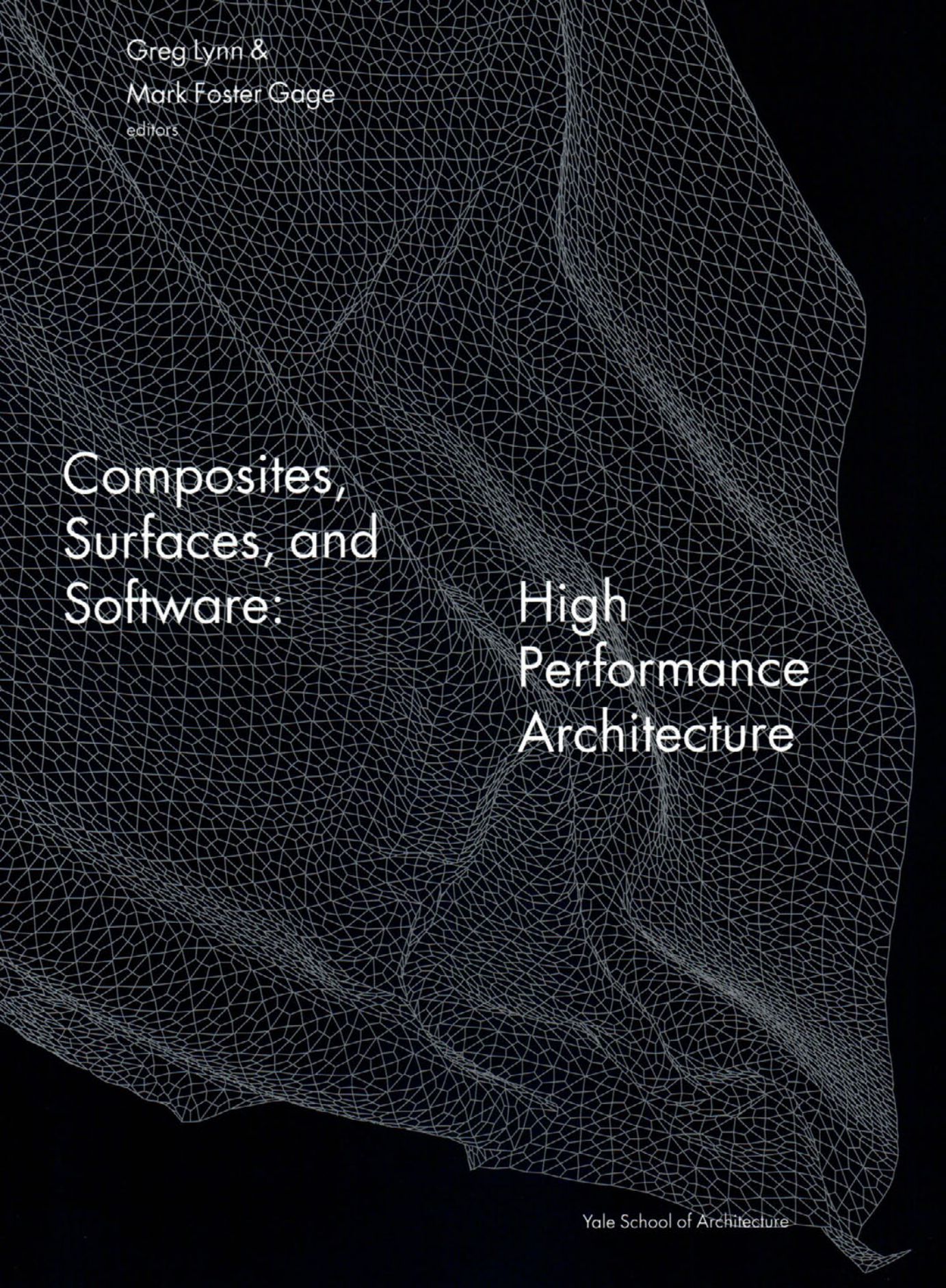
Greg Lynn &
Mark Foster Gage
editors

Composites, Surfaces, and Software: High Performance Architecture



Yale School of Architecture

Mark Foster Gage

Software Monocultures

"It is a mistake to think you can solve any major problems just with potatoes."
—Douglas Adams

Software Monocultures

While originally Incan-cum-Peruvian in its ancestry, it was in Ireland that the potato suffered its greatest humiliation. Imported to Ireland in the late sixteenth century, the potato took firm root in the wet sticky soil that rejected most other significant food-providing forms of agriculture. By 1846 the *Solanum tuberosum,* known in Ireland as the "lumper" variety, had become as Michael Pollan tracks in detail in his 2002 book *The Botany of Desire* virtually the sole remaining variety of spud on the entire island.[1] While this monocultural strategy worked for nearly 250 years, offering easily farmed nourishment to a hungry and growing population, as well as independence from the domineering wheat of the British, in the end it was this same uniformity of genetics that caused the great potato famine that eventually claimed over a million lives and spurred a mass exodus of equally as many between the years of 1846 and 1852. Reliance on only one genetic strain of potato simply assured that a single virus could wipe out the entire crop species, which is precisely what unfolded in this six-year period.

