Computational Design Methods

DAVID LEE
Clemson University

The ACSA Digital Aptitudes Conference celebrates 100 years of architectural discourse. Of parallel importance to the theme of this session, the event will also mark the 50th anniversary of the internet's conception. Indeed, Licklider's concept of the 'Galactic Network'[1] marked a revolutionary shift in thinking about how data sets could be managed and was followed by a series of influential publications that collectively laid the groundwork for the Age of Information.

While computation is not inherently about digital tools, the advent of the Information Age – spawned by the internet and fueled by technology such as, mobile computing, social networking, and GPS – is largely responsible for the current necessity for computational thought in design. Computational thinking being compulsory to the various disciplines that employ information processing, it is critical that architecture schools adopt an attitude that computational thinking be compulsory to the education of the architect. Moreover, it must be engrained in every aspect of a design education, from beginning to advanced design as well as in practice. The following sections document several important concepts of computational design and speculate on its necessary role in design education through examples of coursework delivered at all levels of an architectural curriculum.

[concept 1]

COMPUTATION IS NOT A TREND, IT'S A PARADIGM SHIFT

We must not view computation as something that is a passing period of architectural style. Computation is much more broadly embedded in a cultural paradigm shift toward the use and integration of information in every aspect of our lives. It has been less than twenty-five years since E-mail was first commercialized by Steve Dorner with the Eudora program.[2] In that very brief history, the transfer of information via written messages has shifted from walking to the post office and sending a letter (the hand courier), to sending messages electronically over the internet, to text messaging and a wave of related 'instant messaging' formats. We depend on the scientific advancement of information processing in our everyday activities. And we need to be ready to let go of the nostalgic view that architecture is so sacred that it is immune to technological advancement and cultural change and can remain ignorant of such influences on our everyday lives. After all, architecture is merely a reflection of its environment.

It's important to note that the paradigm shift here, from the Machine Age to the Information Age is one that involves a long period of transformation. At the same time, we are in the midst of this transformation – and have been for a few decades now, to the point where the door we entered is but a faint image in the distance and the door we will exit is growing near. As Reyner Banham observed, the Machine Age's departure from previous historical periods was not simply in aesthetic appearance, but one that reflected entirely different cultural attitude that embraced technological advancements.

> 'The Man Multiplied by the Motor, to use Marinetti's phrase, was a different kind of man to the horse-and-buggy men who had ruled the world since the time of Alexander the Great.'[3]

With automation came new modes of cultural representation that both reflected attitudes toward the opportunities provided by technological advancement as well as the necessity for architecture to respond to the changing needs of its environment.

The automobile, and later the commercial airplane, had an influential impact on our perception of speed and accessibility. Following the thought processes that were born in the Industrial Revolution, those machines have evolved greatly, yet we see their limitations.

The importance of the Age of Information is that it has brought an entirely new kind of speed to humanity. It's a speed that doesn't require physical movement to make things happen and, for that reason, a departure from its predecessors. Email, Social Networks, Instant Messaging, these concepts have dramatically altered the way we live and represent a new kind of speed. Few would suggest that this would imply that the speed implied by the automobile, for example, is now obsolete. Rather, the speed of information has become an integral part of our cultural contemporary and architecture is obliged to respond.

[concept 2]

COMPUTATIONAL TOOLS ≠ OTHER TOOLS : : MODERNISM ≠ MODERN

The Villa Savoye is an excellent example of what we now commonly refer to as the International Style and we understand it, in part, through our knowledge of its historical importance. That is, its design was influenced by the technological and cultural moment it was a part of, aspiring to represent the Machine Age utilizing both new technology and aesthetic decisions. It would not make sense to design the Villa Savoye today' however, because it does not relate to contemporary culture or technology.

As Jessie Reiser writes:

> 'Apologists for modernism – or those who simply want to extend the modernist project by updating their arguments while leaving the architecture unchanged – are in grave error. In their minds, the shifting paradigm is simply yet another shift in discourse, it doesn't affect the object, and the object has no effect on it. Discourse alone merely becomes a more fashionable view of the same universe, thus implying that the early model is but a failure of interpretation."[4]

To follow the argument for architectural movement being inextricably linked and at the same time a result of its contemporary culture would logically suggest that a cultural paradigm shift would potentially yield a different, unique set of design methods. For at least two decades now, we have seen a broad range of attempts by architects to qualify what might be ostensibly linked to computational design. Many of these attempts have been specific to one design element: be it formalism, technical resolution, the conceptual diagram, or another component of a comprehensive design. These were the leaders responding to the need for the Information Age to find its way into architecture. Of course, with the increase in the pace of technological change and the simultaneous increase in accessibility of computational tools, we are now seeing a broader potential for computing in design and design education.

