RECURSIONISM

a methodology of recursive systematic computation

MICHAEL JEFFERS & JORDAN PARSONS

DALE CLIFFORD, PRIMARY ADVISOR
P. ZACH ALI, ADVISORY COMMITTEE
ERIC BROCKMEYER, ADVISORY COMMITTEE
MADELINE GANNON, ADVISOR COMMITTEE
GOLAN LEVIN, ADVISORY COMMITTEE
ZACK JACOBSON-WEAVER, ADVISORY COMMITTEE

The Frank-Ratchye STUDIO for Creative Inquiry

THIS PROJECT WAS SUPPORTED IN PART BY FUNDING FROM THE CARNEGIE MELLON UNIVERSITY FRANK-RATCHYE FUND FOR ART @ THE FRONTIER.

dFAB

THIS PROJECT WAS MADE POSSIBLE WITH EQUIPMENT IN THE CARNEGIE MELLON UNIVERSITY SCHOOL OF ARCHITECTURE DFAB LAB.

INTRODUCTION

Our thesis is the development, study and integration of a systematic computational method that is continuous from generation to fabrication and assembly.

The thesis encapsulates not only the work itself, but its larger application, contribution, and understanding among others outside of ourselves of the direct impact of the work. The work and the thoughts on the work were borne from a series of internal dialogues and discussion of various subject matters, that we have formulated arguments on, and attempted to address and provide proof for these statements through the work.

A Priori vs Systems:
Systematic computation preserves the connection between intent and execution. Erroneous results force refinement of the system, and not the resultant form. Whereas traditional design practice amalgamates conflicting logics into a single object, obscuring the connection of intent to execution.

Craft and Authorship in Computation:
Craft in computation is based on knowledge of low level functionality in involved processes. This knowledge and the computational logic employed in the execution of the system transfers the authorship of the designer to any resultant instantiations if the design intent and logic are congruous.

Construction Process:
Systematic computation engaged with digital fabrication allows for the explicit linking of digital and physical information. Construction logics and nomenclatures emerge based on this relationship, controlling the degree of on site decision making.

First Generation System:
The toolkit was designed as an open framework that can accept and interfaces with a multiplicity of tools, types and functionalities allowing for the incorporation of additional data sets, conditional gates, analytic and evaluative subsystems. It can also be understood as a component of a larger framework that operates across different scales, engaging with higher level logics.

Instantiation as Architecture:
The built instantiation is experienced independently of the logics of its generation. This disconnect leaves the instance in an ambiguous state between architecture and object. Altering qualities and factors in its siting, scale and tectonic execution could reposition it in a more architectural state.

Systematic and Non-systematic Computation:
Non-systematic computation exists as the realization of a priori forms through the use of discrete computational functions without consideration for the direct interplay and communication between them. Systematic computation arises through the direct articulation and acknowledgement of the relationship between discrete computational functions and subsystems.

This thesis was the combined skills and interests of two individuals that had interfaced with computation in an architectural and creative environments, but began with radically distinct views on what it means for the designer and the act of designing with this tool. The discussion began as a conflict of interpretations on what it was giving us, as architecture, through our observations of the aesthetic result of these fetshized new processes. The ambition was to construct and demonstrate a fully computational workflow that would demonstrate the continuity and validity of the design logic through to the built form.

The first step was establishing a very clear and logical means of evaluating our differing views, and how we determine which is correct, or even able to be determined. After which we would reason through how to construct an argument as a thesis that would be supported by our mutual conclusions. Deductive argumentation was adopted to allow an agreed upon framework to bring about truths and falsehoods of our own understandings, weaknesses in justifications, and allow our combined conclusions guide our work. Deductive arguments also allowed us to quickly and plainly state what was assumed, proof for those assumptions, and demonstrate how this led into a particular conclusion. This is emphasized because it is often not a part of

discussing work within architecture. Many arguments go unspoken, assumptions assumed, and strawmen abound. All that is left are reactions and statements of agreement or disagreement appealing to popular convention or the equally popular rejections thereof. It was a long shot, but the hope was that this deductive reasoning, through clear logic the benefits of our method as well as what it implied about design processes, could convince those that previously objected on a reactionary, aesthetic level.

The computational toolkit became the method by which we began to create and discuss an externalized design process. The thought was that if, in theory, the intuition or decision making process of a designer could be extracted, and codified as rule sets, links of dependencies, and relationships of values, then it could exist as coded instructions in an entirely virtual and external interface. All of the sub-arguments about craft in code, authorship and control, systematic focus over high-level determinism, all contribute to this original thought. If the design logic is to be accurately codified, the designer would have to achieve a level of honesty in their intention, which code would reveal with the 'unwanted' results. To change the result would be the a-priori, deterministic, high-level design method that we were proposing an alternative to.

Another key element of the thesis was the linkage of the abstraction of the virtual system to real physical data sets, and the demonstration and evaluation of its success through the construction of the physical object. The geometry and the algorithmic processes that produced must produce it within the constraints of the construction and assembly logics of the physical material capabilities. At the sametime as physical capabilities ran into limits, the rules and generation then had to be redesigned to accommodate different limits or processes of constructability.

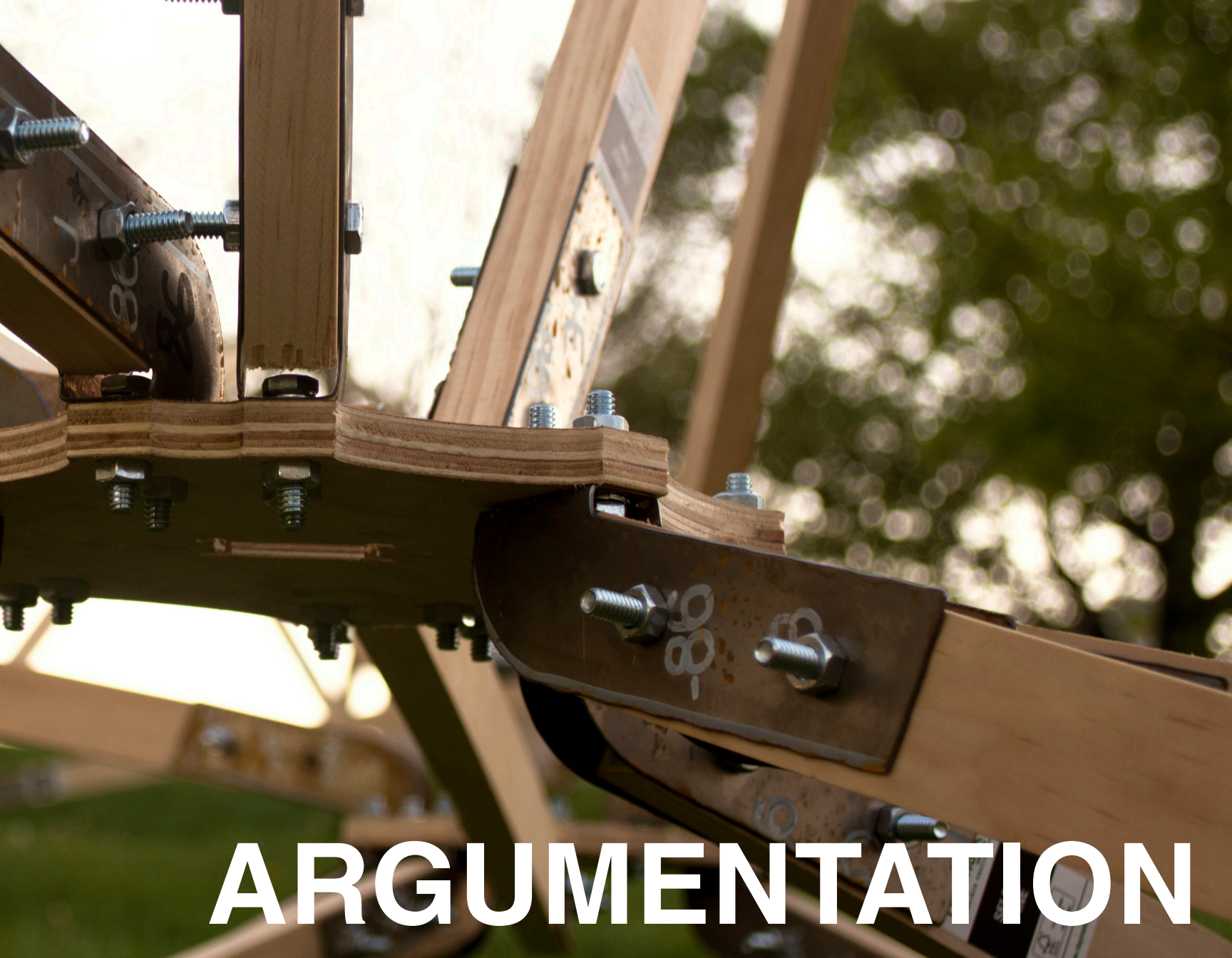
Recursionism should be understood as a word that we are defining as a summation, embodiment and framing of the work that we have done within this thesis. It is our title, but is also a word of its own that we have redefined to encompass our work, and more.

