A generic shape grammar interpreter for discursive grammars. (paper)

Topic: Architecture

Author(s):
Rodrigo Correia
Portugal, Instituto Superior Técnico, Universidade de Lisboa
tecnico.ulisboa.pt
António Leitão
Portugal, INESC-ID/Instituto Superior Técnico, Universidade de Lisboa
tecnico.ulisboa.pt
José Pinto Duarte
USA, Penn State University
stuckeman.psu.edu

Abstract
Shape grammars are generative design tools based on rules that operate directly on shape rather than through symbolic computation [1]. Designs are created by recursively applying rules to an initial shape until a given goal is achieved or no more rules can be applied. Since several rules can be applied to a given shape, it is possible to produce many different designs. To automate rule application, researchers have developed shape grammar interpreters, but current approaches limit the production of three-dimensional shapes and provide little or no control over rule application. This restricts the range of designs that could be generated and, often, produces meaningless designs, that is, designs that do not match the current design context. In this article, we propose a general three-dimensional shape grammar interpreter and explain how shapes and rules are defined, and how the interpreter decides which rule to apply at a given time.
The interpreter developed is aimed at supporting designer's conceptual exploration, and permits post-processing of the computed design by bridging to different computer-aided-design (CAD) applications.
To validate our approach we present the implementations of several shape grammars including the one used in the design module of a specific shape grammar for mass-customized housing, called DESIGNA [2].

rodrigo.correia@tecnico.ulisboa.pt

Key words: shape grammar, mass customization, Malagueira

Main References:

[1] TW Knight, 'Shape Grammars in Education and Practice: History and Prospects', The Department of Architecture School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, 2000.
[2] JP Duarte, "Towards the Mass Customization of Housing: the grammar of Siza's houses at Malagueira", Environment and Planning B: Planning and Design, 32(3), pp. 347-380, 2005.
[3] J Gips, 'Computer Implementation of Shape Grammars', Workshop on Shape Computation, MIT, 1999.
[4] J Heisserman, 'Generative Geometric Design and Boundary Solid Grammars', doctoral dissertation, Carnegie Mellon University, Department of Architecture, 1991.
[5] J Lopes and A Leitão, 'Portable Generative Design for CAD Applications', Proceedings of the 31st annual conference of the Association for Computer Aided Design in Architecture, Banff, Alberta, Canada, pp. 196-203, 2011.

A generic shape grammar interpreter for discursive grammars

Rodrigo Correia[1], António Leitão[2], José P Duarte[3]
[1,2] IST – University of Lisbon, Portugal, [3] Penn State, USA
[1] rodrigo.correia@tecnico.ulisboa.pt, [2]
antonio.menezes.leitao@tecnico.ulisboa.pt, [3] jxp400@psu.edu

Abstract
Shape grammars are generative design tools based on rules that operate directly on shape rather than through symbolic computation [1]. Designs are created by recursively applying rules to an initial shape until a given goal is achieved or no more rules can be applied. Since several rules can be applied to a given shape, it is possible to produce many different designs. To automate rule application, researchers have developed shape grammar interpreters, but current approaches limit the production of three-dimensional shapes and provide little or no control over rule application. This restricts the range of designs that could be generated and, often, produces meaningless designs, that is, designs that do not match the current design context. In this article, we propose a general three-dimensional shape grammar interpreter and explain how shapes and rules are defined, and how the interpreter decides which rule to apply at a given time.
The interpreter developed is aimed at supporting designer's conceptual exploration, and permits post-processing of the computed design by bridging to different computer-aided-design (CAD) applications.
To validate our approach we present the implementations of several shape grammars including the one used in the design module of a specific shape grammar for mass-customized housing, called DESIGNA [2].

Introduction
Initially developed by Stiny and Gips [3], shape grammars allow capturing, creating, and understanding designs. They are generative systems based on rules that work directly with shape computations rather than with symbolic computations. Shapes are conceived as a finite collection of maximal lines [4], and a design solution is created by the application of rules to an initial shape until no more rules can be applied or a solution is found. Because many rules can be applied to any given shape, different solutions can be achieved. To help reduce symmetries in shapes, and to control or limit which rules are applied, labels were introduced. Labels can be added or removed during computation. With parametric shape grammars [4], which allows a shape to be defined in terms of parameters, the range of designs becomes larger. This increased flexibility also entails a more complex implementation because the number of generated design solutions produced can be very large, if not infinite. To fully explore the range of possible design solutions generated by a shape grammar, interpreters have been proposed and implemented [5][6]. These interpreters allow to automate the application of shape rules but still suffer from several problems, ranging from limited shape representations to lack of control of rule applications. Moreover, typically, these implementations do not interface with the traditional tools used by the designer, limiting the ability to continue developing the computed designs.
To overcome some of these limitations, we propose a shape grammar interpreter that 1) is able to represent labeled shapes in two or three-dimensions, 2) permits the definition of rules and the control of its application to shapes, and 3) integrates seamlessly with the designer's workflow by interfacing to his tool of election.

Shape Grammar Interpreter

To represent shapes, we use a graph-based boundary representation introduced by Heisserman [7]. As in Heisserman, a shape is just set of vertices, edges, and facets with respective incident relations between them. This representation is encoded in a graph where the nodes represent the shape elements, and its edges represents the incident relations, or the topology of a shape (e.g. the edges represents the connectedness and adjacencies of facets through vertices, and so on). The geometric information is directly associated to vertices and labels, which encode non-geometric information such as numbers, strings or others and can be associated with any shape elements, such as points, lines, facets, and solids. Labels can also be used to restrict rule application by imposing conditions on the shape generation. While this is a more complex representation of a shape, it also provides for a data structure that allows for queries to be done quickly and inexpensively, e.g. “what are the adjacent facets of an edge?,” “what is the angle formed by two edges?,” or “What is the area of a facet?” Therefore, a shape can be seen as a stack of layers where 1) the lowest layer contains the shape elements and their relations, representing the topology of a shape, 2) the next layer contains three-dimensional points, representing the geometric information, and 3) the last layer contains the labels, representing non-geometric information. Shapes are constructed with Euler operators, which ensure valid topological shapes, for example avoiding the creation of non-manifold shapes like a facet with a hanging edge. Shapes are implemented using half-edge data structure provided by CGAL[2], a library for efficient and reliable computational geometry algorithms and data structures used in academia and the industry. The CGAL library also provides for several numeric and geometric features allowing different degrees of precision and speed. We use these features, making exact queries in order to avoid common problems associated with rounding errors. While this entails some tradeoff in performance, it ensures that geometric tests are always correct, e.g. when querying if a point lies on a line.