The 21st century architect cannot avoid being linked to information systems; specifically, information systems that rely on complex networks of data and are operated with various layers of computational devices. As such, while prior generations could perform tasks with a parallel rule and a lead holder, the 21st century architect must also consider computational tools – and more importantly, how these tools operate conceptually – as their primary means for thinking and designing.

Most importantly, educators need to recognize that the electronic computer's sole purpose is not to quench the call for faster production time and narrow the talent gap with representational tools. Such an interpretation would drastically underestimate its potential and importance. We are at a point where the design and construction representation standards themselves have not only been called into question, but are being reinvented to keep pace with our world. Building Information Modeling is very much about construction documentation for architects – and is more opportunity than alternative, but has very little in common with drafting.

[concept 3]

COMPUTATION IS NOT DIGITAL

nor is an algorithm, nor a parameter

Parametric and Algorithmic are quite popular terms in today's design world and are often associated with digital tooling. While it is true that these most

commonly incorporate electronic media, it is important to recognize that they are not design methods that are inherently digital, rather they simply refer to two variations of a constructive logic.

Digital design, digital thinking, digital tools – these phrases all typically refer to work produced electronically. Because there is no significant difference in their process or organization, they are essentially counterparts to analog production. Computational design involves any organizational thought process that uses computing to solve a problem. In theory, any problem that can be solved computationally can be solved using analog or digital techniques.

Antoni Gaudi famously developed analog computing models for his design of the Sagrada Familia. In a postmortem continuation of Gaudi's project, Mark Burry later developed a digital counterpart to these models by simply applying the same computational methods of evaluating the curvature of catenary arches used by Gaudi. While the computational tools dramatically expedited the project's timeline, the effect was the same.

In the context of design pedagogy, this relationship between digital and analog processes has a profound impact when we consider the highly cautious reaction many design schools had to the advent of the personal computer. At the same time, it is important to recognize that the changes we are observing in the Age of Information, those associated with computational thinking, are representative of a wholly new understanding of our environment. Therefore, it would be irresponsible to suggest embracing computation by integrating it into a line of thinking associated with a pre-computer environment. For years, certainly throughout the 90's where the computer became more of a commonplace at student desktops, the common argument for its use was that it should somehow recreate or represent the draftsperson's lead holder. Computers were 'integrated' into the curriculum by simulating the act of manual drafting, a practice that remains prevalent.

[concept 4]

LEARNING COMPUTATION

let education foster enthusiasm, involvement, and self-reliance

Learning how to do something properly in means going back to fundamentals. When I began my design education I was heavily influenced by Georgy Kepes' Language of Vision. In describing how to begin training one's mind to understand the visual field, he states:

> "We have all been taught, in looking at pictures, to look for too much. Something of the quality of a child's delight in playing with colors and shapes has to be restored to us before we learn to see again, before we unlearn the terms in which we ordinary see…When we structuralize the primary impacts of experience differently, we shall structuralize the world differently."[5]

The process of unlearning poor habits or misunderstood values is as important as the retention and application of new material.

Therefore, it is essential that a student interested in pursuing computational design remove preconceived notions of what a computer is and engage in the act of computational thinking with an open mind.

At the same time, it is important that the student of computational design learn in an environment where opportunities are presented that allow knowledge to be applied in a variety of ways. In outlining a teaching method that creates an active experience for students in the classroom, Senske cites three concepts for the transfer of knowledge: the ability for a procedure to take into account multiple contexts, mindful extraction of information, and metacognition.[6]

[CONCEPT 5]

DON'T LEARN HOW TO USE SOFTWARE, LEARN HOW TO STRUCTURE INFORMATION

give a man a fish and he'll eat for a day, teach him to fish and he'll eat for a lifetime

It's an all-too-common occurrence in design school that students are presented with an assignment that asks them to learn a particular tool or subset of tools from a particular software, but not be instructed as to why those tools might be used in assisting with the assignment. This method encourages students to use the tools, but not to learn how they operate at a conceptual level. At the same time, because the student will only be able to complete a single task with the tools, they will be dis-

couraged to use them in the future because they would have no idea how to create.

