Implementing the Santa Marta Urban Grammar

a pedagogical tool for design computing in architecture

Eduardo Castro e Costa[1], Debora Verniz[2], Siavash Varasteh[3],
Marc Miller[4], José Duarte[5]
[1,2,3,4,5]The Pennsylvania State University
[1,2,3,4,5]{erc15|dvp6|sxv205|mlm693|jxp400}@psu.edu

We present a tool intended to enable non-expert users to apply and manipulate a shape grammar, SMUG, which encodes the urban design of informal settlements such as favelas. Such tool, the Interpreter, was developed considering that students would be its main users, and therefore we consider this grammar implementation to potentially be a multipurpose pedagogical tool since it supports conveying knowledge about urban design, shape grammars and parametric modeling using Grasshopper. This paper focuses on the development of the Interpreter and discusses the results of its use in a design studio, which can better inform subsequent iteration as well as other courses and schools.

Keywords: *Shape grammars, Urban design, Design studio, Parametric modelling,*

CONTEXT

Globally, cities are experiencing significant growth in population coupled with fast urbanization processes (Davis, 2006) - most notably in the global south. These trends of migration towards cities exert enormous pressure on cities giving rise to numerous social, infrastructural, ecological, and urban problems (United Nations, 2015). The formal sector is unable or unwilling to provide enough appropriate dwellings to this growing population, which then resorts to self-build processes, giving rise to informal settlements (Maricato, 2001). These settlements are spatially rich and adjusted to the site, but lack infrastructure and public spaces.

This paper is part of a larger research that explores the use of digital technology to overcome such limitations and promote social integration. Technology is used in two ways: first; to map, analyze, and explain the genesis and morphology of informal settlements. Second; to devise alternative planning and design methodologies to generate planned settlements with the qualities found in informal settlements while avoiding its shortcomings.

The research is supported by a design studio where students are asked to design proposals for planned settlements. The studio aims to take a proactive approach in designing infrastructure systems for settlements or social housing. This presents a host of problems and questions for the participants in the design studio that reflect the complex relationship between residents in informal housing settlements and conventional planning structures that put these peoples and their homes at risk along with associated conflicts with social and political structure in the "formal city." As a design tool and studio, emphasis is being placed on understanding the spatial complexities of informal settlements using shape grammars.

The design studio context comprises five main tasks: (1) To understand the morphology of a case study, favela Santa Marta in Rio de Janeiro; (2) To infer a shape grammar that explains the generation of the case study's morphology; (3) To develop a computer program that implements the shape grammar; (4) To discuss what are the qualities and shortcomings of the informal settlement used as a case study; (5) To manipulate the design parameters in the computer program to generate the desired planned settlements. Steps 1, 2, and 3 are related with the preparation of the studio, while steps 4 and 5 are related with the work done with the students in class.

This paper is focused on the development of step 3, presenting the methodology to implement the Santa Marta Urban Grammar (SMUG), which has been previously published (Verniz & Duarte, 2017).

RELATED WORK

Computer implementation of shape grammars has naturally followed the development of shape grammar theory. While at a certain point the implementation of shape grammar interpreters was disappointing to the shape grammar community (Knight, 2000; Chase, 2002; A. I.-K. Li, 2005), currently, we can find a number of tools that can be considered proper shape grammar interpreters (Stouffs, 2016; Grasl & Economou, 2018; A. I. -kang Li, 2018; Ligler & Economou, 2019). In common, these tools enable users to design shape grammar rules and apply them to designs, waiving the need for coding skills. However, these interpreters are constrained to compute linear geometries, rendering them unable to deal with meshes or surfaces, which becomes a limitation when working with three-dimensional models.