It is loosely defined in other circles, chemical engineering, and even art history as something that is about a recursive property. A lot of the visual representation of recursive structures come from the models of chemical and material sciences in the study of microstructures having various properties and rules to their constructed geometries, then being deduced into an algorithm that generates the self-similar structures.

That being understood, our usage is perhaps a less specific application of this word. Essentially the redefinition removes the visual context and allows it to apply to all recursive processes, or processes that contain recursive elements. We use this word to describe the way we work (method) and the way we think about the way we work (methodology). The system we constructed is intended to be an externalized design process. This system then becomes the object of the design, and therefore we then become designers of that object. This creates a self-similar problem, when we can further step back again and design that system that designs a system that designs an object. Therefore producing a recursive understanding of the meta-design that was the systematic computation method we established. So a self-proclaimed Recursionist would be one that meta-thinks at all levels of meta-thought. To avoid internal stackoverflows, we limited the depth of our recursion to 1.

It is not our intent to recreate a definition of a word. It was merely a title but took on a life of its own as we tried to cohesively package the ideas we advocate for through our thesis.

Deductive argumentation allows for a statement to be tested for validity, and its assumptions weighed in logical truth. This method was used two-fold: in how we evaluated our statements of our thesis, and in the development of our code with respect to the intention of its use and performance. Just as code that produces erroneous results, so too an invalid argument does not imply the intended conclusion.


ARGUMENTATION

IF ARCHITECTURE RESULTS FROM A METHOD THEN THE **VALIDITY OF THE METHOD** HAS A CAUSAL RELATIONSHIP TO THE **VALIDITY OF THE ARCHITECTURE**

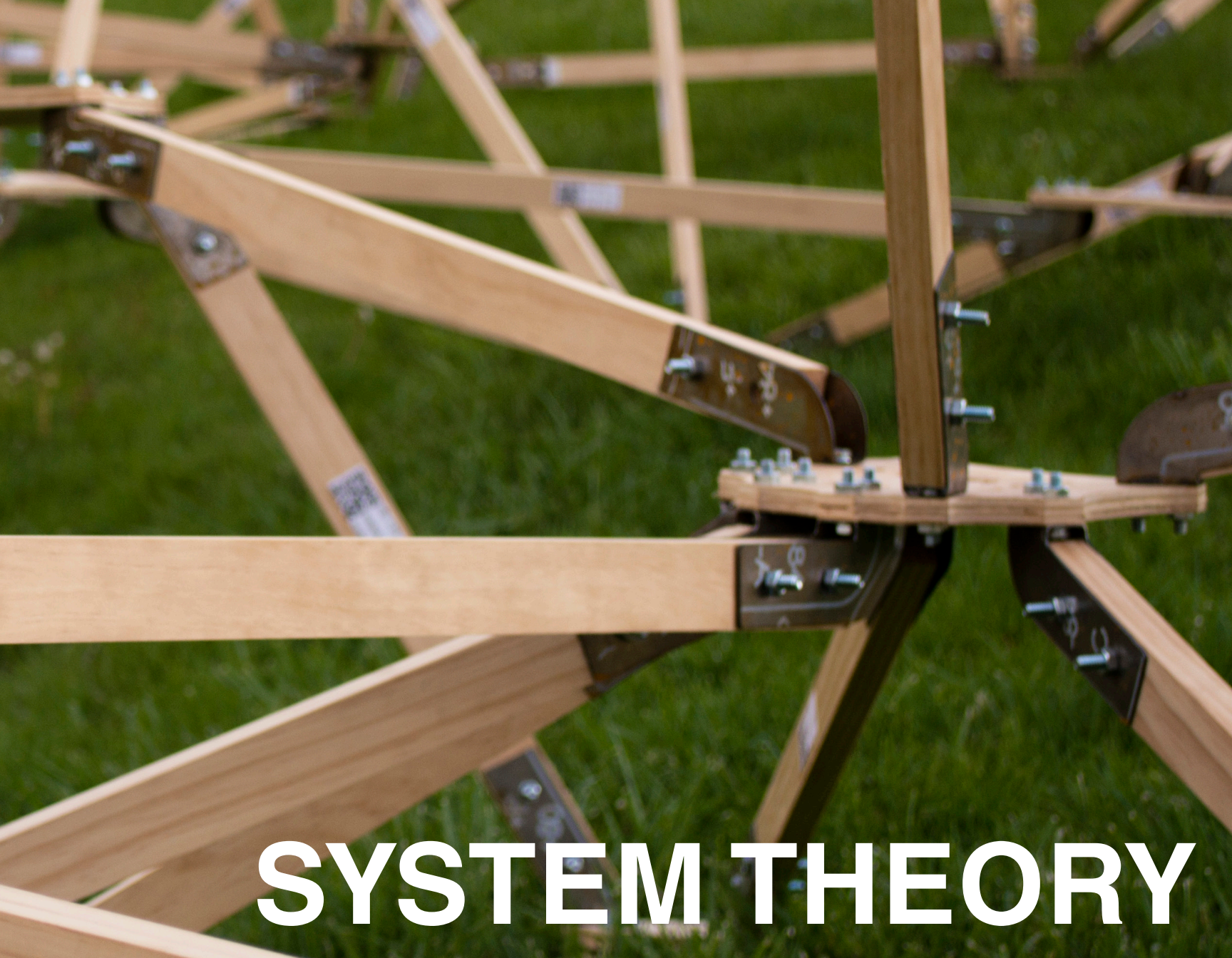
ASSUMPTIONS ARGUMENTS AND CONCLUSION BECOME DATA APPLICATION AND RESULT

OUR THESIS IS THE DEVELOPMENT STUDY AND INTEGRATION OF A **SYSTEMATIC COMPUTATIONAL METHOD** WITH **DIGITAL FABRICATION**

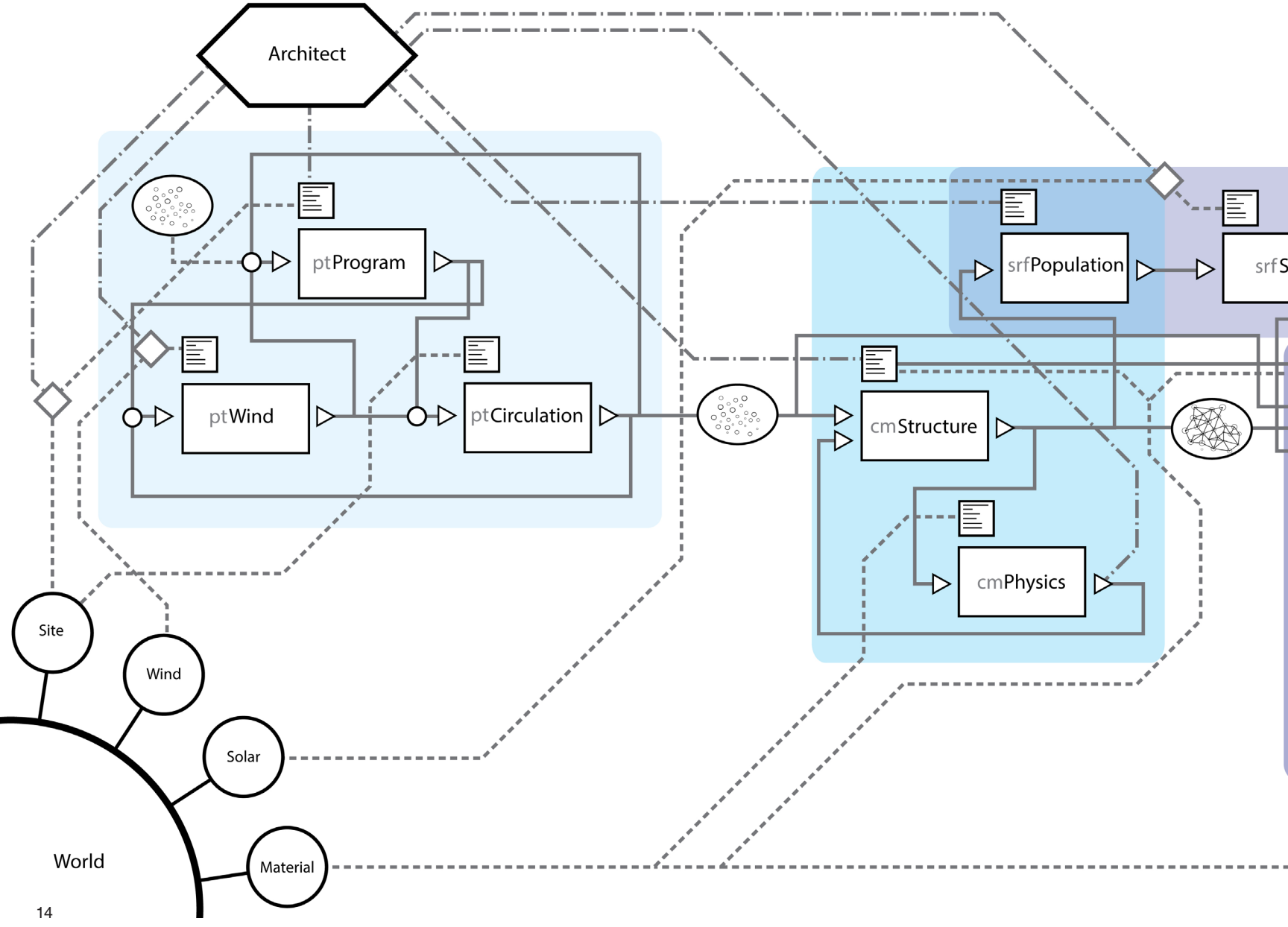
OUR METHOD IS THE RECURSIVE LOGICAL APPLICATION OF DATA AND THE EVALUATION OF ITS RESULT