"Blazer Russet (A8893-1)" as determined by the Potato Variety Management Institute

Most systems of design, production, or analysis have a potato of sorts—the thing on which they rely a bit too heavily, that is questioned perhaps less than it should be, and that because of its sheer productivity, or its seeming innocuousness, exists just out of focus of prying critical eyes. Training at least our nose, if not a critical eye more focused on our own potential oversights, to the profession of architecture we might find that the potato for our contemporary hunger is our software—or more precisely, the standardized software packages on which we rely to do, well, everything. Software, like the electrons it pushes, occupies a multiple form of existence, being both and neither a physical thing as well as series of processes. This characteristic assures that, in a profession as generalized as architecture, software cannot be governed by any specific single group of people. Instead it draws on the talents of a vast range of programmers, debuggers, information technicians, and finally architects for its ultimate effectiveness. Because no single group can assume sole responsibility for its development, use, and therefore oversight in the architectural process, the position that software is maturing to play in contemporary architectural practice remains largely critically unchecked.

Gage / Clemenceau Architects, SolarPod research project, rendering of solar pod, June 2010. Generated using a combination of automotive (Alias Studio), architectural (3d Studio Max), and special effects (Maya) software.

Addressing this lack of such critical oversight becomes increasingly important as the assimilation of digital technology, via software, dramatically continues to alter how the products of the architectural profession are designed, produced, documented, transmitted, approved, tested, and recorded. Despite this range of participants in the software-cum-building equation, ultimately it must be the architect who is responsible for the cultural impact and performance of final building, and accordingly for the biases of the tools and processes from which it was created. And without question, all aspects of the process that transpire under the architects supervision, from initial design to final fabrication have been affected by software. Through this process it is undoubtedly the production of standardized user-friendly architectural software, namely representational Computer Aided Design, or CAD programs, that have currently had the most dramatic impact on the day-to-day workings of the profession, and therefore, the built environment. Software packages intended to provide architects with easy access to the difficult processes of computation have been developed for everything from the aforementioned purely representational CAD systems to advanced parametric Building Information Modeling (BIM) systems that manage not only representation but material quantities, costs, supply chains, and assembly instructions, among other variables, in a single digital model. It must be recognized that although these programs are intended to manage information they have formal biases as well that impact the creative process. These biases arise because the software not only includes but also omits particular formal tools. No tool, software included, is free from leaving traces of its use. Accordingly, across scales and building types, the architectural design and documentation process for construction has been almost entirely rewritten in merely two decades. And, although BIM programs are gaining in popularity, the primary ingredient in this starchy standardized Vichyssoise has been the critically unexamined dominance of representational CAD programs.[2]

An obvious drawback of this particular digital revolution is architecture's dependence on an extremely limited set of software packages that give architects access primarily only to the Euclidean geometries on which the profession has traditionally relied. These CAD programs operate by grouping together simple palettes of tools which allow designers to produce primitive shapes, in two dimensions and increasingly three dimensions, and alter, multiply, deform, and connect these geometries in the service of design. Circles are bisected to create arches, or further segmented into sections that are then recombined to produce more complex curves- albeit all still products of the original primitive. This poses the problem that the emerging built environment is being produced, almost exclusively, by the limited commands and limited geometries available in standardized architectural software packages.[3] The irony here is that while the computer has enabled vast advances in the formal opportunities for architectural design, never before have architects relied so heavily on standardized interfaces of design that obscure these new formal freedoms. This may be inadvertent, but it has certainly limited the design palette of the profession, and, as a result the aesthetic parameters of the built environment are being dictated largely by tools included or omitted by software engineers.[4]

Even with the best intentions, the most popular of these CAD programs, unavoidably, become fertile ground for monocultural development. They offer design and representational tools to architectural offices coupled efficiently with the formatting options to facilitate direct translation of this design language into the dialects of standardized contracting and fabrication. This is not a sinister proposition, as much as it is one governed by a primary interest in economic, contractual, and constructible efficiencies as opposed to ambitions of formal diversity in the service of performance or other forms of architectural innovation. So architecture finds itself today standing tall, and proud of its recent technical advances in its own vast and never changing field of genetically identical spuds. Responsibility here lies not with the software companies, who provide for the demands of the profession, but with the profession of architecture itself. The discipline has done little to challenge the increasingly constricting demands of economic efficiency that devalue architecture as merely a cost-per-square-foot container for program.

The Irish lumper was by contemporary standards a small, bulbous, and rather ugly looking tuber and is entirely unfit to carry the analogy

of a contemporary architectural culture so perfectly geared to meet the expectations of efficient corporate material production and contracting processes. Expectations for standardized building practices in our current moment of the twenty-first century require simple geometries that allow for the use of a limited palette of regularly sized, easily purchasable, and readily deliverable products. From these standardized systems, shapes, and parts, architects attempt to arrange the *appearance* of a customized artistic vision or in more fashionable architectural parlance: a unique solution to a set of given programmatic or other performance-based problems.[5]