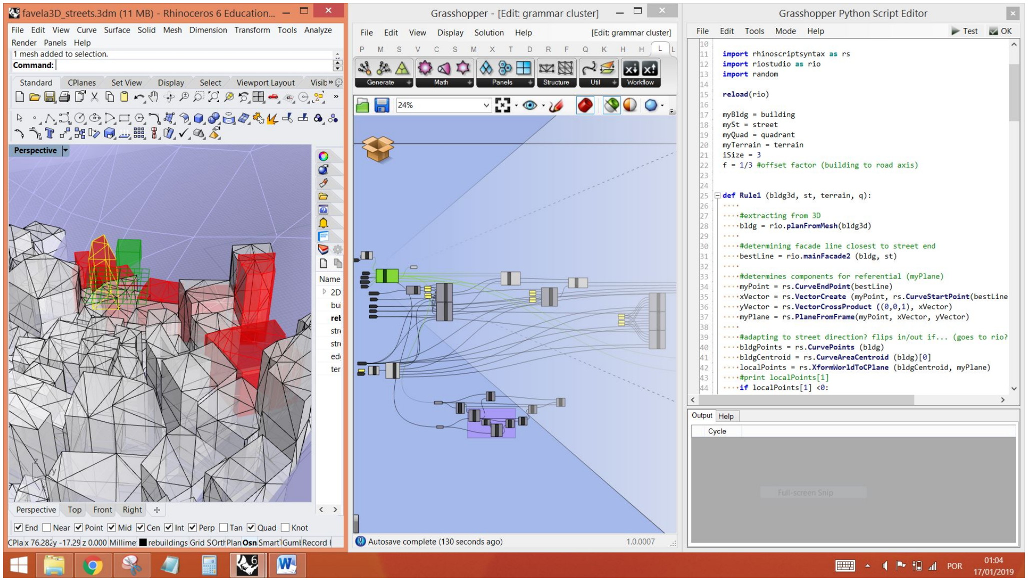
Such tools were considered inappropriate for implementing the Santa Marta Urban Grammar, not so much due to their limitations on computing meshes, but because there is a fair amount of implicit information in the SMUG rules. For example, in rules responsible for positioning a new building based on an existing one, such buildings are represented by their corresponding plans. However, the implementation of such rules will operate on a 3D model of the favela, implying some interpretation of the rule on the part of the interpreter's developers.

In fact, such implicitness is an important aspect of this experiment, as will be discussed along this paper. The implementation process helped reducing it, and thus making the grammar less ambiguous. However, as analytical methods for design research, it is important that shape grammars are allowed to have some implicitness and flexibility, since eliminating it completely to the point where it can be implemented using the aforementioned interpreters can become cumbersome, and typically falls beyond the scope of the research it serves.

In order to accommodate for such flexibility, it is justifiable to produce custom implementations of particular grammars. We find a similar approach in a number of studies applying shape grammars to urban design (Duarte, Rocha, & Soares, 2007; Beirão, Duarte, & Stouffs, 2011; Mendes, Celani, & Beirão, 2014). In these studies, a custom implementation of the grammar allowed researchers to explore and improve the shape grammar in their own terms. The main difference from these previous approaches is that the SMUG implementation was designed towards an easy process of performing transformations to the original grammar (Knight, 1994) in order to better describe other favelas or to explore better ways of developing informal settlements.

THE INTERPRETER

The main objective of implementing the grammar is for students to use it for developing urban design solutions, corresponding to 3d models developed in in Rhinoceros . The grammar was converted into a parametric model whose definition comprises both Grasshopper (GH) components and GHPython scripts (Figure 1). The resulting parametric model was dubbed "the Interpreter", although this term is used loosely. In fact, the Interpreter is not consid-



favela3D_streets.3dm (11 MB) - Rhinoceros 6 Education...
Grasshopper - [Edit: grammar cluster]
Grasshopper Python Script Editor
File
Edit
Tools
Mode
Help
Test
OK
Perspective
Top
Front
Right
Output
Help
Cycle
Autosave complete (130 seconds ago)

```
import rhinoscriptsyntax as rs
import riostudio as rio
import random

reload(rio)

myBldg = building
mySt = street
myQuad = quadrant
myTerrain = terrain
iSize = 3
f = 1/3 #offset factor (building to road axis)


def Rule1 (bldg3d, st, terrain, q):
    
    #extracting from 3D
    bldg = rio.planFromMesh(bldg3d)
    
    #determining facade line closest to street end
    bestLine = rio.mainFacade2 (bldg, st)
    
    #determines components for referential (myPlane)
    myPoint = rs.CurveEndPoint(bestLine)
    xVector = rs.VectorCreate (myPoint, rs.CurveStartPoint(bestLine
    yVector = rs.VectorCrossProduct ((0,0,1), xVector)
    myPlane = rs.PlaneFromFrame(myPoint, xVector, yVector)
    
    #adapting to street direction? flips in/out if... (goes to rio?
    bldgPoints = rs.CurvePoints (bldg)
    bldgCentroid = rs.CurveAreaCentroid (bldg)[0]
    localPoints = rs.XformWorldToCPlane (bldgCentroid, myPlane)
    #print localPoints[1]
    if localPoints[1] <0:
```

Figure 1
The SMUG Interpreter is built on Rhino, Grasshopper and GHPython

ered a proper implementation of a shape grammar since it does not perform automatic shape recognition (Chau, Chen, McKay, & Pennington, 2004). Nevertheless, converting the grammar into a parametric model allows extending its accessibility to wider audiences that are non-specialists in shape grammars (Barros, 2015), such as students. Consequently, the presented implementation of SMUG consisted in translating the Grammar's rules, in an effort to replicate such rules with the greatest possible fidelity to the grammar, following an approach used in previous similar exercises (Figueiredo, Castro e Costa, Duarte, & Krüger, 2013; Castro e Costa, Jorge, & Duarte, 2019).

