Programming Complex 3D Meshes. A Generative Approach Based on Shape Grammars

Prof. Dr. Umberto Roncoroni
Faculty od Design, Universidad Peruana de Ciencias Aplicadas, Lima, Perù
www.digitalartperu.org
e-mail: umberto.roncoroni@upc.pe

Abstract

This article summarizes the results of art based research developed thanks to a grant of the PUCP University of Lima in 2021 2022. It will be described an open source generative solution, based on generative grammars to create very complex and programmable 3D meshes. Analyzing hundreds of models generated with these algorithms, it was found a solution based on the idea of “intelligent meshes”, which change their behavior during the modeling process. This is done using tags, or vertices identifiers, that, like genes, describe the topological characteristics of each vertex and its generative development during the process. Tags can be programmed interactively editing its data with tools provided by the interface or using generative grammars that allow an incredible variety of complex forms and stimulate the user creativity. The research findings also elucidate some important conceptual issues, like the importance of original technology development to defend cultural identity.

1. Introduction

Creativity is a key issue in arts, science and cultural industries, not to mention that it is of the greatest concern for innovative educational programs. But creativity is a difficult topic to be handled properly. It is enough to mention just three problems: creativity is hard to define, explain and measure [1], its aesthetic meaning and aura are jeopardized by postmodern art [2], over production and media saturation, and, last but not least, the disruptive effect of digital media.

To enter directly into the digital matter, today computational creativity, 3D modelling, animation and image processing technologies research, such as generative algorithms or fractals, is occupied by the AI and Machine Learning discourse. But AI, not so much paradoxically, leaves small room to users’ creativity (Colton 2008) and, spreading Anglo-Saxon computational

thinking, is one of the most efficient assets of digital colonization [4].
These are good reasons to develop shape grammars [5] and generative algorithms as a valid alternative [6], for their simplicity, creative power [7][8] and because they offer the possibility to simulate natural phenomena and local artistic traditions, like ethno computation [9][10], intuitively and without black boxes [11][3]. In this paper I will concentrate the attention on software development, visual analysis and artistic practice results. Due to these properties, the generative design tools described in the following paragraphs will be valuable to artists, industrial designers and educators to experiment with new design processes, explore computational creativity as a research or educational tools and to link parametric design with cultural identity. From the production point of view, these algorithms help artists and designers to explore the relationships between forms and new materials also suitable for 3D printers and robotic fabrication.

2. Methodology

This paper is the result of an interdisciplinary artistic research project supported by a grant of the PUCP University of Lima. The research methods expand the art based research framework [12] and consist of: a) Review of papers in the field of Computer Science, Digital Humanities and Digital Art, especially generative design, shape grammars and ethno computation topics; b) Analysis of software for audio-visual creative production (DAWs and 3D Modelling software Rhino and 3D Max); c) Visual analysis of pre-Columbian art; d) Software development using extreme and incremental programming; e) Artistic practice and digital fabrication with 3D printers and a Kuka robotic arm.

3. Results

3.1 Literature and Software Analysis

Papers about computational creativity, generative art and parametric design show that the potential of shape grammars is not fully developed [13]. Besides, there is a lack of friendly and interactive generative applications. On the other hand, plug ins (like EuroRack), programming languages (like Processing), game design engines or DAWs (like Unity or Reaper) that use AI or generative techniques and can be often installed freely, quite often share the same algorithms and lack proper documentation. This is reflected in repetitive and standardized design artefacts.

3.2 Analysis of Natural Forms, Pre-Columbian Art and Shape Grammars Simulations

The capability and potential of L-Systems to simulate natural phenomena is well known [7], so it is not necessary to enter into this topic here. On the other hand, Pre-Columbian and traditional ethnic art shows [14] that algorithmic and natural procedures were commonplace.
As shown in figure 1, there is obviously a computational thinking in the ropes, knots and colours and a creative hypothesis to use them as a linguistic code or interface design metaphor to improve usability in shape grammars applications.

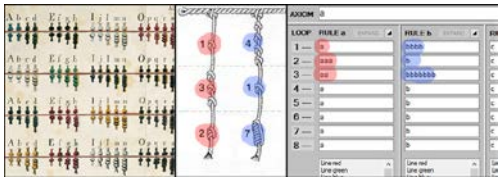

Fig 1. The rule system interface design is similar to quipus, with a bas eline –the axiom- that opens the sequence of rules. The effect of the rule depends on i ts vertical hierarchy, like the quipus' knots. For instance, the generative potential of quipus was investigated by the neapolitan alchemist Raimondo di Sangro [15].