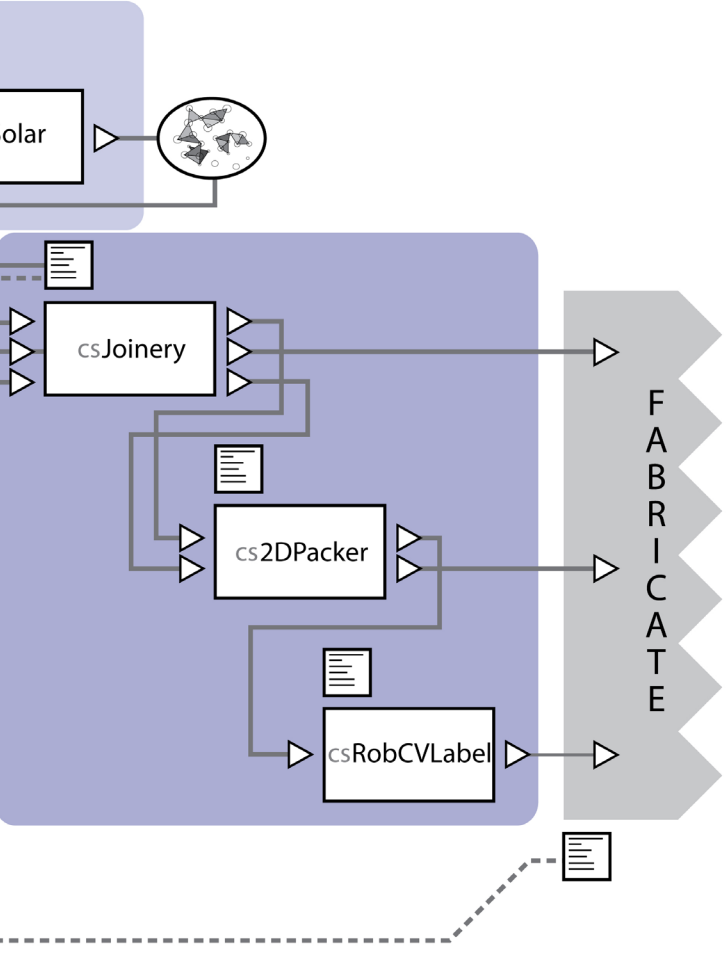
Understanding the design process as a system of variables and relationships that extend from environmental to material was to be the framework for the use of computation. Systems are inherent to the understanding of all things, and the interactions of these things.

This self-aware system, one that the designer is an agent within, is characterized by a series of gates and flows of information from one tool to another. Design happens through the creation and control of a computational tool. The designer is made aware of the tool's operability through its product. Therefore a recursive feedback loop is constructed between the designer and the tool.



SYSTEM THEORY

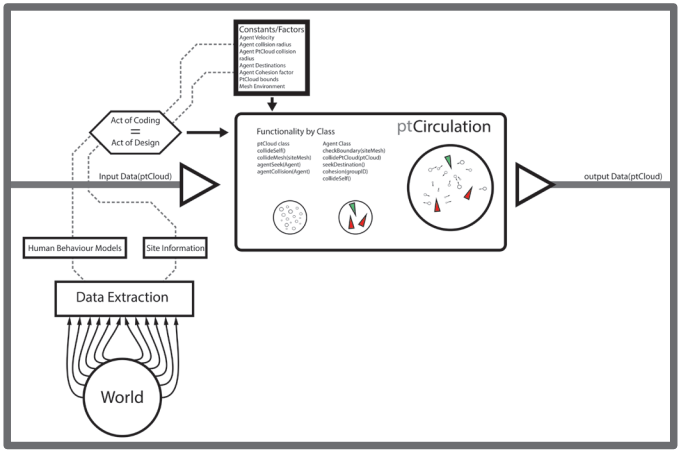

Architect
ptProgram
ptWind
ptCirculation
srfPopulation
cmStructure
cmPhysics
Site
Wind
Solar
Material
World



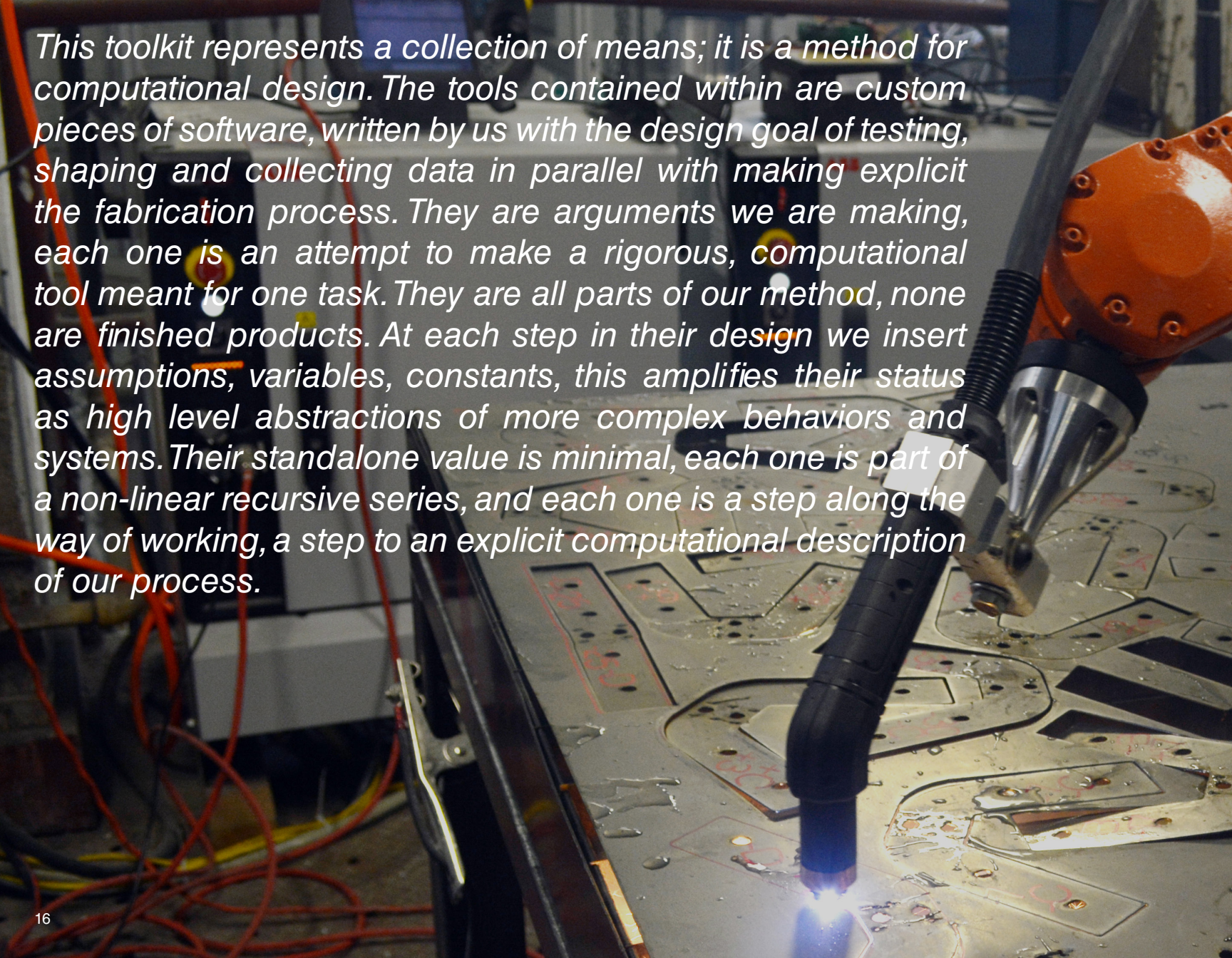
COMPUTATIONAL TOOLKIT AS DESIGN SYSTEM

The actual functionality of each component and the nature of the coupling differences, loops, gates, will be specified in the Computational Toolkit chapter. The creation of this larger framework in which other scripts were hosted began to be a design process of its own. Only some tools can talk to others, and each alters some critical piece of information that they all act upon. How that data is passed to and from and when it moves to the next step, when it loops, is all another higher level functionality. At the point of the presentation of this work, it was us, the architects. A critical introspection occurred upon making this diagram. Where 'Architect' is seen controlling the gates and loops, one can simply supplant another tool, a program that controls other programs. This will later launch a discussion of this toolkit being a 'first generation system' in that as a system itself, it too can be packaged and coupled into yet another higher level larger network of tools and toolkits.