Moreover, since SMUG adopts a bottom-up approach for modeling the growth of a favela, the corresponding GH definition models the behavior of a cellular automaton (CA). According to Clarke (2014, p. 1220):

"a CA has four elements: (i) a grid of cells, each of which can assume a finite number of states; (ii) a neighborhood, over which a change operator applies, usually the Moore (8-cell) neighborhood surrounding a cell in the grid; (iii) a set of initial conditions, that is, an instance of the states for each and every cell in the system; and (iv) one or more rules, which when applied change the state of a cell based on properties or states of the neighborhood cells."

Although the Santa Marta favela does not correspond to a regular grid of cells, we can consider that each cell corresponds to a building. Each building can be surrounded by other neighboring buildings. Each new building is generated from a previously existing building, which is inherently its neighbor, as well as taking into account the remaining neighbors. Moreover, the new building is generated according to a set of rules that are common to all the buildings - namely, SMUG.

Therefore, the Interpreter defines the behavior of the automaton. It operates on one building at a time. A newly generated building can be subsequently used for the next building, featuring a recursive behavior that can be implemented for example using the Hoopsnake plugin for GH. Naturally, such operation can be automatized for generating multiple building geometries. The behavior itself corresponds to an algorithm that follows the rule application order defined in SMUG (), although in some cases such order had to be changed, as explained further below.

Although Buildings are the main elements of the favela, there are two other types of inputs for the Interpreter. The second element type is the Street. Even though favela streets are informal and implicit as the spaces in-between buildings, the implementation of the SMUG implies formalizing streets, since such spaces influence the generation of new buildings. The third and final element it the Terrain, which is particularly steep in Santa Marta and naturally plays an important role in determining the geometry of new buildings. Therefore, as in the grammar, the necessary elements to generate the favela are Buildings, Streets and the Terrain. In the Interpreter, the geometry of these elements acts as inputs for the Interpreter's parametric model.

LEVELS OF DETAIL

While implementing the grammar, special attention was paid to how to present the tool to students, taking into account their different skill levels. As a result, the parametric model develops along four levels of increasing complexity: the main program, the grammar, the rules, and the module. Level One corresponds to the main program, containing a GH clustered component comprising the actual grammar implementation, for which users can specify input parameter values, or define these as random values, for automatic generation of buildings for example (Figure 2). This level includes additional functionalities related to user interface, and targets non-expert users of a parametric model.

Level Two corresponds to the Santa Marta Shape Grammar itself, encapsulated in the clustered component exposed in Level One (Figure 3). This level contains a set of GHPython components containing the definition of the Grammar's rules. Moreover, this level contains remapping structures that enable users to easily define parameter intervals for each