3.2 Software Development, Artistic Practice and Improvement of L-Systems Techniques

Even if a huge amount of research about shape grammars exists, the creative power of symbolic dictionaries, rules and substitution algorithms can be expanded. In existing applications rules are rigid, can't share parameters and programming tools like loops or conditional statements. In previous research [13], were developed improvements to L-Systems dictionary and rule sets to overcome some of their limitations.

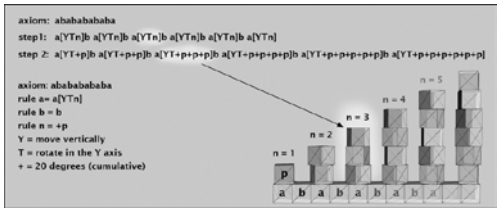

Figure 2. Example of symbols for nested recursive substitution

I will mention here just one of these extensions: automatic symbols ("n") with nested recursion and w ith slave or sub-symbols ("ñ") controlled by the number of instances, or the master symbol hierarchy in the grammar, or by the level of the substitution process. Figure 2 explains a design that is impossible to build with standard L-Systems vocabulary and rules, since it will be necessary to write a par ticular rule for every column to match the number of blocks and their rotation degrees. Symbol "n" sets the hierarchy of the columns in the row and " ñ" sets the corresponding number of objects: for example, the first instance of "n" sets 1 block, the third instance 3 blocks, etcétera. In this way L-Systems are converted in a sort of programming language, like side chain functions, to link the number of bricks to the empty space between them, and t o match the *chakana's* grammar to the position and rotation parameters of the growing spiral (fig. 3).

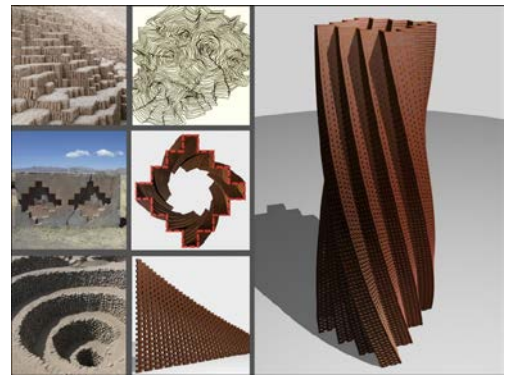

Fig. 3. Left: huaca, Andean cross (chakana), and spirals in the Cantalloq aqueduct. Algorithmic drawing, L-Systems to rotate the chakana and t o match positions with bricks' number. Final L-Systems tower.

3.3 Software Development and Artistic Production

During the research many generative techniques have been ex plored, using

self-similarity, natural processes, and traditional designs' ethno computation. After the generation with different functions and parameters of hundreds of models, were selected two solution that solved the task to create something new. The first is the mesh remix tool set that expands the standard morphing process with additions like masks, side chain modulation, genetic behaviours, shape grammars and cellular automata (fig. 4). The second that will be exposed in the following sections, is the programmable mesh technique.

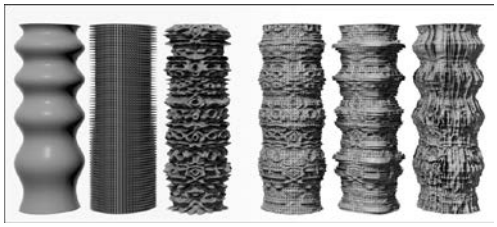

Fig. 4. Left: three meshes (ancestors). Right: three remix modes of ancestors.

3.4 Generative Programmable Meshes

The algorithm that will be described here is based on the idea of a mesh that changes its geometric properties during the generation process. Like in cellular automata and finite state machines, the mesh' vertices act like cells whose values describe topological properties, transformation parameters and other behaviors. In this way the mesh grows like an organic natural process. This is done using "tags", or vertices identifiers, assigned to a pattern of vertices that can be programmed interactively or using L-Systems [6]. This allows for an incredible variety of complex forms, and stimulates the user to experiment freely.

In the first step the user creates a pattern of n points (usually a multiple of 8 to match symmetry and bytes) and allocates their alphanumeric identifiers, the tags. This pattern generates a closed shape with 8 or 4 axis symmetry (fig. 5). Here is where shape grammars and L-Systems come into hand, to create interactively the patterns and change the tags during the process.

