Inês Caetano, António Leitão

INESC-ID/Instituto Superior Técnico, University of Lisbon

PROCESSING FOR ARCHITECTURE

The Processing language was created to simplify the learning of programming by designers and architects. Due to its pedagogical and graphical capabilities, the Processing language has spread among the electronic arts and design communities. Unfortunately, it is much less used in architecture, which relies on CAD and BIM applications that cannot be programmed using this language. To overcome this situation, we propose a solution that joins Processing with these tools. In this paper, we develop a complex architectural example using the Processing language, whose results are visible and editable in different supported CAD and BIM applications.

Keywords:

3D modeling, Processing, Generative Design, Programming.

INTRODUCTION

In the past, programming was considered a highly specialized task in the design and architectural fields. However, nowadays, many designers are already aware of the potential of programming, thus introducing it in their design practices (Burry, 2011). This increasing use of algorithms in the field of architecture allowed the automation of tedious methods, the exploration of generative processes, and the simulation of complex solutions that would be impossible to generate manually. Therefore, algorithms have become extensions of human thinking, overcoming its potential limitations by allowing the exploration and experimentation in an alternative realm (Terzidis, 2003). This algorithmic and rule-based process defines the concept of Generative Design (GD), wherein a wide variety of solutions can be created in a short period of time (Fasoulaki, 2008).

Unfortunately, programming is not trivial (Burry, 2011). If architects and designers have the necessity or interest in learning programming, they have to spend some of their time studying it. This learning process can take more or less time according to the language being studied. In order to facilitate the use of GD, some programming languages were carefully designed with the aim of teaching programming skills to designers and architects. Such is the case of the Processing language (Reas & Fry, 2007).

Processing was inspired by the Design by Numbers project (Maeda, 1999) and it was created especially for designers, although it has also been used in the architectural field. This programming language balances between simple and more advanced features, and is considered a pedagogical language. In fact, Processing was designed mainly to teach computer science and programming to designers without programming experience. With the support of an academic community, this language has grown over the years, and it is already being used in several courses due to its simplicity and to the excellent documentation available (Fricker, et al., 2008).

PROBLEM

Generative Design is extending the role of the architect to also become a programmer, thus requiring not only algorithmic, mathematical, and abstract thinking, but also programming skills. Programming is increasingly being used in architecture and, therefore, creating a methodological shift in architecture's practice by introducing an intermediate step in the architectural work. In this step, architects develop a program that reproduces and generates the idea they have in mind, instead of modelling it manually as they were accustomed to do.

CAD tool manufacturers are already aware of this methodological shift and they have adapted their tools to this emerging reality. Therefore, they have already made available the possibility to program their tools with a variety of programming languages, such as Python, Grasshopper, AutoLISP, and VisualBasic.

Processing, however, is much less used in the architectural practice, despite its attractive pedagogical capabilities. The main reasons for this unfortunate situation are (1) the inability of Processing to interact with the Computer-Aided Design (CAD) and Building Information Modeling (BIM) tools that are typically used by architects, such as AutoCAD, Rhinoceros 3D, and Revit, and (2) its shortcomings in 3D modeling operations and transformations, such as sweeping or lofting.

This is not surprising, because Processing was originally intended for 2D drawings and animations, running in its own programming environment, completely isolated from other applications. It was only recently that Processing was extended with simple 3D operations and ways of exporting the generated designs. However, these are very limited in their capabilities. As a result, although architects can easily learn and use Processing, this knowledge cannot be easily and directly applied in the field of architecture.

In order to overcome this situation, we first need to augment Processing to deal with a wider range of 3D modeling primitives (*cylinder, sphere, cone, etc.)* and transformations *(extrusion, loft, sweep, Boolean operations, etc.)* that are essential in the architectural daily practice. Secondly, we need to enable the users of Processing to generate their results directly into a CAD or BIM tool.