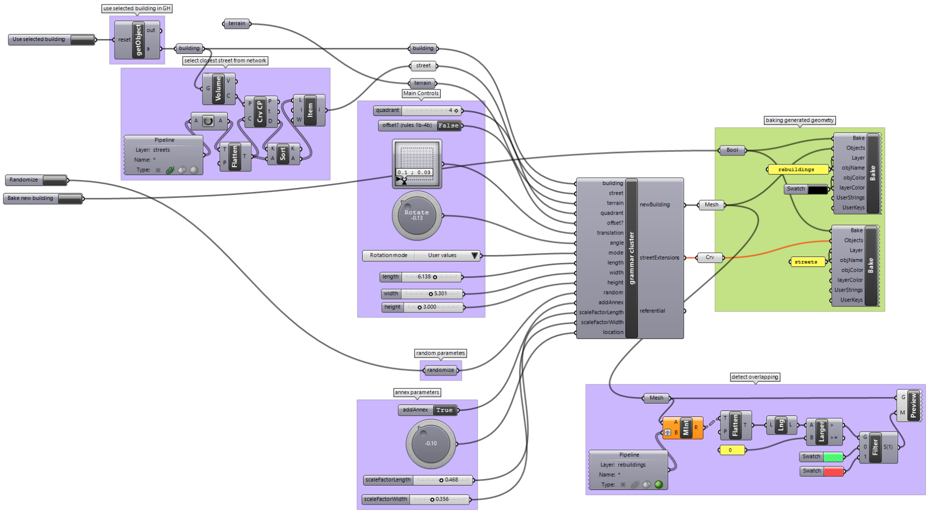


Figure 2
Interpreter Level 1:
Main program

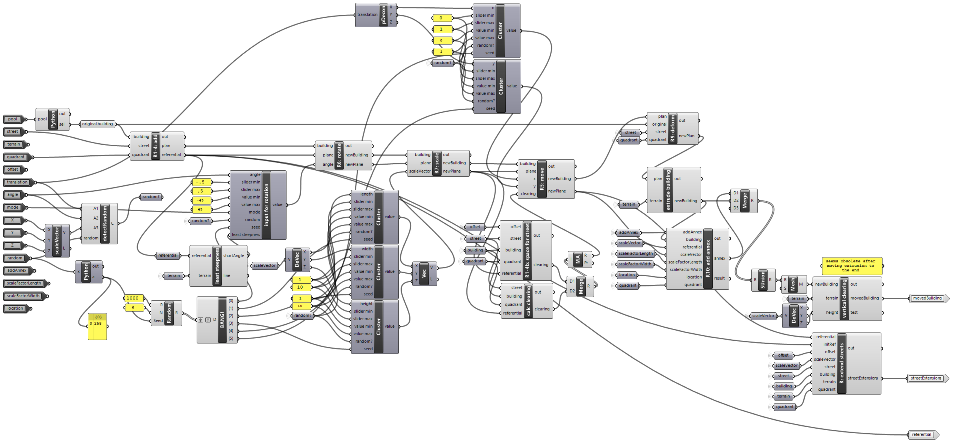


Figure 3
Interpreter Level 2: Grammar

rule. The purpose of this level is not only to better organize the grammar implementation, but to allow manipulating the grammar by adding or removing rules and defining parameter intervals. Such operation is possible for novice Grasshopper users.

Level Three corresponds to the Grammar's rules. Each rule has been encoded into a GHPython script component. By default, operating at this level is recommended for students that know Python. Finally, Level Four corresponds to a Python module (riostudio.py) that is used by the Rule scripts in Level Three. This module includes functions that either are used in more than one rule, or that were considered too complex for students to deal with. Operating at this level is recommended for more proficient Python users.

Being a pedagogical tool, applying levels for increasing expertise was an early and deliberate decision in designing the implementation, so that students with less programming skills could still manipulate the grammar, while students with more skills would still be challenged to explore deeper into the program. Moreover, such hierarchy allowed that at least one of the levels could be adapted to different skills. Indeed, the Python scripts in Level Three can be translated into GH definitions. Therefore, users with intermediate skills in GH could also change the behavior of the Grammar's rules. In order to accommodate for an unknown number of Python users, the team decided that a good exercise for teaching GH to students would be to translate the Rules from Python to GH.

STAKEHOLDERS

An interesting aspect of the implementation of SMUG was the interplay among stakeholders during the implementation process. We can consider four distinct types of stakeholders: a) researchers, who created the grammar; b) developers, who implemented the grammar; c) faculty, who will use the grammar and its implementation for teaching; and d) students, who will use the grammar and its implementation for learning.

In the case of this implementation, students are represented by a student who attended a previous edition of the course, and thus could provide insights on the skills and expectations of his former colleagues. Such input allowed, for example, balancing the researchers' drive towards grammar fidelity.

The input of faculty is also especially relevant. Less concerned with the technical aspects or the implementation's fidelity, faculty are interested in how the implementation can be most effective as a pedagogical tool for teaching about shape grammars, GH and eventually programming in general. And consequently, how can it illustrate - or negate - the potential of such approaches in the real world of design practices. For example, it was a faculty member's idea to constrain how shape rules were implemented, so that they could be translatable to GH, and therefore used to teach about GH.

The interaction between researchers and developers is perhaps the most impactful. Naturally, the researchers' inputs about the grammar define its implementation, but there were a considerable number of times in which the implementation sparked further detailing of the grammar. Such happened because the act of encoding grammar rules into algorithms implies an exact definition of those rules. In fact, the intentional effort towards grammar fidelity actually led to identifying ambiguities in the grammar rules, leading to debate that allowed to fine-tune the rules, as well as their implementation.

In a number of occasions, the developers raised questions about a particular rule that had not been considered when such rule had been defined by the researchers. This attests to the existence of aspects of the shape grammar that are implicit, and while such implicit aspects might be easy to identify by humans, they should be made explicit if they are to be encoded into a computer program (Knight, 2000).

