Constrained, Grammar-Directed Generation of Landscapes

Mark Friedell and Jean-Louis Schulmann

Aiken Computation Laboratory
Harvard University
Cambridge, Massachusetts 02138

Abstract

Constrained, grammar-directed generative processes are investigated as a computational paradigm for scene modeling. This paradigm provides a theoretical framework for constructing modeling systems that perform an active function, generating some components of the scene automatically. We propose the term *active computer-aided design* (ACAD) to refer to this vision of user-computer cooperative scene modeling.

Formal grammars are used to imbue the modeling system with an elementary "understanding" of the kinds of scenes to be created. The grammar interpreter accepts components of the scene created by the system user as constraints on the scene to be generated. This approach to scene modeling harnesses the power of the computer to construct scene detail, thereby freeing the human user to focus on essential creative decisions.

1. Introduction

Scene modeling is frequently a very expensive, time-consuming process. In contrast to the synthetic camera framework for scene rendering, there is no powerful and comprehensive theory of modeling from which general-purpose automated modeling systems may be constructed. As a result, we observe that modeling consumes the majority of the resources expended in producing a "typical" graphics application.

In a few very specialized applications, completely automatic scene generation has been possible [MACINLAY] [FEINER] [FRIEDELL]. In each of these special cases, design decisions are sufficiently well understood (as a result of the research cited above) that no human intervention is required.

In this paper, we explore cooperative user-computer modeling. We envision applications in which it is necessary or desirable to allow human intervention in the design process. The goal is synergistic collaboration in which the human user exercises creative control while the machine rapidly constructs scene detail that is resonant with the principle design decision made by the system user. We refer to this vision of the computer as an active participant in design as *active computer-aided design* (ACAD) to distinguish it from the less complete role of the computer in conventional computer-aided design.

To support technically this vision of ACAD, we propose constrained interpretation of generative grammars as a computational paradigm. Within this paradigm, the generative grammar describes a language of scenes. It is used, in effect, to imbue the modeling with an elementary understanding of the type of scene to be constructed. Components of the scene that are created by the user serve as constraints; the grammar interpreter must select and apply rewriting rules to produce a scene that incorporates the user's design decisions.

This paper explores the essential ingredients of ACAD in the context of modeling three-dimensional landscapes, including the built environment. In section 2, we survey previous efforts to employ grammars as a means of describing classes of objects in computer graphics and other disciplines, and we propose in Section 3 a new type of grammar -- the *landscape grammar* -- for our experimental application area. Section 4 describes how landscape grammars are interpreted to generate scenes that comply with design decisions made externally, e.g., by human system users. Section 5 describes the function, architecture and implementation of an operational prototype system and provides examples of its use.

2. Previous Work

Formal grammars have been used in several very different applications to describe objects and processes to construct them. These efforts may be partitioned into 2 categories: those which manipulate topological structures and those which manipulate geometric structures.

Formal grammars that manipulate topological structures include graph grammars and L-systems. Although most studies of graph grammars are designed to explore their theoretical properties, some forms of graph grammars produce topological structures that can be interpreted graphically to describe physical objects [VOLKER et al.] [HARTMUT et al.].

Lindenmeyer developed L-systems [LINDENMEYER] as a theoretical model of the development of filamentous biological structures. L-systems may be viewed as specialized graph grammars restricted to mathematical trees. Recently, Smith [SMITH] showed how L-systems can be applied in computer graphics to construct topological branching patterns of physical trees.

Efforts to manipulate geometric structures via formal grammars include the very widely known fractal curves and surfaces [FOURNIER, FUSSELL & CARPENTER], and the less well known map grammars and shape grammars. Map grammars are used in theoretical biology to model developments of tissue cultures and cell divisions [PAZ].

Shape grammars are a tool of theoretical architecture. They serve as a mechanism for describing an architectural style by specifying a "language" of buildings [MITCHELL] [STINY]. Shape grammars are not, in general, amenable to implementation within computer systems, but must be interpreted "loosely" by people who can recognize can recognize approximately correct applications for rewriting rules and produce the appropriately altered rule application.

