Teaching Computational Thinking with Processing

Nicholas Senske
Assistant Professor, Univeristy of North Carolina, Charlotte
PhD. Candidate, University of Michigan, Ann Arbor

Summary

There is a developing trend in architecture towards computational design practices. Research shows that the most difficult part of learning to use these unfamiliar tools and methods is not the interfaces involved, but rather the mindset—understanding how to design processes and systems[1]. Computational thinking[2] is the idea that other fields can learn this mindset by drawing upon principles and concepts from computer science.

In the fall of 2009, the University of Michigan at Ann Arbor introduced an experimental digital media course, ARCH 506, with the objective of teaching students computational thinking.

Using the Processing programming language, students learned computer science principles and methods which could serve as a foundation for a critical engagement with advanced architectural topics such as parametric systems, simulation, generative scripting, physical computing, and others

Method

The goal of ARCH 506 is not to learn programming, but rather to learn how to think like a programmer.

However, teaching higher-level thinking skills is a challenge. Students tend to encode material in a rote way, copying examples in their design work without much thought as to how or why they function. To inhibit this, students were led through a series of programming "sketches": rigorous, small-scale exercises.

The objective of these sketches was not design, but rather to explore and abstract computational principles. By engaging with the same principles across multiple contexts, students were more mindful of the processes involved and less likely to merely reproduce patterns from the demonstration material.

What is Processing?

Processing is a programming language, developed at the MIT Media Lab, which is intended for artists and designers. While it is used by many professionals, it was initially designed as a pedagogical tool, combining visual and computational concepts in a simple, yet powerful programming syntax. The idea of prototyping and/or thinking through code as part of one's design process is built into the idioms of the software. Processing files are literally called "sketches".

Unlike scripting, Processing requires no prerequisite knowledge and interoperates with a variety of different media. It is a full programming language and so it is a useful platform for learning computer science principles in a way which should easily transfer to other programming languages and computational tools.