IMPLEMENTING RULES

SMUG infers the implicit rules to generate the shape of specific Brazilian informal settlements that are highly dense, located within the urban area, and that develop on steep terrains. The rules express the decisions that residents must make in order to build within the settlement. The set includes rules to: locate the house according to the geographical and social context; orient and shape the house considering the optimized use of available resources and the immediate physical context; provide proper access to the houses, reshape the house considering changes in its functional organization; and gradually formalize the circulation network (Figure 4) (Verniz & Duarte, (forthcoming)).

In this section, we analyze in more detail the translation of the shape grammar rules into its implementation as Grasshopper clusters, referring to them as Grammar Rules (GR) and Implemented Rules (IR) respectively. Please note that the Interpreter is a work in progress, and thus some GRs are yet to be implemented.

BEFORE THE RULES

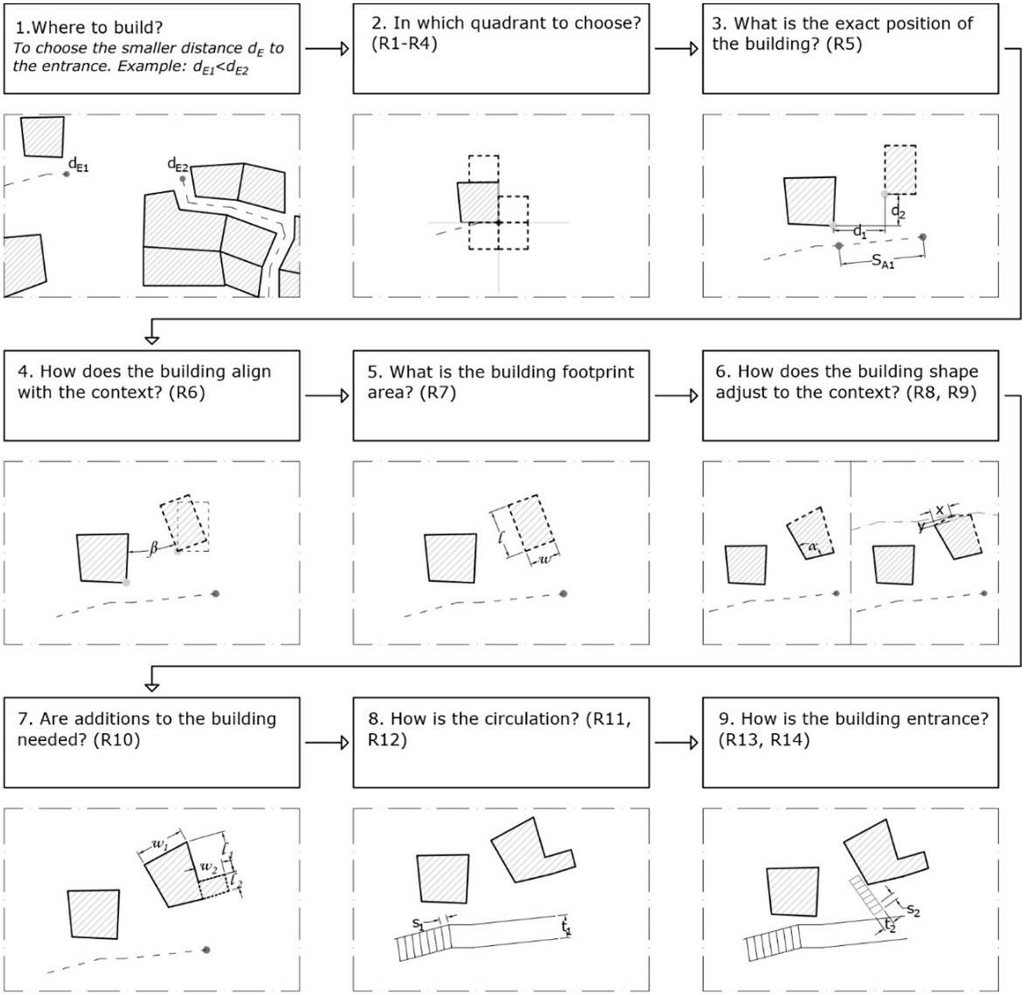
The first GR is applied to a building. However, such building needs to be selected, as illustrated in Figure 4, step 1. According to SMUG, the new building should be placed at the end of the street with the shortest distance to the favela's entrance, at the bottom of the hill. On the other hand, in the Interpreter, for the sake of versatility, every existing building can be considered for extending the favela, by being selected randomly or by the user. The selected building will therefore be used as the left hand side of the first GRs, or as input of the GH cluster corresponding to the respective IRs. Since streets are also used as inputs for generating new buildings, the Interpreter searches for the street segment that is closest to the selected existing building. Note that the Interpreter operates on a 3D model of the favela. Therefore, the streets correspond to line segments on the terrain.

