ABSTRACT

Computation as both method and aesthetic is applied to the architectural design process at multiple scales. This enables complex, performative, and specific solutions uniquely relevant to emerging models for fabrication and construction. Formal and spatial manipulation of architectural envelope, skin, volume and structure occurs indirectly via the design of geometric algorithms. Relationships are both amplified and collapsed together in the manifestation of the hypothetical design of the NASA administrative headquarters in Washington, DC. Process and product merge to create a poetic language of phenomenological affects and patterns. Simultaneously, the systemization and codification of design “problems” facilitates functional, solution-driven architecture. Animation as a medium is exploited to represent parametric relationships while reflecting the realities of perception of time and space. Usage requirements and site conditions carry the baggage of information-based contexts and image-based parameters as fuel for the inherently cyclical process. NASA’s shifting mission, agenda and values are branded non-verbally through the expression and explicit exploitation of both glaringly broad and obscurely minute requirements that through the medium by which they are applied have additional meaning.
Imaging Computerization: Scripting and Animation as Process and Product

by

Carl Lostritto

Thesis submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Master of Architecture 2008

Advisory Committee:
Assistant Professor Michael A. Ambrose, Chair
Dean and Professor Garth C. Rockcastle, FAIA
Professor Karl Du Puy, AIA
Preface

Much of the content of this investigation is ill-suited for print distribution. To view animations and to interact with the author and this content directly please visit: www.0095b6.com/lostritto
This is dedicated to Lisa.
Acknowledgements

The following played essential roles in this process:

Michael Ambrose for his mentorship, infectious attitude and for seeing the potential in everything but being satisfied by nothing.

Ben Callam (www.regulatedlines.org) for sharing his parallel investigations, enthusiasm, curiosity and knowledge.

Marek Hnizda for having an inspiring attitude and entertaining process in his simultaneous work

Lisa Lacharité for proving we can be intellectual as well as life partners.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Foundational parameters</td>
<td>7</td>
</tr>
<tr>
<td>Process Calendar</td>
<td>11</td>
</tr>
<tr>
<td>Scripting and animation as co-catalysts</td>
<td>13</td>
</tr>
<tr>
<td>Animation Algorithm</td>
<td>15</td>
</tr>
<tr>
<td>Animation Craft</td>
<td>16</td>
</tr>
<tr>
<td>Animation and Scripting As Design Process</td>
<td>17</td>
</tr>
<tr>
<td>Design of Algorithms</td>
<td>29</td>
</tr>
<tr>
<td>Catalog of Functions</td>
<td>31</td>
</tr>
<tr>
<td>Meaning in the Scripted Image</td>
<td>53</td>
</tr>
<tr>
<td>Parts and the Whole</td>
<td>63</td>
</tr>
<tr>
<td>Exigence of Information</td>
<td>67</td>
</tr>
<tr>
<td>First Principles</td>
<td>74</td>
</tr>
<tr>
<td>Conclusion</td>
<td>77</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Fig</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Inside</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Between</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>Site from Maryland Avenue</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Site plan</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Preliminary Program</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Program</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Site Corner</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Relationships at Site</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Dispersion of NASA mission focus to physical centers</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Site Railroad Edge</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>Process map as of week 12</td>
<td>13</td>
</tr>
<tr>
<td>10a</td>
<td>Process Map as of week 35</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>Frame from visualizing parameters experiment 1</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Frames from visualizing parameters experiment 1</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>Animation Algorithm</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>Part-Whole Animation Experiment 1</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>Frames from visualizing parameters experiment 2</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Frames from visualizing parameters experiment 4</td>
<td>20</td>
</tr>
<tr>
<td>17</td>
<td>Frames from visualizing parameters experiment 4</td>
<td>21</td>
</tr>
<tr>
<td>18</td>
<td>Frames from visualizing parameters experiment 0</td>
<td>21</td>
</tr>
<tr>
<td>19</td>
<td>Frames from visualizing parameters framework 7</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>Frames from visualizing parameters framework 7</td>
<td>23</td>
</tr>
<tr>
<td>21</td>
<td>Frames from visualizing parameters framework 5</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>Frame from visualizing packing algorithm</td>
<td>24</td>
</tr>
<tr>
<td>23</td>
<td>Frames from visualizing packing algorithm</td>
<td>25</td>
</tr>
<tr>
<td>24</td>
<td>Frames exposing relationships in parabolic skinning</td>
<td>26</td>
</tr>
<tr>
<td>25</td>
<td>Frames exposing relationships in outer skin algorithm</td>
<td>27</td>
</tr>
<tr>
<td>26a</td>
<td>Frames from physical model construction process animation</td>
<td>28</td>
</tr>
<tr>
<td>26b</td>
<td>Frames from physical model construction process animation</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>Public Path</td>
<td>55</td>
</tr>
<tr>
<td>30</td>
<td>Conglomeration of interdependant design experiments in simulated experience</td>
<td>56-57</td>
</tr>
<tr>
<td>31</td>
<td>Conglomeration of interdependant design experiments--west elevation</td>
<td>58-59</td>
</tr>
<tr>
<td>32</td>
<td>Outside</td>
<td>60</td>
</tr>
<tr>
<td>33</td>
<td>Outside at entry</td>
<td>62-63</td>
</tr>
<tr>
<td>34</td>
<td>Analytic west elevation</td>
<td>64-65</td>
</tr>
<tr>
<td>35</td>
<td>Analytic section</td>
<td>67</td>
</tr>
<tr>
<td>36</td>
<td>Analytic plan</td>
<td>67</td>
</tr>
<tr>
<td>37</td>
<td>Analytic plan during parameter evaluation</td>
<td>69</td>
</tr>
<tr>
<td>38</td>
<td>Conglomeration of design endeavors as simulated from the Gallery/Observatory</td>
<td>70</td>
</tr>
</tbody>
</table>
Fig 39. [above] Conglomeration of design endeavors as simulated from approach on D St. SW 70
Fig 40a-c. Anticipatory effect of overlapping topologies 72
Fig 42. [above] exploiting errors captured motion 73
Fig 41. [top] exploiting errors screenshot 73
Fig 41. [top] Manual Construct: Spaces within a neutral structural system 74
Fig 42a-c. [above] Further framing 74
Fig 44. Analytic simulated experience from corner 76
Fig 45. Analytic simulated experience from car on 14th Street 77
Architectural discourse and practice has recently been infused with a sense of urgency with regard to digital media and technology. Architects see other disciplines encroaching on precious territory. Artists, contractors, fabricators, and engineers have begun to perform what had once been activities within the domain reserved for architecture. The line between disciplines had always been blurry. Now, in some cases, the line is nonexistent. Computational design and more broadly digital design, represent the means by which architects can reposition themselves as generalists. Architects are empowered to more directly control more, if not all, of the design process. This condition merits discussion because technology brings not new techniques and knowledge but potentially wholly new foundational approaches to shaping the built environment. Digital revolutionists proclaim, “change or perish,” in reference to the need for architects to break out of the ability patterns of design and embrace the potential for mass customization. Indeed these paradigms promise real, quantifiable solutions to urgent problems. In the true spirit of modernism, however, architects need not accept assumptions or inevitabilities of “problems,” or “solutions,” as they are.

The generalist architect is not concerned with protecting intellectual, technical or consumer territory. Rather, he reaching approach to the incorporation of other disciplines is employed.
Fig 1. Inside
From within an open office volume the outer skins are framed. The view directly out the opening is unobstructed by the skins as view preservation is one of the variables passed to the skinning function.
This investigation does this with regard to computational design and digital animation. As disciplines, cultures are directly associated with the body of knowledge. A programming culture emerged in the 1970s as engineers, mathematicians, scientists, and philosophers erased the lines between their disciplines to respond to new technology. This culture is alive and still evolving. Besides the technical knowledge collectively available with regards to scripting, algorithm structuring, and data structures, there exists a valuable culture of sharing and expandability that is highly and directly valuable to architects.

The culture of digital animation design is a similar parallel. Evolving out of a subset of filmmaking, the culture of animating by hand-drawing cartoon frames borrowed from computer science and translated into a new discipline. Surface modeling emerged as a technique for achieving desired graphic results. A craft that is sometimes similar in the output to architectural design is ultimately a different paradigm. This primacy of the graphic – the perceived – has implications for architecture, a discipline in which the making of form and space is front stage rather than backstage. Possibly more significant is the animators’ connection to, and use of, time.

This investigation places an architectural process in the path of these fluctuating disciplines. A new architecture by a new type of architect can emerge from the assimilation of knowledge, technique, and culture. It is then the role of the pre-existing culture of architecture—a culture of versioning, research, pragmatism, praxis and realization—that is uniquely suited to mediate between these sometimes conflicting agendas. And in doing so the culture of architecture can intelligently and critically engage its own traditions through its evolution.
Fig 2. Between
The space between the building envelope and the outer skin
Fig 3. Site from Maryland Avenue

week 3
Fig 4. Site plan
Located at the edge of the public memorial and museum district, within the southwest federal office core
NASA’s presence in Washington DC is the administrative and public arm of the government organization charged with the following mission: “… to pioneer the future in space exploration, scientific discovery and aeronautics research.” NASA has numerous operational facilities around the country, and of course the international space Station and hundreds of satellites currently in orbit. The NASA headquarters must embody the image of the research and operations that are consistently being engaged by NASA. However the function of the headquarters is purely administrative and political. No actual research or mission operations activities occur at the headquarters. It is the location of the NASA administrator as well as the Chiefs of the four principal organizations or “Mission directorates.” The headquarters is NASA’s interface with lobbyists of the organizations within the executive branch, Congress and the public – tourists as well as academics. NASA is apolitical although the NASA administrator is appointed by the president and the budget is entirely subject to congressional approval.

