Teaching Computational Thinking with Processing

Why is it important for architects to be able to think like computer scientists? At first glance, the two fields might seem unrelated. After all, buildings don't appear to have much in common with software. However, some educational researchers argue that the concepts underlying computer science— as well as the problem-solving and design methods computer scientists use—are fundamental knowledge that every professional should learn. This idea is known as **computational thinking**[1].

An example of computational thinking comes from the field of biology – one of many fields transformed through the integration of algorithmic processes. Today, a biologist might study mutation rates by creating an algorithm to statistically analyze a large dataset of DNA. While computer science is not the biologist's profession, it is part of her toolkit, and its approaches help shape her thinking. The same skills and outlook can benefit architects. Procedural and information-based methods promise to shape the future of design, but only if architectural educators can find a way to introduce computational thinking.

This poster describes projects from a pilot course at the University of Michigan with the objective of teaching students computational thinking. The course used the programming language Processing as its primary form of representation. Processing is a general, flexible language intended for artists and designers, and so it is particularly well suited as a first introduction to programming with no prerequisites[2]. Students used the language to explore computer science topics using visual media. For example, a unit on simulation involved learning about the principal of type inheritance: using a general type to generate specific, relational subtypes. A student might investigate this idea by modeling something simple (like a school of fish), but it is a concept also found in architectural precedents and building information modeling (BIM).

The goal of the course is not to learn Processing or to generate form, but rather to learn how to *think like a programmer*. Understanding topics like iteration and problem decomposition are necessary in order to write programs, but these are also general design concepts that predate and exist apart from computation. Thus, computational thinking can help architects use computers better, as well as affect them even when they are not in front of one.

There is evidence that this occurred with the pilot course. In later semesters, some students reported that working with Processing helped them appreciate and utilize their conventional software better. Others mentioned that it made it easier for them to advance in other computing courses such as scripting or physical computing. Still others used what they learned to generate more sophisticated ideas within their studio projects. This suggests that introducing computational thinking early in the curriculum, in a course similar to the pilot, can be beneficial. Tools and techniques may change, but the fundamentals of computer science – which directly relate to design and computing – remain constant.

[1] http://www.cs.cmu.edu/~CompThink/papers/Wing06.pdf
[2] For example, in order to script in Grasshopper, you need to know how to model in Rhino.