Vision, Modeling, and Visualization (2010), pp. 1–8

Graphical Interface Models for Procedural Mesh Growing

Stefan Menz, Holger Dammertz, Johannes Hanika, Michael Weber and Hendrik P. A. Lensch

Ulm University, Department of Media Informatics, Germany

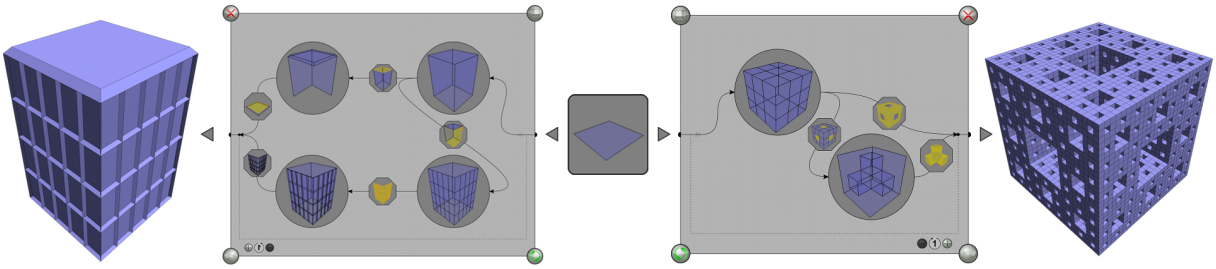

Figure 1: *A basic input model is intuitively transformed into different complex models. The left group of operations depicts the construction plan for a modern building and the right group represents a visual rule to generate a Menger sponge of arbitrary recursion depth.*

Abstract

Procedural modeling allows to create highly complex 3D scenes from a small set of construction rules, which has several advantages over storing the full data of an object. The most important ones are a very small memory footprint and the ability to generate infinite variations of one prototype object by using the same set of rules. However, the problem that procedural modeling imposes on the user is to define a reasonable set of rules to generate a specific object. To simplify this task, we present new interaction metaphors for a graphical user interface and a minimal set of geometric operations that allow the user to efficiently create such rules and the respective models. These metaphors are then implemented in a prototype system and are evaluated by user tests with regard to usability and user performance. The results show that the system enables even inexperienced users to create complex 3D objects via procedural modeling using the presented approach.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [User Interfaces]: Graphical User Interface (GUI), Interaction Styles, Evaluation I.6.3-5 [Simulation and Modeling]: Applications; Model Validation and Analysis; Model Development J.5 [Arts and Humanities]: Architecture

1. Introduction

The growing demand of highly detailed geometric models for feature films and computer games make a purely manual creation no longer feasible. A powerful yet simple technique to create complex models with a very low memory footprint and without the need for offline storage is procedural modeling. A procedural representation usually consists of a *grammar* (i. e. a set of rules) that describes how an object is constructed in general and *parameters*, which specify the properties of a certain instance. Therefore, even a simple grammar, that consists of a basic set of rules, can yield highly complex models by means of recursive expansion. This property is also known as "data amplification" [Smi84].

As powerful as procedural modeling might be, it remains a demanding and unintuitive task to create the rules. To support the user in this process, we propose a novel graphical user interface approach for procedural modeling of geometry. Its main concepts are continuous visual feedback about all user interactions and a very general modeling technique that allows for the creation of a wide variety of models.

2. Related Work

L-Systems were introduced by Aristid Lindenmayer to serve as a formal description for cellular interactions in multicellular organisms [Lin68], but quickly evolved to an efficient formalism for the definition of botanic structures, i. e. plants

and trees. L-Systems represent a class of parallel string rewriting systems and are defined by a formal grammar that defines a set of replacement rules. The evaluation of a L-system starts with the *axiom*, i. e. the base rule, and in each expansion step the current string is parsed linearly and each encountered symbol is expanded according to its respective rule. Further extensions of L-Systems employed more powerful context-sensitive, parametric grammars [PL90], which allow to create complex branching structures and also account for several external influences (gravity, photo-tropism, wind, topiary, etc.) [Han92, PJM94, PK96, DHL*98].
Even though L-systems provide an elegant concept to represent plausible and quite realistic plant models, their textual representation is not very intuitive. Thus a deep understanding of the underlying concepts and processes is required to create rules to generate a desired result.

Besides their successful application for plant models, L-Systems have also been used for the modeling of cities, especially for the underlying street network [PM01]. However, they are not well suited for the generation of complex buildings featuring highly detailed facades, since the main concept behind L-Systems is the simulation of growth in open space [WWSR03]. For this purpose an integrated modeling approach utilizing a *split grammar* was presented [LWW08, MWH*06]. This grammar class represents a derivation of the more general shape grammars introduced in [Sti75, Sti80].

