornament) in rule 6. By specifying the containment flag $c_r = true$, the interval for the symbol needs to be contained in the interval for the rule, which is never the case for the default interval $[-\infty, \infty]$ (as in the example for attribute "blind").

6 Results

We implemented the described grammar and created an example database consisting of about 250 rules and 40 attributes. The grammar comprises 10 basic shapes, 3 out of which (cuboid, cylinder and prism) have split rules defined. The rules were modeled according to several different resources, mainly [Mitchell 1990], [Moser et al. 1985] and our own analysis of architecture in several American and European cities including London, Paris, Atlanta and Vienna. The creation of the rule database took about 2 weeks, but note that all the building designs in this paper were generated from the same set of rules. This means that once the rule database was set up, only high-level attributes were changed in order to derive the different example buildings seen in the figures. The control grammar was set up to steer the derivation so that only architectonically plausible building designs were derived. Most buildings have between 1000 and 100,000 polygons, and the creation of a building takes roughly one to three seconds on an Intel Pentium 4 with 2 GHz. Also note that all snapshots were taken from interactive rendering in a real-time rendering engine.

Currently, we use two types of prototype user interfaces for modeling: The first user interface is the GIS system ArcView, where buildings are stored as footprints with attributes. Our software reads ArcView files and uses each individual building footprint together with its attributes to start a derivation of the grammar (the last scene in the video was generated with this approach). The second user interface is a command line interface, where attributes of an initial shape can be set with commands (e.g. `wall_material=15`, `shop=1` etc.). Some of these attributes are shown in the video as white text overlay.

The attributes in the example database allow the user to specify several aspects of the design and appearance of a building. An important classification of buildings is according to the vertical or horizontal emphasis of the design [Moser et al. 1985] (see Figure 10). Some attributes can be used to specify the symmetry of the building, select a general level of complexity of the façade and specify the building style. Additionally, several attributes can be used to select or exclude specific elements or colors.

Another example shows the power of split grammars when designing buildings of a more complex 3D structure (see Figure 9). Some of the different possible 3D designs we tried include a family house with non-trivial floorplan, a tower built with a cylindrical split rule, and a modern office building with a complex volumetric 3D structure.

Our method also lends itself to an iterative approach to designing buildings. In Figure 10, more and more design goals have been specified from left to right. In Figure 11, we show the derivation of a simple building in a given split grammar.

7 Discussion

This section discusses various design aspects of the framework and details its main contributions to the field of computer graphics.

Architecture for computer graphics. There is an emergent interest for procedural architectural modeling in computer graphics and related fields. Parish and Müller [2001] have only recently used parametric L-systems to model buildings in a huge urban environment. While their approach aims at quickly generating a large number of simple, yet diverse buildings, the focus of this work lies on producing complex geometric representations of individual buildings, which requires a different approach.

Shape as the basis for modeling. Shape grammars have been used in architecture to describe and also create building designs. Even though they are not suitable for automatic and semi-automatic modeling, we believe that the concept of shape is essential for describing architectural design problems, because it opens the field for formal design principles (like split grammars as presented in this paper) to be applied to 3D modeling tasks.

Flexibility. Once a basic building grammar has been created, the user is free to introduce any additional data in the form of attributes or rules in order to guide the derivation, making design an iterative process (see Figure 10). Sometimes it is desirable to create a number of more or less random buildings, and sometimes, a particular building design needs to be achieved. Both approaches are possible in our framework or can be blended, within the requirements of an urban planning application. Note also that although the database of 250 rules used for this paper already provides a significant variety of building designs, a real-world urban planning application will likely require a still larger number of rules (probably in the order of 2000 to 3000). Also, while the system can be useful for a wide range of architectural styles, starting (in a historic classification) from pyramids, Greek and Roman temples, up to modern American skyscrapers and works from modern architects like the New York Five, there are limitations in the complexity of architectural details the system can handle. This includes complex details (e.g., Corinthian capitals in Greek temples), which should be created in a separate modeling package and used as terminal shapes in the grammar, and complex configurations such as for example Gothic windows.

Complexity and usability of the system. Grammars such as a split grammar can be extended by users of the system, usually in a graphical editor, whereas a parametric or hard-coded system would require a rewrite of the underlying framework. However, due to the complexity inherent in architecture, modifying the rules of the grammar is by no means trivial, and requires a certain familiarity with the split-grammar approach. We envision that a specific split grammar will be created by a designer in collaboration with an architect, and that other users of the system will mainly modify the attributes of the given rules and the parameters of the grammar.

Automatic derivation and variation. Our grammar-based approach can mimic a parametric system because of the flexible attribute matching system. The modeling process could then be split into creating the grammar and its attributes, and modifying the attributes to get the desired results within a given design space. The last step can also be carried out by less experienced users of the system, since the derivation in the grammar is fully automatic. It has to be noted that even when a number of parameters is fixed in order to achieve certain design goals, our framework is still capable of producing a large number of varying designs according to these parameters through the stochastic rule selection process.

