

Using Processing with Architectural 3D Modelling

Inês Caetano¹, António Leitão²

^{1,2}INESC-ID/Instituto Superior Técnico

¹ines.caetano@tecnico.ulisboa.pt

²antonio.menezes.leitao@ist.utl.pt

Although programming was considered a specialized task in the past, we have been witnessing an increasing use of algorithms in the architectural field, which has opened up a wide range of new design possibilities. This was possible in part due to programming languages that were designed to be easy to learn and use by designers and architects, such as Processing. Processing is widely used for academic purposes, whereas in the architectural practice it is not as used as other programming languages due to its limitations for 3D modeling. In this paper, we describe the use of an extended Processing implementation to generate three 3D models inspired in existing case studies, which can be visualized and edited in different CAD and BIM applications.

Keywords: *Generative design, Programming, Processing, 3D modeling*

INTRODUCTION

Only recently have architects considered the use of programming in architecture, as they become aware of its potential, thus introducing it in their design practices (Burry 2011). This allowed the automation of tedious tasks, the exploration of generative processes, and the generation of complex solutions that would be difficult and time consuming to produce manually. Therefore, algorithms became extensions of human thinking by overcoming its potential limitations, allowing the exploration and experimentation in an alternative realm (Terzidis 2003). The concept of Generative Design (GD) can be defined by this algorithmic and rule-based process, through which a wide variety of solutions can be created in a short period of time (Fasoulaki 2008).

However, programming is not a trivial task (Burry 2011). In order to facilitate the use of GD, some programming languages were carefully designed or adapted with the aim of teaching programming skills

to designers and architects. Such is the case of the Processing language (Reas and Fry 2007).

BACKGROUND

Processing was created especially for designers without any previous experience in programming. Nevertheless, it has also been used in other fields, including architecture. This programming language is considered as a pedagogical language and, in fact, it has been taught in several academic courses. Therefore, this language has grown over the years and has become increasingly popular due to its simplicity, to the academic community support, and to the excellent documentation available (Fricker et al. 2008).

Still, in the architectural practice Processing is not as used as other programming languages, such as Python, Grasshopper, AutoLISP, and VisualBasic. The main reasons for this unfortunate situation are the inability of Processing to interact with the Computer-Aided Design (CAD) and Building Informa-

tion Modeling (BIM) tools that are typically used by architects, such as AutoCAD, Rhinoceros 3D, and Revit, and also its shortcomings in 3D modeling operations and transformations, such as sweeping, lofting or extruding. This is not surprising, as Processing was originally intended for 2D drawings and animations, running in its own programming environment and completely isolated from other applications.

Only recently was Processing extended with simple 3D operations and ways of exporting the generated designs. Some libraries were also developed to add some of the required operations, such as increasing the range of 3D primitives, creating and controlling irregular shapes, or dealing with some 3D shape transformations. In the Related Work section we will enumerate and describe some of these libraries with more detail.

Unfortunately, these 3D operations are still limited in their capabilities. Therefore, the application of Processing in the field of architecture remains difficult, even though it is easy to learn and use. In order to overcome this situation, we propose an augmented Processing for architects that deals with a wider range of 3D modeling primitives (cylinders, spheres, cones, etc) and transformations (extrusions, lofts, sweeps, Boolean operations, etc), which are essential in the architectural daily practice, and generates their results directly into a CAD or BIM tool.

AUGMENTING PROCESSING

In the past (Correia and Leitão 2015), we proposed a solution to join CAD and BIM tools with the Processing language, allowing architects to develop new designs using this programming language while generating their results directly into a CAD or BIM application. This solution was implemented in Rosetta (Lopes and 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 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, cones, etc., as well as shape forming operations, such as lofting, sweeping and extruding, and also Boolean operations, i.e. subtraction, intersection, and union of shapes.