As shape grammars are rule-based systems, the representation of rules and their application to shapes becomes central to the generation of solutions. However, there are two important issues to consider: (1) shape grammar rules tend to condense too much information in one rule, and (2) manual application of rules is impractical when the exploration of a design requires many rule applications. To overcome these problems, we propose a novel approach for the representation and application of rules. Traditionally, shape grammar rules have an antecedent, which describes the design to which the operator applies, and a consequent, which describes the design that results from the application of the rule. In our representation, each rule is an operator that also has an antecedent but, instead of having a single consequent, the operator will have as many consequents as needed to describe the designs that may emerge from rule application. Besides this representation, we also provide strategies that dictate which resulting designs would be taken into account when applying the subsequent rule. For example, we can apply a rule to all the generated designs, or just explore the application of rules to the first design generated.

In summary, our shape grammar interpreter can be seen as an application built on different levels. In the bottom level there is the shape representation implemented using CGAL data structures and its exact geometric and numeric kernels. In the next level we have the core of the interpreter which implements the rule system and the strategies to apply rules. This level uses the previous level to create, modify, and query shapes. Finally,

[2] http://www.cgal.org/

the last level uses the previous two levels and communicates with standard CAD tools. All these levels are implemented and managed by Rosetta [8]. Rosetta abstracts the use of CGAL with a wrapper, manages the creation of shapes and the definition of rules, and controls the application of rules to derive new shapes. Finally, it supports the communication with different CAD tools so that the generated designs can be visualized in these tools and further manipulated by the designer.

Results

To evaluate the shape grammar interpreter we implemented three shape grammars: 1) an ice-ray grammar, 2) a three-dimensional grammar, and 3) a specific shape grammar for mass-customized housing.

Ice-ray grammar

The ice-ray grammar was first formalized by Stiny [9] to describe the design of Chinese lattices. Using a simple set of rules, Stiny was able to generate typical Chinese ornamental windows. In Figure 1, we show the simplified rules (e.g. without the labels) for the ice-ray grammar as originally defined by Stiny.

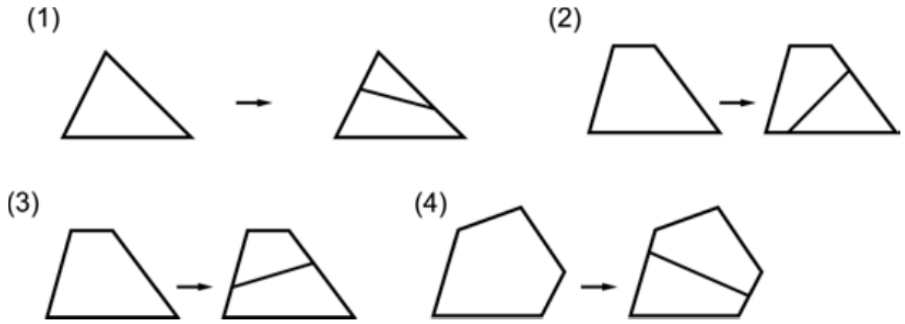


Figure 1: Ice-ray shape grammar rules by Stiny (simplified version).

This grammar manipulates two-dimensional shapes like triangles, quadrilaterals. and pentagons. Therefore, considering that the context to which these rules apply is limited to triangles and convex quadrilaterals and pentagons, the antecedent of the rules simply identifies facets with the required number of vertices, three for triangles, four for quadrilaterals and five for pentagons. The consequent of the rules specifies the placement of new lines, which is done by using the Euler operator for splitting a facet in two.

In Figure 2 is shown the steps taken by our shape grammar interpreter to generate a final design using the ice-ray shape grammar. In this case, rules applications was defined manually, including, inputting values to indicate where new line segment were to be introduced. This Figure illustrates how a user can generate a design and how maximal line representation is handled, e.g. when detecting the facet F, the right edge is defined by the line segments *a* and *b*.

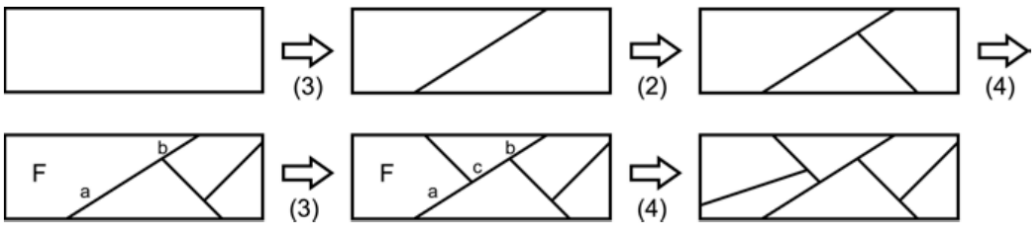


Figure 2: Derivation of a design according to the ice-ray grammar

Examples of additional design solutions following the ice-ray grammar are shown in Figure 3. Note that this figure only shows a small portion of more than two hundred designs that were generated. To give an idea of the potentially very high number of solutions that a simple shape grammar like the ice-ray grammar can generate, in Figure 4 we show all the

258 possible designs from which the designs in Figure 3 were extracted. Note also that, while the examples shown are rectangles, the interpreter is able to apply “ice-ray rules” to any shape with an arbitrary number of edges, thereby extending the interpretation of the rules defined by Stiny. Such an example is shown in Figure 5.