POSITIONING RULES

The first four rules of the grammar locate a new building according to the geographical and social context (Figure 4, step 2). Each of the four rules places the new building in a different quadrant, whose origin is one of the vertices of an existing building's plan. Although implicit in GR1-4, such origin needs to be determined by the Interpreter. In IR1-4, the origin corresponds to a vertex of the building's main facade, which in turn is determined as its vertical surface that is best aligned with the input street. The origin and the main facade determine a plane that will be used as a referential for the remainder of the generation procedure. IR1-4 output not only a new building but also a new street segment, as well as the referential.

This example illustrates situations in which aspects of building generation are implicit in the shape grammar but need to be made explicit for its implementation. As we will later discuss, these situations provide the grammar creators with valuable insights for further developing the grammar.

Besides using different quadrants, each IR adapts the position of the new building according to the ex-



1.Where to build?
To choose the smaller distance d_E to the entrance. Example: d_{E1}<d_{E2}
2. In which quadrant to choose? (R1-R4)
3. What is the exact position of the building? (R5)
d_{E1}
d_{E2}
d_2
d_1
S_{A1}
4. How does the building align with the context? (R6)
5. What is the building footprint area? (R7)
6. How does the building shape adjust to the context? (R8, R9)
β
l
w
α
x
y
7. Are additions to the building needed? (R10)
8. How is the circulation? (R11, R12)
9. How is the building entrance? (R13, R14)
w_1
w_2
l_1
l_2
s_1
t_1
s_2
t_2

Figure 4
Derivation of the Santa Marta Urban Grammar

isting conditions. For example, a building produced by Rule 2 would overlap with the selected existing building. Therefore, the new building is placed adjacent to the existing one. Although, in the current implementation, the quadrant is defined by the user, the Interpreter provides information about overlapping between the new building and existing buildings other than the one selected (Figure 5).

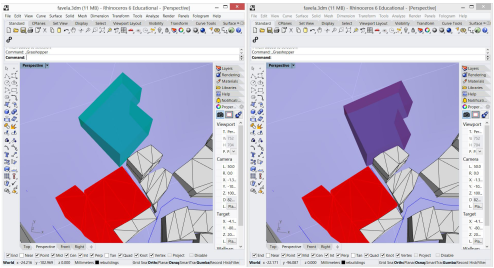

favela.3dm (11 MB) - Rhinoceros 6 Educational - [Perspective]
Perspective
Top
Perspective
Front
Right

Figure 5

Figure 5 - Buildings overlapping signaled with different colors

In the Grammar, GR1 through GR4 correspond to four distinct rules. Moreover, rules GR1, GR2 and GR4 are divided into sub-rules a) and b), the latter enabling space for circulation. However, due to the similarity among all these GRs, the corresponding Implemented Rules IR1-4 were implemented as one cluster, in which the quadrant is a parameter that can be either selected by the user or determined by the Interpreter according to particular criteria. An additional Boolean parameter determines whether or not space for circulation should be added, thus implementing version b of the mentioned GRs.

TRANSFORMATION RULES

Rules 5 through 7 apply affine transformations to the new building generated by the preceding rules, namely translation, rotation and scaling.

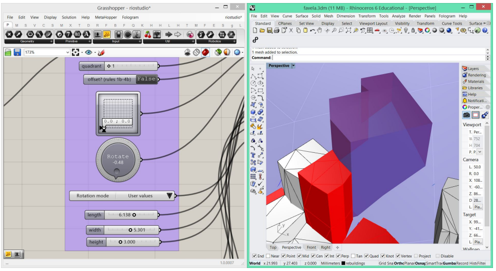
Rule 5 (GR5: Figure 4, step 3) is used for applying a translation to according to a particular input. While implementing the Interpreter, however, we verified that translation of the new building results from a number of different components, such as a) avoiding overlapping with the existing building, b) allowing space for circulation according to Rules 1-4b, and finally, c) accommodating for user input. Moreover, some of the translation components might result from applying subsequent rules. For example, rotating the new building (Rule 6) might cause it to overlap with the existing building. Therefore, in the Interpreter, Rule 5 is applied at the end of the generation procedure (Figure 3). In the Grammar, this would mean that Rule 5 would be applied at the end of the derivation, potentially justifying its renumbering.

