VISUAL PROGRAMMING IN ARCHITECTURE
Should Architects be trained as programmers?

STEFAN BOEYKENS AND HERMAN NEUCKERMANS
K.U.Leuven Dept. of Architecture, Urbanism and Planning (Belgium)

KEYWORDS: programming, CAAD, design process, interface, curriculum

OVERVIEW

Most commercial CAAD applications have online communities of end users, through blogs, discussion forums and learning portals. On many occasions, questions lead to the conclusion that the software can only provide the desired functionality using scripting or programming, to fill in the gaps of the software feature set. However, more than once, people suggest that architects are not programmers and that they should not be responsible for adding this functionality. Similarly, when guiding students through some of their technical design studio problems, a comparable reaction is noticed. This is especially remarkable, considering the fact that these students had prior programming courses, as part of their Bachelor of Architecture curriculum.

There are many different advantages of embedding programming skills into the curriculum of the architect. Burry (1997) witnessed the influence of programming on the design studio work. While the examples from the article might be dated, the relation between programming and design studio assignments is still non-existent in many schools, nowadays. A notable counter-example can be read in (Fricker et al, 2008), describing the use of Processing (Reas and Fry 2007) as programming platform within the architectural courses.

Programming should be part of the main architectural skills, albeit not necessarily in the sense of writing code. The design brief for a building project is commonly described as a series of constraints, parameters and a listing of the desired functionality. While the translation of these constraints into a design usually occurs mentally, there are benefits to transfer this process more explicitly, through the application of scripts or parametric systems, directly in the early
stages of the design.

The problem becomes more complex as the solution under development will have to be continuously adapted. The programmatic design process is not linear, but rather evolves through a cyclic process of modifications to the solution, not unlike the architectural design process. The last few years, however, have displayed a growing number of design tools where the creation of procedural systems can be performed more visually. Examples include Bentley Generative Components, as discussed in (Aish 2005) and Grasshopper for McNeel Rhino3D. The elaborate writing of code is wholly or partly replaced with the visual metaphor of connecting small blocks of independent functionality into a whole system or procedure. Through the application of Graphical User Interface items or widgets, designers can even add interaction to their design, allowing exploration of different alternatives through the modification of parameters controlled by sliders, check boxes or other widgets. Their visual approach assists in creating a functional solution, while at the same time being less dependent on strict syntax and more easily adaptable during the development.

Looking at the individual aspects of programming within CAAD and 3D applications, all the pieces of the puzzle seem to be available, in the form of powerful modeling kernels, implementations of diagram-like interfaces to manage programmed logic, BIM functionality and extensive architectural libraries. However, right now, they seem to evolve almost independently, applied in different applications.

**REFERENCE**