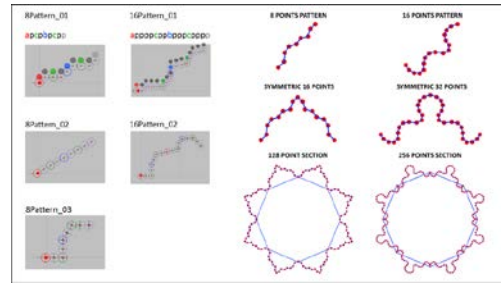

Fig. 5. Left: example of patterns. Left: construction of the mesh sections shape.

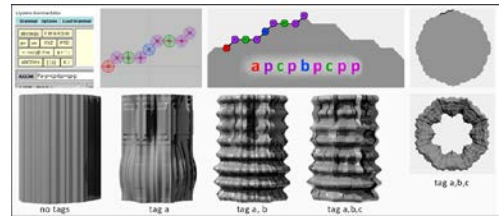

Fig. 6. Using tags in a programmable mesh. Top left: L-Systems grammar, tags pattern and the complete symmetric section shape. Down: adding tags transformations to the linear mesh.

Now, during the mesh construction, every point can be translated, scaled or rotated using their tag parameters, and behave independently or interacting with other tags, considering its XZ position in the section and in its height in the mesh (Fig. 6, 7). In this way every section or slice of the mesh can smoothly change its form without losing the formal coherence of the mesh as a whole. The interactions between points and tags can be done with cellular automata, interactive functions or reading values

from data sets or images. The tag rule set can be processed using the usual shape grammars substitution process embedded in the main function (Fig. 8). These data can be saved and combined with the others using the remix tools describe above.

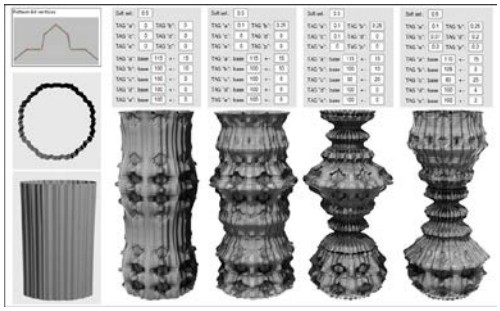

Fig. 7. Left: the pattern, the section and the linear mesh. Right: transforming the mesh with the same pattern and tags but different parameters' values.

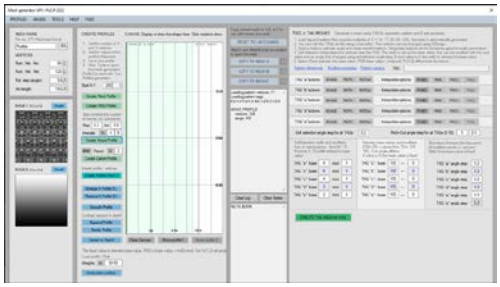

Fig. 8. The workflow is from right to left and from top to down. The right panel configuration depends on the previous choice of the user. The software checks which buttons are enabled, avoiding unnecessary work of the user. Help is included in the panel, improving the concept of the software as a book.

3.5 Technical Issues of Complex Generative Meshes

Generative processes like programmable meshes are highly unpredictable (this is the reason why they are so fascinating). But this comes at the cost of geometrical problems that happens when vertices are heavily transformed and vertices' positions are too rough. In this sense, tags helps to analyze the topological data without performing tests that, when you are working with more than 1,000,000 polygons, slow down the process too much. The software additionally takes charge of other issues that could result in geometric inconsistencies such as face intersections that cause errors or the need to use support material in the 3D printing process.

3.6 Software Development and Interface Design. The Artist as Computer Scientist

Working with complexity, generative processes and art, it results that software development gets very confusing. It is interesting to stress here the different approach to programming of artists and computer scientists. In the present case, extreme and incremental programming paradigms where used, but when the programmers are artists, the development is a lot less linear than expected. While programming needs careful organization and a precise workflow, the artistic experimentation and software development needs improvisation, serendipity and permanent trial and error processes that leads quickly to bugs, undesired effects and ineffectiveness.