In SMUG, Rule 6 (GR6: Figure 4, step 4) is used for applying a horizontally planar rotation to the new building, deriving from topography, when a building is aligned with a line of least steepness, or from the urban context, aligning it with neighboring buildings. In the Interpreter, an angle provided by the user is added as a possible input. Finally, Rule 7 (GR7: Figure 4, step 5) scales the building non-uniformly in the three dimensions according to user input, namely needed square footage. One action that is not contemplated in the grammar is the placement of the building vertically. Similarly to Rule 5, such placement depends on the result of additional rules, and is therefore left for the end of the generation procedure.

MODIFICATION RULES

Two additional rules were implemented. Rule 9 (GR9: Figure 4, step 6) changes the shape of the building by adding a point to its plan. Note that the first seven rules operate on rectangles, and therefore this rule potentially provides morphological richness to the generated favela by enabling the generation of non-rectangular buildings. An interesting aspect of rule 8 is that it was developed by request from one of the students, during the period in which they were exploring the Interpreter. The student's main purpose was to allow a new rotated building to seamlessly attach to the original building (Figure 6). Therefore, in its intent, this can be interpreted as a variation of Rule 9. However, the implemented IR9 can be later generalized for accommodating the more generic behavior of GR9.

Rule 10 (GR10: Figure 4, step 7) adds an annex to the new building. The implementation of its counterpart in the Interpreter (IR10) was drafted in the Interpreter as a scaled-down copy of the new building and attached to the latter, and only for quadrant 1. Later on, IR10 was improved by one of the students, who generalized it for all four quadrants.


Grasshopper - riostudio*
File Edit View Display Solution Help MetaHopper Fologram
Geometry
Primitive
Input
Util
Robotics
173%
favela.3dm (11 MB) - Rhinoceros 6 Educational - [Perspective]
Perspective
Top Perspective Front Right

quadrant 1

offset? (rules 1b-4b) False

0.0 ; 0.0

Rotate -0.48

Rotation mode User values

length 6.138

width 5.301

height 3.000

1 mesh added to selection.

Command:

Figure 6
Figure 6 - Implemented Rule 9

Implementation of these two rules illustrates that there is much space for improvement in the Interpreter implementation, but also that such improvement can gain much from being informed by their actual users. Moreover, the Interpreter can be further completed with implementation of subsequent GRs.

DISCUSSION ON RESULTS

As planned, the Interpreter was presented and made available to about a dozen students in the scope of the design studio during the spring semester of 2019. Using the Interpreter was recommended rather than mandatory, given the various levels of expertise among students on the different technologies in which the Interpreter is based, namely Rhino, Grasshopper and Python.

The results fell short of our expectations. None of the students actually used the Interpreter for the project for their final presentation, although there were some students who explored it during the semester. User feedback was also extremely reduced: only two people responded to a survey held at the end of the semester about the experience of using the Interpreter.

We hypothesize a number of reasons for such low usage of the Interpreter. The first potential reason is that most students in the design studio in question were not proficient in Grasshopper at the beginning of the semester. Although they had GH training during the semester, they have not reached a level of proficiency that would have them use GH in that same semester. Since the most approachable level of the Interpreter requires GH knowledge, such fact might have deterred students of using the tool.

The second potential reason is that the Interpreter was not yet complete. In fact, the Interpreter was admittedly a work in progress during the semester when it was available to students. Moreover, one of the points of having students use an incomplete Interpreter was for understanding what would be more important to develop, in a user-centered approach. However, the fact that the tool still had bug to be corrected might have discouraged students from relying on it for pushing their designs forward.

In spite of the shortcomings of this experiment, they provide important feedback on how to move forward. First of all, it raises the discussion of whether or not the effort of implementing a grammar is by itself justifiable. From the low acceptance of the Interpreter, we conclude that it is not, which raises the subsequent question of how to determine its necessity. In our case, although the Interpreter was not as successful as expected for its use in class, it proved to be considerably useful for clarifying issues in SMUG during its implementation.

Another aspect that seems determinant for assessing the relevance of implementing a shape grammar is the preceding knowledge of its users. The experiment presented in this paper was the first experience most of the students in the design studio had with both shape grammars and Grasshopper. This experiment can be used as an argument for requiring these skills as precedent for attending such design studio courses.

