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

Umberto Roncoroni Osio[1]

[1]Universidad Peruana de Ciencias Aplicadas, Lima, Perù
umberto.roncoroni@upc.pe

Abstract: This article summarizes the results of art based research developed thanks to a grant by the PUCP University of Lima in 2021-2022. An open source generative solution will be described, based on generative grammars, to create very complex and programmable 3D meshes. Analyzing hundreds of models generated with these algorithms, a solution was found 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.

Keywords: Shape grammars, complexity, digital fabrication, ethno computation, generative design

1 Introduction

Creativity is a key issue in the 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 (Carnovalini & Rodà, 2020), its aesthetic meaning and aura are jeopardized by postmodern art (Vattimo, 2000), 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 modeling, 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, it is one of the most efficient assets of digital colonization (Iranil, 2010).

These are good reasons to develop shape grammars (Stiny & Gips, 1972) and generative algorithms as a valid alternative (McCormack, 2004), for their simplicity, creative power (Prusinkiewicz & Lindenmayer, 1990; Pestana, 2011) and because they offer the possibility to simulate natural phenomena and local artistic traditions, like ethno computation (Varma, 2006; Roncoroni & Crousse, 2016), intuitively and without black boxes (Alfieri, 2005; Colton, 2008). In this paper I will concentrate 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 from the PUCP University of Lima[1]. The research methods expand the art based research framework (O'Donoghue, 2009) 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 audiovisual creative production (DAWs and 3D Modeling software Rhino and 3DMax); 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 (Roncoroni, 2022). Besides, there is a lack of friendly and interactive generative

[1] See https://investigacion.pucp.edu.pe/grupos/gries/noticia-evento/concurso-anual-proyectos-investigacion-cap-2021/

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 artifacts.

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 (Prusinkiewicz & Lindenmayer, 1990), so it is not mandatory to enter into this topic here.

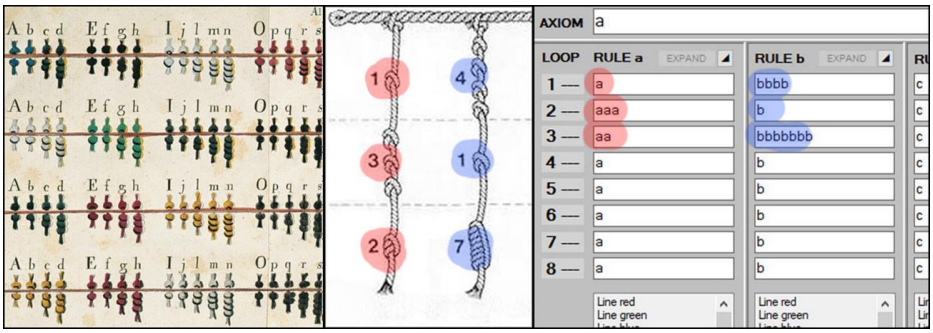

Fig 2. The rule system interface design is similar to *quipus*, with a baseline –the axiom- that opens the sequence of rules. The effect of the rule depends on its vertical hierarchy, like the *quipus*' knots. Source: author (2020).

On the other hand, Pre-Columbian and traditional ethnic art shows (Crousse, 2011) that algorithmic and natural procedures were commonplace. The same could be said about the *chakana* and the *quipus* that will be mentioned in this article, and other ancient artifacts and designs like the *yupana (*Fig. 1). For instance, the generative potential of *quipus* was investigated by the Neapolitan alchemist Raimondo di Sangro (1750). As shown in figure 2, there is obviously a computational thinking in the ropes, knots and colors and a creative hypothesis to use them as a linguistic code or interface design metaphor, to improve usability in shape grammars applications.

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

Even if a huge amount of research about shape grammars already exists, the creative power of symbolic dictionaries, rules and substitution algorithms can be expanded. In existing applications, rules are rigid, they can't share parameters and programming tools like loops or conditional statements. In

previous research (Roncoroni, 2022), improvements were developed to L-Systems dictionary and rule sets, to overcome some of their limitations.

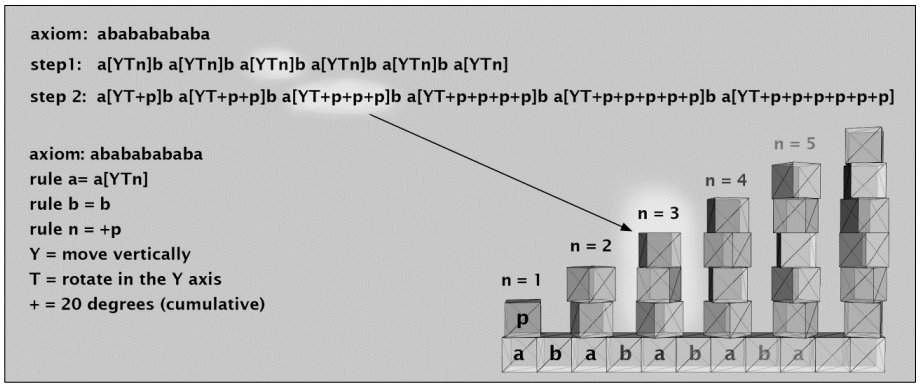

Fig. 3. Example of symbols for nested recursive substitution. Source: Author (2020).

