# SOURCE STUDIO
## Teaching programming for Architectural Design

JULIEN NEMBRINI, GUILLAUME LABELLE, NATHANIEL ZUELZKE, MARK MEAGHER, AND JEFFREY HUANG
*Media & design Laboratory, Ecole Polytechnique Federale de Lausanne, Switzerland*

**KEYWORDS:** programming, studio teaching, scripting, parametric design

The architectural studio framework presented here is based on the use of programming as central form generation reflexive medium (Schon, 1983). Its aim is to teach architectural design while introducing a different approach toward computer tools by enabling students to fully explore variations in their designs through the use of coding for form definition. It proposes the students to reflect on their design process through its confrontation to algorithmic formalization (Mitchell 1990). This results in exercising the synthetic re-thinking of their initial sketch intents to comply with the difficult task of fitting the language syntax. With the proliferation and constant replacement of computer tools among the architectural practice, a shift appears in the attitude towards introducing students to different tools: studio teaching is branded by specific software platforms advocated by the teaching team. A lack of generalized view, independent of commercial CAD software, is problematic for the definition of new teaching tools suited for this constantly evolving situation (Terzidis, 2006).

As a response, the prior intention of the presented studio is to confront the students with the direct use of coding to define their form. This intention builds on a newly developed programming interface for the definition of geometrical form, the ANAR library. This library is built on a similar programming for non programmer's attitude as the open-source framework 'processing.org', initially developed to introduce programming for creative production (Reas and Fry 2007). The library offers a basic 3- dimensional environment implementing a set of low-level elementary functions such as linear transformations, local reference coordinates (Logo Turtles) or curves. In the ANAR framework, geo-

T. Tidafi and T. Dorta (eds)
Joining Languages, Cultures and Visions: CAADFutures 2009

metrical form is only defined through algorithmic instructions. A novel element is related to another through inheritance relationships and every change to parent elements is propagated to descendants. This construction may be complex: by reusing the same values through the expression of relations, parameters are kept to a small limited set of numerical values. These primitive values are implicitly extracted and displayed in the form of sliders that can be dynamically and interactively changed to understand their effect on the overall form.

The proposed architectural typology proposed for this studio – the skyscraper - is intentionally abstract; it suggests an abstract repetition of geometrical elements structured by juxtaposition. The students are encouraged to adopt a bottom-up approach; defining a dynamic module as starting point. An exploration of potential variation of the geometrical construction with a combination of modules leads from abstract compositions to site specific settings. Exploiting parametric inheritance assesses the variational potential of the form through an interactive exploration of the form space. Programming skills teaching is based on concrete architectural examples.

Key concepts such as types, variables, arrays, flow control or functions are contextualized to the field of architecture rather than abstract as commonly presented in computer science. Students are thus stimulated to discover on their own the programming concepts that trigger their interest. They are encouraged to build and share their own tools and explore by copying or extending existing code. Exposed to programming, students develop a critical view on what software is made of, in light of the novel importance of scripting techniques in contemporary architecture, visible among others in the developments of 'non-standard' architectures (Oxman 2008). To this end, the prior assumption to keep away from commonly used interfaces and explore the potential of syntax alone proves to be efficient in channeling innovative designs, despite the overarching challenge to learn a new language to describe architectural form while producing meaningful architectural design. It does not result in creativity impairment, but instead in a shift in design technique providing an alternative point of view that should nowadays be part of the student curriculum.

## REFERENCES

Oxman R "Digital Architecture as a challenge for design pedagogy: theory, knowledge, models and medium", Design Studies, vol.29, nb. 2, pp. 99-120, 2008.

Reas C, Fry B, "Processing: A Programming Handbook for VisualDesigners and Artists", The MIT Press, ISBN-13: 978-0262182621, 2007.

Schön D, "The reflexive practioner [How Professionals think in action]".New York, Basic Books, 1983.

Terzidis K, Algorithmic Architecture, Architectural Press, Elsevier, OxfordUK, 2006

Mitchell WJ, "The Logic of Architecture [Design, Computation, AndCognition]", ED TheMIT Press, Cambridge, Massachusetts, 1990.