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

1. Interactive evaluation, which allows designers to evaluate small fragments of Processing programs in a Read-Eval-Print-Loop (REPL), enabling quick experimentation of the scripts being developed;
2. 3D modeling, an essential extension in order to improve the use of Processing for architecture;
3. Professional CAD, a connection between Processing and CAD/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.

The most significant advantage of the implementation of Processing for Rosetta is the ability to use, in Processing programs, all the 2D and 3D modeling operations available in Rosetta. Given that these operations access the corresponding operations in the CAD/BIM tool being used, this gives Processing the capability to directly create shapes in that tool.

USING PROCESSING IN ARCHITECTURE

Our implementation of Processing is intended for architects that learned Processing and want to use it in their architectural practice. In this section, we develop three architectural examples using Processing:

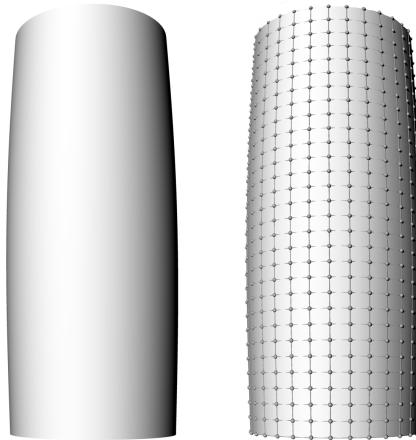
1. Al Bahar Towers;
2. Allianz Arena;
3. Quality Hotel Friends.

Additionally, we discuss the advantages and drawbacks of Processing for the architecture practice. For each of the examples we use Processing running in the Rosetta IDE, thereby allowing the visualization and edition of the generated models in the supported backend applications, including Rhino and AutoCAD.

Al Bahar Towers

Our first example is the Al Bahar Towers in Abu Dhabi, design by Aedas Architects. These towers are characterized by their responsive facade, which is composed by several triangular units inspired in the traditional Islamic element "mashrabiya". Nevertheless, in this paper we will simply focus on modeling these towers facade design, and not on its kinetic properties.

Figure 1
Inner glass surface
(left-side); outer
surface organized
in groups of four
points (right-side).



We divided this example into two different stages. Firstly, we created the inner glass surface with the tower's shape and, then, we explored the outer skin, which is composed by the triangular units. Before exploring each stage, we had to define the set of points that corresponded to the tower's geometry, which together with a surface creation operation produced the inner glass surface (Figure 1, on the left). More-

over, these points were also used to generate the surface mesh of the outer layer, which was then organized into groups of four points (i.e. a squared mesh) so as to facilitate the future development of the triangular units (Figure 1, on the right). In practical terms, this first stage required operations like calculus of matrices and surface normal vectors, arrays of coordinates, and surface creation.

The second stage was the creation of the tower's outer skin with the triangular units. First of all, we defined the geometry of these units and, then, we applied them on the squared mesh defined in the previous stage.

To create the units, we started by calculating the vertices that defined their shape. Then, these vertices were strategically linked using lines, which in turn were used to produce the surfaces. This process is synthesized in Figure 2 by the images A-B-C.

Secondly, the generation of the outer skin consisted in mapping these triangular units along the squared mesh. Finally, we overlapped both inner and outer surfaces so as to produce the final model, which is visible in Figure 2.

Note that, apart from the arrays of coordinates, this second stage required surface creation between curves, an operation that is not available in the original Processing implementation.