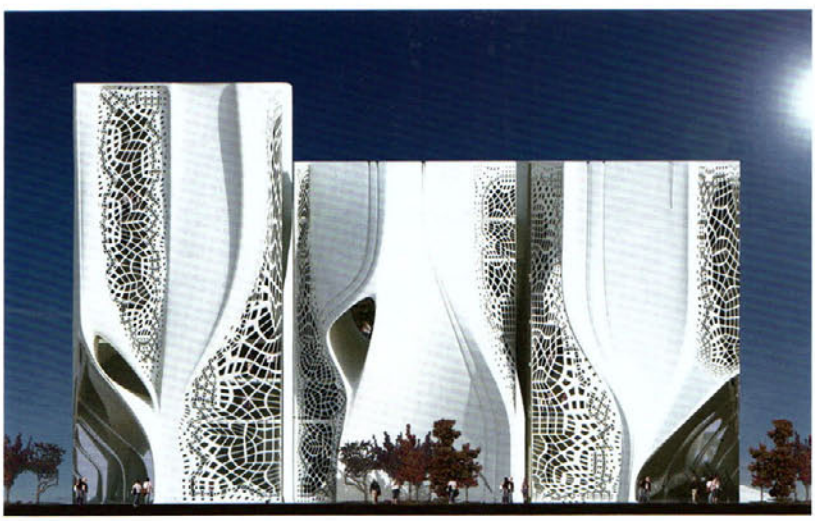
Therefore, our potato of today needs to be of the Russet Burbank variety, which is far larger, more uniform, and therefore more corporately efficient than the historic lumper. The Russet Burbank, to reverse the analogy, is the standard CAD package of potatoes, and as such lends itself more readily to being cut into predictably uniform and inexpensive shapes for us to consume—namely the nearly 28 pounds of French fries that each United States citizen consumes on a yearly basis.[6] The economics of America's love affair with french fries dictates a radical monoculture of tuber uniformity that covers nearly one million acres of land in the United States, a mere dollop compared to the 134 million acres of area currently covered with architecture in the United States. Simply stated, one of the largest land-covering products that humans produce, architecture, is now more than ever a vast monoculture of forms. Our dominant architectural software packages produce this through their very genetics. As the ultimate desire of standardized architectural construction, including factory made windows, suites catalogue issue doors, thirty-foot structural grids, and a whole host of design decisions that default to the properly formatted availability of standardized materials and construction processes, ensure vast formal uniformity across the American landscape.

Again this cannot be considered a problem of the software or building material industries as much as a shortcoming of an architectural culture that demands nothing greater from these immensely technical and productive industries. The introduction of BIM software to the playing field is often described as a significant improvement over the CAD programs of yesteryear. If judged on production and economic efficiency, this is absolutely true, and BIM will likely replace CAD in the future. The problem of this progression is that although BIM software is significantly more complex than the CAD programs, it can manage but not produce as equally complex and innovative forms. That is to say that CAD and BIM programs represent two generations of information management for the profession of architecture, but they are not designed to capitalize on the computer's ability to perform as a design tool. As such, their palettes of tools, and reliance on primitive geometries and simplistic repetitive commands are oriented towards information management and constructability rather than design ends. As the complex interconnections between data and form become more intricate, the sacrifice at the moment is that the shapes and options for producing them are even further simplified. If not by the software, then this is being carried out by its users who are now confronted with significantly more variables that require input, and taking shortcuts on inputting these variables, through simplistic icons for instance, results in the sacrifice of the potential original qualities of the form itself. Of late, the profession seems only too eager to make this sacrifice in order to streamline design and production in the service of efficiency, time, and profit. Architecture now more than ever needs to be defined as the addition of value, cultural or economic, to building and not to be seen as the enabler of its lowest common denominator economic production.

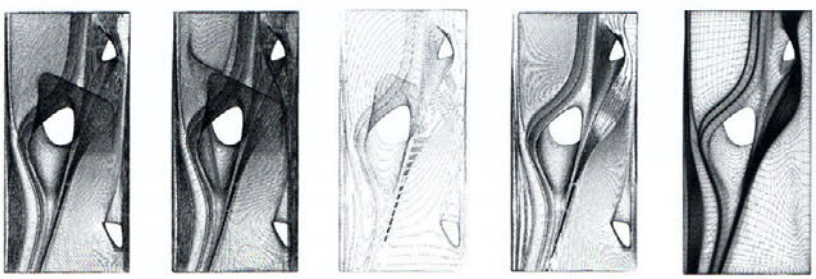
Early resistance to this trend was asserted by a small contingent of architects in the early to mid 1990s, including Greg Lynn of Greg Lynn FORM, Hani Rashid and Lise Anne Couture of Asymptote Architecture, and Jesse Reiser and Nanako Umemoto of Reiser + Umemoto. These architects employed new forms of experimental digital practice to develop strategies to counter the emerging monoculturalism of standardized CAD forms and their associated geometries.[7] This ignited the struggle for the role the computer would occupy within the profession: would it be merely a drawing tool to better organize representations of geometries that are already familiar to us? Would it be a tool to analyze performance criteria in the service of efficiency? Or would it be used to introduce new families of geometry and form in the service of innovation? The original artillery used

to bolster an architecture of innovation was the adoption of software from other industries of design typically unconnected to architecture. Early experiments at the turn of the millennium involved software programs including Alias, Softimage, and later Maya, Rhino, and Mudbox to introduce new formal vocabularies into the discipline of architecture. They provided new palettes of tools that not only were based on the simple geometries dominant in architectural CAD platforms, but that also imported the polygonal and topological tools responsible for character animation in Hollywood movies and similar extra-architectural sources. These more mathematically robust tools were joined by later experiments with software programs that allowed access to other new languages including the precise surfacing tools used in automotive design (AliasStudio), the polygonal faceting languages of origami folding programs (Pepakura), and the mechanism to hold subdivision surfaces together found in later releases of Maya.