Finally, ontogenic grammars are the bases of all object modeling in GMS-1, an experimental generative modeling system [BEYER & FRIEDELL]. With GMS-1, every object description is expressed in two parts: 1.) a generative process, articulated in terms of an ontogenic grammar, that constructs the object's topological structure; and 2.) an explicit geometric interpretation for the topological structure. The theory of ontogenic grammars and GMS-1 were designed to integrate, through a uniform representation scheme, the use of conventional polygonal boundary representation with fractal curves and surfaces, L-systems, particle systems [REEVES], and other, specialized generative modeling processes.

3. Landscape Grammars

yThe words in the language generated by a landscape grammar are not ordinary three-dimensional scene descriptions, but rather *landscape schemata*, which serve as two-dimensional "blueprints" of three-dimensional scenes. Landscape schemata comprise area, line and point features which specify the identities, locations, orientations and abstract shapes of the objects in the scene. A simple transliteration process, described in Section 4, reformulates schemata in terms of conventional scene descriptions that may be rendered by ordinary three-dimensional rendering techniques.

The area features of a landscape schema form a contiguous tiling of polygons delineating geographic regions of various types such as forest, residential area and industrial site. Superimposed over this tiling of area features is a collection of line features describing lineal phenomena such as roads, rivers and power lines. Point features located within area features describe objects such as buildings and trees. Area features and point features are directionally oriented as shown by the vectors in Figure 3.1.

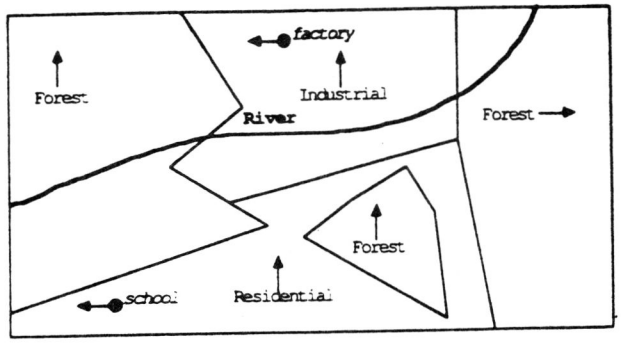


Figure 3.1 A Simple Landscape Schema

Landscape grammars guide directly the manipulation of geometry, rather than develop topological structures whose geometric properties must be later derived. A landscape grammar is described in terms of two types of rewriting rules, one that rewrites area features and another that generates points. As shown in Figure 3.2, the left side of an area-rewriting rule specifies the type of an area feature along with minimum and maximum area requirements. These area requirements are expressed approximately in terms of the minimum and maximum width and height of a box bounding the area feature to be rewritten.

The right side of an area-rewriting rule is a rectangular mosaic of polygonal regions. This mosaic is referred to a *partitioning space* and the polygonal regions are termed *area masks*. The partitioning space is directionally oriented, as is each area mask. Additionally, each area mask is annotated with an area-feature type. An area-rewriting rule may be applied to any area in the schema whose type and

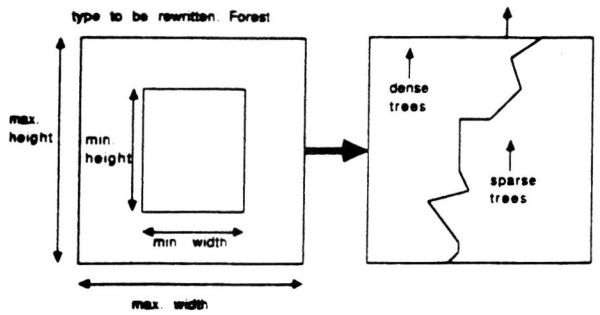


Figure 3.2 Rule for Rewriting Area Features

dimensions satisfy the specifications given by the rule's left side.

When an area-rewriting rule is applied, the rule's partitioning space is rotated so that its orientation vector is parallel to that of the area feature being rewritten. Next, the partitioning space is scaled independently in both orthogonal dimensions to form a bounding box around the area feature as shown in Figure 3.3. Finally, assuming the width and height of this bounding box satisfy satisfy the geometric constraints given by the left side of the rule, new area features are created by finding the polygonal intersections of the area feature being rewritten with each mask in the partitioning space. The types and orientations of the new areas features are given by the area masks.