This site for this investigation is part of the office core of downtown Washington. South of the mall, this district accounts for many of the relatively permanent, by government standards, institutional offices. Significantly expanded and unified in the 1960s, this area operates on a 9-5 Monday through Friday schedule. Large,
Chief Financial Officer
Aeronautics Research
Exploration Systems
Ames Research Center specializes in research geared towards creating new knowledge and new technologies that span the spectrum of NASA interests.

Dryden Flight Research Center
As the lead for flight research, Dryden continues to innovate in aeronautics and space technology. The newest, fastest, the highest— all have made their debut in the vast, clear desert skies over Dryden.

Glen Research Center
Develops and transfers critical technologies that address national priorities through research, technology development, and systems development for safe and reliable aeronautics, aerospace, and space applications.

Goddard Space Flight Center
The mission of the Goddard Space Flight Center is to expand knowledge on the Earth and its environment, the solar system, and the universe through observations from space.

Jet Propulsion Laboratory
The Jet Propulsion Laboratory managed by the California Institute of Technology is NASA's lead center for robotic exploration of the Solar System.

Johnson Space Center
From the early Gemini, Apollo, and Sky Lab projects to today's Space Shuttle and International Space Station programs, Johnson Space Center continues to lead NASA's effort in Human Space Exploration.

Kennedy Space Center
Kennedy Space Center is America's Gateway to the Universe—world in preparing and launching missions around the Earth and beyond.

Langley Research Center
Langley continues to forge new frontiers in aviation and space research for aerospace, atmospheric sciences, and technology commercialization to improve the way the world lives.

Marshall Space Flight Center
Marshall Space Flight Center is world leader in the access to space and use of space for research and development to benefit humanity.

Stennis Space Flight Center
Stennis is responsible for NASA's rocket propulsion testing and for partnering with industry to develop and implement remote sensing technology.

Fig. 6. Program direction, connection, size and population of program leading into the building, within the building and from the building.
ominous forums are the norm here making this area stark outlier in the usually small, very, heterogeneous mix that is typical of Washington. The Department of Education, the Department of the Interior, the Bureau of engraving and printing, the Department of agriculture, the Bureau of weights and measure, the Internal Revenue Service, among others all make their homes here. At the periphery of this zone is the vibrant tourism and arts Center of Washington. Nearly all of the institutional buildings have some form of tourist component. Some, such as the presses at the Bureau of engraving and printing, are quite popular.

The specific location, at the corner of D. Street and 14th Street Southwest is on the cusp between the office core and the tourist district. One block south of the mall on 14th Street resigns the United States Holocaust Memorial Museum. Adjacent to that cultural and monumental epicenter one block to the south is the Bureau of engraving and printing, one block still further south across 14th Street is the site for this investigation. 14th Street is an important court or in and out of the city at the site is the base of the 14th Street Bridge a critical artery between downtown Washington and the state of Virginia. Hundreds of tourists find themselves, sometimes accidentally, near the site en route to a destination. Many thousands more drive adjacent to the site as commuters to and from the suburbs or Reagan National Airport.
At the time of this writing, this land is used for parking, leased by the government. Maryland Avenue, a once prominent diagonal Blvd. Tracing the connection between the United States Capitol and the Jefferson Memorial just across the title basin, terminates at the site. He exists a conglomeration of multiple levels, an undefined ground plane, multiple movement systems, an axis, important sight lines, edge barriers, and direct and indirect access by pedestrians. Beyond being symbolically meaningful as a symbolic location in a symbolic city, the same complex cities that made this site hospitable to developers and builders makes it uniquely suited as parametric fuel for an algorithmic process.

Fig 8. Dispersion of NASA mission focus to physical centers
As estimated from general descriptions made by NASA at www.nasa.gov

Fig 9. Site Railroad Edge
Existing (and assumed permanent) rail line between site and hotel. Photograph taken from terminating traffic circle of Maryland Avenue.
While engaging other disciplines, a choreographed process of returning consistently, regularly, and purposefully to the realm of architecture was choreographed over the course of this investigation. Regular discovery, consistent learning, and adaption of the goals was understood that to occur. Simultaneously, a gradual narrowing focus onto the design of the building was to be a natural and necessary occurrence. The calendars in figure x set forth production goals also. The resultant building, represented by the ‘+’ is and was not to be the final product per se. The cyclical nature of this investigation demands the product be produced throughout the range of interdisciplinary focuses, including during the effort to engage the meaning and potential of media.

The updated calendar reveals a new layer of priority revealed only a conclusion of the investigation. Programming mastery, digital technique, and computational patterns emerged as more than the study of media. Computational design as a method, was the common means by which the disciplines of art and architecture were breached, mediated, and blended. This new diagram necessitated the tracking of critical scripts. Less important than the association with the scripts on this diagram with the specifics of each set of code, is a reality that the scripts gradually were focused on an architectural rather than an abstract or purely geometric output.

**Process Calendar**
Fig 10a. (top) Process map as of week 12
Multi-linear cyclical process periodically plans to re-focus as built proposal before reaching again to more intelligent media experimentations. + symbol represents points at which a building proposal could be exported from the process. This technique is expected to become more viable and therefore more frequent as process evolves.

Fig 10b. (above) Process Map as of week 35
Updated version of calendar maps the outcome. A new layer, algorithm design, becomes necessary to convey unexpected shifts in focus.
Animation as distinctly different from video or motion is defined as the illusion of motion through the repetition of slightly varying discrete instances. Animation, whether digital or analog, consists of a series of static frames. Algorithmic process take, as input, some value and perform computation(s) based on that value. Embedded in whatever calculations take place are, generally speaking, relationships. Figure 11., for example, is the direct result of the execution of a function, the direct application of an algorithm. This particular algorithm is neither complex nor complicated. However, the relationships inherent in the algorithm are not clear given this image in isolation. A critical theoretical as well as functional link is made to vary with a parameter. When an algorithm is placed within an armature of animation the parameter $T$, time, is available as the single value governing the extent to which relationships in the algorithm are applied to the variables passed. The same script is executed $n$ times ($n$ being the number of frames) to generate an animation. As the discrete valued $T$, time, updates with each subsequent frame the, results from that particular execution of that particular frame vary slightly from the previous iteration. Because $T$, by definition a parameter, is not constant, neither are the results. The animation then, viewed as a singular event, portrays, through motion, the relationships embedded in that algorithm. Mark Goulthorpe ironically notes, “Perhaps it is not stating the

**Scripting and animation as co-catalysts**

Animation as distinctly different from video or motion is defined as the illusion of motion through the repetition of slightly varying discrete instances. Animation, whether digital or analog, consists of a series of static frames. Algorithmic process take, as input, some value and perform computation(s) based on that value. Embedded in whatever calculations take place are, generally speaking, relationships. Figure 11., for example, is the direct result of the execution of a function, the direct application of an algorithm. This particular algorithm is neither complex nor complicated. However, the relationships inherent in the algorithm are not clear given this image in isolation. A critical theoretical as well as functional link is made to vary with a parameter. When an algorithm is placed within an armature of animation the parameter $T$, time, is available as the single value governing the extent to which relationships in the algorithm are applied to the variables passed. The same script is executed $n$ times ($n$ being the number of frames) to generate an animation. As the discrete valued $T$, time, updates with each subsequent frame the, results from that particular execution of that particular frame vary slightly from the previous iteration. Because $T$, by definition a parameter, is not constant, neither are the results. The animation then, viewed as a singular event, portrays, through motion, the relationships embedded in that algorithm. Mark Goulthorpe ironically notes, “Perhaps it is not stating the
obvious to suggest that animation animates(!), produces an effect..."³

Basic and fundamental process empowers and a computational design methodology with the potential for spontaneity, surprise and discovery. It is through these means that programming, scripting, and algorithm design generally can be applied has more than tools to solve known problems but rather, mediums by which to explore, design and create. The reverse linkage is available between motion graphics and film as art form and the relationship-laden reality of built architecture. Film and motion graphics can therefore be used as precedent and generator where the previous role is usually limited to that of inspiration.
Every animation in this series is created using this algorithm designed by author. The overall loop runs a set of functions which create and operate on and with diagrammatic elements for each frame (numbered T).

Functions for each parameter are called with relevant arguments. Each of which includes the value of ‘T.’ The parametric variable, time is a requisite of the functions which are designed to produce oscillating results. A critical aspect of the algorithm is that each frame is an entirely clean slate. After exporting the frame image all non-site objects are deleted from the modeling environment. This generate-new-per frame method (as opposed to a translation-based algorithm) allows for limitless parameterization: given this algorithm, any executable function (pre-existing in the software of otherwise designed) becomes parametrically controlled and animated.
Making animations traditionally involves a mix of initiating animation with key-framing—establishing the “start” and “end” values and automating the establishment of discrete instances in between which gradually gradiate between the conditions—and a crafted series of modifications to the relationships of geometries which can be expressed cyclically or gradually over time. The former paradigm places an emphasis (evident even in the positive connotations of the term “key”) on the known extremes. This allows for high degree of control over an exact output from a predetermined animation algorithm. Choreographed output is far from the sought level of design input however, assuming that indeed no bias is placed on and no assumptions made about a particular kind of normalized response design response. Instead a procedure is established wherein design relationships are established then the extremes are met and exceeded through the animation.