Another approach that focuses on architecture, but is also well suited for other model classes is the *Generative Modeling Language* (GML) [Hav05], a stack-based postfix scripting language similar to *PostScript*.

The modeling methods reviewed so far all employed a kind of formal, textual grammar, however this is not the only way to obtain good results. For example, an integrated system for plant modeling is *Xfrog* [LD98], that produces quite realistic plants by employing a procedural approach similar to scene graphs [Str93]. These directed graphs are called *structure graphs* (also *prototype graphs* (p-graph)) and represent the overall structure and the relation of different plant components. In a first step the generic rules described in the p-graph are translated into an *instance tree* (i-tree), which represents all instanced components. Then the i-tree is used to generate the final model. The nodes are used to visualize applied operations with icons; different edge types differentiate normal parent-child relations from recursive ones.
This visualization of components and their according rules, provides a more natural and intuitive interface for the user than the editing of a textual grammar.

Even though the approaches presented so far produce quite convincing results, they also have certain drawbacks. All of them are more or less specialized for a certain class of objects (e. g. buildings, plants, etc.). Additionally, the systems based on textual grammars consequently suffer from their inherent abstract representation and are thus unintuitive for unfamiliar users. Before the user is able to use the system he or she must learn the modeling language and develop a mental model of how the operations work and their influence on subsequent operations. Once the required experience is attained the editing process is still not straightforward, because the user does not receive progressive feedback about his actions. The usual work-flow in this scenario is divided into three steps. First the user makes changes to the build rules, then the rules need to be reevaluated and finally the user must inspect the result for its correctness. If the result is not as anticipated the user needs to make further changes to the rules until the desired result is obtained.

To overcome these limitations, a new system is presented that allows for the modeling of more general objects, while still providing the full potential of procedural grammars. Nevertheless, the system offers a highly intuitive interaction even for inexperienced users. For this purpose, the proposed system is based on the following two approaches.

While split grammars are specialized for architecture, L-Systems can be used more versatile. However, L-Systems usually are evaluated in a two step process. First, the grammar is expanded to a high-level description of a skeleton and in a second step the actual geometry is generated by interpreting this skeleton string. The second step is quite problematic, because the choice of a good skinning heuristic is dependent on the object class and also the handling of features like holes requires additional data structures.
In order to avoid these problems and realize a more general modeling approach our system uses *mesh-based parametric L-Systems* [TMW02, Mai02]. In contrast to classical L-systems, this method uses a growth metaphor to modify a mesh directly by attaching growth rules to individual faces of the model. This natural concept for procedural modeling also integrates more complex features like holes seamlessly, since holes can be interpreted as a growth into the model itself. This approach easily allows for the visualization of intermediate results of particular parts of the model. For the sake of simplicity, only a minimal set of operations is implemented that offer the same complexity as traditional box modeling techniques do.

Model Graphs, an integrated framework for procedural modeling was introduced in [GK07]. Similar to Xfrog, this approach is based the visual data flow pipeline (VDFP) paradigm [Ack82, Mor94, JHM04]. VDFPs visualize the structural dependencies and states of operations and therefore provide an intuitive and a more memorable interface compared to textual programming languages. Since the VDFP paradigm has already been successfully used in the creation of materials [Coo84], it is a reasonable conclusion to employ the same metaphor for the modeling of geometry.

A model graph is represented by an directed acyclic graph (DAG) that forms a network of operations. The operations are visualized as nodes, which store variables and functions for its respective operation. The possibility to input complex

formulas offers a balanced mix of visual and textual programming, instead of a more intuitive purely visual pipeline. An interesting feature that allows to define iterated rules that operate locally on the model and not globally like L-systems do is provided by the `for`- and `while`-operators.

The combination of both systems provides a reasonably general modeling technique as well as a guideline for the development of the interface concepts.

3. Interface Design Principles

The main goal of this work is to offer the user an efficient way to navigate the design space, i. e. the space that spans all currently reachable model variations defined by the current graph and the parameters of all operations.

The Model Graphs system provides a reasonable approach, but it has two vital shortcomings that hinder the interactive visual editing of procedural models. First, it requires a large amount of textual editing for the node parameters, which is reasonable for expert users, but hinders the exploratory learning experience of novel users. In particular, this makes it very difficult to understand graphs build by other users. Second, nodes only visualize the type of operation by icons and therefore lack essential feedback about parameter changes and the resulting influences on other connected nodes. To fix these problematic issues, the presented system avoids textual editing and focuses completely on the constant visual feedback on all user actions. The provided visual cues guide the user interactively and enable him/her to make informed decisions to achieve the desired result. The presented system does not aim to generate perfect results that can be used as they are. Instead it tries to create plausible proxy-objects, that can be used as templates and be modified as necessary.