The solution was, in the first place, to experiment freely with the code at the beginning, and rewrite the entire application also improving the user interface design. Through parallel artistic production, it was found that the best software architecture should be modular, to help the user thorough the process step by step, with every step enabled by

its predecessor and the compatibility of geometric properties. The interface accompanies the workflow with instructions and examples on how to use every function, to make the learning curve as smooth as possible. Finally, considering the open source philosophy of this application, the code was revisited in the literary sense, and considered as a text in its full right.

4. Discussion

Setting apart the artistic and technical benefits, the research findings also elucidate some important concept issues about computational creativity and education.

4.1 Original Technology Research

In the first place, software development and artistic results exposed the importance of original technology research. This infers "reinventing the wheel", in other words, to develop algorithms or functions are already available in internet. The true of this lies in the fact that real innovation comes from the deep understanding and control of every layer of the process; on the contrary, the use and abuse of libraries and ready to use solutions, that can be helpful to speed up production, generate creative constraints –the proper word should be cages- which creative results are not of the artist.

Original technology research is paramount also in the broader cultural domain, to defend cultural identity and correct the ideological biases [9] the commercial modelling solutions for artists, designers and architects. Every single line of code embeds significant knowledge that will unfold completely when all the pieces are put together, giving to the software and to its users cultural definition and power.

4.2 The Black Box Problem and the Benefits of Generative Grammars Solutions

The computational and artistic research results demonstrate that complexity and creativity forms don't need complicated technological solutions; L-Systems, in this sense, have many benefits. First, with some improvements, offer control and flexibility almost like a programming language, but are easier to understand (yet certainly difficult to develop properly). In the second place, L-Systems grammars and codes are transparent, and more intelligible, compared, for instance, with AI algorithms [16] whose deep computational processes are puzzling even for their creators.

I will add that AI can be developed starting from the fundamental idea of meta-medium [17] and can be interpreted as interfaces architecture and design in any application. Also the difficulties of generative design can be limited with a proper interface design and coding style, both help the users to exploit the parameters' creative properties and the aesthetics properties of algorithms [18]. It is important to reckon that many independent and open source solutions are discarded because of lack of documentation.

4.3 Issues in Educational Technology

These topics are particularly relevant when digital tools are used in learning contexts [19]. Generative grammars lingos, like L-Systems, not only can be programmed easily, even without

experience, but also, very much like Turing machines, they can be developed by hand [21] and can be used as methods in analog processes with traditional materials. Even in digital processes, the need of computers appears only in the last step of the design process; in this way machines do not interfere with the development of a creative and critical computational thinking.

In this sense, cultural identity and ethno computation references and r esources, like quipus or the yupana, are not just visual metaphors for interface layouts or artistic installations. Embedded and coded in algorithms and f unctions and supported by analogies in design methods, data structures and computations, cultural traditions come to life to shape contemporary culture as concrete methods, solutions and fabrication tools.

4.4 Conclusions

To finish, I will resume the main concepts and findings of the research, and s ome ideas about its future developments and improvements.

Fig. 9. Generative grammars and programmable meshes can simulate different artistic styles, and hel p to understand their formal processes.

a) Generative grammars proved, through artistic practice, that are very creative tools and that there is no need of machines to foster digital literacy and computational thinking. Using traditional techniques and materials overcomes the techno centric bias that educational technology carry out [11].

b) Cultural traditions, native artistic practices and ethno computation are inherently modular and r ecursive, thus and can be molded with shape grammars and the tag solution discussed here smoothly (fig. 9).

c) Generative art and generative grammars are techniques with a great creative and heur istic potential, as software development demonstrated during the project activities. From the aesthetic and epi stemological point of view, the artistic research validation can be sustained precisely by this heuristic potential, whose evidence is the artistic production and its diffusion in design communities.

d) Software development and artistic practice also discovered some geometric and t opological problems raised by complex generative processes. But the programmable tag mesh solution minimizes this issues and f acilitates the compatibility with digital fabrication and demonstrated that complex forms can improve competences in 3D printing and robotic manufacture and t he possibilities of recycled organic materials (Fig. 10).