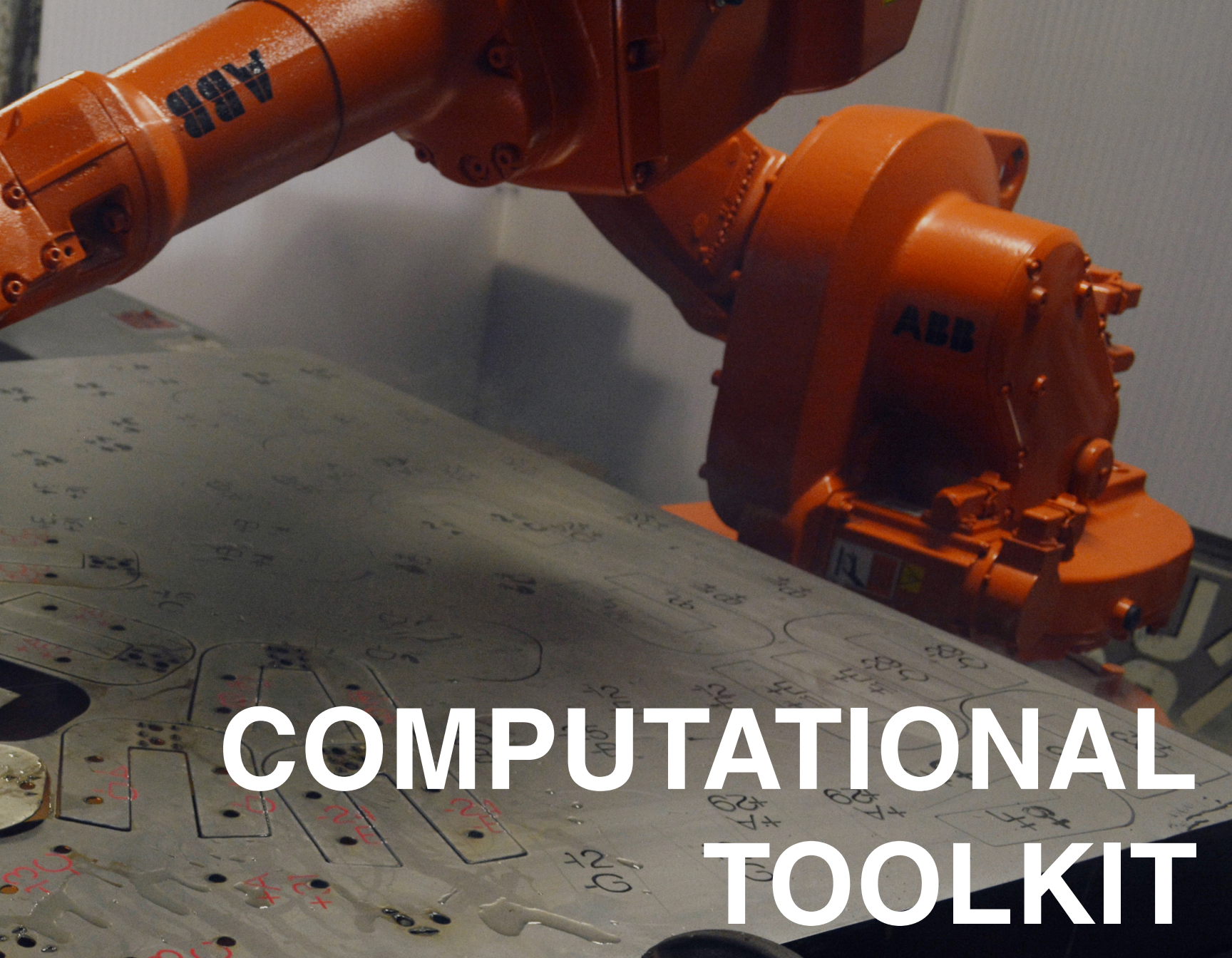
However our goal was to create a valid link from the input information to generate a designed object that was a result of said information. Understanding not only the component but the larger system as a whole in this process of design highlighted the need to apply design efforts at a system scale.



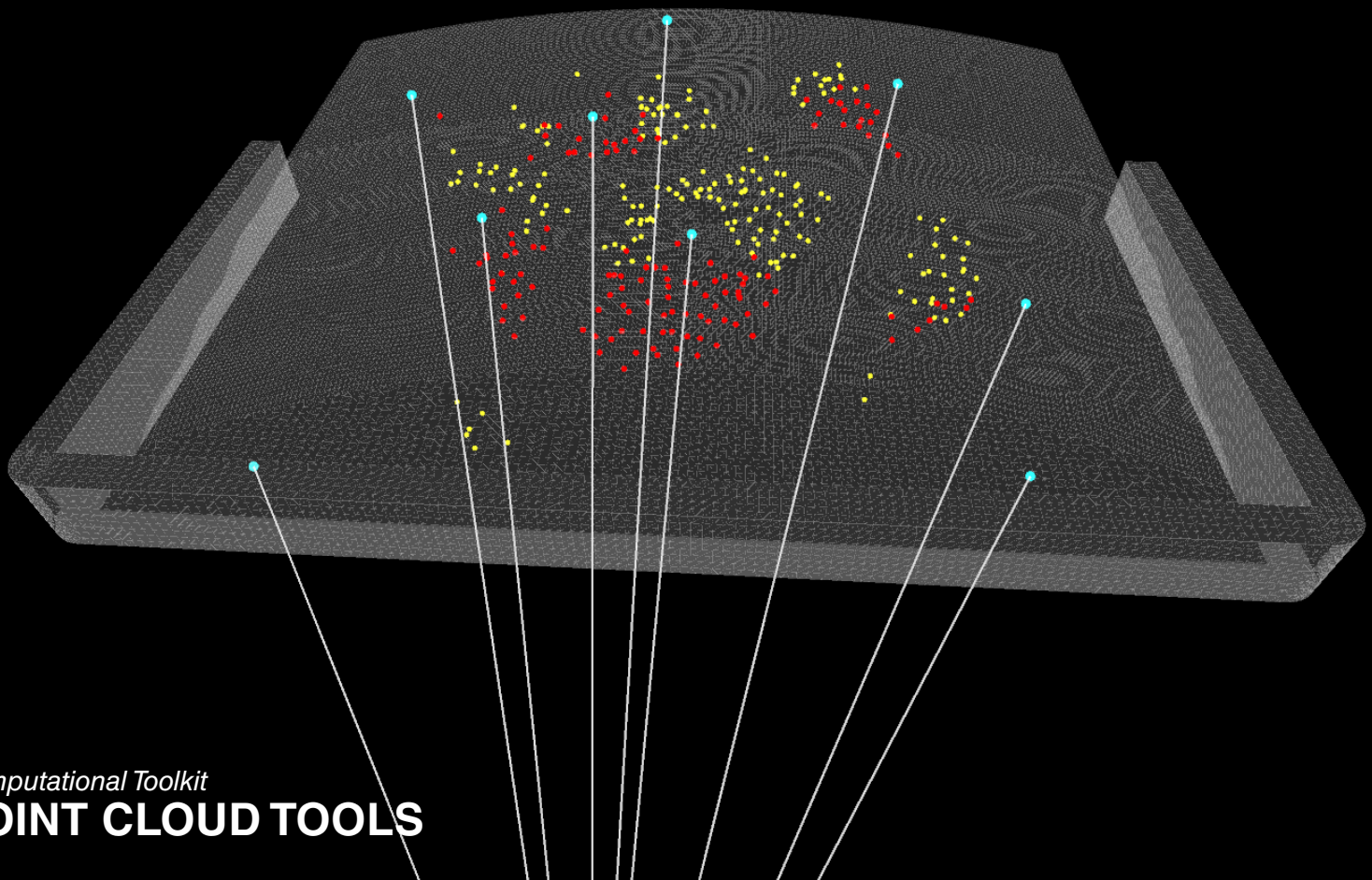
A legend that shows via example how one can read the icons of the larger system diagram.



This toolkit represents a collection of means; it is a method for computational design. The tools contained within are custom pieces of software, written by us with the design goal of testing, shaping and collecting data in parallel with making explicit the fabrication process. They are arguments we are making, each one is an attempt to make a rigorous, computational tool meant for one task. They are all parts of our method, none are finished products. At each step in their design we insert assumptions, variables, constants, this amplifies their status as high level abstractions of more complex behaviors and systems. Their standalone value is minimal, each one is part of a non-linear recursive series, and each one is a step along the way of working, a step to an explicit computational description of our process.