Allianz Arena

Our second example is the Allianz Arena stadium in Munich, designed by Herzog & de Meuron Architects with ArupSport. As in the previous example, we had to first define the overall shape of the stadium. Therefore, in order to obtain this form we used the mathematical formula of the superellipse to produce the cloud of points that corresponded to the stadium's shape. This was then used to produce, first, the underlying surface of the model and, then, to distribute the diamond shaped cells which characterize the facade. Lastly, we produced each of these cells using:

1. an array of coordinates and a closed-line defining the diamond-shaped curve of each cell;

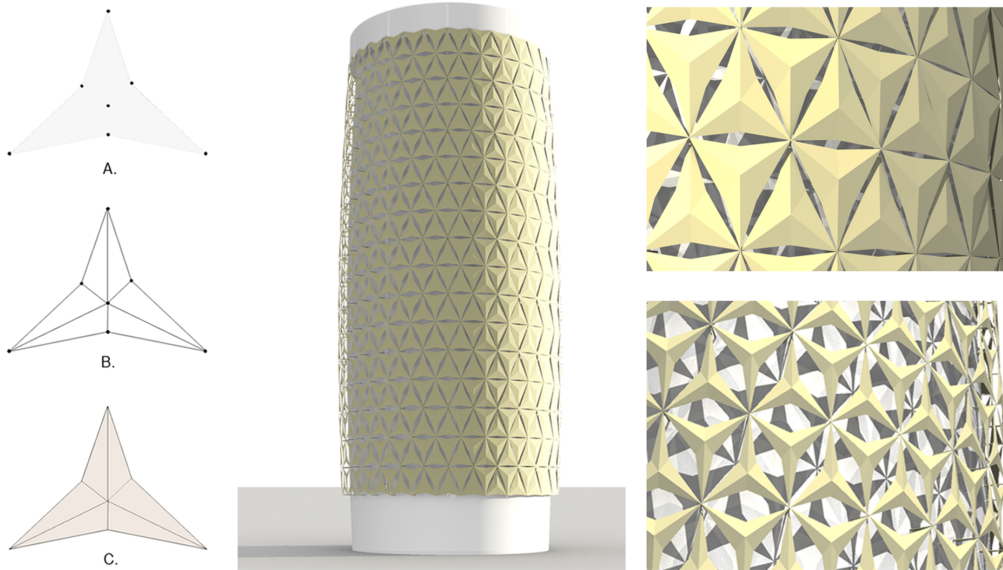


Figure 2
The AI Bahar Tower 3D model using Processing. On the left – the units generation process: A. the main vertices that define each unit shape; B. lines to link the main vertices; C. the creation of surfaces between lines; In the middle - One instance of the generated model; On the right – two shape variations of its facade.

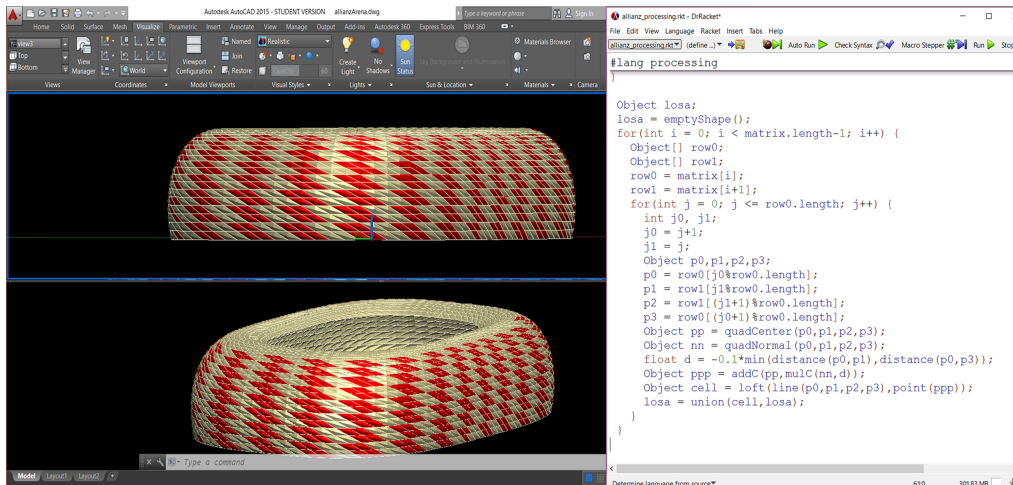


Figure 3
Allianz Arena model: a print screen of the generated model, visualized in AutoCAD, and the Rosetta IDE with the corresponding Processing code.