On one hand, the adoption of software programs from other industries offers a way to overcome the limitations, or at least geometric biases, of architectural design software packages. On the other hand, because these programs come from outside the discipline they lack tools for architectural scale, their techniques for translation into physical form are undeveloped, and the formatting systems are not adequate to manage the vast numbers of components and component types needed for architectural projects. Therefore, the strength of these programs—that they are non-architectural in origin—is also, predictably, their Achilles heel when they are faced with the task of producing work for construction and fabrication. Furthermore, when inexperienced practitioners rely on such programs to produce architectural form, they run the risk of producing non-architectural forms that overtly evidence the original intent of the tools. In inexpert hands, building proposals can adopt the unarticulated continuous faces of animated characters, and architectural givens, like openings for windows become difficult to resolve in sympathy with these new surface languages. A tangential drawback to the use of these tools is the unintended associations with biological forms. Tools designed to produce digital creatures and animals for blockbuster



Gage / Clemenceau Architects, Estonian Academy of the Arts, rendered elevation, March 2008. Designed using Alias Studio—a software program specifically designed to manipulate "Class A" automotive surfaces.



Gage / Clemenceau Architects, Estonian Academy of the Arts, topological analysis image, March 2008. Analysis of curvature complexity for prototype panel fabrication.

movies frequently produce biological-looking constructions for architecture–even when producing such forms was never the intent of an architectural project based on overcoming known limitations of form, whether architectural or animal. This pulls the work into existing discourses of animality, biology, and natural metaphor that are unlikely candidates with which to understand *new* languages of form. And so tools designed to produce automotive surfaces produce architectural projects that look like cars, and tools designed to produce paper origami sculptures began to produce architectural projects that look like paper origami sculptures. These are not fruitless pursuits, only indicators that each of these newly adopted programs is, in fact, a monoculture in its own right that possesses limited tools designed for limited functions. An image of unfolding paper that can be created in the software program Pepakura is perfect for paper which has imperceptible thickness, but it does not translate well into a discipline such as architecture which requires, even revels in through poche, the substantial thickness of its surfaces.

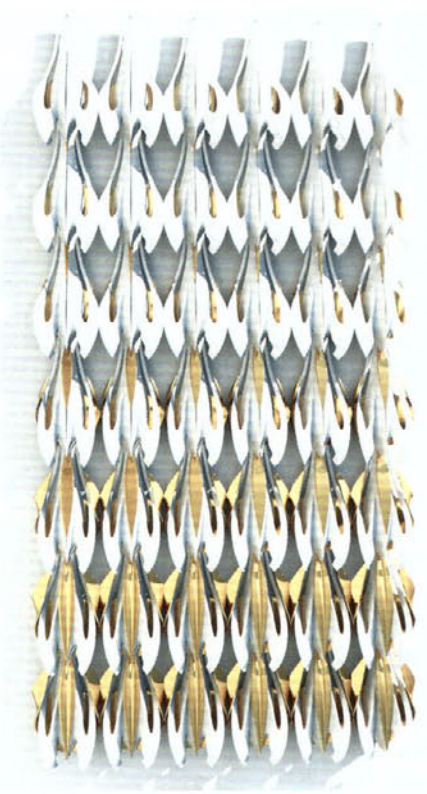
While some of the most provocative developments in recent architecture have come from translating the techniques and tools of one discipline into another via software, following such a model without limits offers what amounts to only a slightly larger variety of software monocultures from which architects can choose. As architects working in this vein wrestle with the translation of the software, they will find in their pursuit of more control that much is to be discovered in the problems that arise. This discovery, however, should be done with a critical understanding of what is at stake and should not be considered (as is currently the fashion in many architectural form-finding exercises today) mere architectural "play." Also, it is hoped that as software companies like Autodesk acquire other design programs such as Maya, Mudbox, and Alias Studio, they will continue to fold the tools of these programs into existing architectural platforms, to create a more formally robust software that maintains architectural applicability.

A parallel computational trend within the profession has emerged in recent years: custom digital scripts are being authored by architects themselves. Albeit difficult and unwieldy to learn, early versions of such customizable software interfaces were present in packages like Maya Embedded Language (MEL) and more recently in RhinoScript. As architectural scripting continues to grow and migrate out of imported software into new intelligences of its own, it introduces a wealth of new formal avenues for architecture, increasing dramatically the ability for the profession to manage complex aggregations of multiple forms—whether varied or graduated and intricately interconnected. From languages of swarms and piles to webs, Voronoi foams, and booleaned bricks, scripting has become the definitive language of variegated multiples assembled into larger architectural wholes. Theoretically this development has been charged with giving architecture access not only to parts but also to a finer grain of small particulates—the stuff from which stuff is made.[8] This stuff however exists within its own world of rules since scripting is, by definition, a myopic process that isolates a problem so that it can be overcome with a limited set of precise and interrelated algorithms. Such myopia risks becoming a new hermetically sealed environment, where outside influence is weak if not non-existent.