Indirect representation (through isolation of variables) of the animation is possible. This allows for relationships to be set between elements in terms of transformative processes that can be themselves repeated to create the animation. Time becomes the canvas on which these relationships are overlaid and overlapped and repeated. These relationships can create new relationships via their explicit representation.
The animate artifact, movie, is a valuable and viable architectural condition. While movies in architecture exist often as indirect representation (as presentation or as generator of static form early in the process), animation can and should be a media in which architecture exists. The tradition “designing the abstraction,” in design studio enriches compositional instincts and promotes a focus on process over product. Tim Durfee and Terry Surjun note regarding their animation studios at Sci-Arc that, “…coursework pursues a prevailing interest in viewing digital tools as vehicles to address not only the techniques of construction but also, in Paul Virilio’s terms, the ‘construction of technique’—that is, the rebuilding of architectural and spatial perception necessitated by ever-evolving contemporary conditions.” In these conditions, representation can be direct and animate. An environment privy to space and time incorporates phenomenology and experience into the abstraction (rather than as methods by which we see and interpret the abstraction).

In many of the animations referenced here, the artifact--the media--is in itself formal. This is in opposition to the paradigm that exists typically architectural animation in which the animate artifact serves to represent and clarify a predetermined formal condition. This narrowly useful attitude is rarely evident in cinema as it is an expressive and communicative medium.
Fig 14. [top] Part-Whole Animation Experiment 1
Exploring the shifting perception of an individual form to field of identical forms. Animation content playable at www.0095b6.com/lostritto

Fig 15. Frames from visualizing parameters experiment 2
Parametric compression and expansion of double-layered surface. Animation content playable at www.0095b6.com/lostritto
Embracing a slight shift in conceptual direction (and in doing so moving closer to the broad agendas of cinematic designers) allows for the production of animations to be just as procedural, systematic and prescriptive in terms of methods. The outcome is, as a result, left uncontrolled and unpredicted. Animations then come to represent a documented, imaged set of relationships. Without animation as a design (and therefore, by necessity, presentation) medium the architect is faced with what can seem to be an impossible burden: Representing geometric relationships is often best achieved through the drawing of
Fig 17 (top). Frames from visualizing parameters experiment 4
"Skinning" a single-layer surface with geometry of that surface parametrically varied with time. Animation content playable at www.0095b6.com/lostritto

Fig 18 (above). Frames from visualizing parameters experiment 0
Camera settings varying with parameter time against static field. Animation content playable at www.0095b6.com/lostritto
Fig 19a, 19b. Frames from visualizing parameters framework 7
Site and building code restrictions vary with parameter time. Animation content playable at www.0095b6.com/iostritto
Fig 20 [top]. Frames from visualizing parameters framework 7
Site and building code restrictions vary with parameter time. Animation content playable at www.0095b6.com/lostritto

Fig 21 [above]. Frames from visualizing parameters framework 5
Skinning algorithm applied to double layer surface animation. Animation content playable at www.0095b6.com/lostritto
geometry in space. The very act of static drawing of geometry though involves the drawing of a singular geometry or set of geometries. Without a willingness to embrace symbolism it becomes difficult to represent relationships without implying a specific manifestation of that relationship. This challenge serves to provide the exigent or at least the fuel and direction for animate experimentation and will conversely re-appear in further explorations into the translation of moving images into static architecture.

Animations can be considered collections of instances. In one sense an animation making explicit a set of design parameters typically found in the generation of design schemes could be considered to be a few hundred “schemes,” (one
scheme for each frame). Process that uses animation though tends to favor an alternative type of design thinking--a spiraling rather than narrowing hierarchy of values. The parti is this singular attitude with a near-inifinitely mutiplicious set of solutions.

It the becomes critical to systematically limit, though editing, honing and deleting in a constructed process that tends toward further branching. The animations are evaluated in terms of their ability to imply (explicitly or implicitly) translation into building or concrete built conditions that can perform in as the animations have. This becomes a scientific process of hypothesizing and testing. Complete building solutions are arrived at early in the process as concrete manifestations are necessary to properly and rigorously analyze and edit initial assumptions. Furthermore, the scripted animation interface promotes the convenient adjustment of parameters and their ability to conference each other.
Fig 24. Frames exposing relationships in parabolic skinning algorithm
Animation parameter controls sectional variation necessary to display discrete instances through surfaces. Animation content playable at www.0996b6.com/lostritto
Fig 25. Frames exposing relationships in outer skin algorithm
Animation parameter controls virtual camera motion necessary to display discrete instances through and around surfaces. Animation content playable at www.009506.com/lostritto
Fig 26a. Frames from physical model construction process animation
Animation content playable at www.00966.com/lostritto
Fig 26b. Frames from physical model construction process animation
Animation content playable at www.0095b6.com/fostritto
Design of Algorithms

Algorithms are, by their nature, repositories of relationships. Comparisons and operations are applied to a given or a set of given values. Equally important is the reality that all algorithms are designed. Technically, code is written. It is a language. The infinite potential and virtually limitless scope of computation is appropriately analogous to a blank canvas of an artist. An important distinction can be made between the application of script and four automation of software—making tedious or repeated processes faster or more convenient—and what is engaged here, scripting using low-level data structures, basic logical Boolean operators, and foundational mathematic operations. In some respects, this represents a distinction between user of software and designer with software. This allows for the distinction between software as tool and digital media as language.

The most successful algorithms are general and efficient. Algorithms do not necessarily reflect how humans tend to approach problem solving, which is through the implementation of heuristics and pattern recognition. There are certain inherently human processes that we perform automatically, such as immediate recollection from memory. These would be very difficult to translate directly into a computational process. Other processes are naturally computational, such as simulations, self-repetition and growth.
The necessity for efficiency is a useful restriction. Thresholds of memory available and computation time are easily reached. For this reason, efficiency is critical. In computer science, efficiencies are classified by order of magnitude equations. They are expressed in terms of n where an order of magnitude n is highly efficient compared to an order of magnitude of n². An order of magnitude of nⁿ is highly inefficient. Measuring efficiency is one way by which Algorithms can be broadly compared with one another. Through these means, patterns in types of approaches emerge. Furthermore, more efficiency is often associated with higher degrees of generalization, meaning the same operational structure can be reused from multiple, sometimes unrelated applications.

Clarity is another valuable restriction that emerges out of certain computational necessities. Often, within an algorithm the same set of operations may be executed over and over. For the sake of legibility, clarity and editability, it makes sense to further parse these repeating operations into independent functions. These functions can then be referenced repeatedly in any number of algorithms. This facilitates the distinction between computational algorithms – operations like sorting, grouping, randomizing, etc. – with generative algorithms that are directly responsible for the generation of forms and space in the modeling environment.

As technical as a discussion of this process can be, there is meaning beyond these issues in "Algorithmic Architecture," Kostas Terzidis points out that, “An algorithm is not only a computer implementation, a series of lines of code in a program, or a language, it is also a theoretical construct with deep philosophical, social, design, and artistic repercussions...a trace back to the origin of design a conceptual activity...a definition of] the inconceivable, impossible or unknown... a historical and critical perspective on the recently emerging are of algorithms in architecture...differentiated from CAD or computer graphics..."
real[3]
\[x[0-2], y[0-2], z[0-2]\]
point[2][n]

real[3][n]
dimensions[3][n]

boolean[n]
dimensions[3][n]
Function pack (arrBlocks, XRange, YRange, ZRange, arrOrigin, gap)

'First, sort array in descending order of volume
Dim newArray()
ReDim newArray (UBound(arrBlocks))
Dim largestVolume
Dim largestID
Dim i
Dim j
For i=0 To UBound(newArray)
    For j=0 To UBound(newArray)
        If (arrBlocks(j)(6)>largestVolume) Then
            largestID=j
            largestVolume=arrBlocks(j)(6)
        End If
    Next
    newArray(i)=arrBlocks(largestID)
    arrBlocks(largestID)(6)=0
    largestVolume=0
    largestID=0
Next

'Position in best possible position
Dim k
Dim n
Dim outofBoundsX
Dim outofBoundsY
Dim outofBoundsZ
Dim shiftnecessary
For k=0 To UBound(newArray)
    shiftnecessary=False
    Call Rhino.MoveObject(newArray(k)(0), newArray(k)(1), arrOrigin)
    newArray(k)(1)=arrOrigin
    For n=0 To UBound(newArray)
        If (n<>k) Then
            If (testIntersect(newArray(k)(0),newArray(n)(0))) Then
                shiftnecessary=True
            End If
        End If
    Next
    Do While ((shiftnecessary)And (((newArray(k)(1)(0)+newArray(k)(2))<arrOrigin(0)+Xrange)))
        Call Rhino.print("inside the x move loop")
        Call Rhino.print(k)
        Do While ((shiftnecessary) And (((newArray(k)(1)(1)+newArray(k)(3))<arrOrigin(1)+Yrange)))
            Call Rhino.print("inside the y move loop")
            Call Rhino.print(k)
            Do While ((shiftnecessary) And (((newArray(k)(1)(2)+newArray(k)(4))<arrOrigin(2)+Zrange)))
                Call Rhino.print("inside the z move loop")
                Call Rhino.MoveObject(newArray(k)(0), Array(0,0,gap), newArray)
                newArray(k)(1)(2)=newArray(k)(1)(2)+gap
                shiftnecessary=testIntersectArray(newArray(k)(0), newArray)
            Loop
        End If
    Loop
    If (shiftnecessary) Then
        Call Rhino.MoveObject(newArray(k)(0), Array(0,0,0), Array(0,gap,0))
        newArray(k)(1)(1)=newArray(k)(1)(1)+gap
        Call Rhino.MoveObject(newArray(k)(0), newArray(k)(1), Array(newArray(k)(1)(0),arrOrigin(1),arrOrigin(2)))
        newArray(k)(1)(2)=arrOrigin(2)
        shiftnecessary=testIntersectArray(newArray(k)(0), newArray)
    End If
    Loop
Next
End Function
sun angle