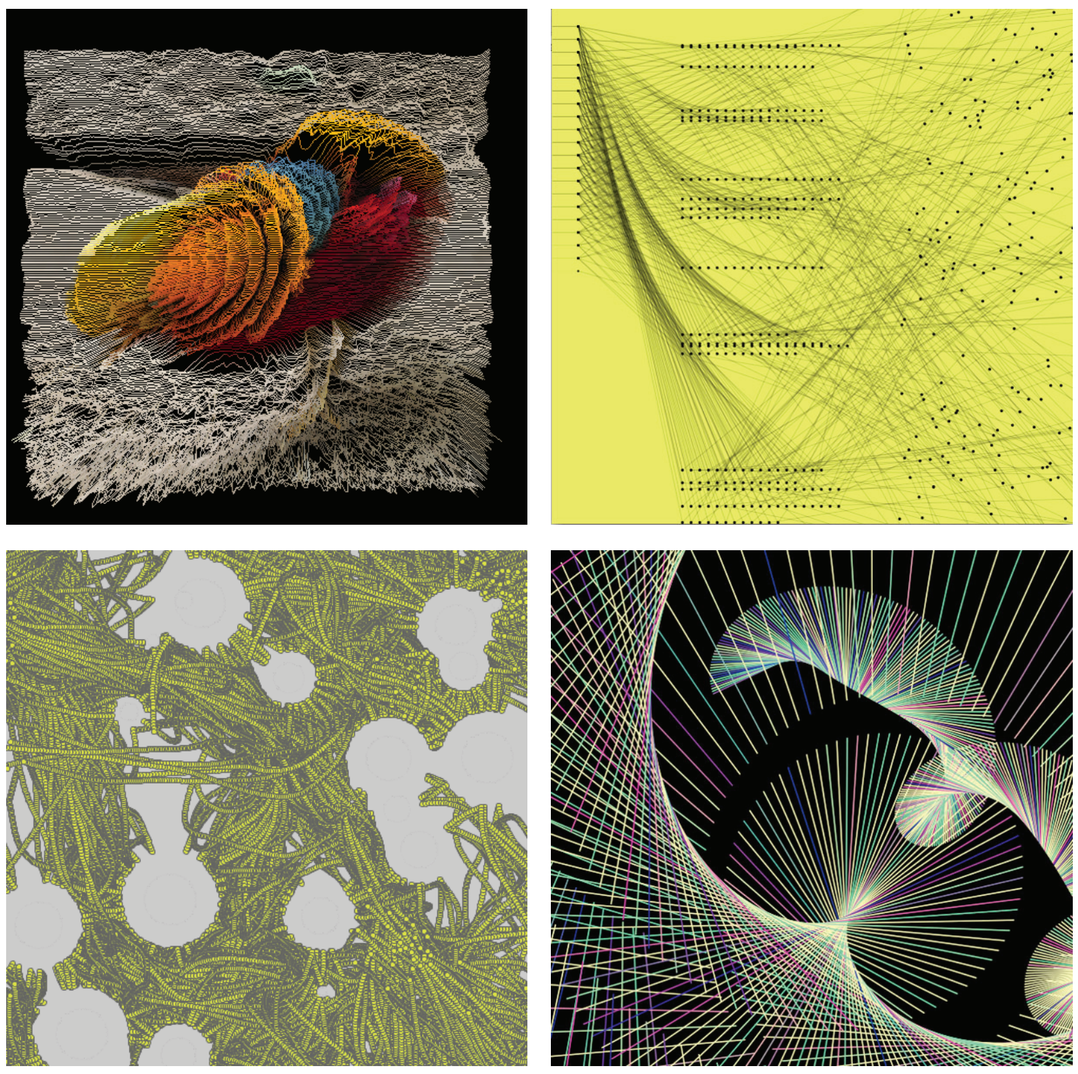
Results

There is evidence that the course succeeded in its objective. In later semesters, many students reported that ARCH 506 helped them appreciate and utilize their conventional software better. Others mentioned that it made it easier for them to advance in computation 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, as a foundation course similar to ARCH 506, can be beneficial. Tools and techniques may change, but the fundamentals of computer science – which directly relate to design and computing – remain constant.

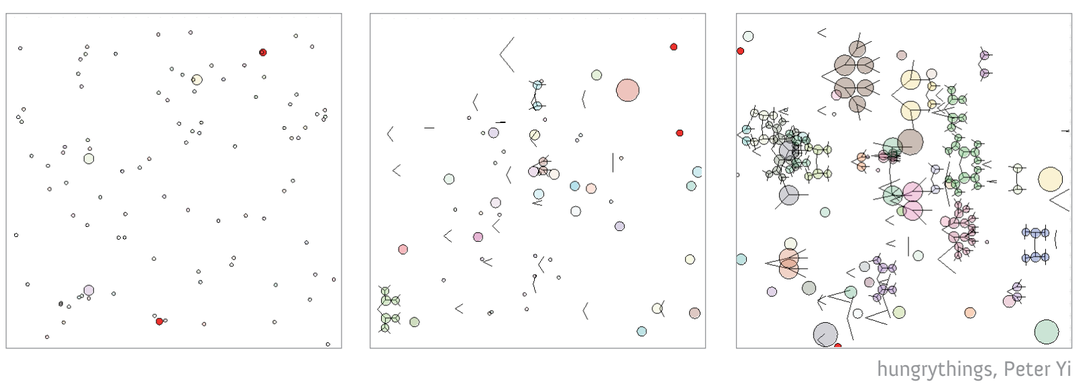
1. Sheil, B. A. (1983). "Coping With Complexity." Information Technology & People 1(4): 295 - 320.
2. http://www.cs.cmu.edu/~CompThink/papers/Wing06.pdf

Why is it important for architects to be able to think like computer scientists?

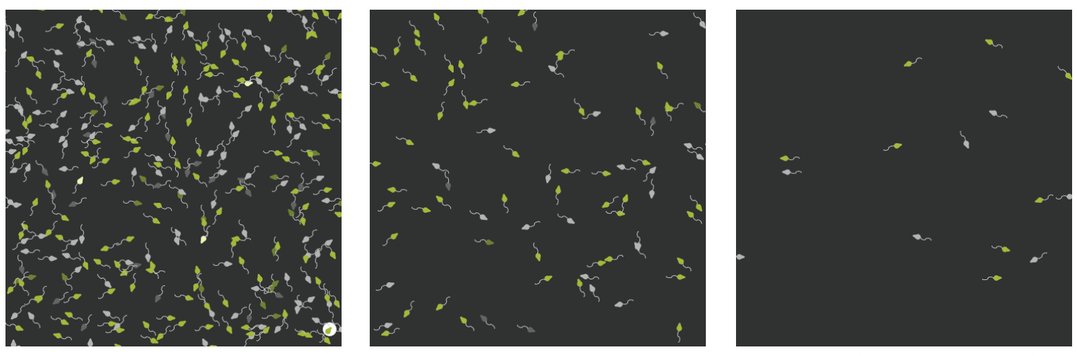


Clockwise, from top-left: Diana Tomova, Vaughn Calandra, Drew Gingrich

The drawing machines unit introduces students to the fundamentals of algorithmic thinking. This includes knowledge of concepts such as variables, iteration, and program flow but also strategies for planning, problem decomposition, debugging, and reuse. These ideas are not only essential to writing effective and thoughtful programs, they apply to design in general, as well.



hungrythings, Peter Yi



Sperm Wars, Jamie Cobb

The simulations unit teaches students the basic concepts of modeling and representing systems: data structures, object reference and inheritance, and separation of data and presentation (among others). Students learn not only how to create complex behavior from simple rules, but also how computer scientists observe, classify, and apply such behaviors to study and solve problems.