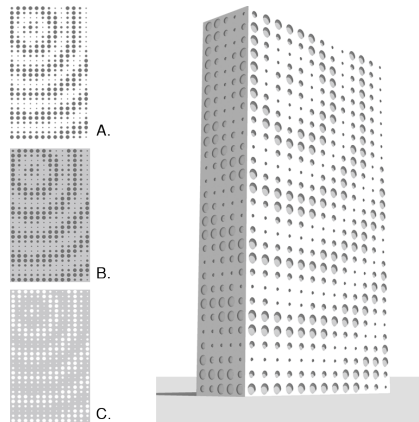
2. a loft operation between this curve and a point placed perpendicularly to it.

To sum up, not only did this model use matrices and vector calculations, it also required the loft operation to generate the stadium envelope pattern. Figure 3 shows an instance of the generated model in AutoCAD and the corresponding Processing code.

Quality Hotel Friends

Quality Hotel Friends, in Stockholm, is our last example, which was designed by Karolina Keyzer+Wingårdhs. This hotel has a straight facade and it is composed by several circular windows of three different sizes. Note that these three sized windows are strategically placed to create the wave effect visible in Figure 4.

Figure 4
Quality Hotel
Friends model: A.
The creation of the
cylinders with
different sizes; B.
overlapping of the
cylinders with the
facade wall; C.
subtraction of the
cylinders from the
wall.



In practical terms, we generated this example in two stages. Firstly, we created the hotel walls with a box operation and, then, we produced the round windows. For this, we had to produce the matrix of points that shaped the facade, and then we organized them into arrays of coordinates. These arrays were then used to distribute the cylinders over the walls, and to control each cylinder radius size. More precisely, each cylinder radius size was controlled by the distance between its positioning point and the attractor point. After generating all the cylinders,

they were subtracted from the walls previously created, therefore creating the windows openings (Figure 4, A-B-C).

Summarily, the modeling of this building required 3D primitives, such as boxes and cylinders, to produce the walls and the circular windows, required both matrices of points and vector calculations to create and place the circular windows, and, finally, the Boolean operation Subtraction to create the windows openings. Figure 4 synthesizes this process and shows one instance of the obtained model using Processing.

Evaluation



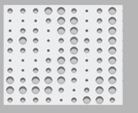


In this section, we summarize the generation process of all three examples and we demonstrate that the modeling extensions added by our solution were crucial in the exploration of these models.

First of all, we synthesized the operations that were necessary for each example and, then, we compared the availability of those operations in (1) the Processing language and Processing Development Environment (PDE), and (2) in our implementation of Processing in Rosetta.

Although our sampling is limited, it revealed the range of operations that are typically used in 3D modeling. After analyzing the information presented in Figure 5, we can conclude that the majority of the operations that are essential for the generation of the previous examples, are not available in the original Processing implementation. Moreover, we predict that models that are more complex than the ones presented in this paper will require even more advanced 3D modeling operations. It is thus important to fully support in Processing the operations that are available in the CAD/BIM tool being used and that is precisely one of the important features of our Processing implementation in Rosetta.

PORTABILITY

Portability is the ability of a program to be compiled or run in a different environment and, in our case, it allows us to produce identical models in different

	EXAMPLE			TOOL	
PRE-DEFINED OPERATIONS					
Matrix of points	●	●	●	✓	✓
Normal vector calculation	●	●	●	✗	✓
Array of coordinates	●	●	●	✓	✓
Superellipse shape		●		✗	✓
Surface creation	●	●		✗	✓
Surface from curves	●			✗	✓
3D Transformations:					
Sweep	●			✗	✓
Loft		●		✗	✓
3D Primitives:					
Cylinder	●		●	✗	✓
Box			●	✓	✓
Boolean Operations:					
Subtraction			●	✗	✓
Union	●	●	●	✗	✓
Intersection	●			✗	✓

● Needed Operation
✓ Available Operation
✗ Unavailable Operation

Figure 5
Synthesis of the needed operations for each example, and their availability in both Processing PDE and in our implementation of processing to Rosetta IDE.