CONCLUSION

In this paper we presented a tool intended to enable non-expert users to apply and manipulate a shape grammar, SMUG, which encodes the urban design of informal settlements such as favelas. Such tool, the Interpreter, was developed considering that students would be its main users, and therefore such development took pedagogical aspects into account. In fact, despite its low usage by the students, we consider this grammar implementation to potentially be a multipurpose pedagogical tool since it supports conveying knowledge about urban design, shape grammars and parametric modeling using Grasshopper. However, in order to fulfill that potential, the hypothesized reasons should be addressed. First, additional training on the technologies used by the Interpreter should be provided prior to its use. Also, the Interpreter should be completed before being made available to students again.

Therefore, we believe that if such measures are addressed, the Interpreter will prove a valuable tool in the next iteration of this design studio.

REFERENCES

Barros, M 2015, *Mass customisation in the furniture design industry: the case of Thonet chairs*, Ph.D. Thesis, University of Lisbon

Beirão, JN, Duarte, JP and Stouffs, R 2011, 'Creating Specific Grammars with Generic Grammars: Towards Flexible Urban Design', *Nexus Journal*, 13, pp. 73-111

Chase, SC 2002, 'A model for user interaction in grammar-based design systems', *Automation in Construction*, 11(2), pp. 161-172

Chau, HH, Chen, X, McKay, A and Pennington, A 2004 'Evaluation of a 3D Shape Grammar Implementation', *Design Computing and Cognition'04*, Dordrecht, The Netherlands, pp. 357-376

Clarke, KC 2014, 'Cellular Automata and Agent-Based Models', in Fischer, MM and Nijkamp, P (eds) 2014, *Handbook of Regional Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1217-1233

Castro e Costa, E, Jorge, J and Duarte, J 2019, 'Comparing Digital Tools for Implementing a Generative System for the Design of Customized Tableware', *Computer-Aided Design and Applications*, 16(5), pp. 803-821

Davis, M 2006, *Planet of Slums*, Verso, London, New York

Duarte, JP, Rocha, JM and Soares, GD 2007, 'Unveiling the structure of the Marrakech Medina: A shape grammar and an interpreter for generating urban form', *AI EDAM*, 21(4), pp. 317-349

Figueiredo, B, Castro e Costa, E, Duarte, JP and Krüger, M 2013 'Digital Temples: a Shape Grammar to generate sacred building according to Alberti's theory', *Future Traditions, Rethinking Traditions and Envisioning the Future in Architecture Through the Use of Digital Technologies*, Porto, pp. 63-70

Grasl, T and Economou, A 2018, 'From shapes to topologies and back: an introduction to a general parametric shape grammar interpreter', *EDAM*, 32, pp. 208-24

Knight, TW 1994, *Transformations in Design: A Formal Approach to Stylistic Change and Innovation in the Visual Arts*, Cambridge University Press, New York, NY, USA

Knight, TW 1998, 'Shape grammars in education and practice: history and prospects', *International Journal of Design Computing*, N/A, p. N/A

Li, AIK 2005 'Thoughts on a Designer-friendly Shape Grammar Interpreter', *Digital Design: The Quest for New Paradigms*, Lisbon (Portugal), pp. 523-528

Li, Alk 2018, 'A whole-grammar implementation of shape grammars for designers', *EDAM*, 32, pp. 200-7

Ligler, H and Economou, A 2019 'From Drawing Shapes to Scripting Shapes: Architectural Theory Mediated by Shape Machine', *Proceedings SimAUD 2019*

Maricato, E 2001, 'Metrópole periférica, desigualdade social e meio ambiente', in Viana, G, Silva, M and Diniz, N (eds) 2001, *O Desafio da Sustentabilidade: um Debate Socio-Ambiental no Brasil*, Fundação Perseu Abramo, SP

Mendes, L, Celani, G and Beirão, JN 2014, 'Meta-PREVI Grammar', *International Journal of Architectural Computing*, 12(4), pp. 459-476

Nations, United 2015, *Urban Solutions: United Nations Human Settlements Programme*, UN, Nairobi

Speller, TH, Whitney, D and Crawley, E 2007, 'Using Shape Grammar to Derive Cellular Automata Rule Patterns', *Complex Systems*, 17, pp. 79-102

Stouffs, R 2016 'An Algebraic Approach to Implementing a Shape Grammar Interpreter', *34th eCAADe*

Verniz, D and Duarte, JP 2017 'Santa Marta Urban Grammar: Towards an understanding of the genesis of form', *Proceedings 35th eCAADe*, Roma, pp. 477-484

Verniz, D and Duarte, JP forthcoming, 'Santa Marta urban grammar: unraveling the spontaneous occupation of Brazilian informal settlements', *Environment and Planning B: Planning and Design*, N/A, p. N/A