Maryland Avenue to Jefferson Memorial sight line

point(s) of interest (pedestrian and automobile) ; associative trajectories

Publicly accessible program position

Publicly accessible program volume

program type (open or closed)

signage zone

base surfaces
real
vector
vector[n];vector[n]

real [3][n]
dimensions[3][n]

boolean[n]

surface solid[n]

surface(u,v)[n]
Function buildSkin()
    Dim surface
    Dim rotAxis
    Dim insideFlag
    Dim insideTextFlag
    Dim thisLayer
    Dim wireLayer
    Dim boolTextSkin
    Dim pointsForThisLine(1)
    Dim textObjects
    Dim newPoint1, newPoint2
    textObjects = rhino.ObjectsByLayer("text")
    Call rhino.EnableRedraw(vbtrue)
    surface = rhino.GetObject("select surface", 8)
    Call Rhino.LayerVisible("mainSkin1", vbfalse)
    boolTextSkin = Rhino.GetBoolean("text", array("text", "false", "true"), array(False))
    thisLayer = rhino.GetLayer("select layer to put surfaces on", , vbtrue, vbtrue)
    wireLayer = rhino.GetLayer("select layer to put wire objects on", , vbtrue, vbtrue)
    Dim rows, cols
    rows = Rhino.GetInteger("rows: " ,20 ,3 , 400)
    cols = Rhino.GetInteger("cols: " ,20 ,3 , 400)
    Dim point
    Dim differencePoint
    Dim x, y, z
    x = Rhino.GetReal("eyeX " ,-100 ,-9000 , 9000)
    y = Rhino.GetReal("eyeY " ,100 ,-9000 , 9000)
    z = Rhino.GetReal("eyeZ " ,20 ,-9000 , 9000)
    Call rhino.EnableRedraw(vbfalse)
    Dim eyePoint: eyePoint = array(x, y, z + .001)
    Dim adjustmentVector
    Dim U, V, t(1)
    Dim a, b, c
    Dim bprev, cprev
    Dim vector
    Dim xyrot()
    ReDim xyrot(rows, cols)
    Dim zrot()
    ReDim zrot(rows, cols)
    Dim k
    Dim f
    Dim pointShift
    Dim normalVector
    Dim fins()
    Dim startFrame
    startFrame = 28
    Dim thisLine1(), thisLine2(), thisLine3(), thisLine4()
    ReDim thisLine1(rows, cols)
    ReDim thisLine2(rows, cols)
    ReDim thisLine3(rows, cols)
    ReDim thisLine4(rows, cols)
    Dim density
    ReDim fins(rows, cols)
    Dim arrFinsExist() ' 0 if no fin in that position, 1 if opposite
    ReDim arrFinsExist(rows, cols)
    For b = 0 To rows
        For c = 0 To cols
            arrFinsExist(b, c) = 0
        Next
    Next
    U = Rhino.SurfaceDomain(surface, 0)
    V = Rhino.SurfaceDomain(surface, 1)
    For f = startFrame To 70
        Call rhino.print("frame " & CStr(f))
        For b = 0 To rows - 1
            t(0) = U(0) + (((U(1) - U(0)) / rows) * b)
            Call rhino.Print("modelingskin row " & CStr(b) & " of " & CStr(rows))
            For c = 0 To cols - 1
                t(1) = V(0) + (((V(1) - V(0)) / cols) * c)
                insideFlag = vbfalse
                insideTextFlag = vbfalse
                point = Rhino.EvaluateSurface(surface, t)
                'call rhino.AddLine(point, eyePoint)
                For k = 0 To Ubound(cuttingObjects)
                    If (Rhino.IsPointInSurface(cuttingObjects(k), point)) Then
                        insideFlag = vbtrue
                    End If
                    Next
                For k = 0 To ubound(textObjects)
If Rhino.IsPointInSurface(textObjects(k), point) Then
    insideTextFlag=vbtrue
End If
Next

If boolTextSkin Then
    If insideTextFlag Then
        insideTextFlag=vbfalse
    Else
        insideTextFlag=vbtrue
    End If
End If

normalVector = Rhino.SurfaceNormal(surface, t)
density=CInt(5*(normalVector(1)*5-normalVector(0)-normalVector(2)))
If density<1 Then density=1
Call rhino.print(CStr(density))
If Not ((b+c)Mod density =0) Then
    insideFlag=True
End If
If (insideFlag=vbfalse)And (insideTextFlag=False) Then
    arrFinsExist(b,c)=1
    vector= Rhino.VectorCreate(point, eyePoint)
    pointShift=rhino.PointSubtract(point,array(1,1,0))
    If f=startFrame Then fins(b,c)=createFin()
    Call rhino.print(fins(b,c)(
    place on surface
    CurveStartPoint(fins(b,c)(3)), Array(point,pointAdd(point,normalVector),eyePoint),eyePoint)
    'orient sy
    CurveEndPoint(fins(b,c)(3)), Array(point,pointAdd(point,normalVector),eyePoint)
    'pivot z
    CurveStartPoint(fins(b,c)(6)), Array(point,pointAdd(point,normalVector),eyePoint)
    CurveEndPoint(fins(b,c)(6)),Array(point,pointAdd(point,normalVector),eyePoint)
    If (f=startFrame) Then -----------------------------
    'connect
    If (b>0) Then
        If (arrFinsExist(b-1,c)=1) Then
            thisLine1(b,c)=rhino.addLine(rhino.
            CurvestartPoint(fins(b,c)(7)),rhino.CurveStartPoint(fins(b-1,c)(7)))
            Call rhino.objectLayer(thisLine1(b,c),wireLayer)
        Else
            bprev=b
            Do While ((Not(arrFinsExist(bprev-1,c)=1)) And
                      (bprev>1))
                bprev=bprev-1
            Loop
            If (bprev>0) Then
                If (arrFinsExist(bprev-1,c)=1) Then
                    thisLine1(b,c)=rhino.
                    addLine(rhino.CurveEndPoint(fins(b,c)(7)),rhino.
                    CurveStartPoint(fins(bprev-1,c)(7)))
                    Call rhino.objectLayer
                End If
            Else
                cprev=c
                bprev=b
                If (bprev>0) And (cprev>0) Then
                    Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And
                              (cprev>0) And (bprev>0))
                        If (bprev>0) Then
                            bprev=bprev-1
                        Else
                            bprev=b
                            cprev=cprev-1
                        End If
                    Loop
                    thisLine4(b,c)= rhino.
                    AddLine(rhino.CurveEndPoint(fins(b,c)(6)),rhino.
                    CurveStartPoint(fins(bprev-1,cprev-1)(3)))
                    Call rhino.objectLayer
                Else
                    cprev=c
                    bprev=b
                    If (bprev>0) And (cprev>0) Then
                        Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And
                                  (cprev>1) And (bprev>1))
                            If (bprev>0) Then
                                bprev=bprev-1
                            Else
                                bprev=b
                                cprev=cprev-1
                            End If
                        Loop
                        If (arrFinsExist(bprev-1,cprev-1)=1)
                            Then
                                thisLine2(b,c)= rhino.
                                AddLine(rhino.CurveEndPoint(fins(b,c)(6)),rhino.
                                CurveStartPoint(fins(bprev-1,cprev-1)(3)))
                                Call rhino.objectLayer
                            Else
                                prevprev=prev
                                prev=prev+1
                                If prev=prev1 And (prev>prev0) Then
                                    Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                              (prev>0) And (prev0))
                                            If (prev>0) Then
                                                prevprev=prev
                                                prev=prev-1
                                            Else
                                                prevprev=prev
                                                prev=prev-1
                                            End If
                                        Loop
                                        AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                        CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                        Call rhino.objectLayer
                                    Else
                                        If (prev=prev1) And (prev>prev0) Then
                                            Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                                      (prev>0) And (prev0))
                                                        If (prev>0) Then
                                                            prevprev=prev
                                                            prev=prev-1
                                                        Else
                                                            prevprev=prev
                                                            prev=prev-1
                                                        End If
                                                    Loop
                                                    AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                                    CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                                    Call rhino.objectLayer
                                                End If
                                            End If
                                        End If
                                    End If
                                End If
                            End If
                        End If
                    Else
                        cprev=c
                        bprev=b
                        If (bprev>0) And (cprev>0) Then
                            Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And
                                      (cprev>1) And (bprev>1))
                                    If (bprev>0) Then
                                        bprev=bprev-1
                                    Else
                                        bprev=b
                                        cprev=cprev-1
                                    End If
                                Loop
                                If (arrFinsExist(bprev-1,cprev-1)=1)
                                    Then
                                        thisLine2(b,c)= rhino.
                                        AddLine(rhino.CurveEndPoint(fins(b,c)(6)),rhino.
                                        CurveStartPoint(fins(bprev-1,cprev-1)(3)))
                                        Call rhino.objectLayer
                                    Else
                                        prevprev=prev
                                        prev=prev+1
                                        If prev=prev1 And (prev>prev0) Then
                                            Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                                      (prev>0) And (prev0))
                                                        If (prev>0) Then
                                                            prevprev=prev
                                                            prev=prev-1
                                                        Else
                                                            prevprev=prev
                                                            prev=prev-1
                                                        End If
                                                    Loop
                                                    AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                                    CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                                    Call rhino.objectLayer
                                                Else
                                                    If (prev=prev1) And (prev>prev0) Then
                                                        Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                                                  (prev>0) And (prev0))
                                                            If (prev>0) Then
                                                                prevprev=prev
                                                                prev=prev-1
                                                            Else
                                                                prevprev=prev
                                                                prev=prev-1
                                                            End If
                                                        Loop
                                                        AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                                        CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                                        Call rhino.objectLayer
                                                    End If
                                                End If
                                            End If
                                        End If
                                    End If
                                End If
                            End If
                        End If
                    End If
                End If
            Else
                cprev=c
                bprev=b
                If (bprev>0) And (cprev>0) Then
                    Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And
                              (cprev>1) And (bprev>1))
                            If (bprev>0) Then
                                bprev=bprev-1
                            Else
                                bprev=b
                                cprev=cprev-1
                            End If
                        Loop
                        If (arrFinsExist(bprev-1,cprev-1)=1)
                            Then
                                thisLine2(b,c)= rhino.
                                AddLine(rhino.CurveEndPoint(fins(b,c)(6)),rhino.
                                CurveStartPoint(fins(bprev-1,cprev-1)(3)))
                                Call rhino.objectLayer
                            Else
                                prevprev=prev
                                prev=prev+1
                                If prev=prev1 And (prev>prev0) Then
                                    Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                              (prev>0) And (prev0))
                                                If (prev>0) Then
                                                    prevprev=prev
                                                    prev=prev-1
                                                Else
                                                    prevprev=prev
                                                    prev=prev-1
                                                End If
                                            Loop
                                            AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                            CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                            Call rhino.objectLayer
                                        Else
                                            If (prev=prev1) And (prev>prev0) Then
                                                Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                                                          (prev>0) And (prev0))
                                                    If (prev>0) Then
                                                        prevprev=prev
                                                        prev=prev-1
                                                    Else
                                                        prevprev=prev
                                                        prev=prev-1
                                                    End If
                                                Loop
                                                AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
                                                CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
                                                Call rhino.objectLayer
                                            End If
                                        End If
                                    End If
                                End If
                            End If
                        End If
                    End If
                End If
            End If
        End If
    End If
End If