Since this work represents a proof-of-concept study it is essential to ensure an efficient and flawless user interaction with the system. Otherwise it could not reliably be determined if the proposed modeling concept is flawed or if poor performance results are purely related to a lack of usability.

To ensure unbiased performance results it is necessary to employ usability engineering techniques. For our system we used a heuristic evaluation approach [Nie94, NM90] realized by paper prototyping [Nie90, Sny03].

A set of possible interaction scenarios were developed in an iterated design process by presenting paper prototypes (screenshots, mockups) to three subject matter experts and incorporating the resulting feedback to refine the interface design and interaction concepts.

4. Proposed Modeling Approach

Based on the model graphs concept, we propose a graph oriented modeling approach for the creation of procedural geometry. For the representation of the geometry graph, further called a GeoGraph, a directed acyclic graph is used. This is chosen, because it is problematic to define how cycles for recursion should be handled. However, the iterated application of operations is an important feature and requires a special handling to integrate in this restrictive graph layout. In general, a GeoGraph is a network graph that supplies a base mesh in a global source node and all operations that alter the input model contribute to a global sink node that collects all components of the output model.

4.1. Graph Structure

Each geometry graph or *GeoGraph* is realized as a network graph with exactly one global source and one global sink, represented by specialized input and output nodes. The input node always propagates a loaded base mesh and the output node merges all arriving geometry input into the final model. Therefore, the described network is a data-flow graph that transports faces along the edges with an overall flow direction from the root (i. e. the input node) to the leaves, which are all connected to the output node.

Each node performs some basic operations on its received faces, i. e. faces are either created (duplicated, new faces), deleted (terminated) or modified (transformed). The respective operations are only applied if they were selected for processing in the previous node supplying them. Selections are mutual exclusive sets of all previously selected or newly created faces. Therefore any node can maximally have n outward connections, where n is equal to the number of previously selected faces and newly created faces.

4.2. Operators

Extrude The extrude operator implements a simple extrusion of a given face of the model into the third dimension. This represents the growing metaphor of the mesh growing approach. Besides its height h, this operator also features a scaling of the extruded face (x_{in}, y_{in}) and allows for the construction of tapered structures (see Figure 2, left). In contrast to other approaches this natural growing metaphor allows to easily define holes in a model, simply by performing a growing step into the model.

Subdivide To complete the growing metaphor also a refinement operator is needed, to allow for the placement of structures and details on different scales. Therefore a simple subdivision operator is provided that refines a face by a defined number along the x- and y-axis (s_x, s_y) (see Figure 2, middle).

Pyramid As a growth terminator a pyramid operator was implemented that marks a face terminated by generating a pyramid on the face. In fact, this is not a new standalone operator, but merely a specialized version of the extrude operator, where the extruded face is collapsed to a single point (see Figure 2, right).

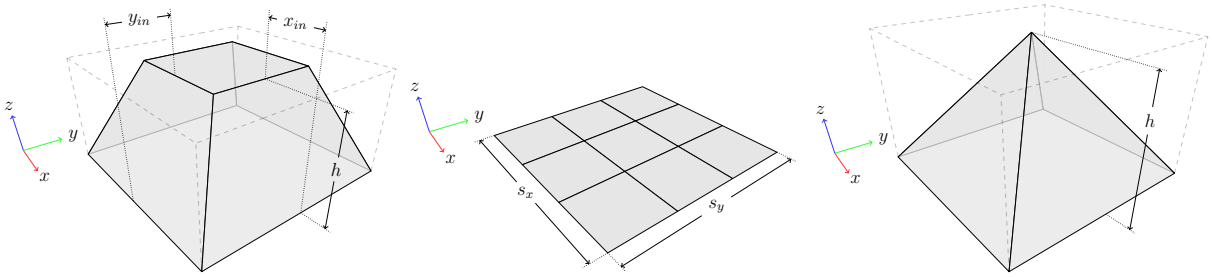

Figure 2: *The three supported operations extrude, subdivide and pyramid (left to right) with their respective parameters.*

Grouping and Iteration Even though a GeoGraph has been defined as an acyclic graph it is essential to support recursive operations so that complex features can be generated on different scales. For this purpose a specialized group node is introduced which can encapsulate a set of connected nodes and allows for the iterated evaluation of this subgraph.