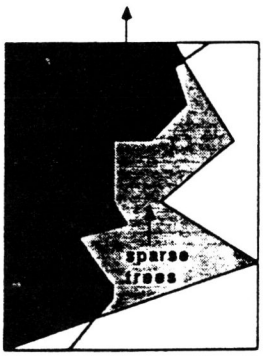


Figure 3.3 Rewriting Forest Area Feature in Figure 3.1

Rules for generating point features are similar to those for rewriting area features, and their left sides are identical. Instead of a partitioning space on the right side, however, a point-generation rule has a *generation space*. A generation space is an oriented rectangular region that is populated by generation points, each of which is oriented and annotated with a point-feature type, as shown in Figure 3.4.

When a point-generation rule is applied, the generation space is rotated and scaled as described above for partitioning spaces. Assuming the generation space satisfies the spatial constraints given by the left side of the rule, point features are created in the schema corresponding to the types, locations and orientations of the generation points in the rule. The application of a point-generating rule is illustrated in Figure 3.5.

3.1 The Organization of Landscape Grammars

Except for the simplest circumstances, a landscape grammar described by an unstructured collection rules is both incomprehensible and unmodifiable. We achieve better results by organizing area-rewriting rules into three kinds of reusable, higher-level abstractions: *replacements*, *subdivisions*, and *residues*. Graphical notations for these abstractions are used to create a diagram of the structure of a grammar, which facilitates its design, coding, testing and later modification.

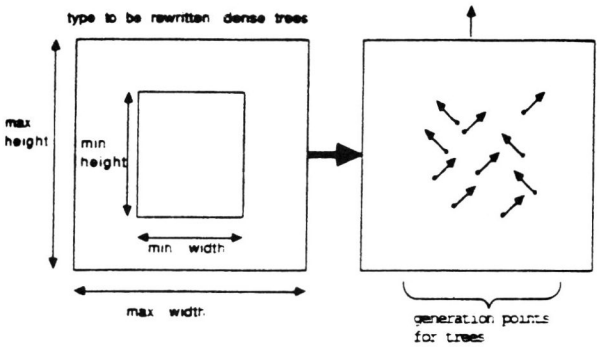
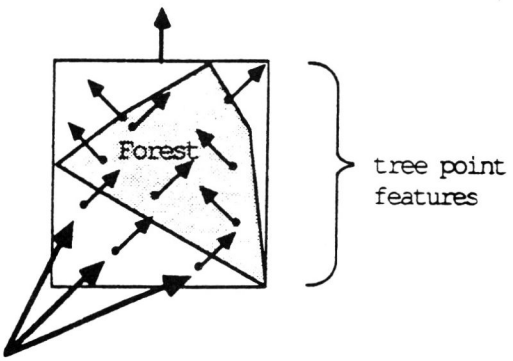


Figure 3.4 Rule for Generating Point Features



these generation points do not produce point features in schema

Figure 3.5 Applying a Point-Generation Rule

Replacements transform one area feature into a fixed number of other area features. They are ordinarily coded with a single area-rewriting rule and are described diagrammatically with a hyperedge. Figure 3.6 illustrates the replacement of a residential block by space for a single-

family dwelling, 2 horizontal roads, and 2 vertical roads.

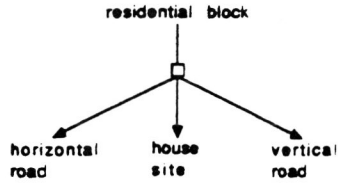


Figure 3.6 A Replacement

Subdivisions transform an area feature of one type into 1 or more area features of another type. This mechanism is used to convert a single area feature of unknown size into a collection of area features of whose dimensions are bounded by known quantities. We use a double-headed arrow to describe a subdivision, as shown in Figure 3.7, which depicts a residential area being subdivided into a collection of residential blocks.

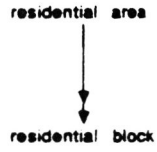


Figure 3.7 A Subdivision

Residues replace one area feature by another, with the possible generation of 1 or more additional, residual area features of a third type. Figure 3.8 shows a residue that replaces a horizontal road with a pavement area feature, producing some number of road-shoulder residue features.