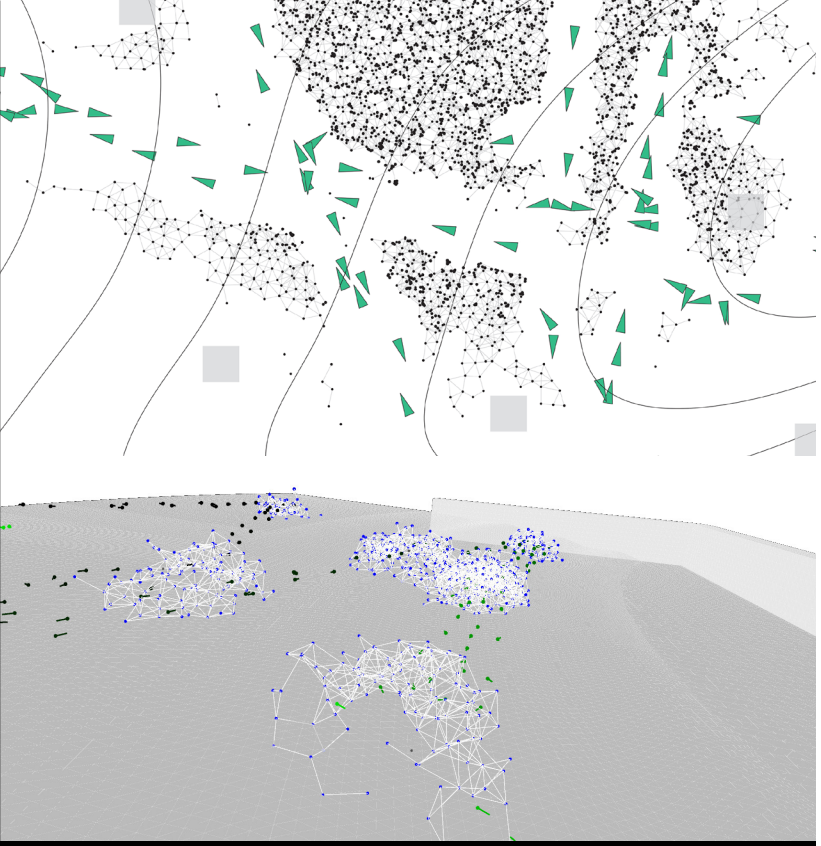


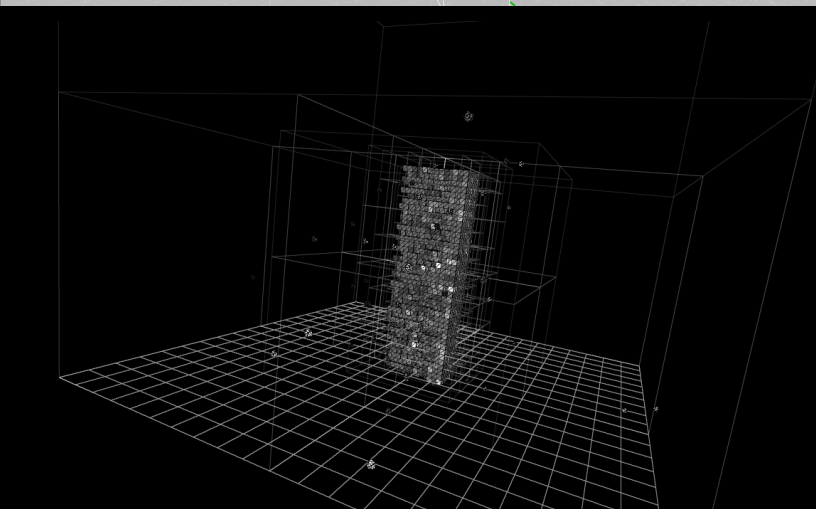
COMPUTATIONAL TOOLKIT



Computational Toolkit

POINT CLOUD TOOLS

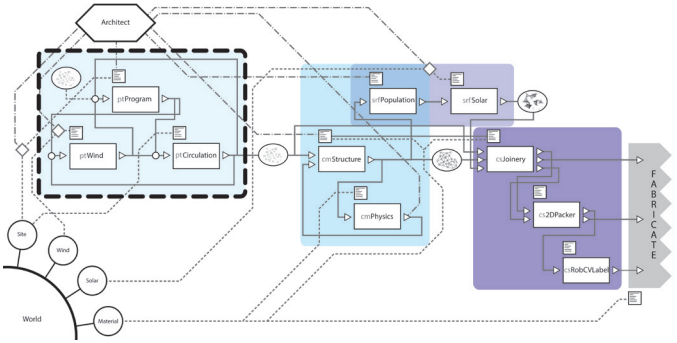




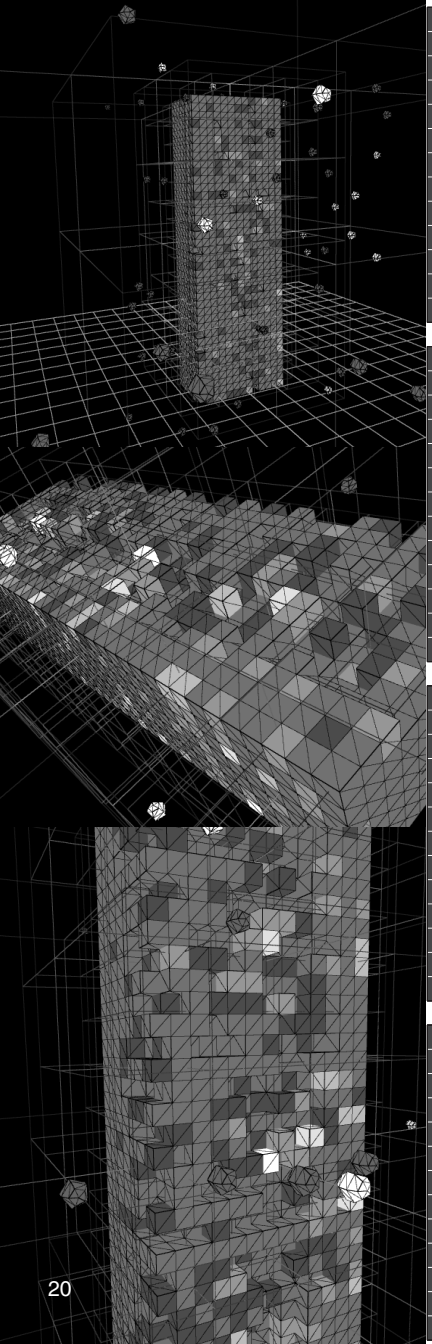
POINT CLOUD: LOW LEVEL INFORMATION

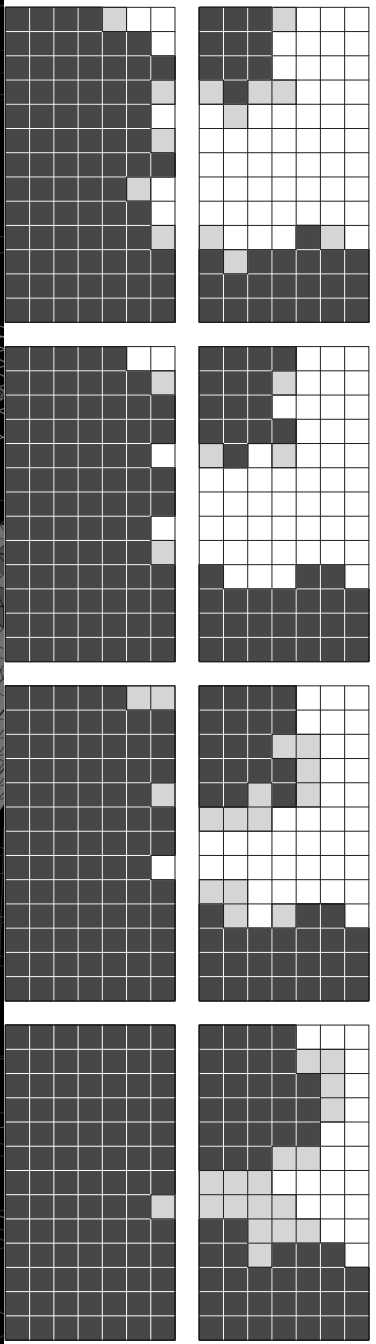
The point is the lowest level of geometric representation of spatial data. Its location is only significant when there is a reference coordinate system on which to base its location, or other points in which it has a series of spatial relationships with. It is at this stage multiple points becomes understood as not a collection of singular points, but of the relationships of one to the others. We define this abstract material as the PointCloud. It is the medium by which other external information imposes various forces and manipulates the distribution of this PointCloud.

After understanding the point, and points as a series of relationships, the PointCloud then becomes an abstract of a larger suggested volume. However this is an interpretive leap. A cluster or group being read as a consistent whole is a technical fallacy, the points still only exist as singular objects. Internal attraction and repulsion governs the distribution. External forces can act on the exterior, forcing waves of transformation through the Cloud. Each point fights to achieve equilibrium. This equilibrium state became the time at which the problem reached a solution-state. The stimulatory nature of these time-based solvers suggested a nebulous result and loosely connected to original input and parameters. However it became clear even the dynamic equilibrium of the output of these tools were most valuable when they communicated with each other.



19.1 - 3
Diagram of circulation script frame(top left). Some snapshots of Point-based tools running(left).





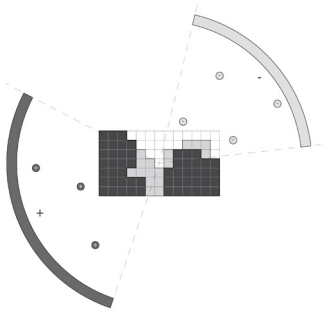
SIMPLE ENVIROMENTAL MODELING

WindErode was an early attempt to make a tool which dealt with low level data, a point cloud, to form it and imbue more information into through the simulation of a natural process. The PointCloud was often seeded and depicted as a solid form, one that would be eroded by the oncoming wind particle objects. This was our first tool, and the point cloud had no method to be initialized or given any particular bias. The chosen form, seen in the images, was for our own understanding of the performance of the script.

