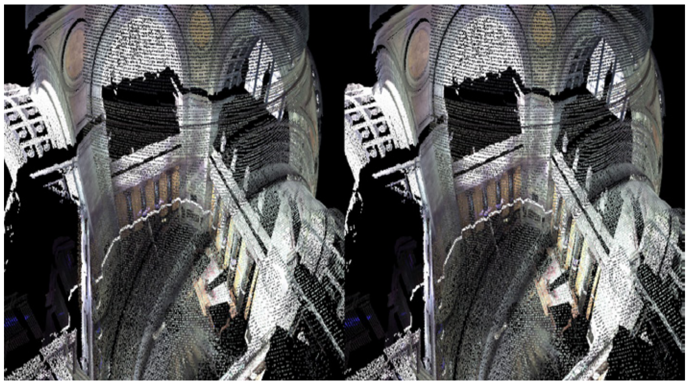
Columbia University GSAPP
Visual Studies Sequence A
Instructors Toru Hasegawa & Mark Collins

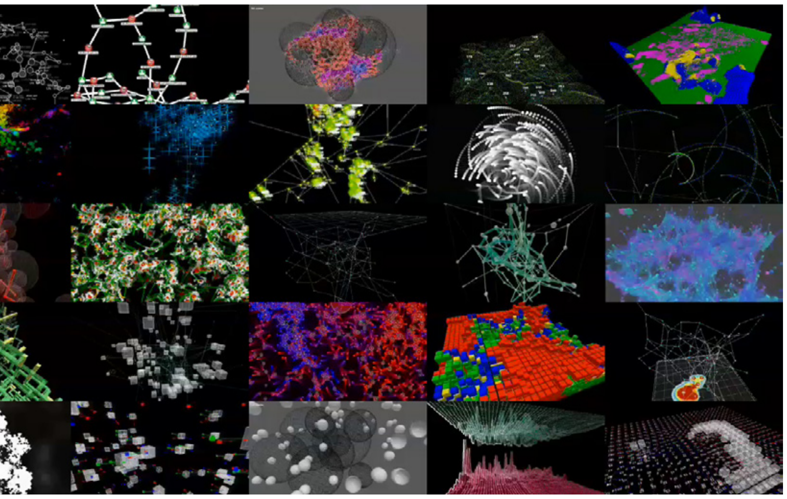
Search: Advanced Algorithmic Design

Architects work with computers every day but rarely achieve a true synthesis of machine automation and human design thinking. This workshop promotes simple, visual programming as a vehicle for designers to take their ideas to the next level by collaborating with the computer at its most potent level: code.

We teach programming through a series of workshops, competitions and project work. We work in Processing, an open source programming environment created for visual thinkers working in two and three dimensional media. No programming experience is required.


Stereographic (3D) point cloud of Low Library, Search:AAD Spring 2010

Design computation encompasses a broad range of approaches and techniques. This course examines fundamental concepts as well implementation. Using object oriented programming and Processing as a starting point, the seminar explores the practice and promise of these new means of design conception through a process of experiment, play and rewiring. These workshops support a focused research led by the student in the exploration of an emerging topic or technology in design computing. The results of the class are visualizations that reveal the hidden algorithmic logic and generative potential at work inside each code.


Mosaic of Search:AAD student work.

Course Goals and Methodology

This workshop will explore generative design methodologies through the application of algorithmic techniques. We will be looking at fundamental coding principles (recursion, feedback, modularity and I/O) while working within an object-oriented framework, opening the door to complex simulation and animate formation. Artificial life, material intelligence, interactivity, and other second-order principles will be approached from the vantage point of "dynamics" and "search" – or the introduction of directed intelligence into a dynamic process of making.

Students will develop a focused inquiry into a specific algorithm and formal target. By casting a wide net, we hope to see opportunities for developing a critical stance towards algorithmic 'tooling.'

Processing

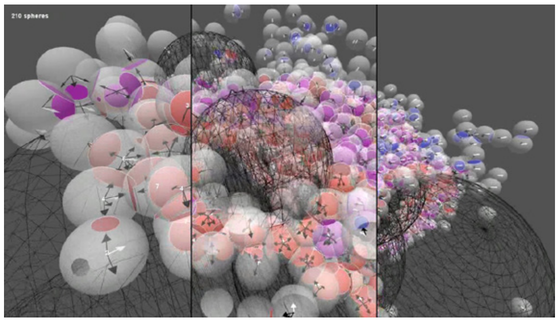
Why do we need to open up the black box of computation? Using ready-made software, even extended with scripting, we still remain embedded in the universe of the software. A project, especially a building, is its own universe of variables, materials, means and agency. Search promotes the authorship of new, unique software to explore design problems and possibilities specific to individual projects. In constructing our own software we will draw upon a fundamental understanding of computational language, a primary and necessary "literacy" for design innovation.

Object Oriented Programming

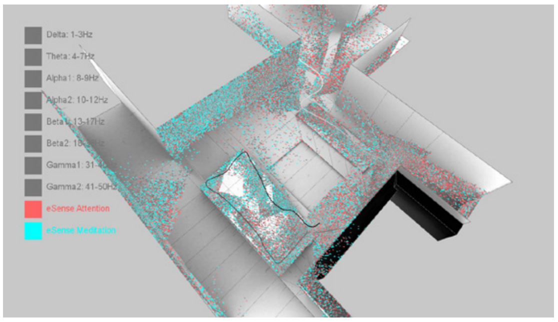
Object-oriented programming (OOP) is a crucial part of the seminar's approach to algorithms. Object oriented programming gives us a rich, flexible language to model complex systems and visualize their outcomes. Each student will develop a series of custom objects that interact and extend the objects introduced through workshop and 3rd party libraries (such as those used to interface with Kinect, GIS data, video, or other potential outputs and inputs).No prior scripting experience is required.

Please see www.proxyarch.com/wiki for previous work and more information.

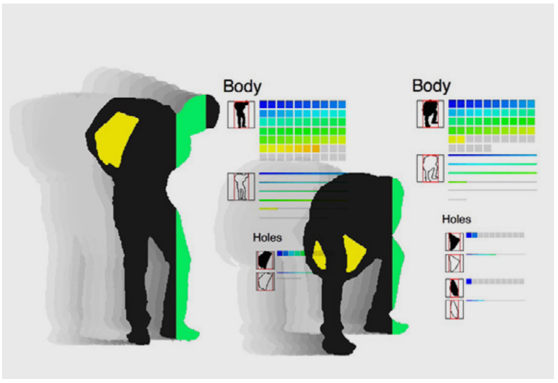
Recrusive geometry at work in nature. Recursion is created through self-reference, either in geometry or transformation.



Physics based modeling in Processing.



Mapping brainwave data onto a BIM model.



Processing visualization of body analytics using the OpenCV library.