SOLUTION

We proposed a solution that joins CAD and BIM tools with the Processing language (Correia & Leitão, 2015) and allows architects to (1) develop new designs using this programming language and (2) generate their results directly into a CAD or BIM application. This solution was implemented in Rosetta (Lopes & Leitão, 2011), an Integrated Development Environment (IDE) for generative design. One main advantage of Rosetta is its emphasis on portability and, unlike other development environments, Rosetta supports scripts using different languages (AutoLISP, JavaScript, Python, Racket, and Scheme) and generates identical models in all supported CAD and BIM applications (AutoCAD, Rhinoceros 3D, SketchUp, Revit, and ArchiCAD).

In order to make Processing more suitable for the needs of architects, 3D modeling extensions to the Processing language were also implemented in the Rosetta IDE. These extensions include several operations for basic 3D modeling, such as boxes, spheres, cylinders, etc., as well as shape forming operations, such as lofting and sweeping.

In addition to supporting the traditional syntax and semantics of the Processing language, our solution extends Processing in three directions:

(1) Interactive evaluation, which allows designers to evaluate small fragments of Processing programs in a Read-Eval-Print-Loop (REPL), thus enabling a quick experimentation of the scripts being developed;

(2) 3D modeling, an essential extension in order to improve the use of Processing in CAD and BIM tools;

(3) Professional CAD, a connection between Processing and CAD or BIM tools, supporting the generation of designs in those tools, without suffering from the problems that typically occur when designs are imported from different applications.

EVALUATION

Our solution is intended for architects that learned the Processing language and want to use it in their architectural practice, more precisely, in the production of 3D models. Due to the large number of Processing tutorials available, it is natural to expect that architects will try to use and adapt some of the material presented in those tutorials as a starting point for more advanced modeling efforts. This situation was taken in consideration and, in this section, we explore a complex architectural example that starts from one simple trigonometry tutorial available in the Processing web site. This trigonometry tutorial explains the generation of a sine curve using Processing primitive operations running on the Processing Development Environment (PDE). Although it presents a trivial example of the use of sine curves, it only produces results aimed for a 2D visualization.

Marriott Hotel in Anaheim

It is noteworthy that there are several buildings and facades inspired by sine curves and, indeed, sine curves have large applicability in the architectural practice. Therefore, it becomes relevant to extend Processing's sine curve example to 3D modeling and, in particular, to generate a more architectonic result. In order to provide a more realistic evaluation of our solution, we developed a model based on a real building, the facade of the Marriott Hotel in Anaheim, using the Processing language running in the Rosetta IDE.

The facade of the Marriot Hotel in Anaheim is composed by several sinusoidal balconies with opposite phases. This produces an offset between sine curves, which creates a complex visual effect. The relevance of the sine curve in the development of this example becomes evident when we notice that it is needed to model almost all the facade elements, including the slabs, guards and handrails, and the vertical walls.

In this section we describe, step-by-step, the generation of the 3D model of this facade. To this end, we combined the original features of Processing with our proposed extensions, enabling a more expressive modeling approach, whose results are visible (and editable) in any of the supported CAD and BIM applications, such as Rhino5, AutoCAD, SketchUp, and Revit.

In order to generate the 3D model of the Hotel's balconies, we started by developing the sine curve in the tutorial, since it was then required for most of the following operations. After the implementation of the sine curve we proceeded with the development of the facade elements.

Balconies Slabs and Guards

First of all, we developed the slabs that compose each level of balconies. To this end, we used operations such as lines and splines to define the contours of the slabs: the lines were used to create the regular faces of the slabs, whereas the spline was used to define the sinusoidal contour. Then, we used the operation Surface to create a surface between the contours previously defined and, after that, we applied the operation Extrusion to the obtained surface in order to give a certain thickness to it. As a final result, the resulting volume of the previous operations (Fig1.a) corresponds to one slab.

The next stage was defining each balcony's handrail and guard (Fig1.b). For that we also used a sine curve defined by a spline, but this time to work as a path for a sweep operation. As a result, the sweeping of a circular section or surface along this sinusoidal path created the sinusoidal handrails.