If you understand how computers work, you're going to be much more adept with using computers. A software tool uses some form of computation, but using the tool does not make a design computational. It was simply made with that tool.

Different software is written to solve different kinds of problems. If you know how to use a given palette of software, you will be able to solve the problems the software was written to solve – and only those problems. The problem with this scenario is that the problems change. If you learn how to write software programs, you can solve any problem.

John Maeda writes:

> 'Users of tools are much more prevalent than the makers of tools. This imbalance has traditionally been rooted in the vast difference in skill levels required for using a tool compared to making a tool… A strange and reverse phenomenon is in motion today: As programming becomes easier and more accessible, the tools for expression are becoming more complex and difficult to use.[7]

In our hunt to create design software to solve the widest range of problems with the most comprehensive set of tools, we have inadvertently overcomplicated software. Many of the 'advanced' modeling software we see are now so densely packed with tools that they stunt creativity by the sheer amount of time it takes to master that set of tools…only to have a newer, slightly different set of tools be introduced in a more recent version of said software.

Interestingly, the most profound discoveries and marked change are being made in programming, by remarkably simple tools. Companies such as Facebook and Twitter are obvious examples, here, of how simple and accessible programming can create massive change when placed in an open-source and/or ubiquitous computing environment. Programming IS structuring information.

[concept 6]

COMPUTATION EVOLUTION

Learning to design with computation must evolve throughout a curriculum

Computing is a mindset that the current generation of students are already engaged with because it is embedded in their everyday activities and has been for some time already before they enter design school. They have email accounts, use social media such as Facebook and Twitter, have GPS enabled smart phones, and a host of other technology all before they have their learner's permit to drive. Therefore, it should be possible to engage computational thinking in all coursework, regardless of its position along the timeline of a curricular map.

An important consequence of this logic is that of perception. If computation is treated as a 'given', embedded in education, it will be viewed as ancillary. If it is elevated to the status of 'special topics' coursework, and that alone, it will risk being viewed as extraneous. For this reason it is imperative that computational thinking be introduced at the very beginning of the curricular map. If it is developed by a progression through both design studios and other theory and technology coursework, it will fully be embedded. If courses exist within a curriculum where computation is not reinforced, by discourse or by technology, there will always be a sense that it is an option, as if it were a choice between architectural styles or aesthetics.

[delivery 1]

METAMAPS

In their introduction to the 2006 book, An atlas of Radical Cartography, Alexis Bhagat and Lize Mogel claim: "This Atlas is an atlas and not the atlas. Rather, it is one of many possible atlases, given the abundance of artists, architects and others using maps and mapping in their work."[8]

Radical Cartography is an appropriate phrase for describing how one might map their experiences in space and time. A map offers us a simple diagram of place, primarily though metric values. It can offer possible solutions for moving from one place to another and highlight certain geographical features, buildings, landmarks and other inanimate things. But what does a map tell us about experience? How might a map tell a story about place that doesn't exist in physically measured features?

Metamaps, as the name implies, involve a higher order of representation than a cartographic exercise. Metamaps are self-referential and dynamic vi-