3.2 Theoretical Observations

Although the rules in a landscape grammar are applied in a context-free fashion, the degree to which this limits the generative power of landscape grammars is not clear. In particular, context-free multi-pointed hypergraph grammars (CFMPHGG) [HABEL & KREOWSKI] can be reformulated in terms of landscape grammars by replacing hyperedges with area features and nodes with edge boundaries between area features. Since the generative power of CFMPHGG is known to include some, but not all, context-sensitive string languages, landscape grammars are at least that powerful.

The practical limits of landscape grammars are also not yet fully understood, and we are still learning how write rules for increasingly complex landscape languages. To date, our principal challenge has been to avoid contextually inappropriate rule applications. For example, a specific landscape grammar might include a point-generation rule to create a church on a corner building lot in a residential community. The grammar might also include a rule to create an automotive service station on the same type of corner lot. Although both rules make sense in isolation, we might prefer that contiguous corner lots in the generated landscapes not be used to construct a church and a service station side by side.

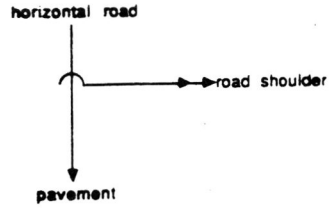


Figure 3.8 A Residue

To avoid this kind of difficulty, we would expand the set of area-feature types in the grammar to include specific corner lots for churches and service stations, and devise replacement structures in the grammar to ensure that these two special types of corner lots may never be continuous. Essentially, contextual applicability predictates are stated implicitly in the context-free rules. Occasionally, when the applicability of a rule depends on a complex context, this approach is difficult to follow. We are presently exploring disciplines for writing grammars that are intended to make this process easier and more reliable. Ultimately, however, it may be necessary to employ a context-sensitive grammar. This option would unfortunately make the grammar interpreter more difficult to construct and the interpretation process potentially much more expensive.

4. Constrained Interpretation

The task of the grammar interpreter is to create a schema for a landscape that is

1. in the language of landscapes defined by the landscape grammar; and
2. compliant with all of the user's design decisions.

The user's design decisions are expressed in the form of an initial, undeveloped schema, which we refer to as the *start schema*. The start schema specifies the identities, locations, and orientations of any area, line or point features that user chooses to include. From a language-theoretic perspective, the start schema is the starting configuration for the generative process described by the grammar. The final schema is then a word in the language defined by the grammar that consists only of "terminal symbols" understood by the schema transliteration process. The output of the transliteration process is a conventional three-dimensional scene description. The user's design decisions, captured in the start schema, may be viewed as constraints on the behavior

of the generative process described by the landscape grammar.

Since each area feature in the start schema is developed in a context-free fashion by the scene grammar, area-feature constraints are satisfied easily. Point features in the start schema are much more difficult to accommodate. We must ensure that the final schema contains all user-specified point features and that they are embedded in area features only as allowed by the point-generation rules of the landscape grammar. To guarantee that these requirements are met, the application of rewriting rules is carefully restricted.

Consider an area feature, A, containing a point feature, P, in a partially developed schema. Assume that rule R is chosen as a candidate to rewrite A and that the dimensions of A satisfy the geometric constraints on the applicability of R. R can now be applied to A only if for each new region that would be created, N_i, one of the following conditions is met:

1. N_i does not contain P;
2. N_i is allowed to contain P directly (which can be determined by examining the point-generating rules of the scene grammar); or
3. N_i may be developed into a subschema that correctly includes P (development of this subschema might require many applications of area-rewriting rules).

In principle, a geometric theorem prover is needed to determine whether the requirements for applying rule R are met. We have found in practice, however, that a much simpler and faster technique requiring only a symbolic theorem prover and a relatively simple point-in-polygon inclusion test works very well in almost all cases.

For each N_i, we first use the inclusion test to determine if P lies within N_i. If it does not, clause 1 is satisfied. If it does, we then check the point-generating rules of the grammar to see if N_i is allowed to contain P directly. If so, clause 2 is satisfied.