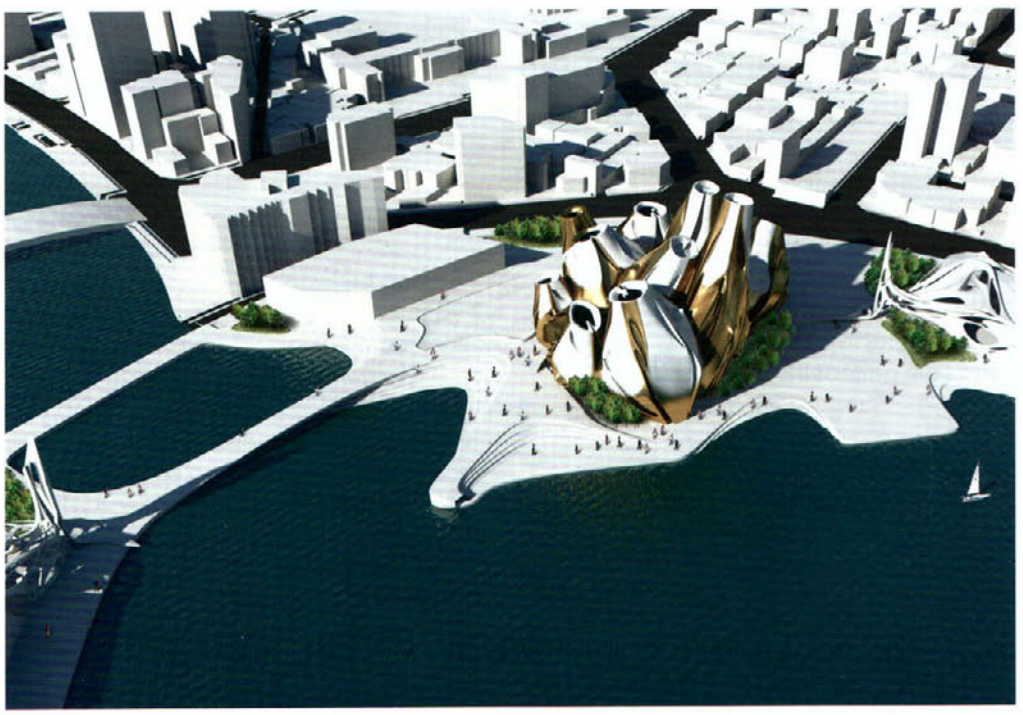
So scripting does not translate into a comprehensive language that facilitates the emergence of architecture, although that is certainly the hope and sometimes the claim. The diversity of components that contribute to an architectural project in the fullest sense demands not only changes in formal and scalar degree, as scripting in its current form provides in spades, but also changes in component and material type, with which these processes currently have more difficulty. That is to say that the component arraying and modifying strategies employed in the acts of scripting produce variations that are sometimes vast in scale and shape, but these scaled and reshaped components are always grandchildren of an original geometric primitive, or data set of points, and are beholden to the limitations inherent in that original form or network, as well as to the script's ability to translate data accordingly.

The techniques of scripting offer architects the ability to write their own formal grammars and associated rules, but in doing so, architects forgo the infiltration of foreign expertise into the protected fickle codes of digital conduct which govern these scripting efforts. If the

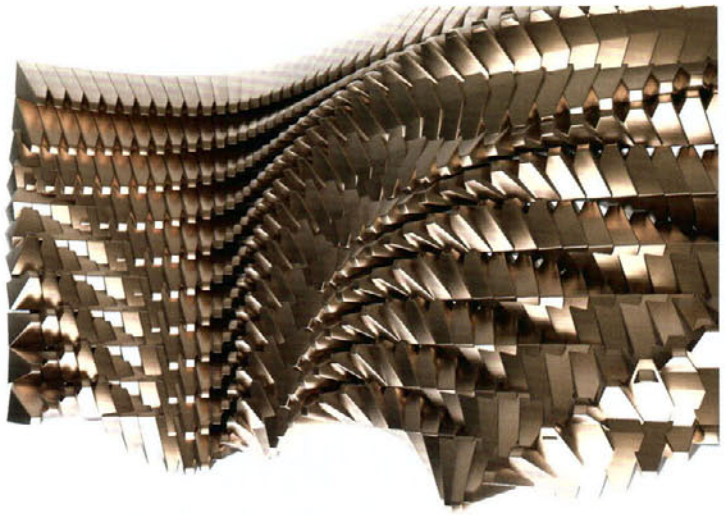
use of software from other disciplines bears the Achilles heel of formal languages with intelligences other than those of architecture, then the drawback of the scripted product is that it exists with no expert intelligence from any design discipline other than that inherent in the original code or parametric data set. The *ex nihilo* act of scripting conducted by an individual architect as opposed to a suite of professional computer programmers also runs the risk of being an amateur endeavor. Although worthwhile, architects as scripters are generally the product of architectural schools rather than computer programming departments and are accordingly inexpert at the acts to which they are dedicated, particularly when compared to actual experts in those disciplines. Because these tools are so new to architecture, amateur knowledge in a field of no existing knowledge is perceived as expert, even though this may not be the case. A more promising model is one of collaboration between experts in architecture and programming—each versed in the nuances of their respective professions—to produce architectural form that exhibits expert intelligence from both disciplines.