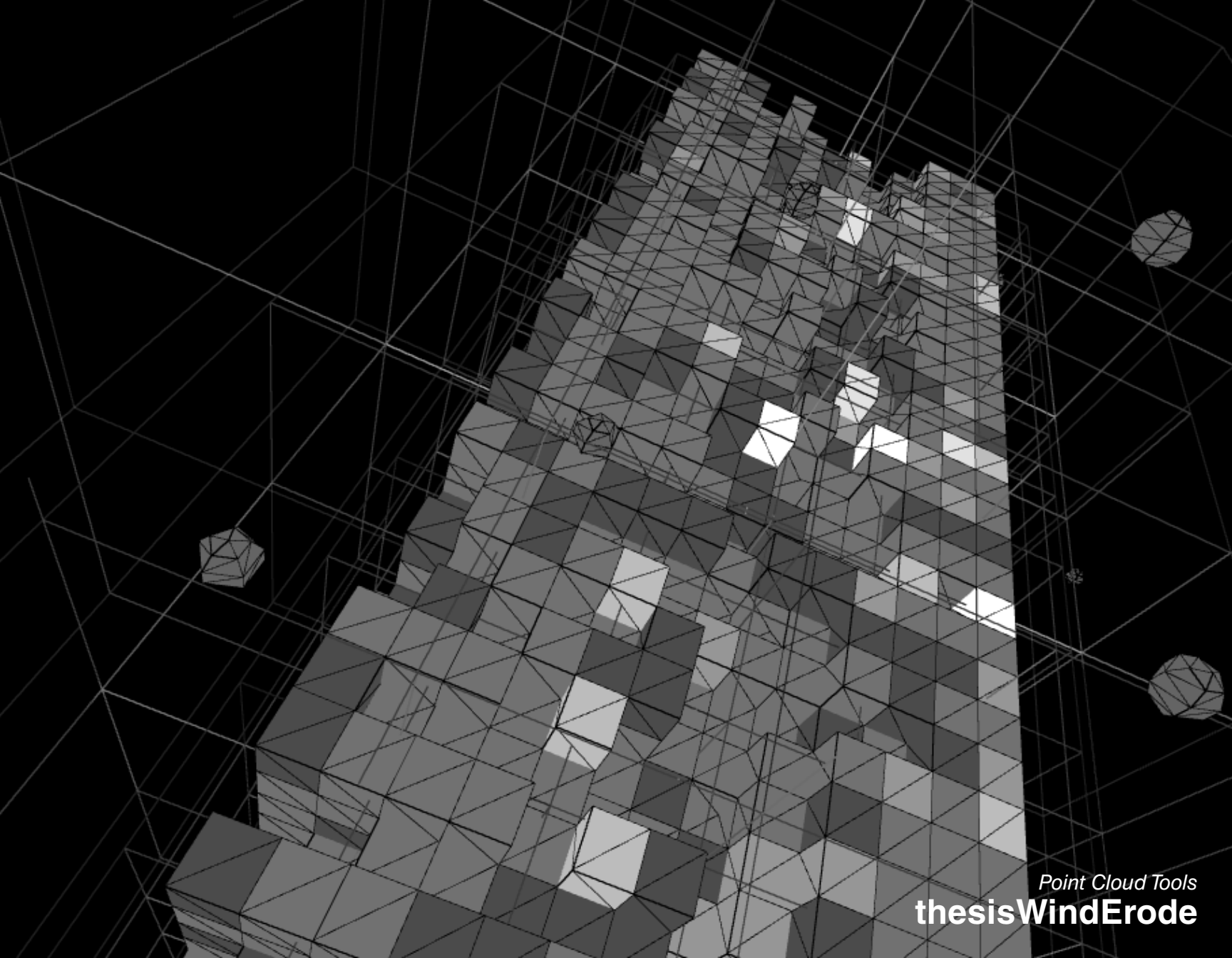
Our analysis of winds acting on a generic form derived a few criteria. Letting wind into the space is good during warm months, bad during cold months. Assuming a number of things, our site is in the northern hemisphere, the space will be unconditioned, etc. Using weather data, degree days, wind direction averages for the year, we were able to create two source angular domains that described the possible directions of wind during summer or winter. Wind 'particles' would then fly towards the object. Upon collision, quite unlike real wind, it would negatively or positively affect the point with which it collided.

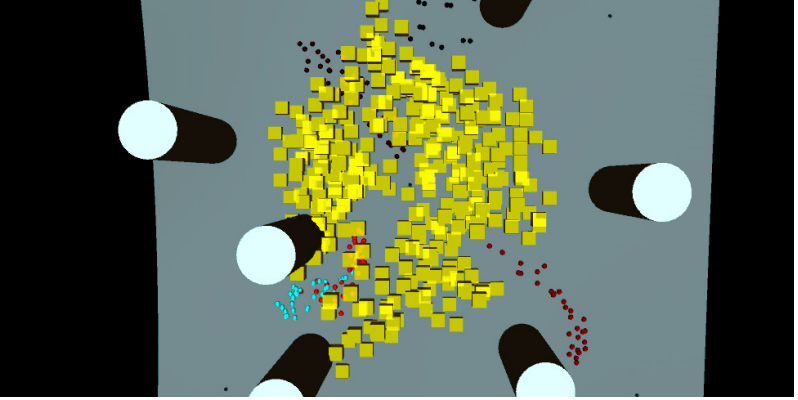
This sets up a basic value system, a scoring method, to determine the balance of wind on a point of the PointCloud. If the particle is from one of the cold wind angles, it would positively score the point. If it was from the warm angles, it would subtract from the point's score. If the score dropped below a threshold, it would be removed, and points behind would be left in the breeze. The behaviour this sets up is that winter winds would reinforce the edges of the point cloud exposed to their influence, while summer winds would carve into and, in some climates, through the mass of points. However, there is most always some angular overlap. The magnitude of how many particles came from which direction could be proportionally manipulated and based on things like speeds, or which was of more importance.

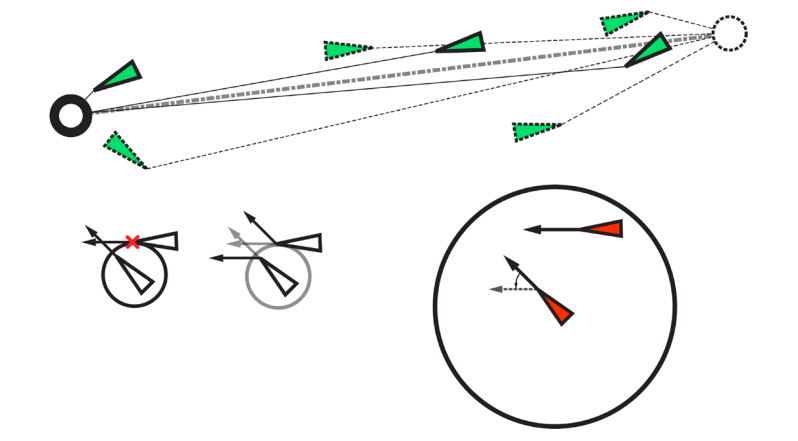
Environment: Java/Processing
Category: Point Cloud
Input: Points
Output: Points

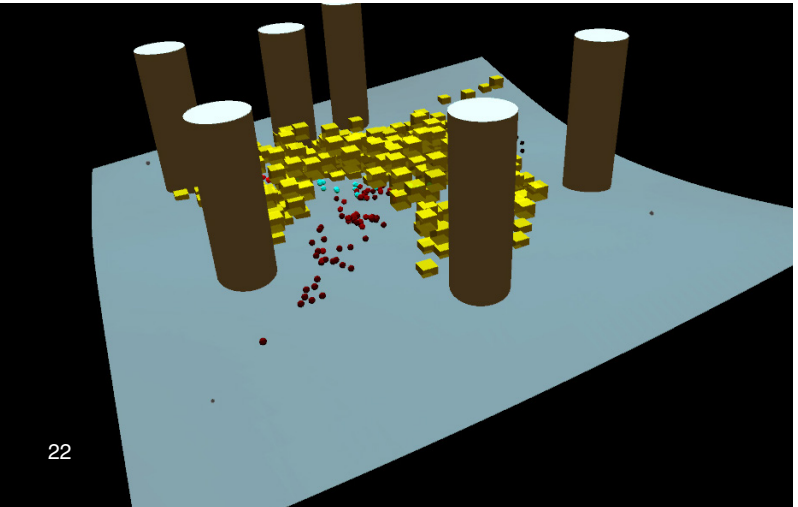


20.1 - 5
Screen captures(far left).
Section diagrams over time as Wind script erodes(left).