If neither cause 1 nor clause 2 is satisfied, we are faced with the more challenging task of determining whether N_i could be developed into a subschema such that some area feature in the subschema would correctly contain P. We attempt to make this determination by considering the converse of what is required, specifically, that no such subschema could be developed from N_i. We attempt to prove that this is the case with a comparatively simple symbolic theorem prover that uses the rules of the landscape grammar as inferences in a forward-chaining fashion. Because this is just a symbolic theorem prover, only the identities of area features are reasoned about; their geometric properties are ignored completely. If we succeed in proving the converse of what is required to satisfy clause 3, then rule R cannot be used to rewrite area feature A. If the proof fails, however, we presume optimistically that the required subschema could be generated. Of course, this presumption could be false, since the failure of the proof does not necessarily mean that the conditions for meeting clause 3 can, in fact, be met. When these presumptions occasionally fail in practice, user intervention is required to avoid generating an incorrect landscape.

5. An Experimental Prototype

We have implemented an experimental landscape generator which we use as a tool in our exploration of active computer-aided design technology. This system accepts from the user the identities, locations and orientations of area, line and point features which the user wishes to include in the landscape, and then generates a complete scene that is consistent with these design decisions. The top-level architecture of the system is shown in Figure 5.1. The Landscape Composer accepts design decisions from the user and produces a complete landscape schema using a user-specified landscape grammar. The Landscape Instantiator transforms the schema into a conventional three-dimensional graphical description which may be rendered by an ordinary renderer.

The Composer establishes a start schema from the user's design decisions through the use of two kinds of unembedding operations, whose effects on the start schema allow the development of area features to proceed independently. Any area feature that contains a line feature is split along the line feature and any area feature that surrounds another area feature is split to eliminate the surrounding relationship. The results of these unembedding steps for the example in Figure 3.1 are shown in Figure 5.2.

The Interpreter develops a complete schema in two sequential phases: area-feature development and point-feature generation. During the first phase, an area feature is selected arbitrarily for rewriting. Next, all rewriting rules that are applicable to the type of the selected area feature are collected, and one is chosen probabilistically, based on the rules' assigned priorities. If the geometric applicability criteria of the rule are satisfied by the area feature, and if the

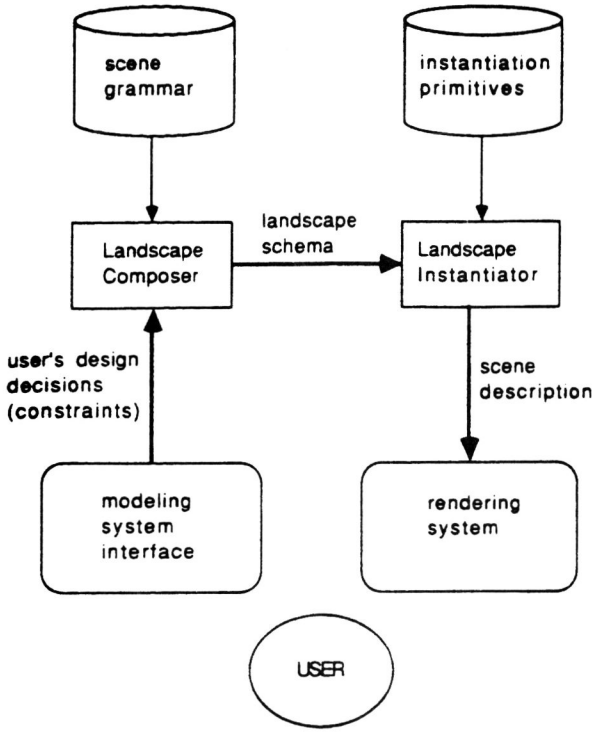


Figure 5.1 Architecture of Landscape Generator

area feature contains no point features, then the rule is applied. If the area feature does contain point features, the Interpreter's embedded theorem is used to verify, as discussed in Section 4, that the application of the rule is compatible with the point features. If for some reason the selected rule cannot be applied, another rule is chosen. If no rule can be applied, the area feature is marked as a terminal feature and becomes part of the final landscape schema.