CAD tools, such as Rhino, AutoCAD, or SketchUp, among others. Using our solution, it is now possible to (1) explore 3D architectural models using the Processing programming language, since the modeling operations required are already available, and (2) to visualize the obtained models in the different CAD or BIM tools that are essential for architects. Moreover, it allows the designer to easily change the CAD tool that he wants to use (Figure 6).

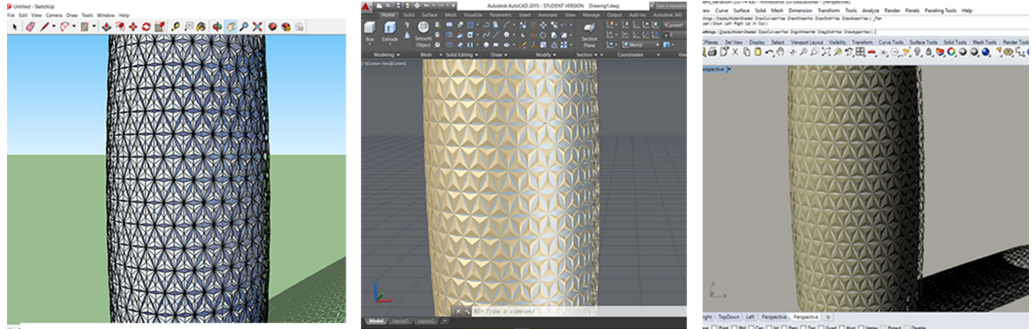
Finally, Rosetta also promotes portability across the supported programming languages, allowing

the combination of Processing with the other supported programming languages such as Autolisp, Python, Racket and Javascript.

RELATED WORK

There are already some libraries that improved Processing to deal with more 3D modeling extensions. Shapes 3D [6] is a library that extends processing with a set of 3D shapes, including ellipsoids, toroids, helices, and others. iGeo [5] is a 3D modeling software library in Java, which has an interface special-

Figure 6
The Al Bahar tower's
model using
different CAD
applications
(SketchUP,
AutoCAD and
Rhino5).



ized for processing called piGeon. This library includes vector operations, NURBS curve and surface geometries, polygon meshes, and 3D model file I/O. ANAR+ is a geometry library for Processing (Labelle et al. 2010) that was intended to be a programming interface supporting shape exploration based on parametric variations. Toxiclibs [4] is an independent library collection for computational design tasks with Java and Processing, which supports 2D/3D vectors, spline curves, 4x4 matrices, intersection tests, mesh container, and OBJ and STL exporters. Finally, ComputationalGeometry library [3] allows Processing to deal with dynamic mesh generation and rendering, including isometric contours and surfaces, boundary hulls and skeletons.

One of the problems with these libraries is that none of them allows architects to directly interact with the most used CAD/BIM applications. Therefore, they have to resort to import procedures which, frequently, lose some of the geometric information or transform it into undesirable forms (e.g. converting smooth surfaces into meshes). In addition, some 3D operations are still missing, including Boolean operations and shape transformation operations. Our solution, besides extending processing with 3D modeling operations and primitives, also allows the interaction between Processing and the CAD and BIM applications typically used by architects.

Dynamo [1] and Grasshopper [2] are programming languages that are also quite popular among

the architecture community, mainly for those who do not have any programming experience. Both have the advantage of interacting with at least one of the CAD or BIM tools that are used in architectural practice.