Foremost the group node is designed to define more complex compound rules consisting of a subgraph of multiple basic operators (see Figure 3). The contained nodes can be viewed and edited as in their normal state, but the group node also allows to collapse the subgraph to a single node that represents the overall operation of the rule in a preview image. This not only leads to a more simplified representation, but also gives a better overview of the overall graph structure since the user can eliminate unnecessary clutter and regains more real estate for display of other parts of the graph.

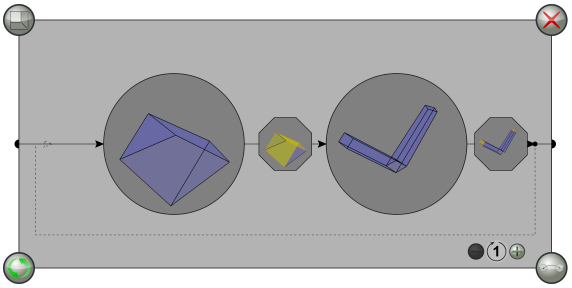

Figure 3: *A build rule for a simple branching structure as it is used in the coral test scenario. This rule can be recursively applied to the selected branch tips by simply increasing the iteration count (bottom right).*

Since each group node acts as a decoupling mechanism between the overall graph and the encapsulated subgraph, this interface property can be used to define different action for the internal and external side. For instance, a group can handle recursive operations on the internal side while respecting the global acyclic constraint. This way a rule can be recursively applied to the selected faces, which are directly fed back to the entry point of the group node. Furthermore the recursion of a node only is stored with exactly one integer in the model file, because all necessary construction information is encoded in the graph connectivity.

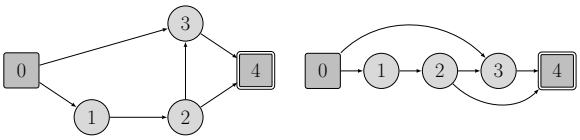

Figure 4: *Evaluation order of a topologically sorted graph and its more intuitive inlined representation.*

To avoid multiple evaluation of nodes or an complex schedule strategy the GeoGraph sorted topologically to obtain the sequential evaluation order of all nodes; since the graph is acyclic it is guaranteed that a topological sort is possible. So each node is processed exactly once, whereas group nodes may iterate multiple times before they provide input for their respective child nodes.

4.3. Parameter Randomization

Up until now each evaluation of a GeoGraph always produced the exact same result. In general, the natural look of an object is determined by a slight variation of features on different scales, especially for organic structures this is essential. To generate such variations an imperfect factory metaphor is implemented by a parameter randomization mechanism. With this functionality the user is able to generate a set of similar yet distinct objects (e. g. trees composing a forest) by randomizing parameters of selected model rules within definable bounds. However, the evaluation of randomized models remains deterministic to avoid completely random results and a new model instance is only generated if it is explicitly requested.

5. Interaction Metaphors

5.1. Assembly Line Processing

Since procedural formalisms represent objects as construction plans instead of the geometric data representing a certain model, it is a very natural way to view the modeling process as a visualization of an assembly line with distributed workstations that construct the different components that make up the final object. Each operator node resembles an independent workstation that performs a specialized, basic

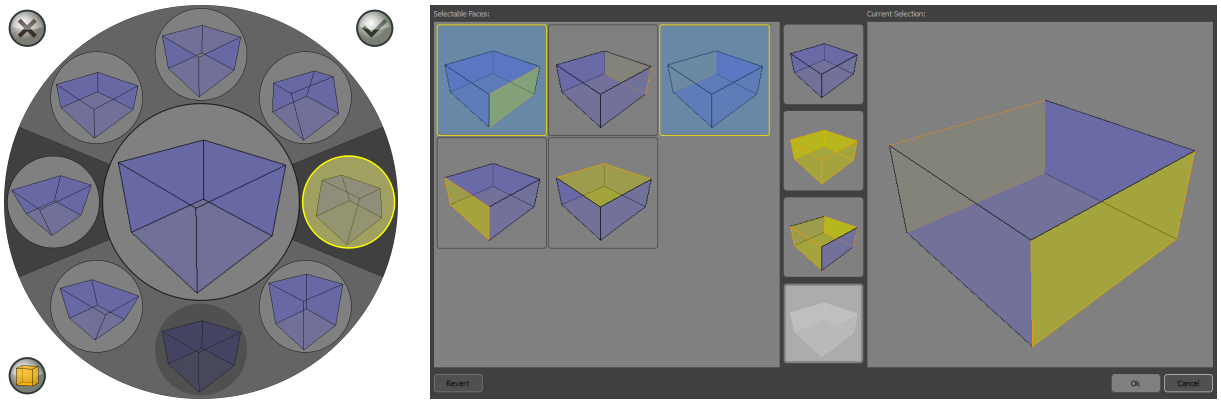

Figure 5: *On the left the variation view is shown. Its previews to the left always display the result of a decreased parameter and their respective counterpart to the right visualizes the result of an increased one. The selection editor on the right consists of individually visualized previews of all selectable faces, the four selection presets and the combined visualization of the current selection (left to right in the right dialog).*