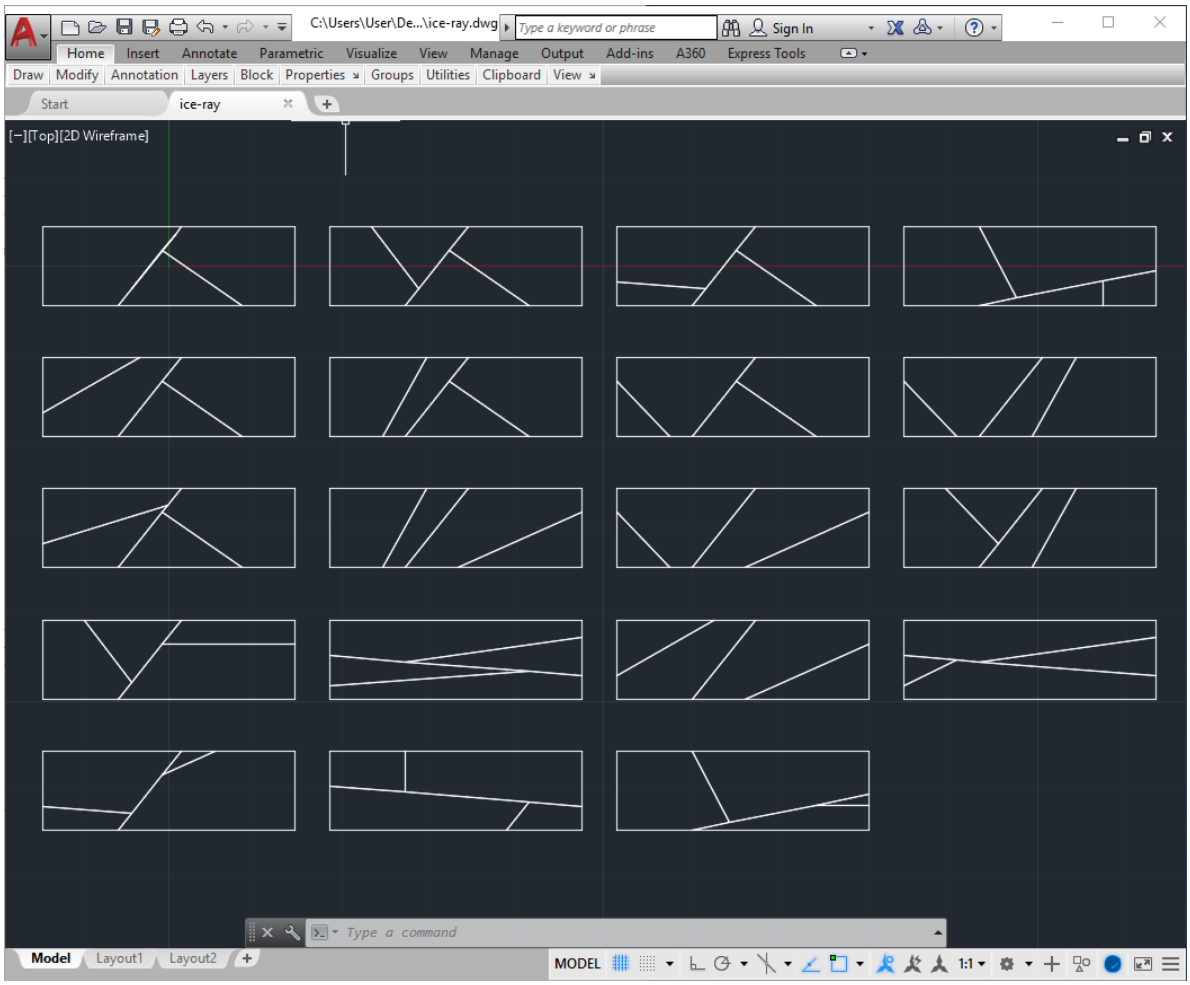


Figure 3: Different design solutions using the ice-ray grammar (filtered example).

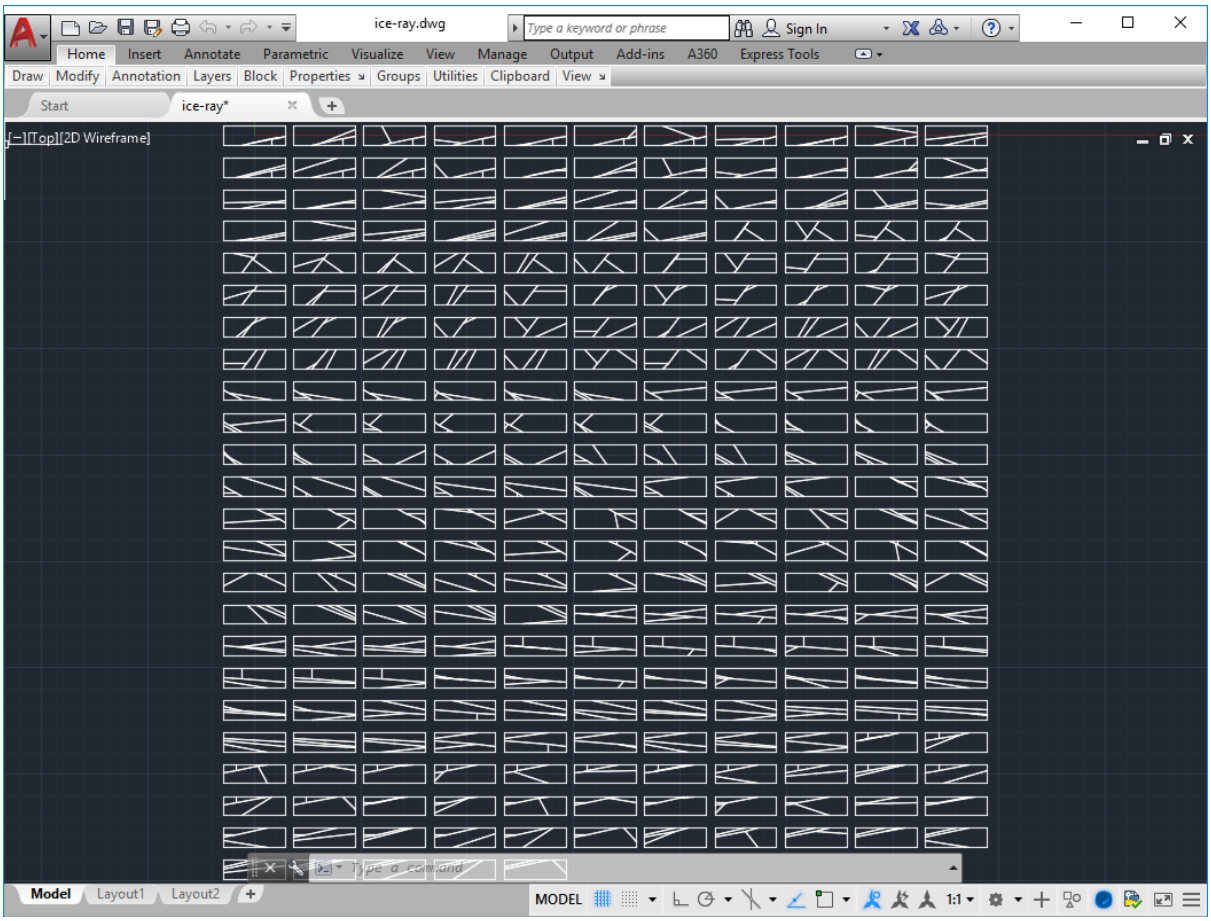


Figure 4: Different design solutions using the ice-ray grammar.