Growth Control. We found that a significant problem with most design grammars that allow automatic derivation (most notably, L-systems) is that it is difficult to prevent objects from growing into each other. This might be hardly noticeable for tree generation, but in architectural modeling it is not tolerable. Another consequence is that local decisions in such grammars may have global impact, which makes it difficult to construct a grammar to obtain a certain goal. The split-grammar formalism presented in this paper has been created to deal with exactly this problem, where shapes created in a rule are always required to be contained in their parent shapes.

Rule selection. To our knowledge, this is the first paper to address the problem of rule selection for grammars with large rule

databases. This is important because for larger rule databases, it becomes increasingly unlikely that all designs derived in the grammar make sense. Through the control grammar and the attribute matching system, such designs can be weeded out at an early stage, and a number of design goals can steer the derivation process in the presence of multiple matching rules.

8 Conclusions and future work

In this paper we have presented a framework for the automatic modeling of architecture, and demonstrated its applicability by using it to create buildings in several different building styles. The system offers great flexibility in the amount of control to be exerted on the design process. High-level design choices can be made through a number of simple attributes, but it is also possible to influence low-level design decisions by manipulating the grammar itself, either by modifying rule attributes or by creating new design rules. The system therefore adapts to the data that is available, one of the most important requirements in any architectural modeling approach.

Our future work on the application side includes the design of an interactive editor for larger environments and a reconstruction of a major American city. Further, we aim at the synthesis of modern architecture—as for example in the style of the American architect Richard Meier—using a shape grammar developed by our collaborators in the department of architecture as a starting point. We are also currently evaluating additional types of rules which can further enhance the expressiveness of the grammar. For example, a merge rule could be used to merge adjacent shapes of the same dimension—however, care has to be taken not to unnecessarily complicate the grammar. We see the most potential for future enhancements in the use of images as training data for creating and improving the grammar.

Acknowledgements

We acknowledge contributions from Greg Turk, Peter Bleier, Andreas Voigt, Heiner Hierzegger and Ari Lamstein. This research was also supported by the Austrian Science Fund (FWF), contracts no. P-13867-INF and J2151.

References

ALEXANDER, C., ISHIKAWA, S., AND SILVERSTEIN, M. 1977. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York.

ALEXANDER, C. 1965. A city is not tree. *Architectural Forum 122 22*, 1, 58–61.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, H. Rushmeier, Ed., 11–20.

DICK, A., TORR, P., RUFFLE, S., AND CIPOLLA, R. 2001. Combining single view recognition and multiple view stereo for architectural scenes. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, IEEE Computer Society, Los Alamitos, CA, 268–274.

DOWNING, F., AND FLEMMING, U. 1981. The bungalows of buffalo. *Environment and Planning B 8*, 269–293.

DUARTE, J. 2002. *Malagueira Grammar – towards a tool for customizing Alvaro Siza's mass houses at Malagueira*. PhD thesis, MIT School of Architecture and Planning.

FLEMMING, U. 1987. More than the sum of its parts: the grammar of queen anne houses. *Environment and Planning B 14*, 323–350.

HILLIER, B. 1996. *Space Is The Machine: A Configurational Theory Of Architecture*. Cambridge University Press.

JEPSON, W., LIGGETT, R., AND FRIEDMAN, S. 1996. Virtual modeling of urban environments. *PRESENCE 5*, 1, 72–86.

KARNER, K., BAUER, J., KLAUS, A., LEBERL, F., AND GRABNER, M. 2001. Virtual habitat: Models of the urban outdoors. In *Third International Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Imaging*, 393–402.

KONING, H., AND EIZENBERG, J. 1981. The language of the prairie: Frank lloyd wrights prairie houses. *Environment and Planning B 8*, 295–323.

LEGAKIS, J., DORSEY, J., AND GORTLER, S. J. 2001. Feature-based cellular texturing for architectural models. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 309–316.

LEYTON, M. 2001. *A Generative Theory of Shape*. Springer-Verlag, Berlin.

LYNCH, K. 1960. *The Image of the City*. MIT Press.

MARCH, L., AND STEADMAN, P. 1974. *The Geometry of Environment*. MIT Press.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proceedings of ACM SIGGRAPH 96*, ACM Press, H. Rushmeier, Ed., 397–410.

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

MOSER, F., MAYERHOFER, R., AND FREI, W. 1985. Charakteristik der Stadtgestalt Wien – Grundlage für Stadterneuerung und Wohnbau. Technical report, Institute of Local Planning, Vienna University of Technology.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press, E. Fiume, Ed., 301–308.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1991. *The Algorithmic Beauty of Plants*. Springer-Verlag.

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *Proceedings of ACM SIGGRAPH 94*, ACM Press, A. Glassner, Ed., 351–358.

REALVIZ, 2002. Image modeler product information. http://www.realviz.com.

RIBARSKY, W., WASILEWSKI, T., AND FAUST, N. 2002. From urban terrain models to visible cities. *IEEE Computer Graphics & Applications 22*, 4, 231–238.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications 21*, 3 (May/June), 53–61.

SHUBNIKOV, A. V., AND KOPTSIK, V. A. 1974. *Symmetry in Science and Art*. Plenum Press, New York.