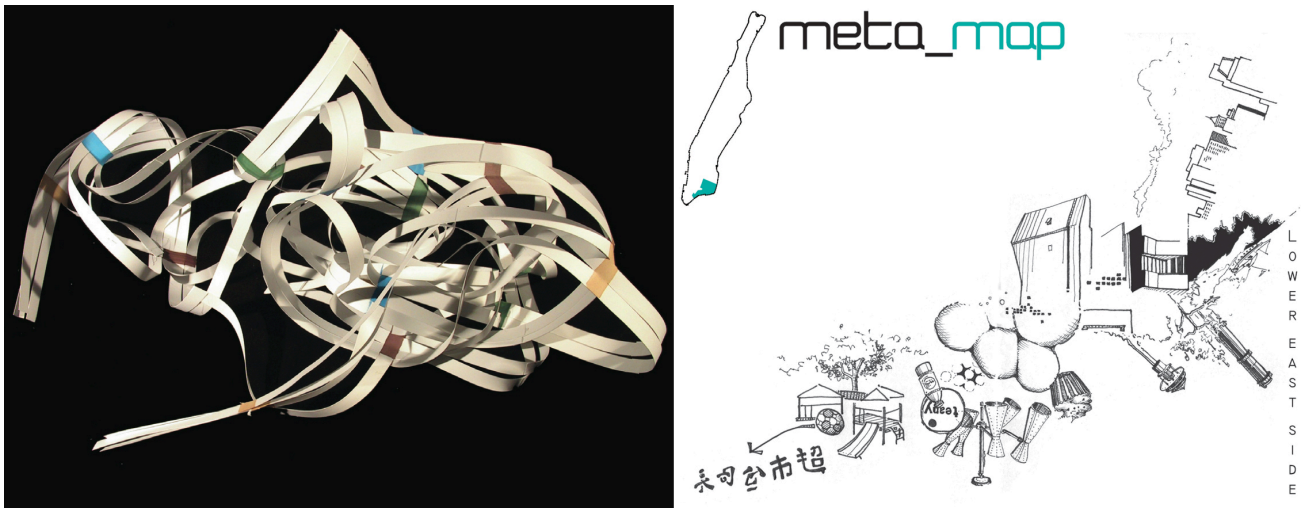

Figure 1. Student projects.

sualizations involving complex sets of information. A metamap is a living organism that can grow and change in time and refresh itself as its referenced information migrates, mutates or simply becomes defunct.

Topological map of space paired with a mental map of space. Neither is bound by Cartesian limits, yet both accurately represent a complex set of networked information in space and both can be understood as a map. In this project, students were asked to map their experience of new and foreign places during a study abroad program.

The goal of the project was to convey an understanding that cartography need not be limited to a metric study, to introduce the concept of network topology, and to use associative or relational information as a visualization tool.

[delivery 2]

ENVIRONMENTAL CONTROL SYSTEMS

One obvious change in architectural thought that has occurred in parallel with the Information Age is the notion that architecture itself, through time-based systems, can physically and environmentally respond to our needs. Although the field of robotics has transformed to a highly evolved state of artificial intelligence research, at its root there are very simple fundamentals that can be applied in an architectural context.

This project asked one simple question: How can architecture respond to an environmental condition? With no prior knowledge of robotics or computer programming, students were each given a condition such as temperature, light, distance, mo-

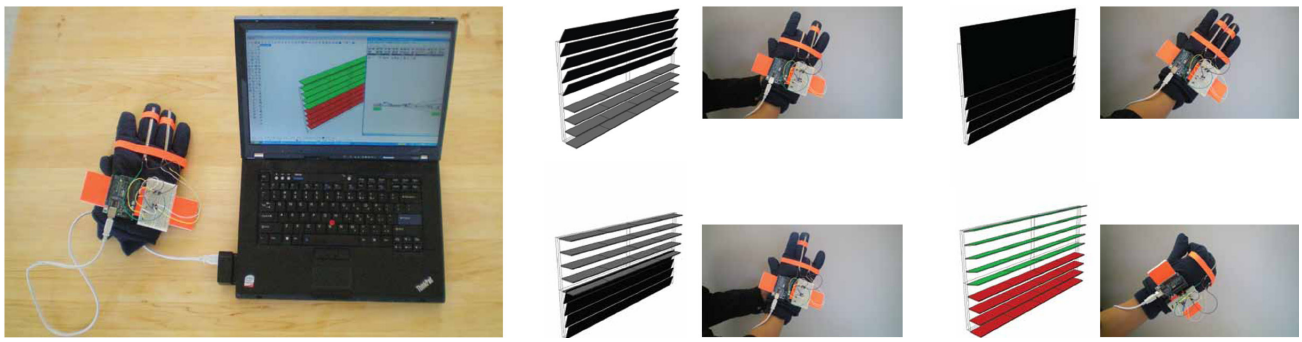

Figure 2. Student projects. Sensor-driven environmental information informing user-programmed parametric device.a human navel.