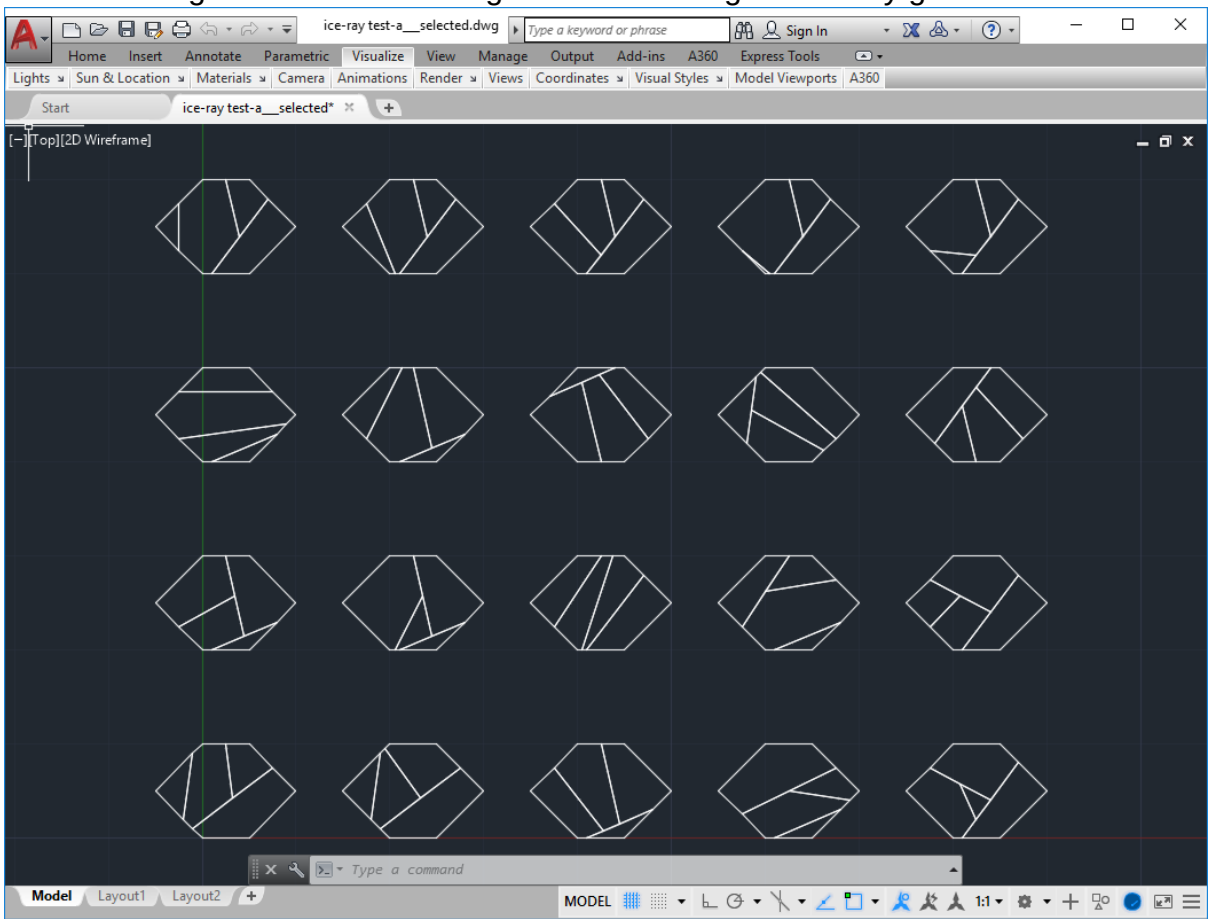


Figure 5: Design solutions generated by an extended ice-ray grammar.

Three-dimensional grammar

The second shape grammar implemented fulfilled two objectives: 1) show that our approach to shape representation could handle both two-dimensional shapes and three-dimensional shapes, and 2) show how flexible rule application strategies could be and the consequences of using different strategies for the generation of designs. To define this grammar we followed a simple idea: taking any shape facet, grab its center point and pull it out in the direction of the normal of the facet to a certain distance *C*, generating as many new facets as the number of edges of the original facet, all diverging from the center of the facet. This grammars has only one rule, the designs are 3D objects, and its facets form the vocabulary of shapes in the grammar. In practical terms, the rule determines that any facet shape found in the design is transformed into a pyramid with the apex at some perpendicular distance from the centroid of the shape.

Figures 6 and 7 show some resulting designs using two different initial shapes, a tetrahedron and a cube, and different values of the *C* parameter. Note that only one rule and different parameters was applied following the same rule application strategy.