operation on the currently available and for processing selected resources. A set of nodes that is grouped together forms a macroscopic workstation that is able to perform a more sophisticated and complex operation on its input. To complete this view, the edges are the respective construction orders of an operation that regulate which components are processed in other workstations.

5.2. Variation View

For a complex task as modeling it is essential to provide the user with immediate feedback about his actions, so that he can develop an understanding of the parameters and their influence on the model.

In design galleries [MAB*97] a concept for setting multi-dimensional parameters was presented. This approach formed the basis of the development of the variation view, a preview based property editor for the setting of node parameters that actively supports the user in the exploration of the design space of the current model (see Figure 5, left).
Each one-dimensional parameter (e. g. height, subdivision step, etc.) is represented as a single parameter axis. All parameter axes of a node taken together span the local design space of the node. A parameter axis is visualized by two opposing buttons, which preview the respective results a decrease or increase of the selected parameter would generate. According to layout conventions in western countries, negative parameter changes are placed on the left and positive ones on the right side. While the node always visualizes the final result of its operation, it is possible to toggle the complexity of the parameter previews between a full evaluation of all input faces and only of a single input face. The number of representable parameters is limited by the resulting size of the preview for each parameter. But since each basic operator currently features only a limited set of parameters, this is in general not an issue.

The variation view aims to provide an exploratory learning approach that supports the user to develop a deeper understanding of the different operator settings. However, the variation view is not intended to completely replace a traditional property editor, but rather to augment it with a visual alternative for novice users. Once a user has gained enough experience and is able to make informed decisions without the need to rely on the variation approach he or she can set parameters more efficiently and more precisely with a traditional property editor.

5.3. Selection Editor

In a conventional 3D-picking approach, the user would navigate around the object and directly select faces of a mesh. However, implementing the selection editor this way would disturb the overall feel of the user interface, since the previews in the nodes are only static images. To ensure a consistent user interface the selection dialog is also image-based and allows for the interactive selection of faces. This implicitly gave us the opportunity to investigate this novel approach, especially regarding its usability as an alternative selection approach for geometry (see Figure 5, left).
The dialog displays all selectable faces on the left, each thumbnail visualizing the selection of a single face. On the right the combined preview of all currently selected faces is shown. Also a set of reasonable quick selectors is provided to allow for easy selection of none, all and the complement of the current selection. Furthermore, a more specialized selector is implemented, which selects all remaining faces, i.e. all faces that are not yet selected by other outgoing edges.

Currently, the selection process operates on prototype indices only, i. e. all top faces generated in an extrude operation have the same prototype index and can only be selected in total. Therefore, it is not possible to create sub-selections of individual faces right now.

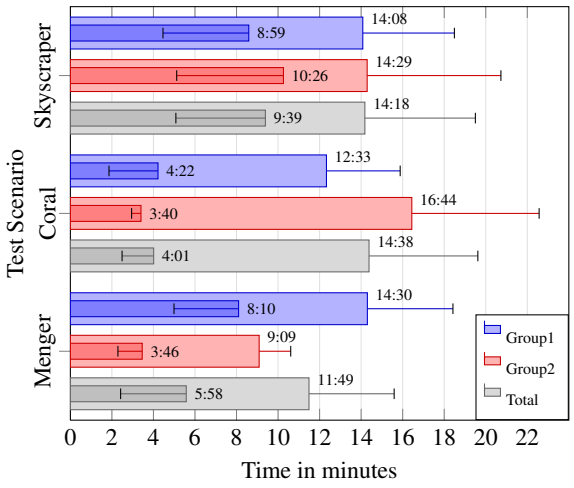

Figure 6: *Average time taken to complete the different test cases for group 1 (old group node, blue) and group 2 (new group node + additional tutorial, red). The gray bars represent the time averaged over both groups. The inner bars represent the average time required to complete the sub-task of creating the basic model build rule. The error bars represent the variances of the respective measured times.*

6. User Tests

The study was conducted with a total of 18 test subjects, which were all students of computer science to ensure a common knowledge of computer interaction. Since it was anticipated, that the concept of the repeatable group node might be confusing the test was performed with two test groups with 9 users each. The first group worked with the original version and the second group was provided with a slightly redesigned group node and received an additional tutorial regarding its iteration functionality.