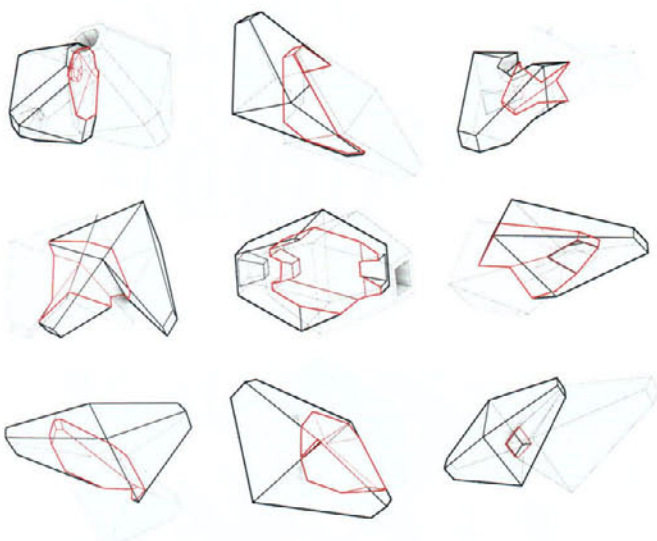
Gage / Clemenceau Architects, Kaohsiung Pop Center, graduated rain-screen detail, July 2010. Skin produces variations in color and geometry based on complexities of curvature—calculated in a custom designed RhinoScript.



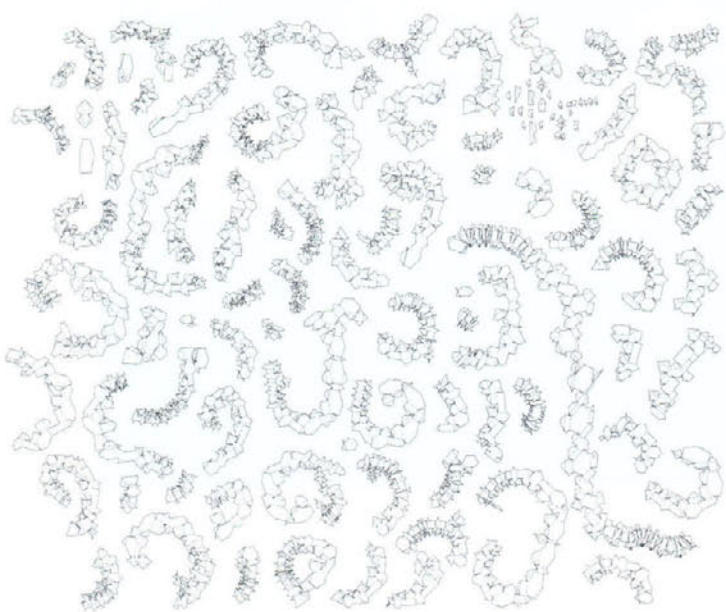
Gage / Clemenceau Architects, Kaohsiung Pop Center, aerial rendering, July 2010. Generated using a combination of automotive (Alias Studio), architectural (3d Studio Max), and special effects (Maya) software.



Gage / Clemenceau Architects, Parawall Research Project: an investigation of Parametric Aggregation Strategies for Complex Architectural Surfaces, perspective rendering, 2009. Individual bricks are repositioned parametrically to respond to various gravity and force loads generated in a retaining wall scenario. Research conducted using Rhinoscript and Maya MEL, Maya Embedded Language.



Gage / Clemenceau Architects, Parawall Research Project: an investigation of Parametric Aggregation Strategies for Complex Architectural Surfaces, joint analysis, 2009. Individual bricks are booleaned from neighboring bricks to produce geometries conducive to stacking. Research conducted using Rhinoscript and Maya MEL, Maya Embedded Language.



Gage / Clemenceau Architects, Parawall Research Project: an investigation of Parametric Aggregation Strategies for Complex Architectural Surfaces, unfolded brick fabrication drawing, 2009. Individual bricks are unfolded in the Origami software program, Pepakura, to be lasercut and assembled from flat materials.

A fusion of these strategies to avoid becoming a monoculture is emerging between software importing and scripting that offers some hope in assuaging the deficiencies of both. Recently some programs have introduced simplified scripting platforms that encourage designers and programmers to customize the performance abilities of the programs themselves. Simplified graphic interfaces like Rhino's new scripting interface, Grasshopper, make modifications and scripting tools more accessible to architects who are unfamiliar with the language of writing custom scripts. However, there are drawbacks to this approach as well. The introduction of simplified palettes and organizational relationships reduces the potential for open-source customization of the program. So another monoculture, albeit one of icons and pull down menus that help build scripts instead of icons that perform transformational functions, is born. This makes such a program better suited for educational use as an introduction to relationships between form and code. Certainly a full collaboration between disciplines would result in more fruitful outcomes.

Many of the aforementioned systems include parametric intelligence, currently architecture's most *au courant* topic. The early scripting program, Bentley's Generative Components, as well as a parametric scripting program currently in development by Autodesk are efforts to build software programs that are more directly dedicated to the instant visual manipulation of form via custom scripts. Parametric intelligence provides for architectural form to be derived and altered using interconnected data sets, as opposed to transformation of the form by the architect. With parametrics the modification or addition of data, when linked with form, has the capability to alter or produce forms without requiring any willful input from the architect. This is touted as a positive development for architects who define architecture as the solution to cleverly identified problems.[9] In a perfect yet very rare parametric world, a data set can lead to the production of an "ideal" form that best suits the performance criteria required by the data. Following from this, if a collection of forms is produced that best suits the performance criteria required and then one form is modified, the other forms in the set will update accordingly in order to best match the new configuration to the criteria of the altered data. This method requires architectural form to be the product of posing and solving problems through data set relationships. The form is limited by the parameters set in the equation, and like writing script it is inherently isolated within one discipline, as it is only through the problem itself that expertise can be derived. Therefore, all architectural intelligence that can be expected from the result must be written into the problem as it is first established. The other noticeable difficulty with the strategy of parametric intelligence is that the intelligence must have an initial primitive shape or similar network of point locations on which to operate because form rarely emerges *ex nihilo*—even one derived from the demanding expectations of massive amounts of architects' accumulated data. As such, parametric design strategies rely on the development of primitives that in turn are modified by the associated data and therefore are limited to the languages of primitives available within the software, like CAD, or architectural intelligences of the given data sets themselves.