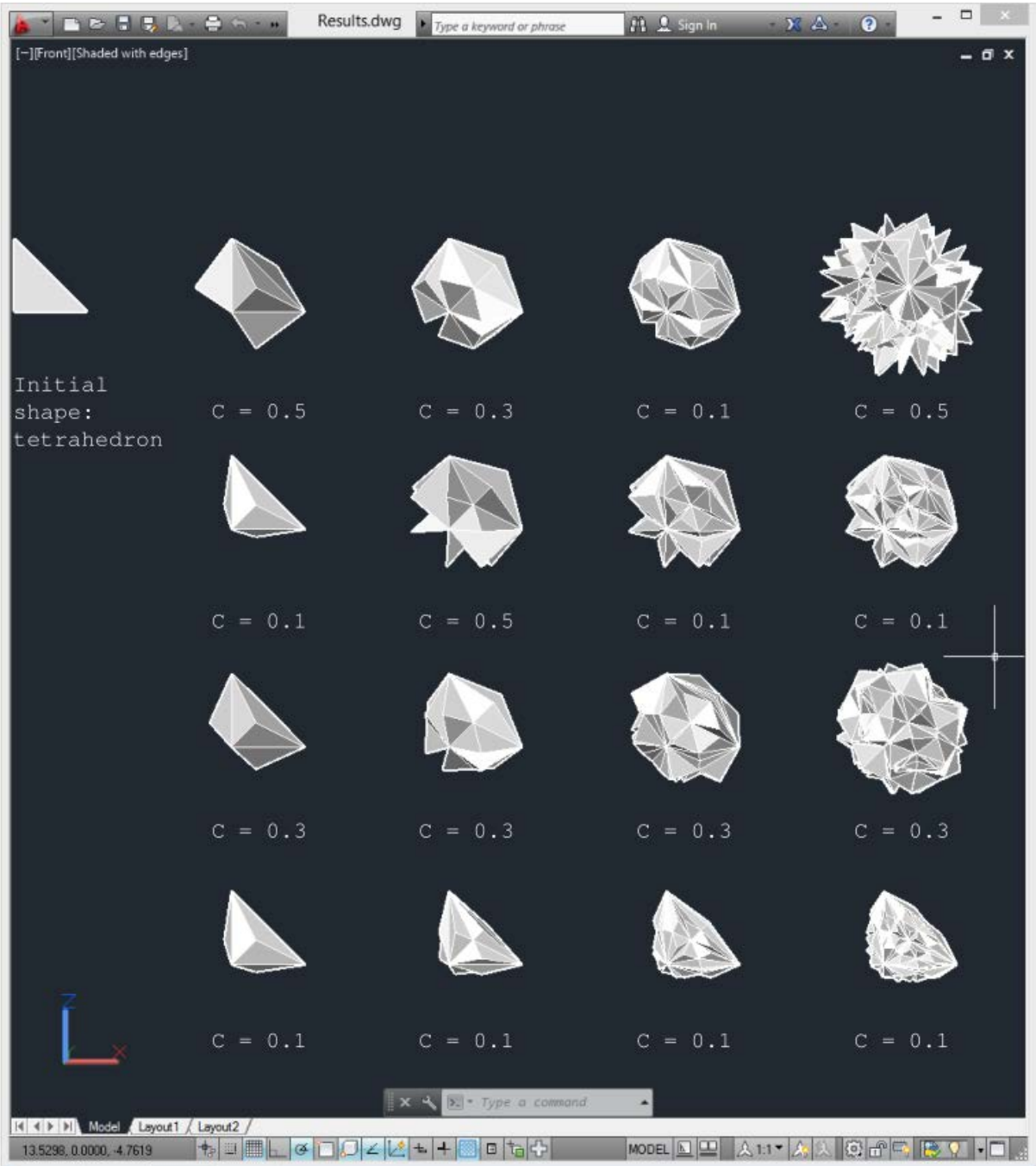


Figure 6: Design solutions from the three-dimensional grammar (initial shape: a tetrahedron)

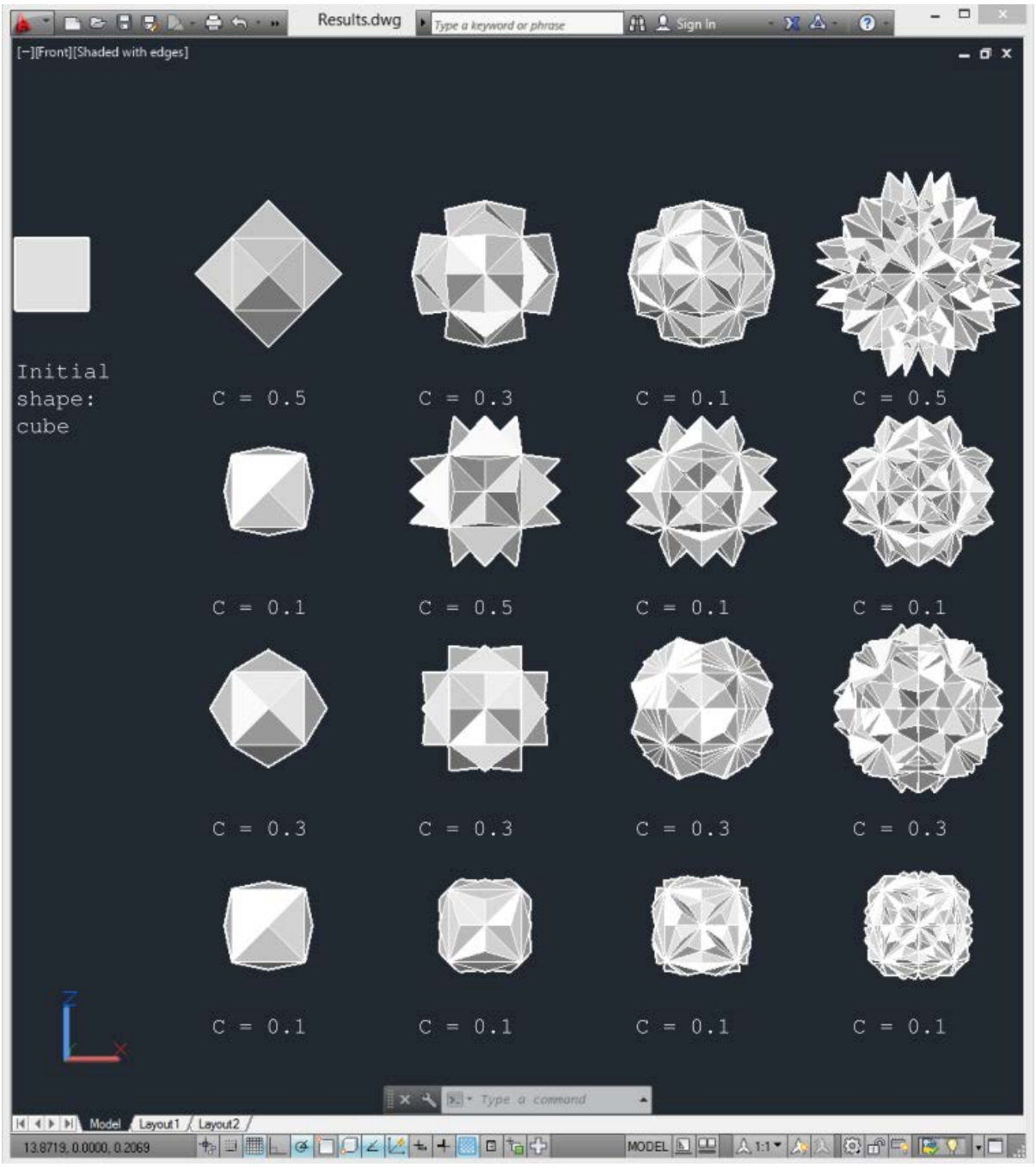


Figure 7: Design solutions from the three-dimensional grammar (initial shape: a cube)

To illustrate the outcome of the use of combined strategies in rule application, Figure 8 shows an example of a combined strategy. In this figure it is shown a cube where the initial shape was transformed using one strategy and, thereafter, a different strategy. More concretely, we first used a strategy that applied the grammar rule to all of the six initial facets of the cube and then we switched to another strategy that applied the grammar rule to the last facet generated.