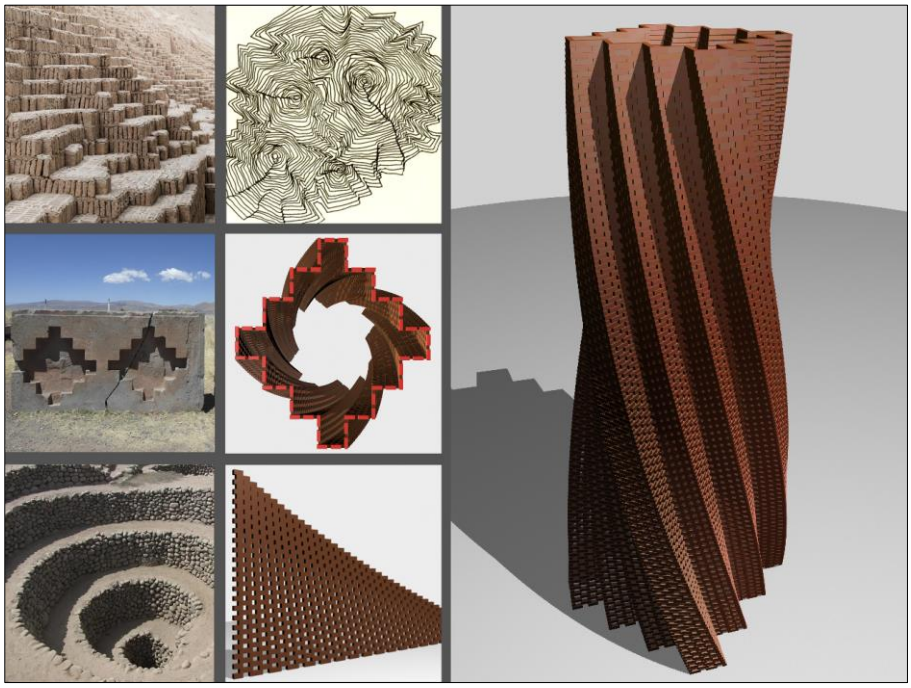

Fig. 4. Left: *huaca*, Andean cross *(chakana)*, and spirals in the Cantalloq aqueduct, near Ica, in Perù. In the middle, Algorithmic drawing, L-Systems' grammar to rotate the *chakana* and to match positions with bricks' numbers. Left: Final L-Systems tower. Source: Author (2018).

I will mention here just one of these extensions: automatic symbols (“n”) with nested recursion and with 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 3 explains a model that is impossible to build with standard L-Systems vocabulary and rules, since it would be necessary to write a particular 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, etcetera. 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 to match the *chakana’s* grammar to the position and rotation parameters of the growing spiral (fig. 4).

3.4 Software Development and Artistic Production

During the research many generative techniques have been explored, using self-similarity, natural processes, and traditional designs’ ethno computation. After the generation with different functions and parameters of hundreds of models, two solutions were selected 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 behaviors, shape grammars and cellular automata (fig. 5). The second that will be exposed in the following sections is the programmable mesh technique.

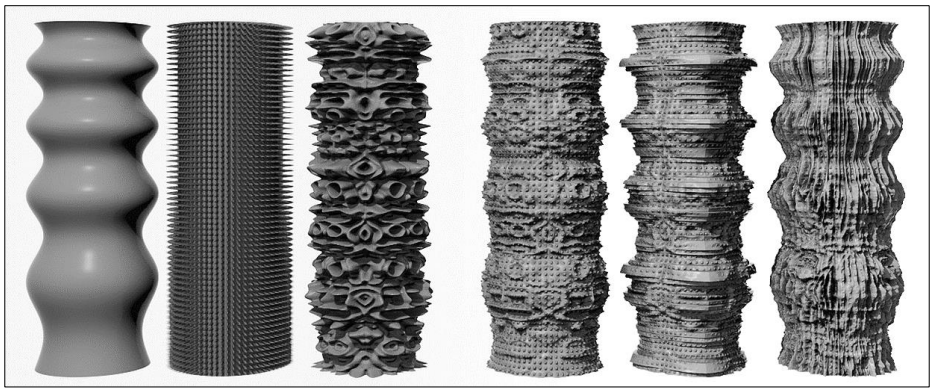

Fig. 5. Left: ancestors meshes. Right: three remix modes. Source: author (2022).