Importers, Scripters, and Parametricists. It is of course a mistake to even initially separate these efforts into such carefully segregated groups, as the groups and tools are actually more intertwined than such partitioning suggests. Nonetheless, all certainly fall under the rubric of architectural design systems emerging from "the digital." It is slightly ironic that each group seeks primarily to overcome the standardized digital tools that currently dominate the formal output of the profession. Even more ironic is that in doing so according to the myopic limitations of their own solutions, each, further isolate their methodologies from the discourse of architecture at large, which needs to be more encompassing than any of these solutions can alone suggest.

The first decade of this new century of architectural design is off to a good start, and our options are increasing, as is our expertise. While these options, taken as a suite are currently much more of a monoculture than a rainforest of diversity, the possibilities offered by all three trajectories are aiming in the right direction. At the moment, however, it is clear that the inclusion of these strategies into the profession increases the variety of types of practice possible but rarely diversifies the practices themselves. In fact, the nature of

architectural propriety has produced a contemporary discourse carried out in segregated monocultural books and equally monocultural symposia, where scripters meet with scripters, importers meet with importers, and parametricists with parametricists. At its most extreme, each group is publishing edited books of allies' work, and congratulating itself on its particular brand of digital toolmanship, even asserting that it is the future of the profession while the other strategies clearly are not.

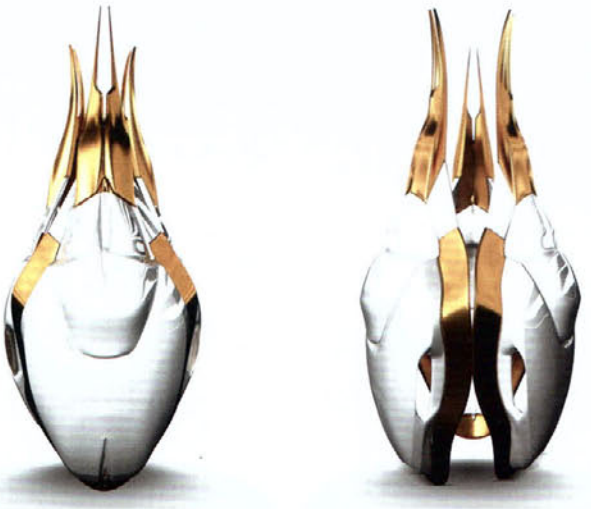
It is important to consider not only how these developments fit into contemporary architectural practice but also to position them in relation to the architectural discipline in its fullest historical and cultural sense. The Renaissance concept of the architect/artist as solitary genius, originally created to separate these artistic castes from that of the more common craftsmen with which they had been previously associated, is mostly to blame for this manner in which architects have mistaken the role software might play in the profession.[10] Architecture in the Post-Alberti Renaissance was a culture of individuals. It is even more so today as both architects and the public recognize vast formal vocabularies by the names of singular practitioners. One has only to describe a building as Meisian or in the vein of Peter Eisenman, Zaha Hadid, or Rem Koolhaas to call to mind an entire corpus of projects and buildings which hundreds, even thousands of people, were involved in producing. That is to say that architects are historically invested in producing a recognizable monoculture in their body of work, which we more artistically refer to as an architectural "signature." Signatures, in this architectural sense, govern the selection and management of particular geometries, materials, colors, design tropes, and historical references.

A tragedy of the digital project, all strains considered, is that signatures are no longer determined by these historical architectural variables and instead, are now being largely governed by software selections and individual technical discoveries. A particular imported software tool or even the simplest Rhino script can produce results that are mistaken as architectural signatures because of their initial formal novelty. Software importers may remain loyal to particular software packages and even particular icons within those packages. Scripters can squeeze an entire family of projects, small and large, from the most common digital codes that are as frequently found in online clearinghouses as they are self-authored. At the moment there is an entire generation of architects who choose to be defined by a narrow range of software monocultures. These monocultures already are producing a predictable sameness in formal, programmatic, and sustainable projects alike. So inculcated are we in the practice of individual signatures that architects now curate identity through these discovered, and closely guarded, software tools. The tools of computation promise to offer architecture the greatest infusion of formal possibility ever seen by the profession. Yet the model of appropriating monocultural software processes for the purpose of developing clear monocultural signatures assures an ironically vast uniformity should the current methods of architectural software use be maintained.