1. prev1=pрев1; prev0=pрев0; prevprev=prev
   If (prev=prev1) And (prev>prev0) Then
       Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                 (prev>0) And (prev0))
         If (prev>0) Then
             prevprev=prev
             prev=prev-1
         Else
             prevprev=prev
             prev=prev-1
         End If
     Loop
     If (prev=prev1) And (prev>prev0) Then
         Do While ((Not(arrFinsExist(prev-1,pрев-1-pрев-0)=1)) And
                   (prev>0) And (prev0))
             If (prev>0) Then
                 prevprev=prev
                 prev=prev-1
             Else
                 prevprev=prev
                 prev=prev-1
             End If
         Loop
         AddLine(rhino.CurveEndPoint(fins(b,c)(3)),rhino.
         CurveStartPoint(fins(prev-1,pрев-1-pрев-0)(3)))
         Call rhino.objectLayer
     End If
```

```
If (bprev>0) And (cprev>0) Then
  Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (cprev>1) And (bprev>1))
    If (cprev>1) Then
      cprev=cprev-1
    Else
      cprev=c
    End If
    If (bprev>0) Then
      bprev=bprev-1
    End If
  Loop
  If (arrFinsExist(bprev-1,cprev-1)=1) Then
    newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
    newPoint2=rhino.CurveStartPoint(fins(b,c-1)(7))
    thisLine2(b,c)= Rhino.OrientObject (thisLine2(b,c),array(rhino.CurvestartPoint(thisLine2(b,c)),rhino.CurveEndPoint(thisLine2(b,c))), array(newPoint1,newPoint2) ,2)
  End If
End If
Else
  'connect
  'Call rhino.print("in the else for second frame")
  If (b>0) Then
    If (arrFinsExist(b-1,c)=1) Then
      newPoint1=rhino.CurvestartPoint(fins(b,c)(7))
      newPoint2=rhino.CurveStartPoint(fins(b-1,c)(7))
      thisLine1(b,c)=Rhino.OrientObject (thisLine1(b,c), array(rhino.CurvestartPoint(thisLine1(b,c)),rhino.CurveEndPoint(thisLine1(b,c))), array(newPoint1,newPoint2) ,2)
    Else
      bprev=b
      Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (bprev>1))
        bprev=bprev-1
      Loop
      If (bprev>0) Then
        If (arrFinsExist(bprev-1,cprev-1)=1) Then
          newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
          newPoint2=rhino.CurveStartPoint(fins(bprev-1,cprev-1)(3))
          thisLine4(b,c)= Rhino.OrientObject (thisLine4(b,c), array(rhino.CurvestartPoint(thisLine4(b,c)),rhino.CurveEndPoint(thisLine4(b,c))), array(newPoint1,newPoint2) ,2)
        End If
      End If
    End If
  End If
  If (c>0) Then
    If (arrFinsExist(b,c-1)=1) Then
      newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
      newPoint2=rhino.CurveStartPoint(fins(b,c-1)(7))
      thisLine3(b,c)= Rhino.OrientObject (thisLine3(b,c),array(rhino.CurvestartPoint(thisLine3(b,c)),rhino.CurveEndPoint(thisLine3(b,c))), array(newPoint1,newPoint2) ,2)
    End If
  End If
End If
Else
  'connect
  'Call rhino.print("in the else for second frame")
  If (b>0) Then
    If (arrFinsExist(b-1,c)=1) Then
      newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
      newPoint2=rhino.CurveStartPoint(fins(b-1,c)(7))
      thisLine1(b,c)=Rhino.OrientObject (thisLine1(b,c), array(rhino.CurvestartPoint(thisLine1(b,c)),rhino.CurveEndPoint(thisLine1(b,c))), array(newPoint1,newPoint2) ,2)
    Else
      bprev=b
      Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (bprev>1))
        bprev=bprev-1
      Loop
      If (bprev>0) Then
        If (arrFinsExist(bprev-1,cprev-1)=1) Then
          newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
          newPoint2=rhino.CurveStartPoint(fins(bprev-1,cprev-1)(3))
          thisLine4(b,c)= Rhino.OrientObject (thisLine4(b,c), array(rhino.CurvestartPoint(thisLine4(b,c)),rhino.CurveEndPoint(thisLine4(b,c))), array(newPoint1,newPoint2) ,2)
        End If
      End If
    End If
  End If
  If (c>0) Then
    If (arrFinsExist(b,c-1)=1) Then
      newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
      newPoint2=rhino.CurveStartPoint(fins(b,c-1)(7))
      thisLine3(b,c)= Rhino.OrientObject (thisLine3(b,c),array(rhino.CurvestartPoint(thisLine3(b,c)),rhino.CurveEndPoint(thisLine3(b,c))), array(newPoint1,newPoint2) ,2)
    End If
  End If
End If
Else
  'connect
  'Call rhino.print("in the else for second frame")
  If (b>0) Then
    If (arrFinsExist(b-1,c)=1) Then
      newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
      newPoint2=rhino.CurveStartPoint(fins(b-1,c)(7))
      thisLine1(b,c)=Rhino.OrientObject (thisLine1(b,c), array(rhino.CurvestartPoint(thisLine1(b,c)),rhino.CurveEndPoint(thisLine1(b,c))), array(newPoint1,newPoint2) ,2)
    Else
      bprev=b
      Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (bprev>1))
        bprev=bprev-1
      Loop
      If (bprev>0) Then
        If (arrFinsExist(bprev-1,cprev-1)=1) Then
          newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
          newPoint2=rhino.CurveStartPoint(fins(bprev-1,cprev-1)(3))
          thisLine4(b,c)= Rhino.OrientObject (thisLine4(b,c), array(rhino.CurvestartPoint(thisLine4(b,c)),rhino.CurveEndPoint(thisLine4(b,c))), array(newPoint1,newPoint2) ,2)
        End If
      End If
    End If
  End If
  If (c>0) Then
    If (arrFinsExist(b,c-1)=1) Then
      newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
      newPoint2=rhino.CurveStartPoint(fins(b,c-1)(7))
      thisLine3(b,c)= Rhino.OrientObject (thisLine3(b,c),array(rhino.CurvestartPoint(thisLine3(b,c)),rhino.CurveEndPoint(thisLine3(b,c))), array(newPoint1,newPoint2) ,2)
    End If
  End If
End If
Else
  'connect
  'Call rhino.print("in the else for second frame")
  If (b>0) Then
    If (arrFinsExist(b-1,c)=1) Then
      newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
      newPoint2=rhino.CurveStartPoint(fins(b-1,c)(7))
      thisLine1(b,c)=Rhino.OrientObject (thisLine1(b,c), array(rhino.CurvestartPoint(thisLine1(b,c)),rhino.CurveEndPoint(thisLine1(b,c))), array(newPoint1,newPoint2) ,2)
    Else
      bprev=b
      Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (bprev>1))
        bprev=bprev-1
      Loop
      If (bprev>0) Then
        If (arrFinsExist(bprev-1,cprev-1)=1) Then
          newPoint1=rhino.CurveStartPoint(fins(b,c)(7))
          newPoint2=rhino.CurveStartPoint(fins(bprev-1,cprev-1)(3))
          thisLine4(b,c)= Rhino.OrientObject (thisLine4(b,c), array(rhino.CurvestartPoint(thisLine4(b,c)),rhino.CurveEndPoint(thisLine4(b,c))), array(newPoint1,newPoint2) ,2)
        End If
      End If
    End If
  End If
  If (c>0) Then
    If (arrFinsExist(b,c-1)=1) Then
      newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
      newPoint2=rhino.CurveStartPoint(fins(b,c-1)(7))
      thisLine3(b,c)= Rhino.OrientObject (thisLine3(b,c),array(rhino.CurvestartPoint(thisLine3(b,c)),rhino.CurveEndPoint(thisLine3(b,c))), array(newPoint1,newPoint2) ,2)
    End If
  End If
End If
CurveStartPoint(fins(bprev-1,cprev-1)(3))
OrientObject (thisLine2(b,c), array(rhino.CurveStartPoint(thisLine2(b,c)), rhino.CurveEndPoint(thisLine2(b,c))), array(newPoint1, newPoint2), 2)
If (thisLine2(b,c)=Null) Then Call rhino.print("fuck")