3.5 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 meshes' 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 (McCormack, 2004). This allows for an incredible variety of complex forms, and stimulates the user to experiment freely.

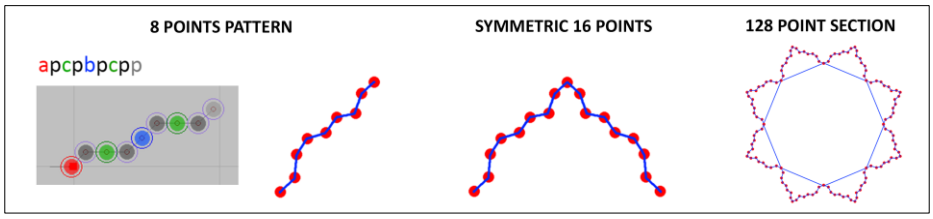

Fig. 6. Left: example of patterns. Left: construction of the mesh sections shape. Source: author (2022).

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. 6). Here is where shape grammars and L-Systems come into hand, to create interactively the patterns and change the tags during the process.

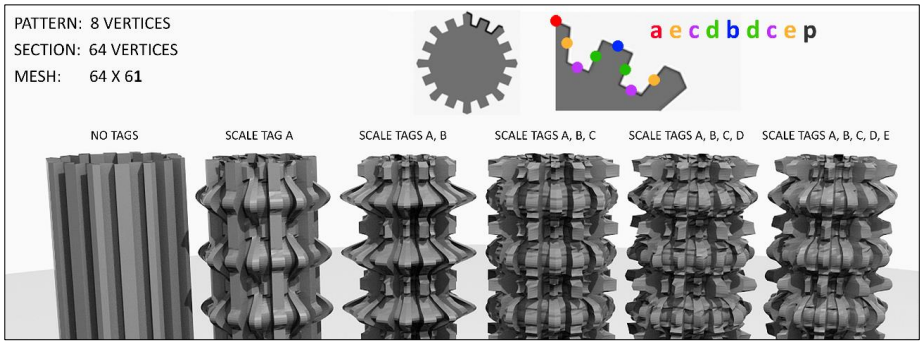

Fig. 7. Top left: L-Systems grammar, tags pattern and the symmetric section shape. Down: adding tags transformations to the linear mesh. Source: author (2022).

Now, during the mesh construction, every point can be translated, scaled or rotated using their TAGs parameters, and behave independently or interacting with other TAGs, considering its XZ position in the section and in its height in the mesh (Fig. 7, 8). 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 TAGs rule set can be processed using the usual shape grammars substitution process embedded in the main function (Fig. 8). This data can be saved and combined with the others using the remix tools describe above.

3.6 Technical Issues of Complex Generative Meshes

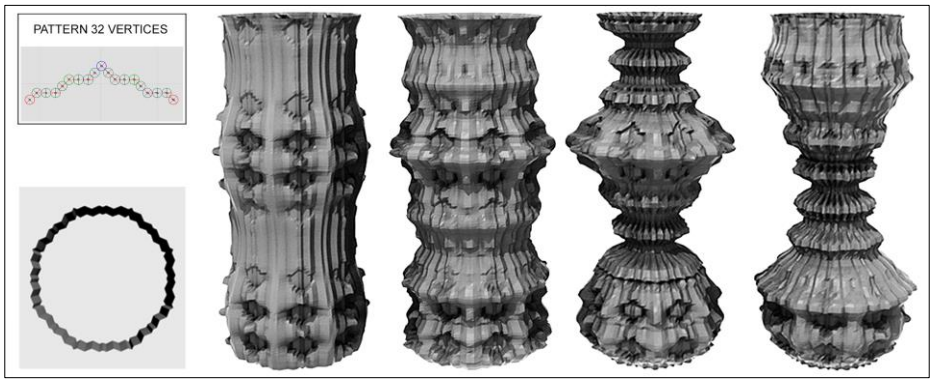

Fig. 8. Left: the pattern, the section and the linear mesh. Right: transforming the mesh with the same pattern and TAGs but different parameters' values. Source: author (2022).

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 happen when vertices are heavily transformed and vertices' positions are too rough. In this sense, TAGs help to analyze the topological data without performing tests that, when working with more than 1,000,000 polygons, slow down the process excessively. 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.