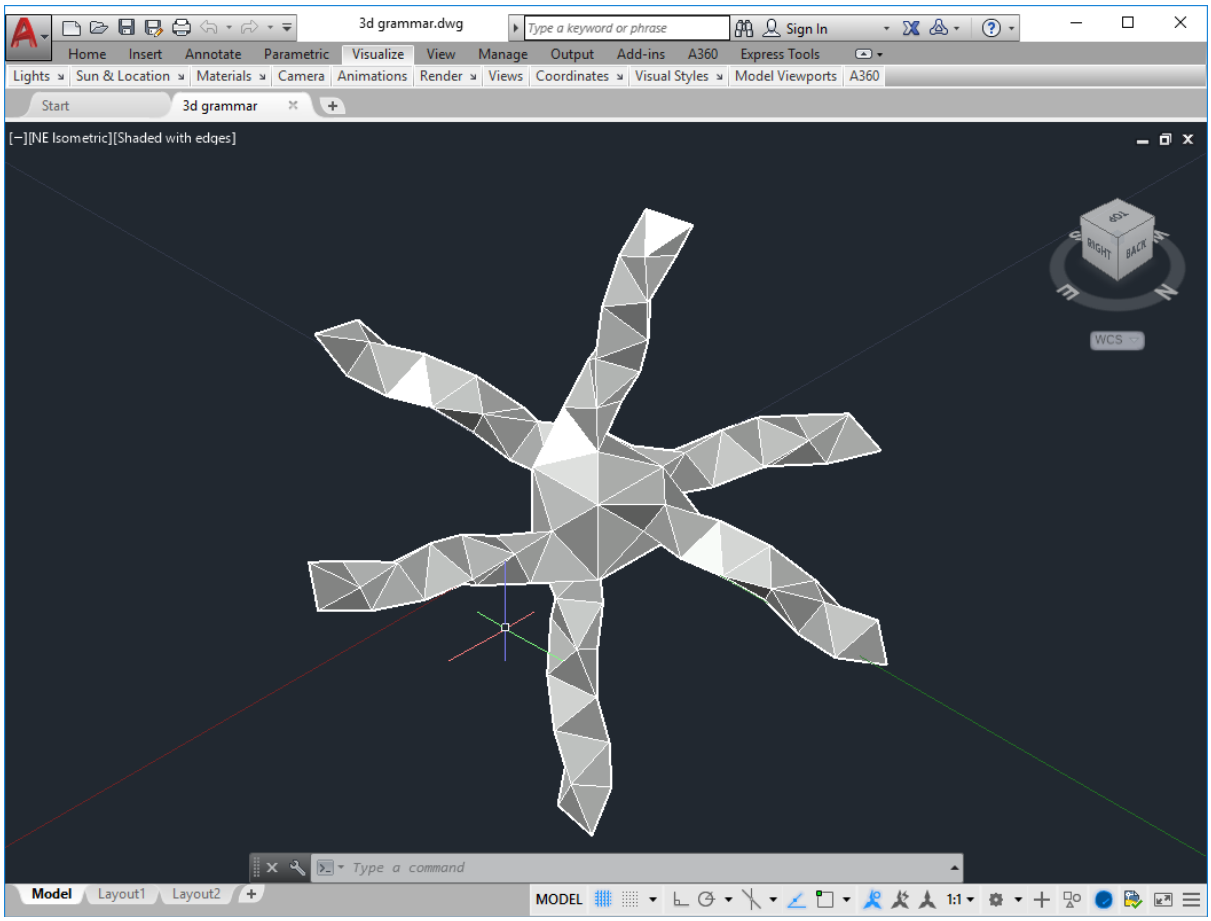
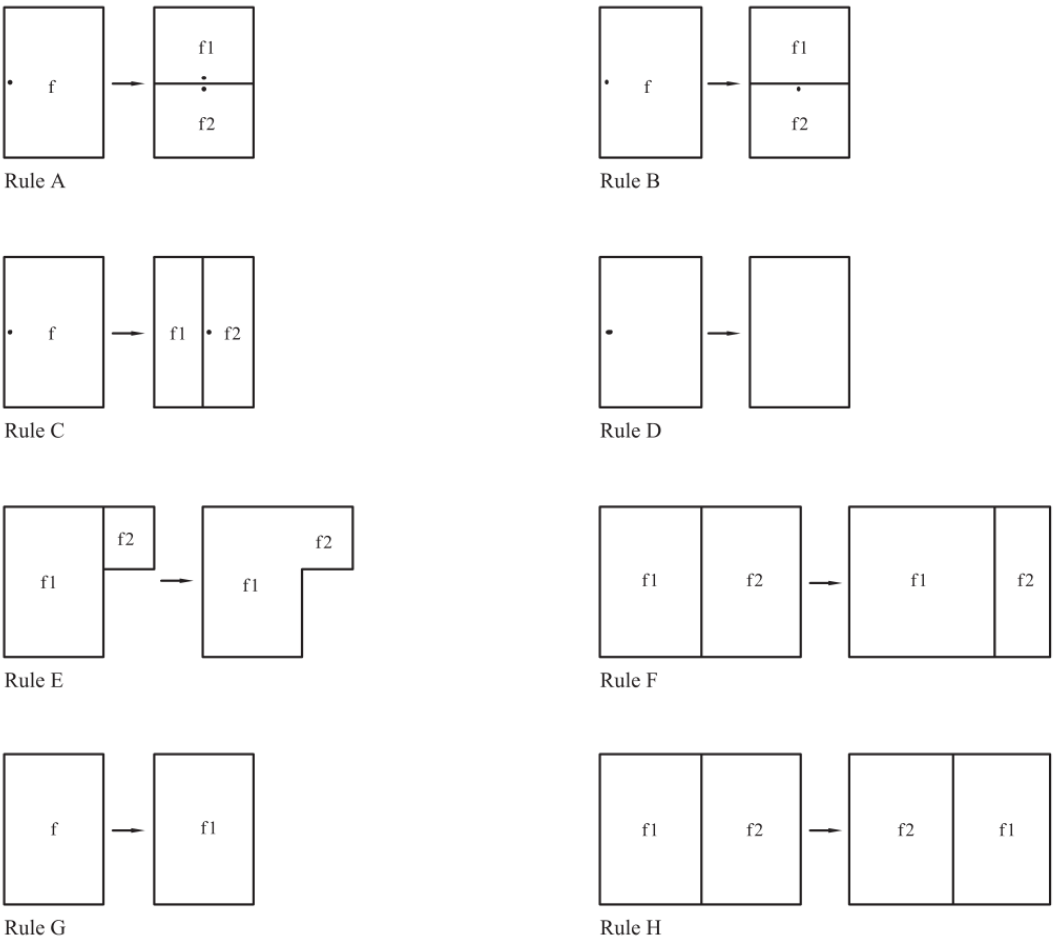