Do While ((Not(arrFinsExist(bprev-1,cprev-1)=1)) And (cprev>1) And (bprev>1))
If (cprev>1) Then
cprev=cprev-1
Else
cprev=c
If (bprev>0) Then
bprev=bprev-1
End If
End If
Loop
If (arrFinsExist(bprev-1,cprev-1)=1) Then
newPoint1=rhino.CurveEndPoint(fins(b,c)(6))
newPoint2=rhino.CurveStartPoint(fins(bprev-1,cprev-1)(3))
thisLine3(b,c)=Rhino.OrientObject (thisLine3(b,c), array(rhino.CurvestartPoint(thisLine3(b,c)), rhino.CurveEndPoint(thisLine3(b,c))), array(newPoint1, newPoint2), 2)
If (thisLine3(b,c)=Null) Then Call rhino.print("fuck")
End If
End If

End If

Next
Next

' set layers
If (f=startFrame) Then
For b=0 To rows-1
For c=0 To cols-1
If (arrFinsExist(b,c)=1) Then
Call rhino.objectLayer(array(fins(b,c)(1),fins(b,c)(2),fins(b,c)(3),fins(b,c)(5),fins(b,c)(6),fins(b,c)(7)),"hide_wire")
Call rhino.objectLayer(array(fins(b,c)(0),fins(b,c)(4)),thisLayer)
End If
Next
Next
End If

' export image
Call rhino.deleteObjects("stop")
Rhino.EnableRedraw vbTrue
Call Rhino.Command("_ViewCaptureToFile"&" skin1_c"& CStr(f)& "jpg d r a Enter")
Rhino.EnableRedraw vbFalse

' reset rotations
For b=0 To rows-1
For c=0 To cols-1
If (arrFinsExist(b,c)=1) Then
Call rhino.RotateObjects(array(fins(b,c)(0),fins(b,c)(1),fins(b,c)(3),fins(b,c)(6)), rhino.CurveEndPoint(fins(b,c)(2)),-zrot(b,c), rhino.VectorCreate(rhino.CurvestartPoint(fins(b,c)(2)),rhino.CurveEndPoint(fins(b,c)(2))))
Call Rhino.RotateObjects (fins(b,c), rhino.CurveStartPoint(fins(b,c)(5)), -xyrot(b,c))
End If
Next
Next

eyePoint(1)=eyePoint(1)-7.5
eyePoint(2)=eyePoint(2)+.8
Call rhino.Print("eye point: (" & CStr(eyePoint(0)) & ", " & CStr(eyePoint(1))& ", " & CStr(eyePoint(2)))
Next
End Function
latitude;longitude

publicly accessible program position

publicly accessible program volume

program type (open or closed)

program bounds

base
Option Explicit
' Script written by <insert name>
' Script copyrighted by <insert company name>
' Script version Saturday, March 22, 2008 11:08:23 PM

Call Main()
Sub Main()
Call skin2Solar
End Sub

Function skin2Solar
Rhino.EnableRedraw vbFalse
Dim sunPoint: sunPoint=array(27500,-12500,100000)
Dim vectorToSun
Dim tempSet: tempset=Rhino.AllObjects
Dim skin2_a
Dim f
For f=0 To UBound(tempSet)
If (Rhino.ObjectLayer(tempSet(f))="skin2_a")Then
skin2_a=tempSet(f)
End If
Next
Dim arrFocusLines(14)
Dim increment: increment=280/14
Dim i
For i=0 To 14
arrFocusLines(i)=rhino.addLine(array(-20,44+i*increment,82),array(435,44+i*increment,82))
Next
Dim resolutionX: resolutionX=10
Dim incrementX: incrementX=340/resolutionX
Dim focusPoint
Dim vertexPoint
Dim p
Dim t
Dim j
Dim arrCurvePoints()
ReDim arrCurvePoints(0)
Dim arrCurves(10)
Dim closePoint
Dim testLine
Dim currLength: currLength=0
Dim guideLine
Dim contourLines()
Dim contourFlag
For i=0 To ubound(arrFocusLines)
contourFlag=vbfalse
For j=0 To 455 Step 45.5
testLine=rhino.addline(array(-20+j,Rhino.CurveEndPoint(arrFocusLines(i))(1),0),array(-20+j,Rhino.CurveEndPoint(arrFocusLines(i))(1),200))
If (Not isNull(Rhino.CurveSurfaceIntersection(testLine, skin2_a))) Then
focusPoint=array(-20+j,Rhino.CurveEndPoint(arrFocusLines(i))(1),88)
closePoint=rhino.SurfaceClosestPoint(skin2_a, focusPoint)
vertexPoint=rhino.EvaluateSurface(skin2_a, closePoint)
p=Rhino.Distance (focusPoint, vertexPoint)
call rhino.print(“p: “ & CStr(p))
t=0
currLength=0
Erase arrCurvePoints
ReDim arrCurvePoints(0)
Do While(currLength<=16)
arrCurvePoints(t)=array(2*P*(t*.05)+vertexPoint(0),p*(t*.05)*(t*.05)+vertexPoint(1),vertexPoint(2))
If (ubound(arrCurvePoints)>1) Then
If Not (ubound(arrCurvePoints)=2) Then call rhino.deleteObject(arrCurves(j/45.5))
arrCurves(j/45.5)=rhino.addcurve(arrCurvePoints)
currLength=Rhino.CurveLength(arrCurves(j/45.5))
ReDim Preserve arrCurvePoints(ubound(arrCurvePoints)+1)
t=t+1
End If
End IF
End If
End If
End If
GuideLine=rhino.addLine(vertexPoint,array(vertexPoint(0),vertexPoint(1)+5,vertexPoint(2)))
GuideLine=Rhino.ScaleObject(GuideLine, vertexPoint, array(-1,1,1) ,vbfalse)
arrCurves(j/45.5)=Rhino.ScaleObject(arrCurves(j/45.5), vertexPoint, array(-1,1,1) ,vbfalse)
arrCurves(j/45.5)=Rhino.OrientObject(arrCurves(j/45.5), array(vertexPoint,array(vertexPoint(0),vertexPoint(1)+5,vertexPoint(2))), array(vertexPoint,Rhino.VectorAdd (vertexPoint, vectorToSun)))
arrCurves(j/45.5, vertexPoint,-90)
If (contourFlag=vbfalse) Then
ReDim Preserve contourLines(ubound(contourLines)+1)
contourLines(ubound(contourLines))=arrCurves(j/45.5)
contourFlag=vbtrue
Else
ReDim Preserve contourLines(ubound(contourLines))
End If
Line's delete(ubound(contourLines)+1)
Next
Call rhino.DeleteObject(testLine)
Next
Call rhino.AddLoftSrf(contourLines)
Next
Rhino.EnableRedraw vbTrue
End Function
property lines

publicly accessible program position

publicly accessible program volume

program bounds

week 26-28
point[2][n]

real [3][n]

dimensions[3][n]

real[3][8]

in

out
property lines

publicly accessible program position

publicly accessible program volume

program bounds
point[2][n]

real [3][n]
dimensions[3][n]

real[3][8]
program bounds

week 19-21
Meaning in the Scripted Image

A cursory analysis of car commercials reveals that American society is not isolated from trends in architectural design. Mainstream society, the media and, by extension, a powerful class of image-makers within our society (other than architects and graphic designers), television advertising designers, are perhaps illiterate in the language of architecture but apparently highly appreciate social value of its image.