Grasshopper is a visual programming language for Rhinoceros. This language is widely used by beginners as it allows them to quickly develop, test and visualise small programs due to the friendly and intuitive environment. Recently, it was also extended to interact with Revit and ArchiCAD via independent plug-ins. Unfortunately, Grasshopper has scalability problems: the more complex the user program gets, the more difficult it becomes to understand and maintain it. The use of clusters can mitigate this problem but, in practice, it is rare to see programs that take advantage of the idea. The plug-ins for interacting with other tools besides Rhinoceros are very experimental and not yet widely used. For the purposes of this work, the major drawback of Grasshopper is that, although it also supports some textual programming languages, the list does not include Processing.

Similarly, Dynamo is a plug-in for Revit strongly influenced by visual programming languages. It is also based on a workflow of nodes and connections that creates BIM objects in Revit. Dynamo share the same advantages of Grasshopper, but it also suffers from the same limitations, including the scalability problem and the fact that it does not support the Processing language.

Comparing to Rosetta IDE, Grasshopper and Dynamo might be more intuitive and user-friendly, particularly for those without programming experience. Nevertheless, to overcome their limitations when developing more complex models, architects are forced to use textual programming languages. As a consequence, architects end up dealing with textual languages even when they were not supposed to. Moreover, programs that combine both visual and textual programming languages tend to be less organized and more difficult to understand.

On the other hand, although Rosetta requires a more profound initial investment in learning the Processing language, it allows the development of programs with higher complexity and more levels of abstraction.

CONCLUSION

Processing is a simple and pedagogical programming language and, hence, easy to learn by designers with no previous programming experience. However, when it comes to the architects' work, Processing shows its limitations:

1. Lack of 3D modeling operations;
2. 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. The examples developed in the paper illustrate the extension and adaptation of Processing to the architectural practice, not only by enabling the use of the most needed 3D operations and transformations, but also by allowing a direct connection with the most used CAD and BIM applications.

It is noteworthy that the majority of the 3D primitives and operations used to develop the examples presented in this paper could not have been done using exclusively the features provided by the original Processing language, as it does not provide lofts, sweeps, cylinders, and other crucial operations for ar-

chitectural modelling.

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

ACKNOWLEDGEMENTS

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

REFERENCES

- Burry, M 2011, *Scripting Cultures: Architectural Design and Programming*, John Wiley & Sons, Ltd, Publication, U.K.
- Correia, H and Leitão, A 2015 'Extending Processing to CAD Applications', *Real Time: Extending the Reach of Computation - Proceedings of the 33th eCAADe*, Vienna, Austria
- Fasoulaki, E 2008, *Integrated Design: A Generative Multi-Performative Design Approach*, Ph.D. Thesis, Massachusetts Institute of Technology (MIT)
- Fricker, P, Wartmann, C and Hovestadt, L 2008 'Processing: Programming Instead of Drawing', *Architecture in Computro - Proceedings of the 26th eCAADe*, Antwerpen (Belgium)
- Labelle, G, Nembrini, J and Huang, J 2010 'Geometric programming framework, ANAR+: geometry library for processing', *Future Cities - proceedings of the 28th eCAADe*, ETH Zurich, Zurich, Switzerland
- Lopes, J and Leitão, A 2011 'Portable Generative Design for CAD Applications', *ACADIA 11: Integration Through Computation - Proceedings of the 31st ACADIA*, Banff, Alberta
- Reas, C and Fry, B 2007, *Processing: A Programming Handbook for Visual Designers and Artists*, MIT Press, MIT, USA
- Terzidis, K 2003, *Expressive Form: A Conceptual Approach to Computational Design*, Spon Press, New York
- [1] <http://www.dynamoprimer.com>
 - [2] <http://www.grasshopper3d.com/>
 - [3] <http://thecloudlab.org/processing/library.html>
 - [4] <http://toxiclibs.org/>
 - [5] <http://igeo.jp/p/>
 - [6] <http://www.lagers.org.uk/s3d4p/index.html>