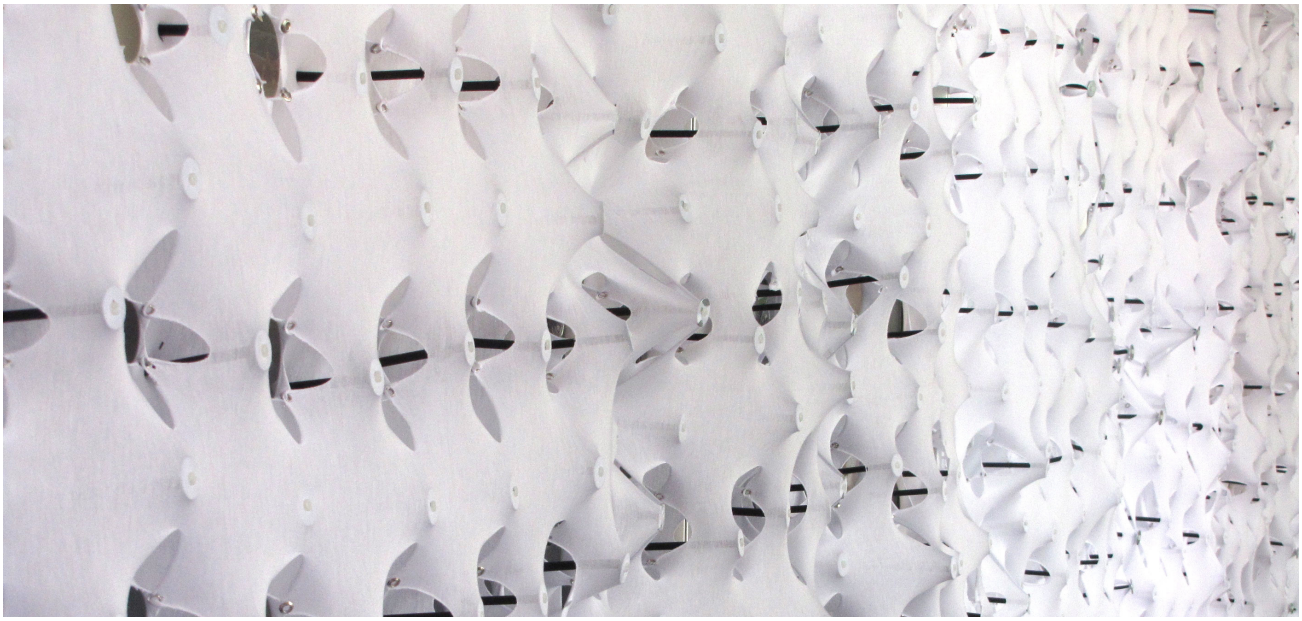

Figure 3. Student projects. Fabricated responsive privacy screen.

tion, or flexibility and a sensor capable of detecting that condition.

The goals of the project were to understand the necessity of information transfer from one medium to another, to introduce the concept of active environmental systems, and to understand that networked and real-time information are not reserved for the internet.

[delivery 3]

MATERIAL INTELLIGENCE

This project investigated analog and digital computing in tandem as a method to simulate material properties, form-finding solutions, and multiple layers of complex sets of information. Two simultaneous projects were investigated in groups. In the first, the process began in an analog format, testing the material properties of stretch fabric and effectively creating analog computing machines. The process then moved to a dynamic digital simulation of the physical properties of the material. Students, working as a team, added multiple layers of environmental information to respond to a programmatic condition of a privacy screen for administrative offices. The design created a variable aperture system using the form-finding solutions they arrived at earlier combined with a tectonic solution for its assembly. In the second, a digital model of a simple action of folding a strip of paper to form a conic section was modeled digitally by constructing algorithms.

The goals of this project were to understand that computing complex information can easily be achieved in an analog format, to understand how to transfer analog information to a digital setting, and to understand the fundamentals of associative modeling.

[delivery 4]

INFORMATION ACROSS MEDIUMS

An image may be considered a representation of an idea. Information can be extracted from that image and used to clarify the idea or intention of the image. It is also possible to take information out of its native context and insert it into another, foreign context while retaining the essential message of the information. Students first create a 'spray painting' using aluminum foil and spray paint. They then transfer the information embedded in the painting to other mediums.

This goals of this project are to introduce basic digital tooling skills, to introduce the notion that information can transfer across mediums, to introduce how referenced information can be used to affect otherwise unrelated conditions.