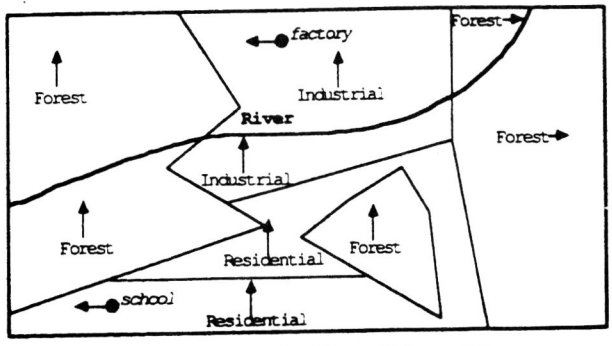


Figure 5.2 Unembedding of Figure 3.1

Point-feature generation is applied to only those area features that do not already contain point features specified by the user. For each of these initially empty area features, a process similar to that for area-feature development is used to select and apply a point-generation rule. After area-feature development and point-feature generation are complete, the final schema is passed to the Landscape Instantiator.

The Instantiator is a simple process that assembles a three-dimensional scene description using the landscape schema as a "blueprint." For each area feature, an appropriate color or texture definition is selected from a table associated with the Landscape Grammar. Point features are replaced by corresponding three-dimensional object descriptions. Typical point features include houses of various kinds, factories, office buildings, churches, trees, cars, boats, etc. For some types of point features, a stochastic process is used to select among different graphical instantiations to create a sense of natural variation. For example, this technique may be used to instantiate the trees in a forest.

5.1 Examples

Consider first a very simple example: a grammar describing, in coarse detail, suburban developments of tract houses. The example grammar expresses these scenes only in terms of houses, yards, roads, and soil along road shoulders, and is almost as trivial as the architectural vision of the developers who undertake tract-house projects.

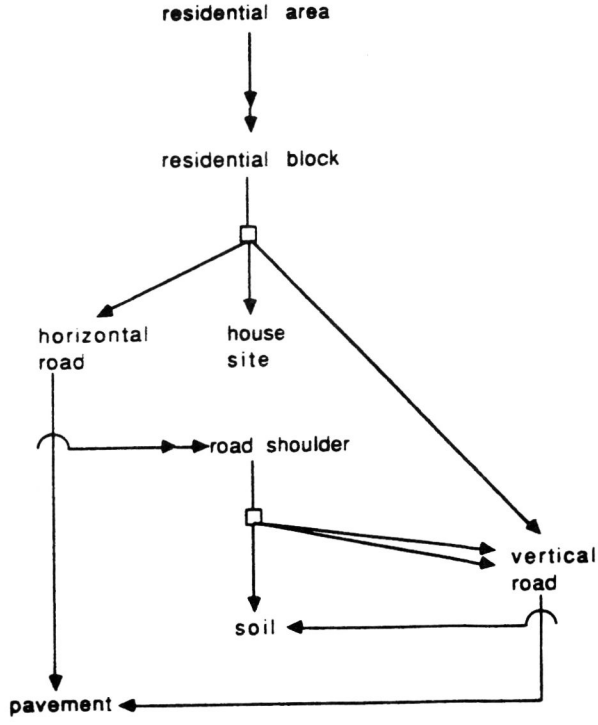


Figure 5.3 Structure of Example Grammar

The organizational structure of the grammar is shown in Figure 5.3. Figure 5.4 shows graphically the area-rewriting rule that replaces residential blocks with yards, vertical roads

and horizontal roads. The textual form of the rules for the residue structure that transforms vertical roads into pavement and road shoulder soil is given in Figure 5.5. Note the use of recursion in this residue structure and the limit case of the recursion that is triggered by the rules' geometric applicability predicates. Finally, Figure 5.6 presents a point-generating rule for creating a house. The scene language described by this very simple grammar permits almost no variability the generated scenes. The only design decisions permitted to the user are the selection and placement of a few different types of houses.