Each test participant had to complete an introduction and two out of three test scenarios, which were assigned in random order. The test cases were chosen carefully to form a representative set of commonly used models and to verify the proposed general modeling approach. The test cases provide a model from the fields of architecture (skyscraper), fractals (Menger sponge) and organic structures (coral).

The user test was concluded by answering a short usability questionnaire. The statements used in this usability questionnaire represents a revised and shortened combination of the IsoMetrics^S [WHG97] and the Computer System Usability Questionnaire (CSUQ) [Lew95]. The questionnaire as well as the other materials were supplied in German to avoid possible misunderstandings and to reduce the cognitive impact on the users.

Performance Even though this test does not represent a comparative study, the assignments were timed to perform a qualitative estimate of anticipated modeling times. To distinguish between the performance for pure modeling and for the iterated application of rules two independent times were measured. The first time was taken when the user completed the basic build rule and the second time was taken when the whole model was finished. In Figure 6 the average times are plotted for each test scenario separated into group 1 that tested the reference system and group 2, which tested the improved version. Also the average times for the whole population are listed.

For the skyscraper test case the total completion time is consistent. The rule completion time varies slightly between the two groups. The higher average time to complete the building rule with respect to the other two cases is explainable by higher complexity of the model.
In the coral scenario both times are considerably lower for the second group. A possible explanation for this might be the fact that the second group gained additional modeling experience by completing the tutorial regarding the usage of the group node.
A strange result could be observed for the Menger case. While the time for the build rule was relatively consistent, the overall completion time increased notably for the second group. This surprising result implies that the additional tutorial regarding the group node worsened the actual usage of the group node instead of improving it. However, since also the first test group spent about 2/3 of the total time to complete the modeling of the recursion, it can be assumed that this specific test case is subject dependent and related to a lack of spatial sense (recursion passthrough trick).

Interview Feedback In addition to the questionnaire, the users also provides verbal feedback in the course of the test procedure, which revealed a list of minor and more severe usability problems.

It was pointed out that it would be more beneficial if the currently selected node would be previewed in the 3D viewport instead of always displaying the final model. A possible solution for this problem is to simply provide two separate 3D-previews, one for the final model and one for the currently selected node.

A related observation was that some users tried to navigate in the static preview of the selection editor, since the automatic camera positioning algorithm sometimes picks disadvantageous settings. For the required selection tasks this approach works reasonably well, however for larger amounts of selectable faces this approach is likely to be confusing and thus inefficient.

As already stated, the Menger test case turned out to be the most problematic scenario. Since this was already noted in an early stage of the test, the affected users were additionally interviewed about their problems with this task. While most users knew where the required faces where produced, they did not know how to correctly select them for further processing. Therefore, some users tried to model the recursion explicitly instead of connecting the first node directly to the

group node (see Figure 1). Such an edge seemed to take a shortcut in bypassing the extrude operator of the build rule and in turn violated the mental model about operation order.

A severe usability problem was the full evaluation of the nodes in a group, which complicated the selection process. Many users were forced to disable the iteration, change their selection and then reactivate the iteration. This work flow is obviously cumbersome, but can be easily improved by leaving the capsuled nodes in their basic state and provide external previews about the evolution of different iteration steps.

Interestingly, some users developed an inverse mental model of the GeoGraph, where the nodes were considered as state snapshots of the model and the edges as the corresponding transitions between these states.

Questionnaire Evaluation Despite some outliers, the evaluation of the questionnaire yielded overall very positive results. In Figure 7 the results of the evaluation are visualized as the mean with a confidence interval of 95%. Almost all of the positive statements are consistently oriented around 6 (agree) and similarly all negatively phrased statements lie around 2 (disagree).

The high variance of the variable *no severe errors* (Q_{17}) can be explained by the prototype status of the system, causing a small amount of system crashes. If a severe error occurred the user test was temporarily halted until the old system state was restored from the most recent backup of the GeoGraph.

The variance in variables like *available functionality* (Q_{06}), *long learn process* (Q_{21}) and *easy to learn* (Q_{23}) can be explained by too vaguely phrased statements, which were interpreted differently. However, the most essential usability measures like *inconsistent design* (Q_{15}), *simple error recovery* (Q_{18}) and *easy undo-able actions* (Q_{20}) exhibit almost no variance. These results prove that the system was well designed with respect to usability and that the measured user performance directly correlates with the proposed modeling approach for the editing of procedural geometry. Furthermore, the overall satisfaction with the system (Q_{01}) yielded a good result around 5.5, but since this is purely subjective measure it must be handled with care.