Figure 8:Combination of rule application strategies to a cube.

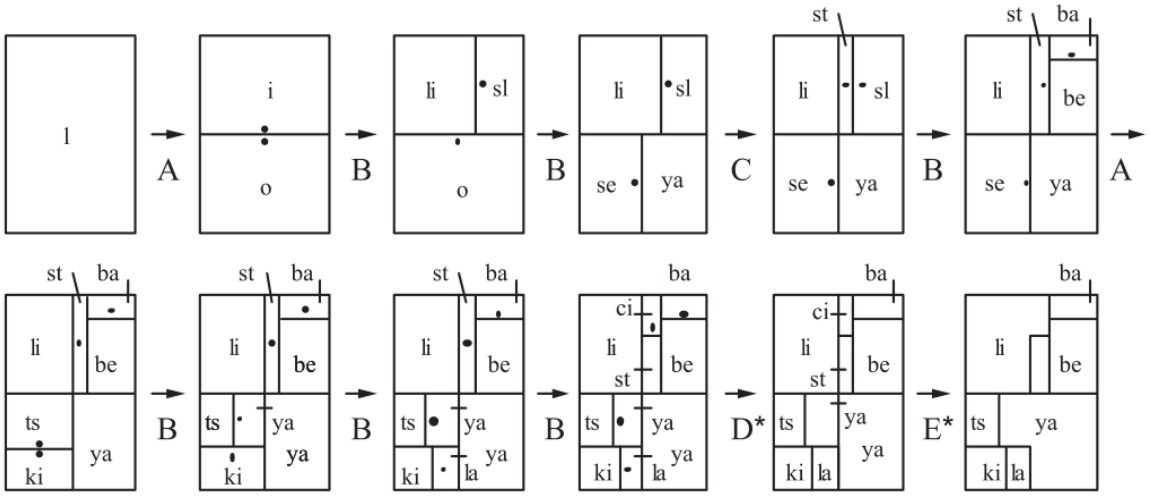
Mass-customization of housing grammar

The final shape grammar implemented was a specific grammar for mass-customization of housing, called DESIGNA [2]. This shape grammar is the designing grammar used for experimentation and evaluation of a discursive grammar [2]. The shape grammar implemented is a parametric shape grammar for the generation of housing solutions that match given design briefs. This particular shape grammar encodes the rules laid out by the Architect Álvaro Siza for the design of the Malagueira houses, in Évora, Portugal.

To better understand the complexity of this grammar, the representation used, and the solutions generated, an example with several oversimplified rules and a partial derivation of an existing layout of a house is shown in Figure 9. The set of rules include rules for: 1) dissection [Rules A, B, C], 2) connecting [E], and 3) extending [F] rectangles, 4) delete labels [D], 5) assigning a function [G], and 6) permuting functions [H]. Note that one of the labels in these rules, the • symbol, marks where the next dissection may occur. Therefore, Rules A, B, and C, dissect a space in two, but while with Rule A we can continue dissecting the two resulting spaces, with Rules B and C we can only dissect one of the two spaces. The derivation shown in the figure starts with the initial lot, the shape with the label *I*, which is dissected in two using Rule A, creating the inside and the outside zones, with labels *i* and *o*, respectively. Then, after applying Rule B, the inside is dissected in the living (*li*) and the sleeping (*sl*) functional zones. This process continues until a layout is found or no more rules can be applied.



(a) Rules of the simplified Malagueira grammar implemented



(b) Partial derivation of an existing layout of a house

Figure 9: Simplified Malagueira grammar (a) and derivation of a house following the grammar (b)[3].

A design solution in this grammar is a two-dimensional layout representing the floor plan of a house. This mimics the way of an Architect could start the design of a house. A design is, thus, a rectangle, where shapes are facets, representing the spaces of a house.

[3] The * represents that the same rule was applied several times. The meaning of labels in this figure is as follows: *l* lot, *i* inside zone, *o* outside zone, *li* living zone, *sl* sleeping zone, *se* service zone, *ya* yard zone, *be* bedroom, *ba* bathroom, *k*i kitchen, *ts* transitional space, *la* laundry, *pa* pantry, *ci* circulation, *st* stairs.

Because the underlying shape representation in our interpreter is a graph, encoding both the topology and geometric information of a shape, we do not need yet another data structure to represent the topology of the house. For example, if a facet with the label *kitchen* is topologically connected to facets with the labels *living-room* and *yard*, then we know that the space *kitchen* is related with the space *living-room* and the space *yard*. This kind of information is important to avoid creating a bedroom near a kitchen or similar scenarios. Rules in the grammar are based on the dissection of rectangles, which is defined by a set of *parameters* whose values of these *parameters* are restricted to avoid the generation of housing spaces that do not comply with existing housing regulations (PAHPA), e.g. a space narrower than 2.20 m. Figure 10 shows some designs obtained by applying the rules of the grammar with the interpreter.