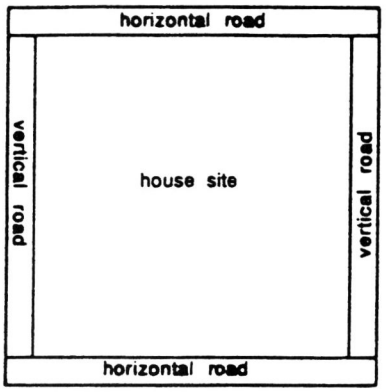


Figure 5.4 Replacement of Residential Block

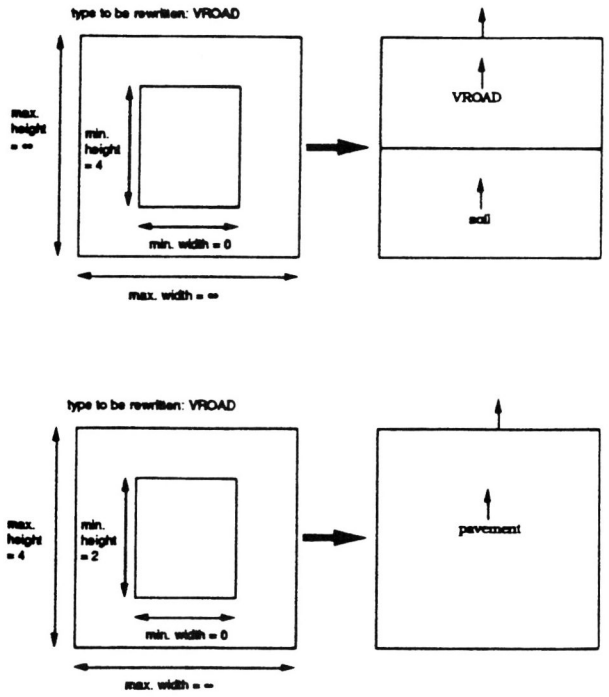


Figure 5.5 Residue Structure for Vertical Roads

A somewhat more complex grammar, described in approximately 300 lines of text, was developed from this simple example and used to create the scenes in Figures 5.7 and 5.8. This extension of the example grammar defines a much more variable landscape language and permits the user to make many more design decisions, thereby enabling the specification residential areas, industrial areas, forests, bodies of water, houses, industrial buildings, trees and roads.

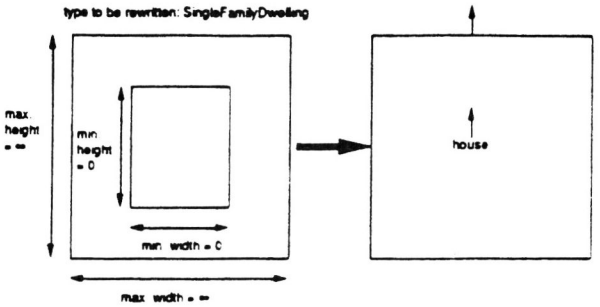


Figure 5.6 Point-Generating Rule to Create a House

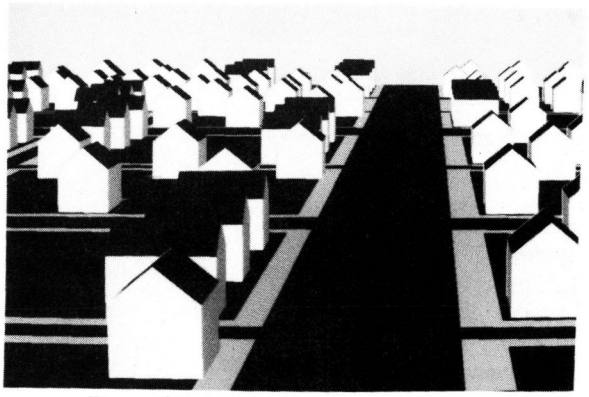


Figure 5.7 ACAD-Generated Scene 1

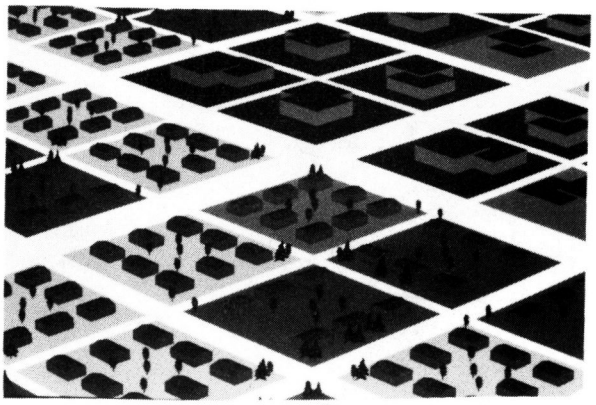


Figure 5.8 ACAD-Generated Scene 2

6. Conclusions and Future Research

We have explored constrained, grammar-directed generative processes as a theoretical model for graphical scene modeling. This model encompasses fully automatic, completely manual, and, most important, cooperative user-computer scene generation.