After that, this same sine curve was used to place the vertical elements of the guards as well. More precisely, we used the curve to compute a list of points to then control the placement of these elements. In practical terms, their distribution is done along the sine curve and the distance between each element depends on the number of vertical elements of the guard. In practical terms, a higher number of elements originates smaller spacings and vice-versa. These elements were generated using the *cylinder* operation, in which the starting point parameter corresponds to the previous list of points, the height parameter depends on the handrail's height value, and the radius parameter is controlled by the size selected by us.

Facade Walls and Levels

As already mentioned, the sinusoidal movement of the balconies alternates between two phase values, 0 and ϖ, which results in a sequence of intercalated undulations along the Hotel's facade. Consequently, this alternation conditions the geometry of the division walls belonging to each level, since their definition is made according to the limits of both upper and lower slabs.

In order to generate the division-walls belonging to each level of balconies we used the *cuboid* operation, which produces an irregular parallelepiped shape. As in the previous examples, the sine curve was also necessary to control the limits of these walls and, thus, defining their shape. As these elements are connected to the upper and lower slabs, they follow two sine curves with alternated movements (Fig1.c). As a result, we defined the shape of these walls using two different sets of points, where each one corresponds to one of the sine curves.

In practical terms, the set of points of the upper curve was used to control the top geometry of the walls, and the points of the lower curve defined the shape of their bottoms. In the end, the generated walls follow the alternated movement of the corresponding upper and lower slabs (Fig.1d).

After the previous steps, we have all the elements that compose one single balcony. Now, we just need to repeat this process to create the Hotel's facade, which is composed by several balconies. For this, we implemented a loop within the method *draw*, which repeated the process multiple times until the stop condition was reached (Fig. 1 - right image).

Note that none of the 3D elements of this example could have been done using the features provided by the original Processing language because it did not provide (1) extrusions, (2) sweeps, and (3) cylinders. In addition, this model was created directly in Rhinoceros 5, but it could also be generated in any of the CAD and BIM tools supported by Rosetta.

To sum up, this example illustrates the extension and adaptation of Processing to the architectural practice, not only by enabling the use of the most required 3D operations and transformations, but also by allowing a direct interactivity with the most used CAD and BIM applications.