7. Conclusion and Future Work

We presented an approach for the intuitive and efficient generation of procedural models and the user tests yielded very good results and an overall positive feedback. With our system, previously untrained users were able to create complex models from scratch in a short amount of time. In contrast, explicit modeling techniques with modeling software as well as the creation of textual rules is cumbersome and requires a lot of effort and experience.

Two important research topics need further investigation. The first is the design of an optimal set of geometric operators, which are easy to understand and provide a complete procedural expressiveness in an intuitive way. The other is how to incorporate a level of detail (LOD) concept in the existing group node to allow for dynamic content refinement and the application of features on different scales.

References

[Ack82] ACKERMAN W. B.: Data flow languages. *IEEE Computer 15*, 2 (1982), 15–25. 2

[Coo84] COOK R. L.: Shade trees. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 223–231. 2

[DHL*98] DEUSSEN O., HANRAHAN P., LINTERMANN B., MĚCH R., PHARR M., PRUSINKIEWICZ P.: Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), pp. 275–286. 2

[GK07] GANSTER B., KLEIN R.: An integrated framework for procedural modeling. In *Spring Conference on Computer Graphics 2007 (SCCG 2007)* (2007), pp. 150–157. 2

[Han92] HANAN J. S.: *Parametric l-systems and their application to the modelling and visualization of plants*. PhD thesis, University of Regina, Canada, 1992. 2

[Hav05] HAVEMANN S.: *Generative Mesh Modeling*. PhD thesis, Technische Universität Braunschweig, Germany, 2005. 2

[JHM04] JOHNSTON W. M., HANNA J. R. P., MILLAR R. J.: Advances in dataflow programming languages. *ACM Comput. Surv. 36*, 1 (2004), 1–34. 2

[LD98] LINTERMANN B., DEUSSEN O.: A Modelling Method and User Interface for Creating Plants. *Computer Graphics Forum 17*, 1 (1998), 73–82. 2

[Lew95] LEWIS J. R.: IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction 7*, 1 (1995), 57–78. 6

[Lin68] LINDENMAYER A.: Mathematical models for cellular interaction in development: parts I and II. *Journal of Theoretical Biology 18* (1968), 280–315. 1

[LWW08] LIPP M., WONKA P., WIMMER M.: Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics 27*, 3 (2008), 102:1–10. Article No. 102. 2

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design galleries: a general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 389–400. 5

[Mai02] MAIERHOFER S.: *Rule-Based Mesh Growing and Generalized Subdivision Meshes*. PhD thesis, Vienna University of Technology, Austria, 2002. 2

[Mor94] MORRISON J. P.: *Flow-Based Programming: A New Approach to Application Development*. van Nostrand Reinhold, 1994. 2

[MWH*06] MÜLLER P., WONKA P., HAEGLER S., ULMER A., GOOL L. V.: Procedural modeling of buildings. *ACM Transactions on Graphics 25*, 3 (2006), 614–623. 2