STINY, G., AND MITCHELL, W. J. 1978. The palladian grammar. *Environment and Planning B 5*, 5–18.

STINY, G. 1975. *Pictorial and Formal Aspects of Shape and Shape Grammars*. Birkhauser Verlag, Basel.

STINY, G. 1980. Introduction to shape and shape grammars. *Environment and Planning B 7*, 343–361.

STINY, G. 1980. Production systems and grammars: a uniform characterization. *Environment and Planning B 7*, 399–408.

STINY, G. 1982. Spatial relations and grammars. *Environment and Planning B 9*, 313–314.

TELLER, S., 2001. Mit city scanning project: Fully automated model acquisition in urban areas. http://city.lcs.mit.edu//city.html.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press, K. Akeley, Ed., 479–488.

WEYL, H. 1952. *Symmetry*. Princeton University Press.

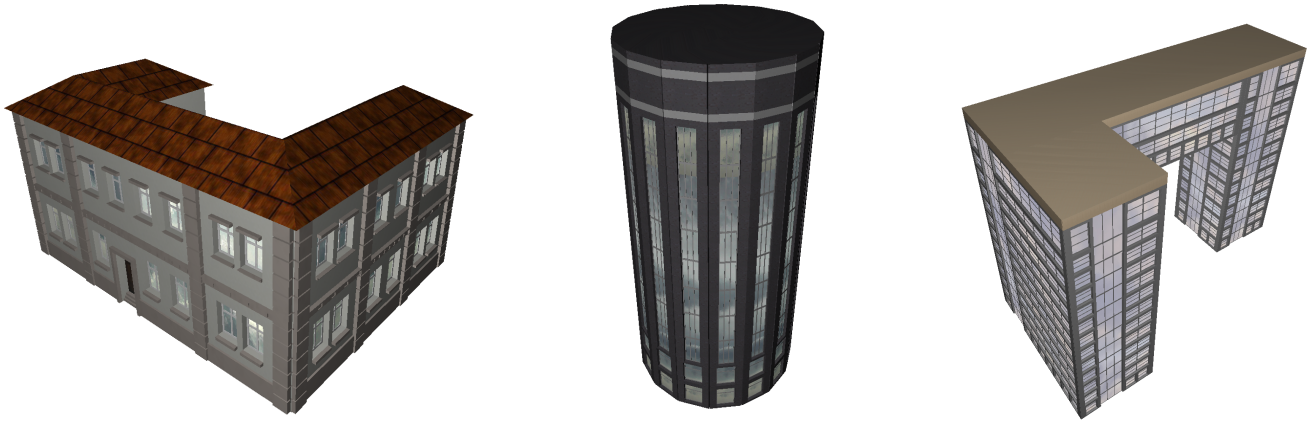


Figure 9: Some examples demonstrating the variety of buildings that can be modeled using the approach presented in the paper. Left: A family home. Note the principle of vertical and horizontal coherence in the ornamentation , and the three-dimensional nature of the floor plan and the façades. Middle: Example for a cylindrical split. Right: Split grammars are powerful enough to describe buildings with non-trivial volumetric arrangements.

Figure 10: This figure shows the principle of iterative refinement in design. The left building does not have any specific attributes, the middle building has the attribute “shop” and “color variation” applied, and the rightmost building has—in addition to the ones for the middle building—attributes for vertical emphasis on the first and last column and horizontal emphasis on the second and third floor.

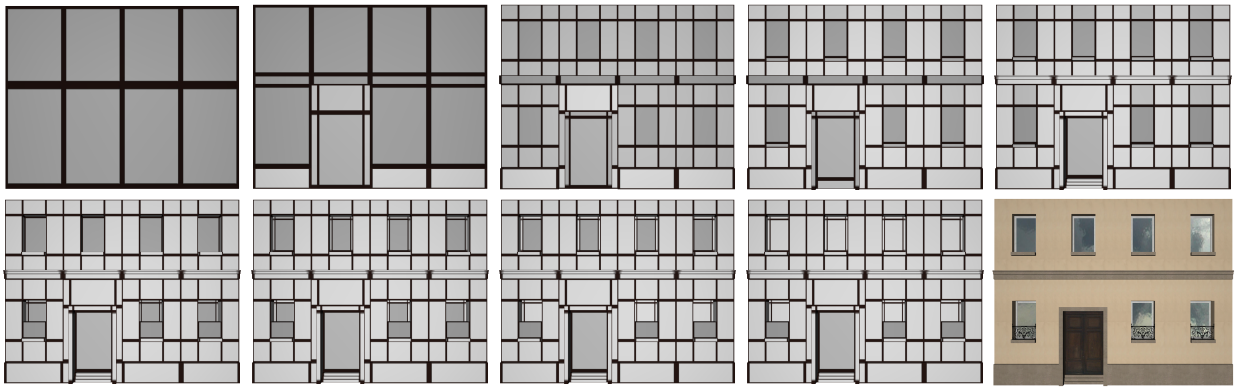


Figure 11: A sample derivation in our grammar; individual steps in the derivation are coded in gray levels.