Point Cloud Tools
thesisWindErode







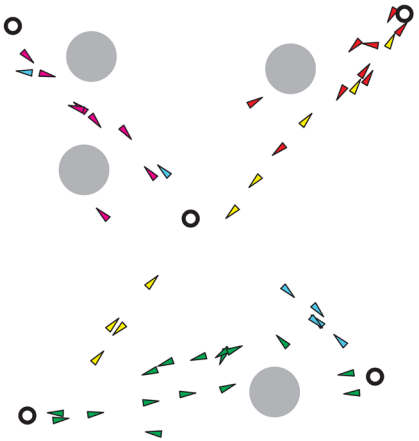
AGENT BASED CIRCULATION MODEL

Circulation acts on a point cloud with a high level abstraction of human movement behavior. It has a series of destination points, and agents who move between those points, pushing the base point cloud around based on the generated circulation path.

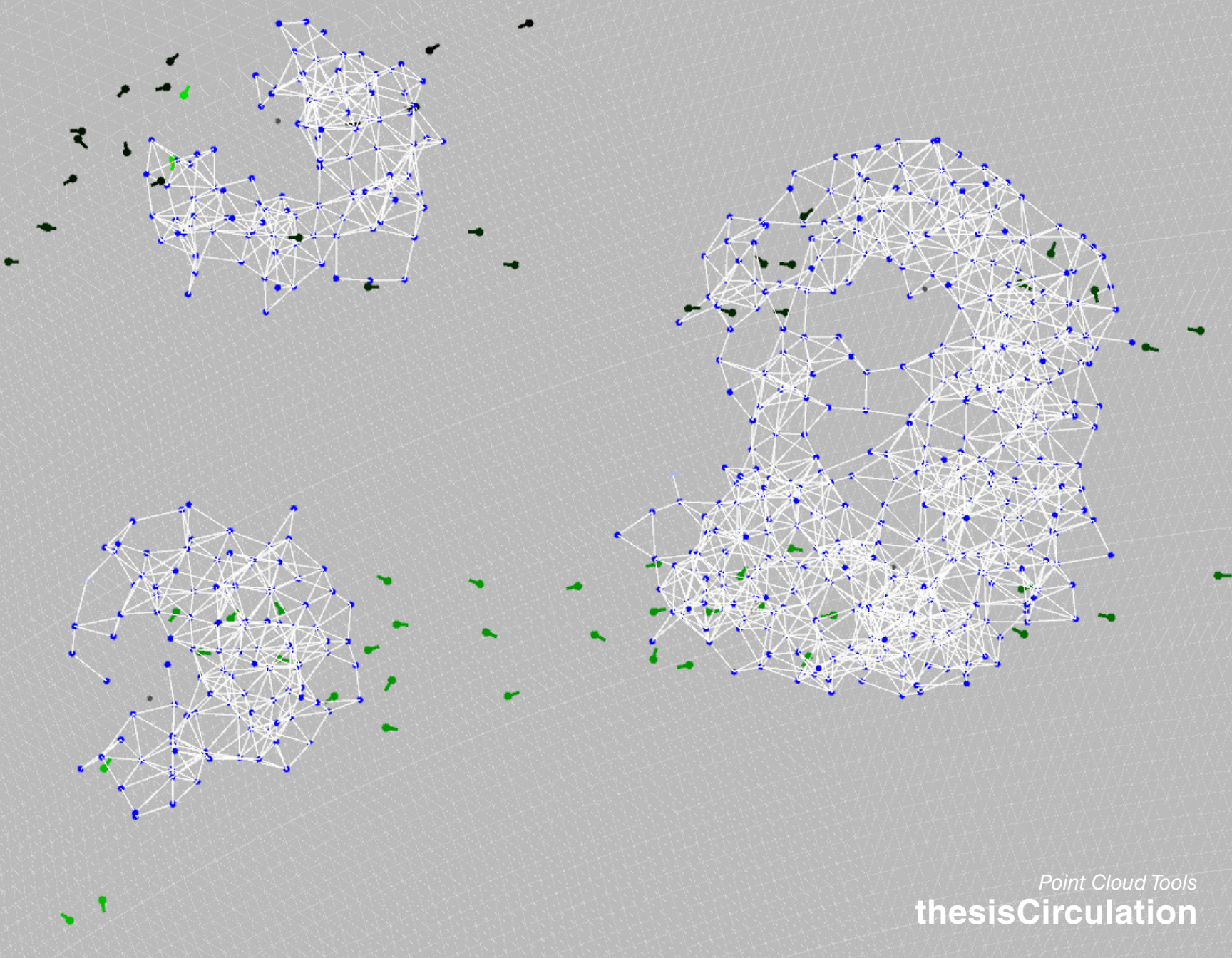
Agents embody a few key abstractions of the human psyche to best achieve a goal-directed circulation model. Each agent has a boundary, and will avoid collision between itself and other agents or environmental obstacles. Each agent was assigned a group to which it would be attracted to travel with. Each group had a pair of destinations that they would travel between.

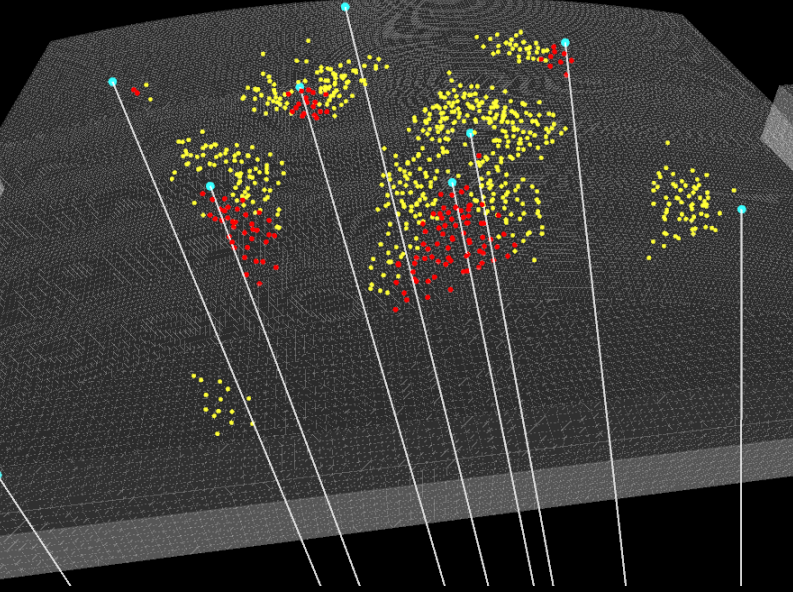
This fundamental behaviour mimics quite closely the site circulation that we observed of students between classes on campus. While other sites would pose radically different motives, patterns, and behavioural criteria, the campus allowed these factors to govern most human foot traffic in our sites of interest.

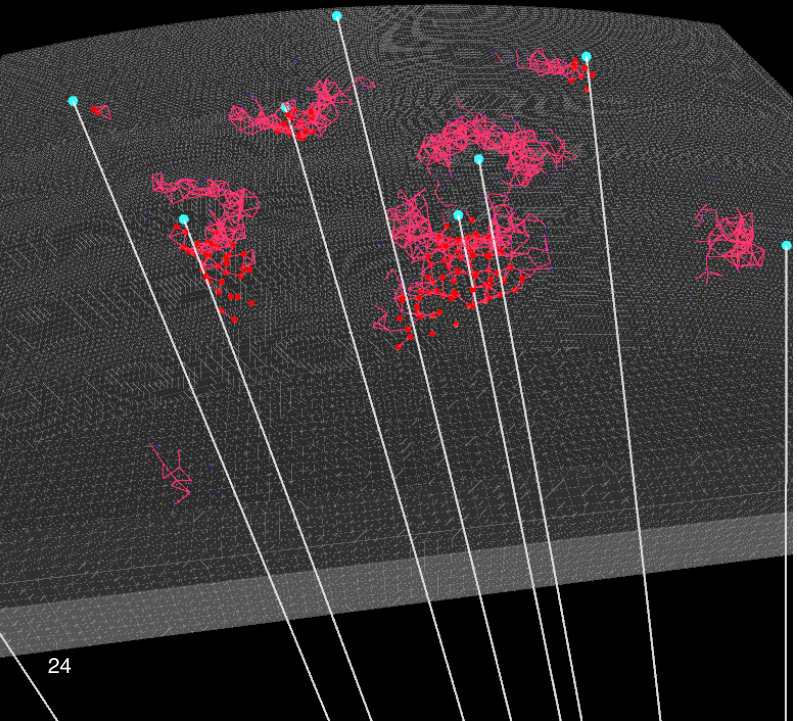
Environment: Java/Processing
Category: Point Cloud
Input: Points
Output: Points



22.3 - 6
Early iterations of Circulation showing agents carving through Point Cloud while negociating obstacles