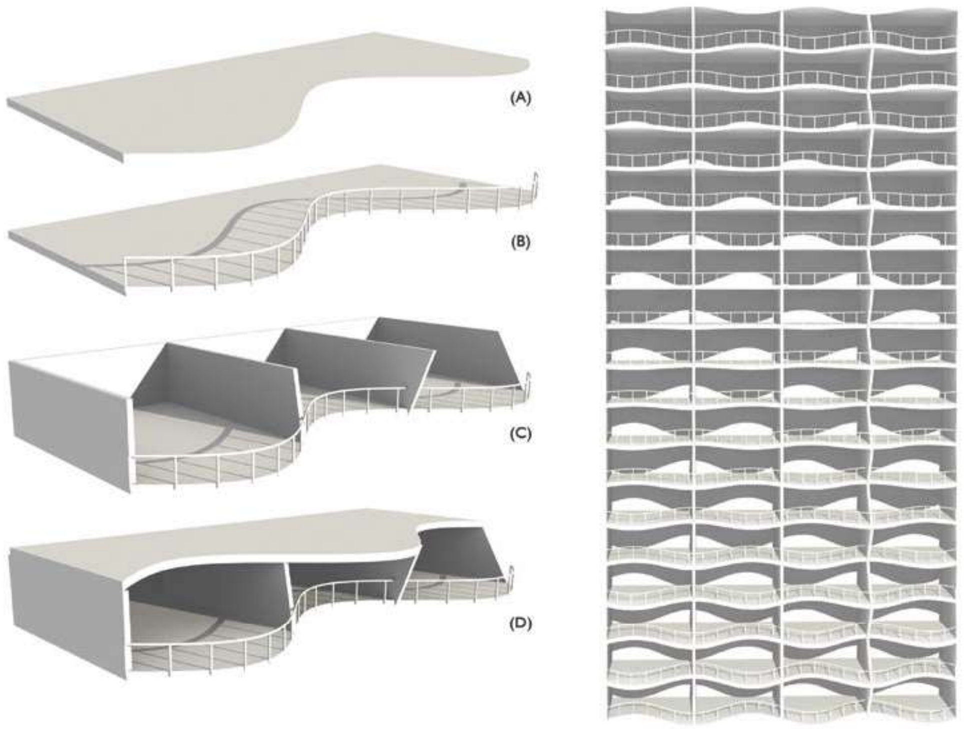

Fig. 1. On the right: An instance of the model inspired in the Marriott Hotel in Anaheim, which was totally implemented using Processing. On the left: the development stages. The sine curves defined (A) the slabs' shape, (B) the geometry of the guards, i.e. the handrails and the placement of the vertical elements (cylinders), and (C) the shape of the division walls.

Source: Authors' own work.

CONCLUSION

Generative Design has been changing the way architects work. In fact, it promoted the introduction of an intermediate step in the architects' daily practice, which corresponds to the development of a program capable of reproducing the ideas they have in mind. Therefore, in order to use GD, architects need to learn programming.

Processing is a simple and pedagogical programming language and, hence, easy to learn by designers with no previous programming experience. Therefore, it empowers the creativity and design exploration of its users. However, when it comes to the architects' work, processing shows its limitations. The main reasons for this situation are (1) its lack of 3D modeling operations and (2) the difficult combination and interaction of Processing with the most used tools in the architectural practice. Our solution overcomes these two barriers, by augmenting Processing with new design abstractions and operations, and by connecting it with several CAD and BIM tools.

In this paper, we explained how we combined the Processing language and the Rosetta IDE, and we demonstrated how the use of this combination supports the generation of more complex design solutions by using a practical example. In addition, we also demonstrated that it would be much harder to implement a similar solution in the original Processing environment.

Using our solution, it is now possible to explore 3D architectural models using the Processing programming language, since the modeling operations required are already available, and the model generation can be directed to several CAD or BIM tools that are essential for architects.

A final advantage of our solution is that it also allows architects to combine Processing with the different programming languages provided by Rosetta IDE, such as Python and Scheme. This allows Processing to move from its comfort zone - the design environment – into the more complex architectural environment.

ACKNOWLEDGMENTS

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013.

References

[1] Burry, M. (2011). Scripting Cultures: Architectural Design and Programming. . UK: John Wiley & Sons Ltd Publication.

[2] Correia, H., & Leitão, A. (2015). Extending Processing to CAD Applications. Real Time: Extending the Reach of Computation, Volume 1. Vienna, Austria: 33th eCAADe Conference.

[3] Fasoulaki, E. (2008). Integrated Design: A Generative Multi-Performative Design Approach. USA: Massachusetts Institute of Technology (MIT).

[4] Fricker, P., Wartmann, C., & Hovestadt, L. (2008). Processing: Programming Instead of Drawing. Architecture in Computro. Antwerpen, Belgium: 26th eCAADe Proceedings.

[5] Lopes, J., & Leitão, A. (2011). Portable Generative Design for CAD Applications. . ACADIA 11: Integration Through Computation. Banff, Alberta: Proceedings of the 31st anual Conference of the Association for Computer-Aided Design in Architecture (ACADIA).

[6] Maeda, J. (1999). Design by Numbers. Cambrige, MA, USA: MIT Press.

[7] Reas, C., & Fry, B. (2007). Processing: A Programming Handobook for Visual Designers and Artists. Massachusetts Institute of Technology, USA: MIT Press.

[8] Terzidis, K. (2003). Expressive Form: A Conceptual APproach to Computational Design. New York: Spon Press.