Fig. 10. Complex meshes to adjust 3D printing process

Setting apart technicalities, this computer interdisciplinary research also enlighten some interesting concepts about computational creativity and the relationships between computational creativity and education.

a) Writing our own functions and giving up the cut and paste of software libraries may seem excessive, since requires hard work and a sort of "rediscovering the wheel" process. But this is necessary for true digital literacy, technological innovations and creativity. In fact, the control over these pieces of knowledge (algorithms, processes and parameters), we eventually miss using libraries lightly, is the key to add aesthetic value and originality to our projects.

b) It should be paid a lot more attention to the cultural aspects of software and interface design. Software is a complex cultural object with many layers of meaning that still we are not taking advantage as such. For educational and artistic purposes of computational thinking and creativity, the artistic research enlightened the differences between coding and software. Software is more than writing code, includes interactivity, the coherence between ends and means, cultural biases, issues about the distribution of information of knowledge. So far, software as cultural object needs much more humanities than sciences.

4.5 Further development

Generative design methods like shape grammars and techniques like programmable meshes can be indefinitely developed and improved from the computational, aesthetic and educational point of view. I will mention some lines of research in digital humanities that seem particularly important: to develop interface designs and human-machine interaction strategies for creative purposes; explore software as a text, in the sense defined by [20], that gathers technical and creative means, data, concepts and audiovisual resources; and finally, strategies and programs to improve the interdisciplinary formation of artists as inventors and scientists.

References

References

[1] Carnovalini, F. y Rodà A. (2020). Computational creativity and music generation systems: An introduction to the state of the art. *Frontiers in artificial intelligence 3*.

[2] Vattimo, G. (2000). *La società trasparente*. Milán: Garzanti.

[3] Colton, S. (2008). Creativity versus the perception of creativity in computational systems. *AAAI Spring Symposium: Technical Report*, pp. 14-20.

[4] Iranil, L., Vertesi, Dourish, J., Kavita, P. and Grinter, R. (2010). *Postcolonial Computing: A Lens on Design and Development*. Irvine: University of California.

[5] Stiny, G., and J. Gips. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. In O. R. Petrocelli (ed.), *The Best Computer Papers of 1971*, pp. 125–135.

[6] McCormack, J. (2004). Generative Modelling with Timed L-Systems. In J. S. Gero, ed. *Design Computing and Cognition*. Berlin: Springer, pp. 157–175.

[7] Prusinkiewicz, P. and A.

Lindenmayer. (1990). *The Algorithmic Beauty of Plants*. Berlin: Springer.
[8] Pestana, P. (2011). Lindenmeyer Systems and t he Harmony of Fractals. *Proceedings of the Chaotic Modeling and Simulation International Conference*, pp. 449-456.
[9] Varma, R. (2006). Making computer science minority friendly. *Communications of the ACM 49*(2), pp. 129-134.
[10] Roncoroni, U and V . Crousse. (2016). La v irtualidad aumentada: procesos emergentes, arte y medios digitales. *Artnodes*, 17.
[11] Alfieri, A. (2005). L-system Fractals: an educational approach by new technologies. *Quaderni di Ricerca in Didattica* (Mathematics), 25:2. Palermo: University of Palermo.
[12] O'Donoghue, D. (2009). Are We Asking the Wrong Questions in Arts-Based Research? *Studies in Art Education,* 50: 4, pp. 352-368.
[13] Roncoroni, U. (2022). Electronic Music and Generative Remixing: Improving L-Systems Aesthetics and Algorithms. *Computer Music Journal*, 45:1, pp. 55–79.
[14] Crousse, V. (2011). *Reencontrando la espacialidad en el arte público del Perú.* Tesis presentada para la defensa del grado de Doctor. Universidad de Barcelona, Barcelona.
[15] di Sangro, R. (1750). *Lettera apologetica dell'Esercitato accademico della Crusca*. Naples: Gennaro Morelli.
[16] Wyse, L. (2019). Mechanisms of artistic creativity in deep l earning neural networks. E n *Proceedings of the International Conference on Computational Creativity*.
[17] Kay, A. (1984). Computer Software. *Scientific American* 251(3), pp. 52–59.
[18] Fishwick, P. (2006). *Aesthetic Computing*. Cambridge, Massachusetts: MIT Press.
[19] Stig Møller, H. (2017). Deconstruction/Reconstruction: A Pedagogic Method for Teaching Programming to Graphic Designers. In Soddu, C. (ed.) *20h Generative Art Conference GA2017 Proceedings*.
[20] Barthes, R. 1997. The Death of the Author. In S. Heath, trans. *Image, Music, Text*. London: Fontana Books.
[21] Roncoroni, U. (2015). *Manual de diseño generativo*. Lima: Fondo Editorial de la Universidad de Lima.