3.7 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 were 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 quickly lead 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 step-by-step process, 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 conceptual 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 truth 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 - where creative results are not of the artist. Original technological research is paramount also in the broader cultural domain, to defend cultural identity and correct the ideological biases (Varma, 2009) of 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. Firstly, 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 (Wyse, 2019) 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 (Kay, 1980) and can be interpreted as interfaces architecture and design in any application. Furthermore, 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 (Fishwick, 2006). It is important to realize 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 (Stig Møller, 2017). 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 (Alfieri, 2005) and can be used as methods in analog processes with traditional materials. Even in digital processes, the need for 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 resources, like *quipus* or the *yupana*, are not just visual metaphors for interface layouts or artistic installations. Embedded and coded in algorithms and functions 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

Lastly, I will resume the main concepts and findings of the research, and some ideas about its future developments and improvements.

a) Generative grammars proved, through artistic practice, that they are very creative tools and that there is no need for machines to foster digital literacy and computational thinking. Using traditional techniques and materials overcomes the techno-centric bias that educational technology carry out (Alfieri, 2005).

b) Cultural traditions, native artistic practices and ethno computation are inherently modular and recursive (Khajuraho & Vinayak 2018), thus, can be molded with shape grammars and the tag solution discussed here easily (fig. 9).

c) Generative art and generative grammars are techniques with a great creative and heuristic potential, as software development demonstrated during the project activities. From the aesthetic and epistemological 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.

Fig. 9. Generative grammars and programmable meshes can simulate different artistic styles, like medieval Hindu architecture and decorative art, and help to understand their fractal processes. Left: source: Khajuraho & Vinayak (2018). Right: author (2022).

d) Software development and artistic practice also discovered some geometric and topological problems raised by complex generative processes. But the programmable tag mesh solution minimizes these issues and facilitates the compatibility with digital fabrication, and demonstrated that complex forms can improve competences in 3D printing and robotic manufacture, as the *Subdivided Columns* of Michael Handsmeyer (2019) already demonstrated, and the possibilities of recycled organic materials. Technicalities apart, this computer interdisciplinary research also enlighten some interesting concepts about computational creativity and the relationships between computational creativity and education.

e) Writing our own functions and giving up the cut and paste of software libraries may seem excessive, since it 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, through 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.

f) A lot more attention should be paid to the cultural aspects of software and interface design. Software is a complex cultural object with many layers of meaning that we are still 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, it includes interactivity, the coherence between ends and means, cultural biases, issues about the distribution of information of

knowledge. So far, software as a 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 Barthes (1997), that gathers technical and creative means, data, concepts and audiovisual resources; and finally, strategies and programs to improve the interdisciplinary formation of artists like inventors and scientists.

References

Alfieri, A. (2005). L-system Fractals: an educational approach by new technologies. *Quaderni di Ricerca in Didattica* (Mathematics), 25:2. Palermo: University of Palermo.

Barthes, R. (1997). The Death of the Author. In S. Heath, trans*. Image, Music, Text*. London: Fontana Books.

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*.

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

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.

di Sangro, R. (1750). *Lettera apologetica dell'Esercitato accademico della Crusca*. Naples: Gennaro Morelli.

O'Donoghue, D. (2009). Are We Asking the Wrong Questions in Arts-Based Research? *Studies in Art Education,* 50: 4, pp. 352-368.

Fishwick, P. (2006). *Aesthetic Computing*. Cambridge, Massachusetts: MIT Press.

Handsmeyer, M. (2019). Tools of Imagination. In Viana, V. (Ed.) *Geometrias' 19. Book of Abstracts*. Porto: Aproged.

Kay, A. (1984). "Computer Software." *Scientific American* 251(3), pp. 52–59.

Khajuraho, T. & Vinayak, S. (2018). Trends in Fractal Dimension in Laxman and Kandariya Mahadev Temples. *International Journal of Applied Engineering Research*, 13, (3) pp. 1728-1741.

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

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

Pestana, P. (2011). “Lindenmeyer Systems and the Harmony of Fractals.” *Proceedings of the Chaotic Modeling and Simulation International Conference*, pp. 449–456.

Prusinkiewicz, P. & Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Berlin: Springer.

Roncoroni, U. & Crousse, V. (2016). La virtualidad aumentada: procesos emergentes, arte y medios digitales. *Artnodes*, 17.

Roncoroni, U. (2015). *Manual de diseño generativo*. Lima: Fondo Editorial de la Universidad de Lima.

Roncoroni, U. (2022). Electronic Music and Generative Remixing: Improving L-Systems Aesthetics and Algorithms. *Computer Music Journal*, 45:1, pp. 55–79.

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.

Stiny, G., & Gips, J. (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.

Varma, R. (2006). Making computer science minority friendly. *Communications of the ACM 49*(2), pp. 129-134.

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

Wyse, L. (2019). Mechanisms of artistic creativity in deep learning neural networks. In *Proceedings of the International Conference on Computational Creativity*.