Car commercials are set generally in four types of environments. 1.) the landscape: either pristine or rugged. 2) historic city or pseudo historic environment –This seems to be done to either exaggerate the car or car’s balance of timeless value and contemporary design or to validate sociocultural context by which the car allegedly should be associated. 3.) the non-environment: infinitely black or white background, no horizon line 4.) Contemporary Architecture, which is a surprisingly regular choice for these directors. In car commercials, the visual focus is obviously the car. Any architectural backdrop is used, as is implied by the term, as a kind of stage set. The architecture is hard to recognize often, even to architects.

An assumption made based on the apparent logic of selling cars is that the associative image of the architecture becomes very important in framing the image of the car. Any and all references to the specificity of a building or space are either ignored or obscured. There is purposefully no sense of
Fig 29. Public Path
Processional connection between gallery space and research port
Fig 30. Conglomeration of interdependant design experiments in simulated experience from Maryland Avenue
place defined by the architecture. Additionally, the architecture and architects involved are not entirely well known throughout society (maybe no living architect is), so the directors have no incentive to have the car associated directly with a specific work of architecture or an architect. This process is entirely about image. It provides a unique framework by which to organize architecture today. If we trust the directors in the terms by which they earn their livelihood and claim expertise we can associate a building’s use in commercials with their image’s ability to register with the current visual trends of society. On some level then they communicate a fashionable image—an image of technological
Fig 31. Conglomeration of interdependant design experiments—west elevation
Fig 32. Outside
Simulation of the net effect of the building as sign, display and shield as experienced from 14th St. SW

week 33
advancement, innovation, trendiness but also a certain degree of comfort.

The most regular work of architecture in car commercials in the summer of 2007 was, by far, the California Transit Authority building designed by Thom Mayne of Morphosis. The Ford motor company commercials made use of this building highly, including the entire line of Mercury car adds (including print ads) released during a roughly three-month period over the summer. When the architecture becomes the stage set, as the Transit Authority has for these commercials we can analytically label it a “cool” building. Despite the fact that the thought of fashion invading the elite realm of architectural design can make some architects nervous, even outraged, the condition exists.

Use of architecture in film (here the definition of film is admittedly very broad in that includes TV commercials) generally also reveals something about the spatial qualities of the built work which are much more parallel with the goals of architectural design—that is, to be a good set for shooting film, the architecture must provide spatial datums for movement to be recorded again. The Transit Authority building, with its spatially coherent fore court with variably defined edge, serves this need well.
Fig 33. Outside at entry
Simulation of the net effect of the building as sign, display and shield as experienced from 14th St. SW at D St. SW

week 34
Fig 34. Analytic west elevation
The net effect of the conglomeration of design outcomes
An important distinction can be made between the discrete function and a set of functions that work together or in sequence on or with overlapping variables. One immediate application for computational design is to implement scripting early in the design process using a limited set of known variables along with some parameters, to generate a system or a piece of architectural form. This investigation however, uses a computational design environment. Many functions reference the same variables, the output from one function may serve directly as an argument for another function. Relationships, meaning, dimension, and geometry are all information that is shared and influenced by the discrete scripts. The goal of this methodology is to arrive and architectural results using the output from scripts as a meaningful, complete spatial proposition. Furthermore, this process allows the regular reintegration of new variables and relationships based on the ability for the functions to interact.

“From the general scheme to the particular detail, the modernist project deals methodologically and architecturally almost exclusively with top-down hierarchy. We do not reject the concept of hierarchy but rather use it in a new way. We work within a hierarchy that is not simply nested in scale and distinct from the orders that lie above and below it. Rather, we are using organizational principles that promote communication across scales, in which the particular is able to affect the general and vice versa. This requires a methodology that involves both top-down and
A potential conflict arises in a desire to maintain an output that reveals the nature of the processes involved in generating output. This does not necessarily mandate the communication of a specific relationship but rather that type of processes. NASA, for example, aims to communicate a culture of computation and systematization generally, rather than any one of dozens of critical, building or environmental relationships. In conflict with agenda, can be the tendency for the dissolution of the strength of the output of one algorithm due to the overwhelming restraint imposed on the algorithm and numerous other functions.

Even with this sustainable approach to design competition, it would be impossible and undesirable to script the output of an entire building scheme. This investigation promotes a continued and regular overlay of human intuition on to the computational design process. Some design processes are inherently human and do not necessarily benefit from the deconstruction of heuristics into algorithm. The human component to this design process is also critical in recognizing unexpected but desired output from the scripts that were an indirect result of the computations. These moments allowed for the linking of computerized processes with formal, visual implications. The metal fabric surface making up the figural elements on the front façade, for example, represents at the visible kink in the surface he shift during the execution of the algorithm from a Boolean if statement resulting in true rather than faults. Understanding the visual potential of this computerized condition results in the protection of this Boolean condition in the further advancement of the algorithm.

The important subtext here is that the human hand is truly present in the computational design as well as an improvisational approach. More important than human versus computer is logic verses personality. When these two forces are not in conflict they advance one another.
Fig 35 (top). Analytic section
The net effect of the conglomeration of design outcomes

Fig 36 (above). Analytic plan
Exigence of Information

There now exists more information stored as bits in digital form than the equivalent knowledge stored in the brains of all living humans on earth. While it is true that, “Justification for architectures that claim a link between data and a graphic representation of data are in error.” Value, certainly in the case of NASA can be associated with the nature of data in terms of quantity, and the representation of data as a signifier for technology culture. If the role of information in our society is established as critical, NASA can communicate it power and importance by revealing its distillation and interpretation of mass data. This is more than just a veil though, NASA fundamental methodology is rooted in retrieving and understanding new information.

For many of the world’s inhabitants, access to information is no longer the most significant hurdle in our pursuits of knowledge. Governments, economies, and social status only mildly restrict the extent to which we can communicate with one another. Information is largely available, if not free. An individual with no money, no possessions, no personal network of any kind can, for example, access the complete works of William Shakespeare, get a stock quote, or read the latest from a leading Japanese Newspaper at a public library sooner than he or she could obtain a loaf of bread from a grocer.

Information is no longer a precious commodity.
Instead, the ability to sort, filter, discern interpret and distill information becomes critically valuable. Situations in which information is available in enormous, even comprehendible, quantities arise often. Control over this information becomes necessary before it can be of value to humans. The Google search engine is an example of such a commodity. The fact that a search for “stock tips” yields over one hundred nine million results is phenomenal, but not useful. In fact, hundreds of millions of stock tips are, by nature, of no value to the hypothetical investor. Reading hundreds of millions, let alone millions, let alone one hundred thousand stock tips over
Fig 38. [top] Conglomeration of design endeavors as simulated from the Gallery/Observatory

Fig 39. [above] Conglomeration of design endeavors as simulated from approach on D St. SW
the course of my entire life. Google.com is of course, incredibly valuable because of its filtering and sorting algorithm, which allows users to be confident that the results are the most relevant out of the millions available. In this example, quantity is overwhelmingly extreme but it is not irrelevant. As a hypothetical investor I would much rather have the top ten stock tips out of hundreds of millions of stock tips than simply ten stock tips.

Pamela Martin and Associates, which the United States government alleges was a prostitution service in Washington, DC, was nicknamed the “D.C. Madam,” by the news media. In July, 2007 the supposed entirety of the phone records held by the D.C Madam were released. The list constituted some 88,000 phone calls in the order that the calls were placed. When printed these records weighed 46 lbs. The records were also made public over the internet, in TIFF format. While some worried individuals initially made hurried admissions of guilt, the fallout from the publication of these calls was essentially non-existent. It is beyond most human beings attention span and physical ability to read through thousands of lines of numbers. The information was available, in a format that is understandable, yet was useless. Many guilty individual’s phone numbers were there, in the public domain but where shielded by the volume of other numbers available. Later that same month, another independent website had indexed most of the phone records and made them searchable. It is now possible, at this new website, www.dcphonelist.com, to search for any phone number in the database. The results allow a journalist, lawyer or curious citizen to then reference a specific page of the original document. Now the information has value.

In other situations, knowledge and value are achieved directly from the aggregation and expression of large quantities of data. It is in this way that NASA is most relevant in this century. Designing NASA’s image is important. It should communicate the essence of its values, goals and methods. In this investigation,
information representation has implications on
the architectural process as well as the specifics
of the test case.

This design process explores how information,
when visualized, can become fuel for design.
There are many facets to this exploration. The
most direct, literal method involves translating
data in such a way that the original, intended
value of the data can be readily understood. It
is this transformative process that data can be
associated with scale and units, the process
by which data becomes information. The next,
often less rational, layer of this process involves
translating the data such that emergent visual
conditions can be realized. This process can
sometimes reveal patterns, or more subtle
relationships. It can also obscure and distort to
such an extreme that any perceived link to the
original data is purely symbolic or metaphoric.
The process of transformation itself might be
more directly communicative than the data. It is
this later sub of visualizations that is of specific
issue in this process.

What gives a process of visualization its exigence
beyond direct communication? Abundant
examples exist of architectural design derived
from a relevant or irrelevant data sets. When
these data relate to per formative parameters
the visualization (or even direct physical
manifestation) can rightfully bypass an aesthetic
defense. A design process can also make use of
the image of data to communicate meaning more
vastly relevant than the data itself. Moreover,
visual and spatial affects sometimes require
raw data as the fuel for transformation. Great
effort is made, for example, to simulate highly
variable but non-random (and therefore non-
noisy) variable data in the motion-graphics art
and advertising.