Our vision of user-computer cooperative modeling places the computer in an active role, creating detail in the scene that resonates with creative choices made by the human user. We suggest that such cooperative modeling systems be referred to as *active computer-aided design* (ACAD) systems.

In this paper, we have described an experimental ACAD system for modeling landscapes, including the built environment. We have discussed the type of scene grammar which this system uses and how the system represents and accommodates design decisions made by the system user.

At present, we are expanding the ideas presented in this paper in three ways. First, we are investigating the user of conformal mappings in rewriting rules for manipulating geometry. Potentially, this would allow structures such as the partitioning space and the generation space discussed in Section 3 to conform precisely to feature being manipulated. Second, we are exploring representations for rewriting rules that manipulate directly three-dimensional features rather than using an intermediate representation such as the landscape schema. Third, we are exploring ways to "tighten the loop" of interaction between the user and the system, possibly allowing the user to guide more precisely the system's choice of rewriting rules.

Scene modeling is the most expensive part of most computer-graphics applications. In stark contrast to rendering technology, modeling technology has not improved significantly during the last 15 years. We believe that this lack of progress is due, in large measure, to the absence of a comprehensive theoretical framework for modeling -- one that is comparable to the synthetic camera theory of rendering. This paper explores the foundations of a theoretical perspective for modeling from which algorithms and architectures for active computer-aided design may be developed and integrated.

7. References

[BEYER & FRIEDELL]
Beyer, T. and Friedell, M. "Generative Scene Modelling." *Proceedings of Eurographics '87*, Elsevier Science Publishers (North Holland) 1987.

[FEINER]
Feiner, S. "APEX: An Experiment in the Automated Creation of Pictorial Explanations." *IEEE CG&A*, November 1985.

[FOURNIER, FUSSELL & CARPENTER]
Fournier, Alain, Don Fussell and Loren Carpenter, "Computer Rendering of Stochastic Models", *Communications of the ACM* 25, 6, pp. 371-384 (June 1982).

[FRIEDELL]
Friedell, M. "Automatic Synthesis of Graphical Object Descriptions." *ACM Computer Graphics*, 18, 3.

[HABEL & KREOWSKI]
Habel, Annegret and Kreowski, Hans-Jorg. *Some Structural Aspects of Hypergraph Languages Generated by Hyperedge Replacement.* Lecture Notes in Computer Science #247, Springer-Verlag 1987.

[HARTMUT et al.]
Hartmut, E., Nagl, M. and Rozenberg, G. *Graph-Grammars and Their Application to Computer Science.* Lecture Notes in Computer Science #153, Springer-Verlag 1983.

[LINDENMAYER]
Lindenmayer, A. "Mathematical Models for Cellular Interactions in Development, Parts I and II." *Journal of Theoretical Biology*, 18, pp. 280-315, (1968).

[MACKINLAY]
MacKinlay, J. "Automating the Design of Graphical Presentations of Relational Information." *ACM Transactions on Graphics*, 5, 2.

[MITCHELL]
Mitchell, W. J, "Synthesis and Style." *Proceedings of the International Conference on the Application of Computers in Architecture, Building Design, and Urban Planning.* Berlin, (May 1979).

[PAZ]
Paz, A. "Geometry Versus Topology in Map Grammars." In Hartman, Nagl, and Rozenberg (Eds.), *Graph-Grammars and Their Application to Computer Science.* Lecture Notes in Computer Science #153, Springer-Verlag 1983.

[REEVES]
Reeves, William, "Particle Systems-A Technique for Modeling a Class of Fuzzy Objects," *ACM Computer Graphics* 17, 3, pp. 359-376 (July 1983).

[STINY]
Stiny, G., "Introduction to shape and shape grammars," *Environment and Planning B*, 7, pp. 343-351 (1980).

[SMITH]
Smith, A. R. "Plants, Fractals, and Formal Languages." *ACM Computer Graphics*, 18, 3.

[VOLKER et al.]
Volker, C., Hartmut, E. and Rozenberg, G. *Graph-Grammars and Their Application to Computer Science and Biology.* Lecture Notes in Computer Science #73, Springer-Verlag 1976.