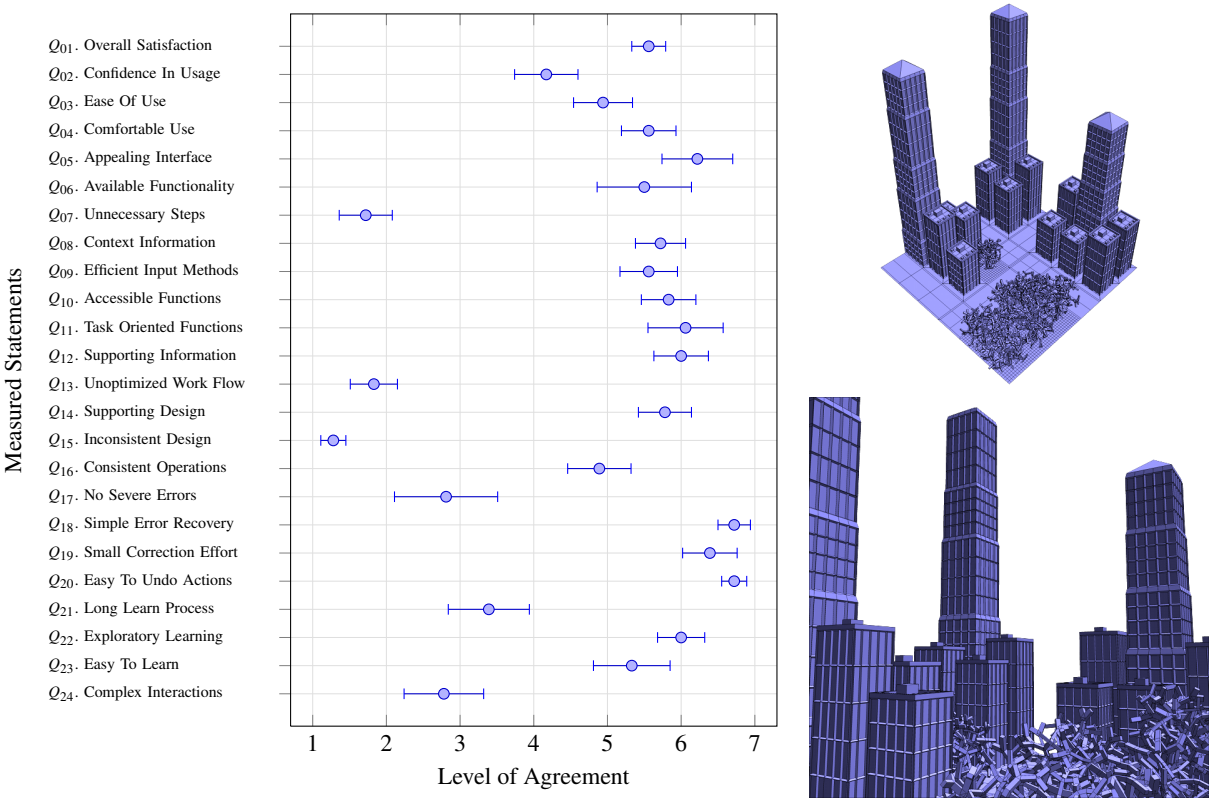

Figure 7: *On the left side, the questionnaire results are plotted as the mean with a confidence interval of* 95%. *The scale ranges from* 1 *(strongly disagree) to* 7 *(strongly agree). Also, the original statements from the questionnaire are abbreviated. On the right you can see two views of a more complex model of a city, which was created by an expert user in approximately 30 minutes.*

[Nie90] NIELSEN J.: Paper versus computer implementations as mockup scenarios for heuristic evaluation. In *INTERACT '90: Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction* (1990), pp. 315–320. 3

[Nie94] NIELSEN J.: *Heuristic evaluation*. John Wiley & Sons, Inc., New York, NY, USA, 1994, pp. 25–62. 3

[NM90] NIELSEN J., MOLICH R.: Heuristic evaluation of user interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems* (1990), pp. 249–256. 3

[PJM94] PRUSINKIEWICZ P., JAMES M., MĚCH R.: Synthetic topiary. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), pp. 351–358. 2

[PK96] PRUSINKIEWICZ P., KARI L.: Subapical bracketed l-systems. In *Selected papers from the 5th International Workshop on Graph Gramars and Their Application to Computer Science* (London, UK, 1996), pp. 550–564. 2

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990. 2

[PM01] PARISH Y. I. H., MÜLLER P.: Procedural modeling of cities. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 301–308. 2

[Smi84] SMITH A. R.: Plants, fractals, and formal languages. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 1–10. 1

[Sny03] SNYDER C.: *Paper Prototyping: the fast and easy way to design and refine user interfaces*. Morgan Kaufmann, 2003. 3

[Sti75] STINY G. N.: *Pictorial and formal aspects of shape and shape grammars and aesthetic systems.* University of California, Los Angeles, 1975. 2

[Sti80] STINY G. N.: Introduction to shape and shape grammars. *Environment and Planning B 7*, 3 (1980), 343–351. 2

[Str93] STRAUSS P. S.: IRIS Inventor, a 3D graphics toolkit. *SIGPLAN Notices 28*, 10 (1993), 192–200. 2

[TMW02] TOBLER R. F., MAIERHOFER S., WILKIE A.: Mesh-based parametrized l-systems and generalized subdivision for generating complex geometry. *International Journal of Shape Modeling 8*, 2 (2002), 173–191. 2

[WHG97] WILLUMEIT H., HAMBORG K. C., GEDIGA G.: *IsoMetrics^s: Questionnaire for the evaluation of graphical user interfaces based on ISO 9241/10 (short version)*, 1997. 6

[WWSR03] WONKA P., WIMMER M., SILLION F., RIBARSKY W.: Instant architecture. *ACM Transaction on Graphics 22*, 3 (July 2003), 669–677. Proceedings ACM SIGGRAPH 2003. 2