In the emerging twenty-first century, after several decades of rehearsing old scenarios of architectural practice with our new digital tools, architecture needs to now adopt new modes of practice that capitalize on the complexity and contemporaneity of these tools. In doing so, it assures that the formal and organizing gifts of the computer are not distilled into a series of selfishly protected monocultures, or ubiquitously accessible icons, but instead are allowed to flourish throughout the profession in the service of greater diversity and opportunity that reflects an architectural culture of actual ideas. Architects need to resist the proprietary mentality of software and begin to use these tools not as ideas in and of themselves but in the service of more encompassing architectural agendas. Such a mentality will necessarily need to contradict the master narrative of individual signatures in favor of a more expert and cross-disciplinary collaborative approach to the profession. Instead of the identity of an architectural office being determined by the original digital skills of its founders, architectural offices should forgo software allegiances in favor of an exploratory ethic to use software to address the newer, more complex, and more important questions which architecture should be authoring and addressing in the emerging territories of the twenty-first century. Our problems and concerns are not the same as those of the various Modernisms of the early twentieth century on whose geometries,

processes, and expectations of efficiency we still largely dwell. Our software tools and the geometries and processes with which we design should not merely be more efficient ways of organizing and building the forms of yesteryear. Instead they should empower a new generation of conceptual thought, theoretical speculation, sustainable responsibility, and formal production. Architectural offices must develop a culture that uses digital expertise in the service of great ideas and does not rely on the software by virtue of its novel production to function as a proxy for such ideas, particularily when the underlying biases of such software remain unexamined. Only then will architecture bring more options to bear on the larger ambitions of our new millennium than a handful of mere potatoes.

Gage / Clemenceau Architects, SolarPod research project, x-ray drawing illustrating panel seams calculated based on material size constraints. June 2010. Generated using a combination special effects (Maya) software rendering combined with Autocad 3D compositing.



Gage / Clemenceau Architects, SolarPod research project, closed pod elevations, June 2010. Generated using a combination of automotive (Alias Studio), architectural (3d Studio Max), and special effects (Maya) software.

Gage / Clemenceau Architects, SolarPod research project, open pod perspective, June 2010. Generated using a combination of automotive (Alias Studio), architectural (3d Studio Max), and special effects (Maya) software.

1

For an insightful history of the potato see Michael Pollan, *The Botany of Desire: a Plant's Eye View of the World* (New York: Random House, 2001), 183–238.

2

William Mitchell in his essay "Thinking in BIM" outlines the development of BIM software and correctly notes that preceding CAD packages had begun to evolve by "shifting their emphasis from the construction and editing of two-dimensional drawings to the construction an maintenance of complete and consistent three-dimensional models, by replacing rigid geometry with object-based parametric modeling, by providing increasingly sophisticated facilities for associating non-geometric properties with geometric entities, by integrating engineering analysis software and by supporting sharing and transfer of information among various members of design and constructions teams." As CAD programs developed these abilities, however, their efficacy was still largely based on Euclidean geometries. While parametrically interconnected to data sets and possible to be visualized in three dimensions, the icon and primitive based systems for drawing geometry in CAD remained formally simple. See William Mitchell, "Thinking in BIM," *A+U: Architecture and Urbanism* (August 2009), 10–13.

3

Certainly any tool, including CAD comes with its own bias towards production, but never before in the history of architecture has the same tool been so uniformly used to govern such a significant majority of the process from conceptual design through design documentation and furthermore be accepted almost without question by the profession at large. William Mitchell refers to some historic examples of earlier tools and their biases in his "Thinking in BIM" essay. See page 11 specifically.

4

That is not to say that computation via CAD did not increase the formal diversity of the profession, only that it fails to capitalize on the exploitation of the computer's ability to provide dramatically new formal languages. CAD largely kept previous geometric mentalities and coupled them only with new systems of drawing and management that did capitalize on the strengths of computation.

5

Although I have my own doubts on architecture's reliance on mapping problems as a primary vehicle for the production of form, see Gage, Mark Foster. "In Defense of Design," *Log*, no.16 (Spring/ Summer 2009), 39–45.

6

Statistics are taken from "French Fry Consumption in USA on the Decline for First Time" in *Quick Frozen Foods International* (July 1, 2002).

7

It should be noted that while Frank Gehry's office was also pivotal in the introduction of computation into the profession, the manner in which they transfer handmade physical models into the computer is more allied with the use of the computer as a management tool as opposed to a design tool, which was ultimately the goal of these practitioners.

8

For further reading that links processes to form via scripting intelligence see Sanford Kwinter's afterward essay entitled "Seven" in Benjamin Aranda and Chris Lasch, *Tooling* (New York: Princeton Architectural Press, 2006), 92–93.

9

See note five.

10

For the definitive source on the Renaissance reading of artists and architects as a separate class of artistic and melancholic genius see Wittkower, Margot and Rudolf Wittkower. *Born under Saturn, The Character and Conduct of Artists: a Documented History from Antiquity to the French Revolution* (New York: New York Review, 2007).

Featuring

Chris Bangle

Founder of Chris Bangle Associates SRL

Lise Anne Couture

Architect and founding partner of Asymptote Architecture

Greg Foley

Of Visionaire, V Magazine, and Thank You Bear. Illustrator, designer, and creator

Mark Foster Gage

Principal of Gage/ Clemenceau Architects

Frank O. Gehry

Architect and founding partner of Frank O. Gehry & Associates

Bill Kreysler

Founder of Kreysler & Associates (K&A), a custom molder of fiber reinforced products

Greg Lynn

Architect and principal of Greg Lynn FORM

Adriana Monk

Founder and Design Director of amDESIGN

William Pearson

Technical Director at North Sail's 3DL® manufacturing plant in Minden, Nevada

Published by Yale School of Architecture
www.architecture.yale.edu
Distributed by W.W. Norton & company
www.wwnorton.com

ISBN 978-0-393-73333-4

Printed in Canada $45.00 US