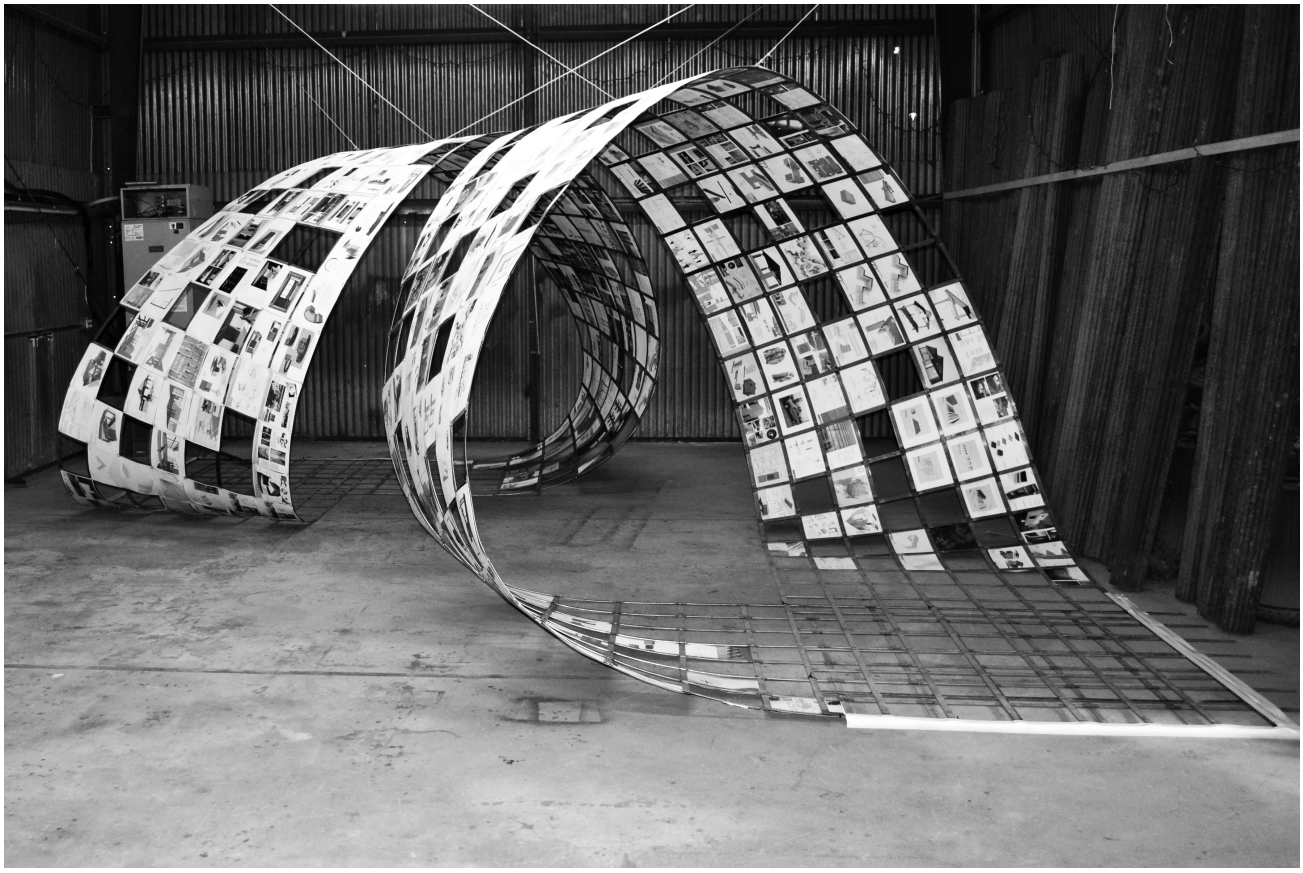

Figure 4. Student projects. Conic plank lines fabricated with welded steel ribbons