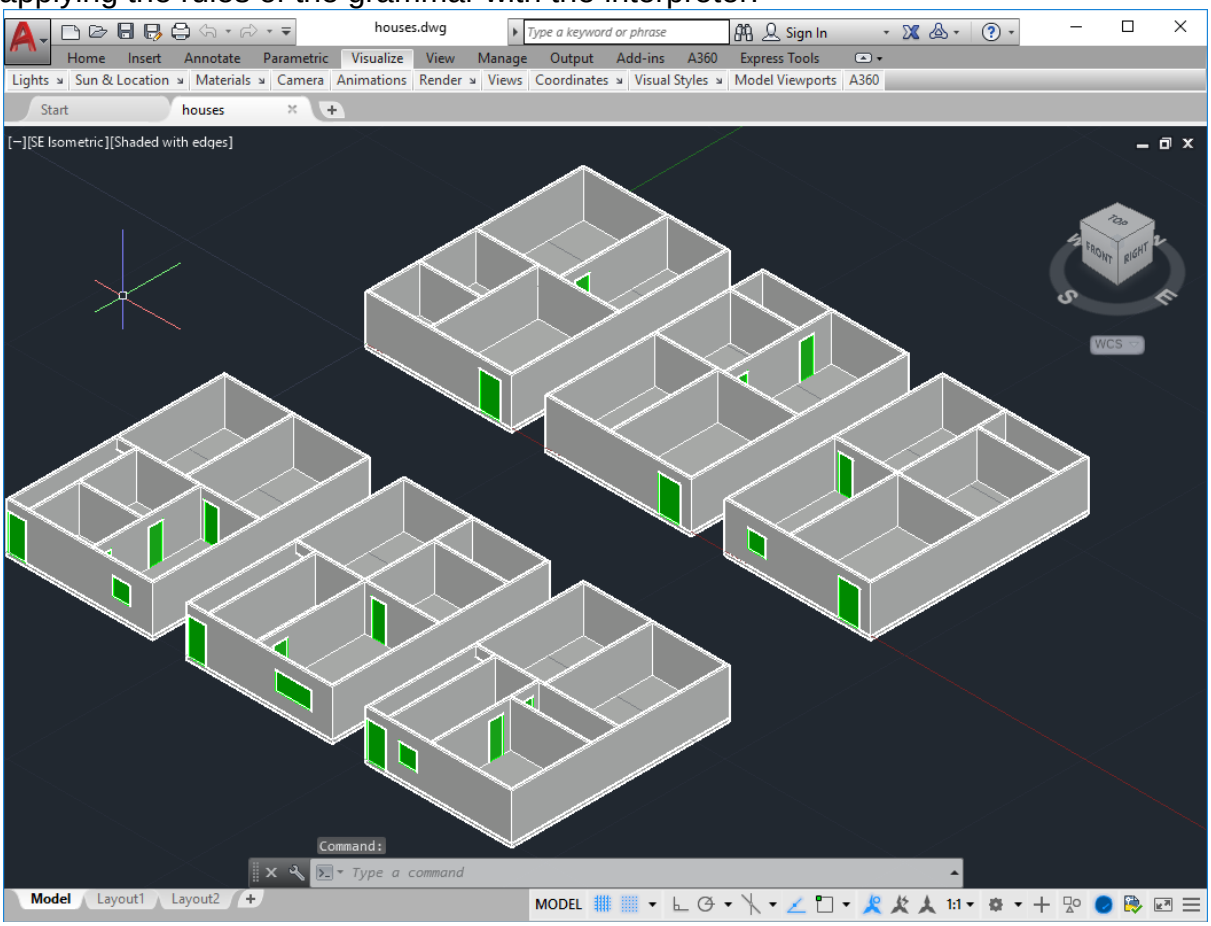


Figure 10: Design solutions after the Malagueira housing shape grammar.

Conclusions

We described the implementation of a new shape grammar interpreter that goes beyond the capabilities of existing interpreters as it is generic (supports the development and implementation of different grammars); it uses embedding (sub-shape recognition); it works with both 2D and 3D shapes; it supports parametric shape rules and conditional statements; it includes labels and, to a limited extent, descriptions; it permits different rule application strategies; and it integrates seamlessly with different CAD applications. Previous interpreters included some but not all of these features. The features of the interpreter permit to implement in a certain discursive grammars, that is, grammars that generate designs matching given criteria.

We discussed the representation of two- and three-dimensional shapes, the implementation of shape grammar rules, and the use of different rule application strategies. We also showed how our implementation can integrate with common CAD tools, allowing the designer to develop further the generated solutions. We believe that these features result in a versatile tool that a designer can integrated in his or her workflow, without changing it, but augmenting it. Moreover, with the interpreter the designer can generate a diverse range of complex shapes difficult to obtain using manual processes.

Because of the explosive number of solutions that a parametric shape grammar can generate, it was necessary to introduce a pruning mechanism in the implementation. This allows to greatly reduce the number of solutions, as some wouldn't even be appropriate for consideration by a human user. Examples of this situation are the ice-ray grammar where designs could contain exceedingly small shapes, and the Malagueira grammar, where unrestricted rules application could lead to the generation of rooms that were too small.

Finally, we believe that by providing the all the basic features listed above, including the ability to integrate seamlessly with different CAD tools, our interpreter promotes the use and portability of shape grammars across different CAD applications. The next logical steps for future work would be to expand the ability to work with descriptions and introduce weights, thereby increasing the capability to implement discursive grammars.

References

[1] Knight, TW 2000, 'Shape Grammars in Education and Practice: History and Prospects', The Department of Architecture School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA.
[2] Duarte, JP 2005, 'Towards the Mass Customization of Housing: the grammar of Siza's houses at Malagueira', Environment and Planning B: Planning and Design, 32(3), pp. 347-380.
[3] Stiny, G., & Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. C V Freiman (ed.) Information Processing 71 (Amsterdam: North-Holland), (pp. 1460-1465).
[4] Stiny, G 1980, 'Introduction to shape and shape grammars', Environment and Planning B 7(3), pp. 343-351.
[5] Gips, J 1999, 'Computer Implementation of Shape Grammars', Workshop on Shape Computation, MIT.
[6] Chau, H, Chen, X, McKay A, de Pennington, A 2004, 'Evaluation of a 3D shape grammar implementation', Proceedings of the First International Conference on Design Computing and Cognition, pp. 357-376.
[7] Heisserman, J 1991, 'Generative Geometric Design and Boundary Solid Grammars', doctoral dissertation, Carnegie Mellon University, Department of Architecture, Pittsburgh.
[8] Lopes, J and Leitão, A 2011, 'Portable Generative Design for CAD Applications', Proceedings of the 31st annual conference of the Association for Computer Aided Design in Architecture, Banff, Alberta, Canada, pp. 196-203.
[9] Stiny, G. (1977). Ice-ray: a note on the generation of Chinese lattice designs. Environment and Planning B , 4, 89-98.