NASA’s image is highly intertwined with data.
NASA’s mission and current operations relate
largely to the collection of data. This is a radical
shift from the NASA image of the 1960’s for
example, which was task-driven. Going to space,
orbiting earth, landing on the moon, etc—these
are all task-oriented milestones. (More about NASA’s priorities, structure, trends discussed in another chapter). These tasks were highly visible, highly imagable in and of themselves. Arguably, these milestones where some of the most imagable moments in America’s history. The “earth from the moon” image is potentially the most famous image of the 20th century. Now though, these milestones are, when specific, less relevant because they are less stunning. NASA has not stopped innovating though and the institution’s academic, scientific and cultural value has not diminished. Instead, it is harder to quantify and demands a less literal method of communication. NASA’s feats relate no longer
to singular achieved goals but to a quantity of captured information. NASA captures and records nearly inconceivable quantities of data about the Earth and our universe and, through research and innovation seeks to apply meaning to and with this quantity. Thus, the architecture of NASA cannot communicate the reality, the essence of NASA but conveying any one set of data. Instead, the goal is to manifest the poetic nature idea of complex, nearly infinite data.
Neither the medium nor the parameters exploited by the medium are novel in this investigation. The goal is not to invent new relationships but to revisit the most basic of relationships, the first principals, that have been long taken for granted by a profession and industry that has (by both habit and design) embraced pattern-based solutions to categories of conditions. Patterns and best practices can, through their own internal logical intelligences unfortunately stifle the potential for performative agendas and wholistic design decisions. Simplicity, recognizability and apparent convenience can mask design complacency and operational mediocrity. First principle design can be characterized as producing a result that has a sense of inevitable simplicity. This investigations seeks to instead to respond to these simple, often uninspiring parameters in a manner that is ultra-responsive, efficient and richly intertwined. Complexity is not sought, but embraced as the simple parameters become dynamic interrelationships.

First principals are rarely novel. Designing based on first principal forces places a burden therefore on the process to satisfy an agenda of innovation. Designing with algorithms promotes and even requires a first principle approach. The designed computational processes must compute something and a result based on first principles has, by definition, value. Beyond that though, this approach affords the algorithms the most potential to influence the most critical aspects of the design.
Fig 44. Analytic simulated experience from corner
Fig 45. Analytic simulated experience from car on 14th Street
Conclusion

This design research has demonstrated that spatial and aesthetic effects can be meaningfully synthesized within the architectural design process. The outcome is more than a digital, computerized, or computational aesthetic. Instead, the result is output and description of the design process simultaneously. This process has established the primacy of topology. To that end, artfulness is available in the efficient solving of problems and in the creation of discrete relationships.

The most foundational lesson is that a computational design process is most fruitful when restrictions, parameters, and complexity are embraced simultaneously and early. The abstraction that is the lifeblood of the designer must exist then through representation and synthesis rather than simplification pre-patterning. The formal parti would circumvent and bias to the point of irrelevancy a computational strategy. The converse warning is that without classification, codification and evaluation though rigorous theory (and even post-rationalization) to balance praxis, a computational strategy can narrowly spiral directly into stagnation though over-specificity.

A connection of methods, practices and techniques of programming as well as animation culture to formal and spatial implications have potential for application in design pedagogy. With
the incorporation of programming languages that can control and output what is traditionally manually controlled by human designers as well as that which is by nature computationally complex (and outside the reasonable realm of human calculation) can lead to a richer, literal understanding of both languages. Deconstructing a technique into an algorithm (or the reverse—making visual an existing algorithm) is both a clarifying learning experience a means for exploring topological relationships embedded in the process of achieving a designed outcome rather than the outcome itself. This opening of floodgates into unlimited parallel or tangential investigations exposes meaning of the problem itself or the meaning of the media used to explore and exploit that problem.

It was observed that as much as there is a culture of computational design independent of architecture, there is a culture within architecture that resists the idea that methodology can successfully shift with technology. The results are broad biases against digital media or the desire to shift the discourse away from the specifics applications to a more comfortable, arms-length discussion about the philosophy of computers (paradigms that fascinated science fiction authors half a century ago, for example, before the nature of computation was foundationally understood should not be necessary to preface a discussion about digital methodology). As much as this discourse rests comfortable within a historical continuum, especially with regards to Modernist thinking, it is inappropriate and counter-productive to engage the work in entirely stagnant terms. It must be assumed that with new methodology can expand the body of architectural knowledge. In critiquing the “use” of computers.

Equally relevant is the observation that although almost all architects--that is broadly speaking, students of architecture included--use computers in their design processes. Terzidis explains further why these architects, again generally speaking, are ill-qualified to critique a computational process and can only do so in philosophically or
analogically in noting, “The dominant mode of utilizing computers in architecture today is that of computerization entities or processes that are already conceptualized in the designer’s mind are entered, manipulated, or stored on a computer system. In contrast, computation or computing, as a computer-based design tool, is generally limited....some venture into manipulations or criticisms of computer models as if they were products of computation....mouse-based manipulations of 3D computer models are not necessarily acts of computation.”

With the presentation of a building, one immediate avenue available to evaluate the process is to judge that outcome, and by proxy the preceding explorations, on its own merits in isolation of the context of the design process. The process of scripting necessitates that some functional requirements be satisfied directly. It is important then, to regularly evaluate the output using additional standards. The public review was not an exception. Orthographic drawings and simulated experiential graphics promoted an engagement with the architecture by any standards. Analogous to a scientific blind study, a review of the resultant building totally independent of any knowledge of its design process would be one fair way to evaluate the process without bias. This type of presentation was part of the necessary proof that the outwardly stretching research had implications for architecture in the narrow sense.

It is also reasonable to conclude that to some degree the resultant building is inconsequential and ultimately irrelevant given that this type of process – computational, temporal design – inherently results in a set of solutions rather than a single ultimate solution. Even if theoretically realized as a constructed work of architecture, the temporary, adjustable nature of the building systems as well as the assumed transience of any one current program from make any one current instance, at least partially, inconsequential. Specific formal and spatial effects achieved directly and indirectly from the specific design process can be examined through an architectural
lens without the requisite of a complete building proposal. It is not necessary to read, study, and understand every detail, or any detail potential, of the logistics of the computational design media. Instead, algorithms and animations are manifest directly as image. These documents are themselves scripted artifacts, simultaneously reflecting early experiments in the meaning of media and representing a potential architectural outcome.

Some important questions are raised at the conclusion of this research with regard to the potential for broader implications and narrower applications. To what extent can or should the processes investigated here be applied to circumstances or architectures other than those, such as the NASA headquarters, which automatically imply through the nature of pre-existing scientific or technological cultures, a predisposition towards technological solutions. To what extent, for example, can these processes be applied successfully and profoundly—but gently too—to a small portion of the design or architecture rather than at multiple scales many levels and in many intensities?

As in any research, as much can be learned from the failures as well as the successes. The nature of designing through the production of functions led to the ability to generalize about certain types of approaches negatively. In some cases a distinction between what is satisfying or pleasurable as a design investigation theoretically, and what is ultimately useful or sustainable in an architectural process is necessary. This is essential in noting exactly how and in what form crossing disciplines is fruitful. In an open, intellectual society diversification of ideas is appropriately embraced by its very nature. This process though allows for the critique of that diversification. In an intellectual and technical climate in which anything is possible, it is more valuable to be able to test, evaluate, and eliminate than to indiscriminately follow technology.

There are inherent benefits to design scripting, generally speaking. Design algorithms tend
towards mass customization of forms that are
directly translatable to a mass customization
paradigm of fabrication and construction.
Componentization, a critical cultural necessity for
NASA, is a critical component of a sustainable,
performative architecture. These paradigms
lead to sustainable economic models in which
innovation, rather than standardization can
be profitable. A parallel benefit is the potential
elimination of the obscurely guarded ego of the
architect, often seen as at-odds with the basic,
fundamental needs of the client or the realities of
the construction industry.

Scripting and animation are open ended and
broadening mediums. Algorithms are, by definition,
expandable and reusable. Connecting scripts
with animation further multiplies the potential for
expanded investigations as unexpected extreme
variations on parameters or the resultant are
brought to light. This paradigm is the most
exciting, intellectually sustaining aspect of this
research – that solution-driven design processes
and an artful exploration of the meaning of media
can be one and the same. This is, after all, the
foundational plight of architecture, to infuse logic
with beauty and art with structure.
Notes


3. “Misericord to a Grotesque Reification” by Mark Goulthorpe. Page 57


5. Algorithmic Architecture by Kostas Terzidis. Prologue Page xii


8. in English at http://home.kyodo.co.jp/


5. Algorithmic Architecture by Kostas Terzidis. Prologue Page xi
Bibliography


Koolhaas, R., Hommert, J., Kubo, M., & (Firm), P. (2001). Prada / [OMA/AMO rem koolhaas ; editorial team, rem koolhaas, jens hommert, michael kubo]. Fondazione Prada,.


Maeda, J. (1999). Design by numbers / john maeda.MIT Press,


   Thames & Hudson,.

   semiotics : Structuralism, post-structuralism, and beyond / robert stam, robert
   burgoyne, and sandy flitterman-lewis.Routledge,.


Thomas, M., & Penz, F. (2003). Architectures of illusion : From motion pictures to
   navigable interactive environments / edited by maureen thomas, francois penz.
   Intellect,.