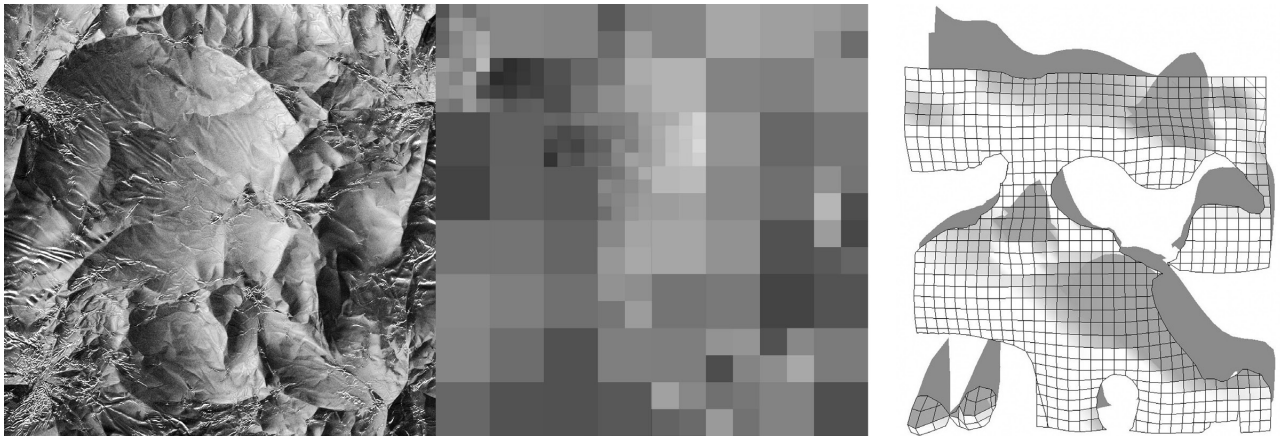

Figure 5. Student Projects. 'Spray Painting', translation to image, representation of average brightness values, translation to heightfield model.

[delivery 5]

ALGORITHMIC DESIGN FOR DUMMIES

Stand up.
Turn to your left.
Walk ten steps.
Open Door.
Congratulations! You've just executed an algorithm. Understanding the importance of procedure and iteration in the design process is an important lesson for beginning design students.

Asking students to understand algorithmic design process by emulating highly complex (and all-too-often overcomplicated) designs doesn't do much for knowledge retention. To learn and be able to apply knowledge to any situation one must begin with the fundamentals.

This project takes an essential element to computational design and introduces it at the level of common knowledge. Once this is established, more complex concepts are introduced, such as conditional statements and recursion.

But there is a wall to my left. My algorithm should read:

Stand up.
Turn to your left.
Walk ten steps or until you reach an object.

But there is no door! My algorithm should read:

Stand up.
Turn to your left.
Walk ten steps or until you reach an object.
If there is a door, open door.

Writing expressions and solving problems through an iterative process that allows an algorithm to evolve or adapt.

After students become familiar with the concept of writing algorithms to solve problems, formal relationships are introduced by engaging the students with visual communication and composition. Because the same logic that was used to write the simple expressions outlined above can be used to generate objects in a spatial field, students are able to quickly move through variations of designs.

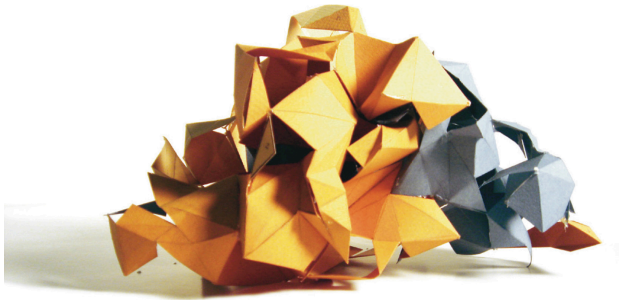

Figure 6. Student Projects. Analog model following algorithmic procedures

The goal of this project is to introduce computing methods such as expressions and algorithms, to introduce iterative process and procedural development, and to work with composition in a rigorous manner.

ENDNOTES

1 Waldrop, M. Mitchell. The Dream Machine: J.C.R. Licklider and the revolution that made computing personal. (New York: Penguin, 2002), 259-261.
2 Engst, Adam C. "TidBITS : InterviewBITS with Steve Dorner." TidBITS. Feb 14, 2000. August, 2011. http://tidbits.com/article/5800
3 Banham, Reyner. Theory and Design in the First Machine Age. (Praeger Publishers, 1967).11.
4 Reiser, Jesse. Atlas of Novel Tectonics. (Princeton Architectural Press, 2006). 24.
5 Kepes, Georgy. Language of Vision. (Dover, 1995). 11.
6 Senske, Nicholas. "A Curriculum For Integrating Computational Thinking" in Cheon, Janghwan, Steven Hardy and Tim Hemsath, ed. ACADIA Regional 2011: Parametricism SPC. University of Nebraska Lincoln, College of Architecture, 2011: 94-95.
7 Maeda, John. Creative Code. (New York, Thames and Hudson, 2004). 113.
8 Bhagat, Alexis and Lize Mogel. Atlas of Radical Cartography. (Journal of Aesthetics and Protest Press, 2008).