Point Cloud Tools
thesisCirculation





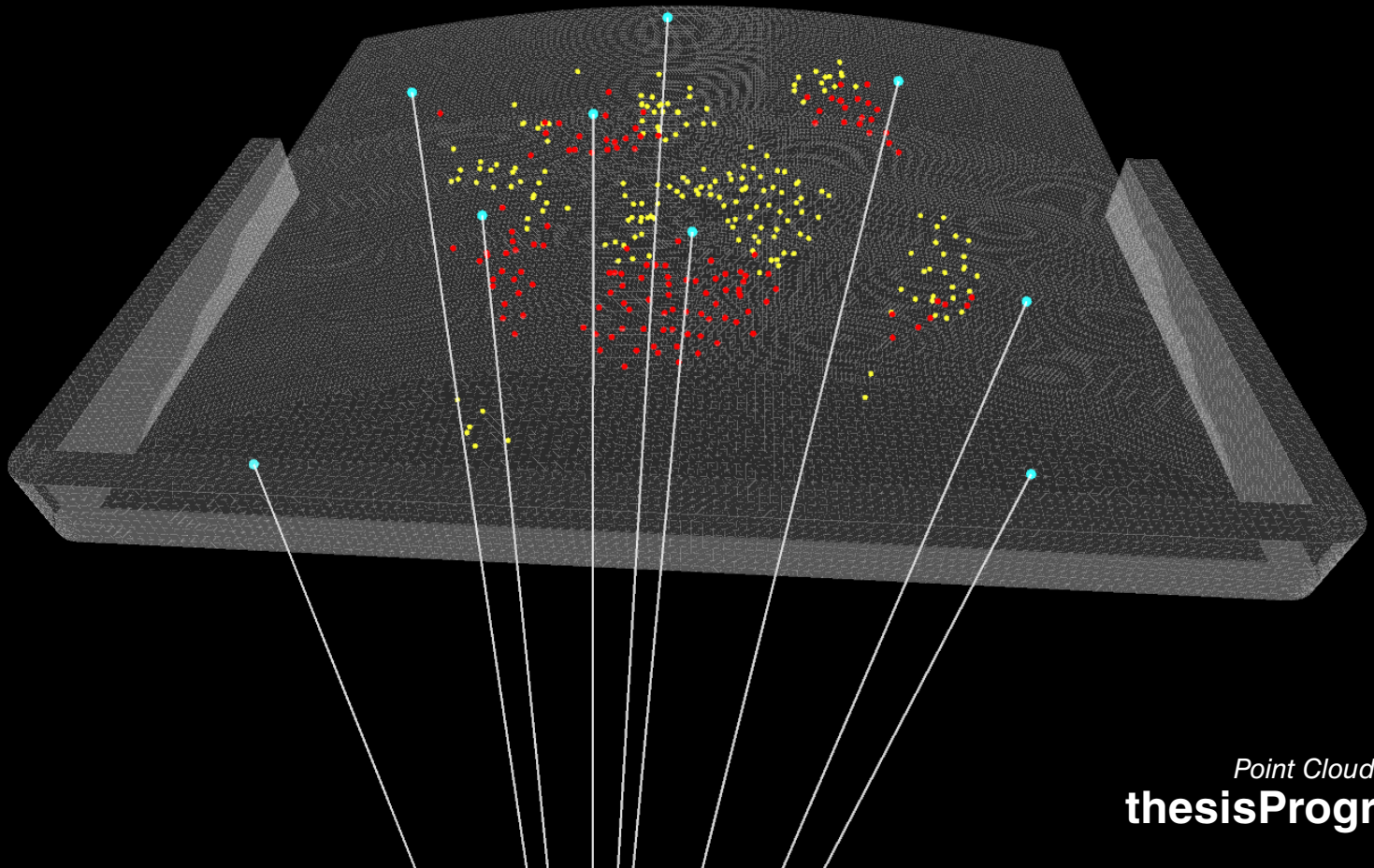
SIMULATED USE AND VIEW

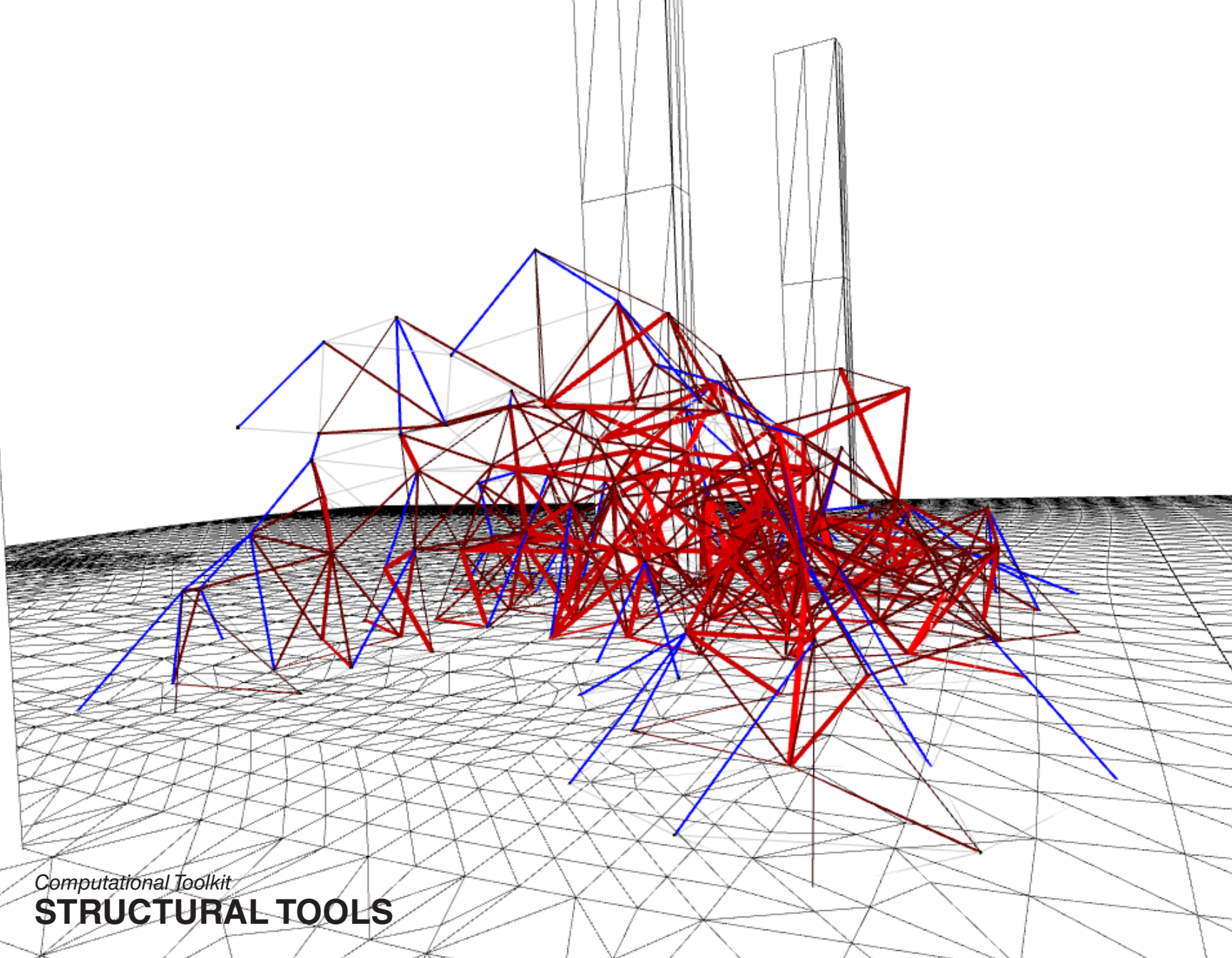
Program was the tool designed to address issues of usage, program of the space. It was clear that it would be a simple matter of hosting space for people to gather in some number, and allow them to feel enclosed but still have visibility outward, perhaps to something of interest.

Attractor nodes would be static locations that agents in Circulation would circulate between, and agents in Program would gather and meet at. The agents also had a view of interest, if that view was blocked they would move to regain line of sight. The points in PointCloud would then be attracted to the agents, but the radius of the agents began to force the points to pile up and around each other to achieve equilibrium. As the agents would settle, the PointCloud would gather and begin the first suggestion of spatial enclosure.

Environment: Java/Processing
Category: Point Cloud
Input: Points
Output: Points

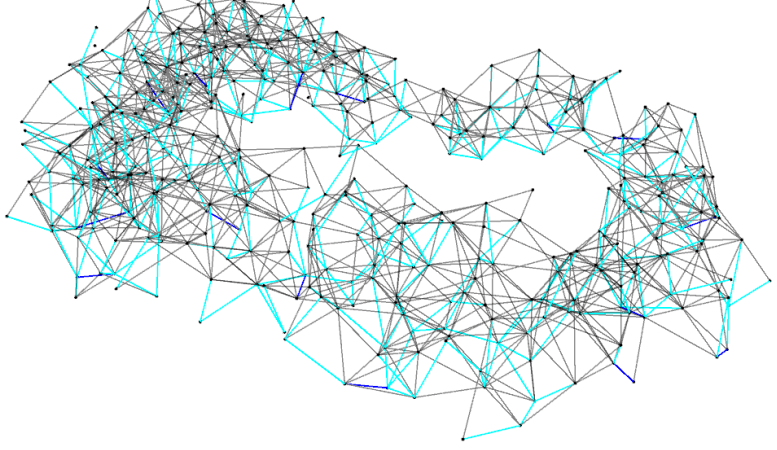
24.1 -2
Red shows the nodes that are in the cone of vision and are under a repullsion force


Point Cloud Tools
thesisProgram



Computational Toolkit

STRUCTURAL TOOLS

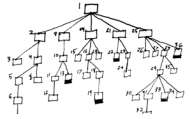


STRUCTURE: CONNECTIVITY & RELATIONSHIPS

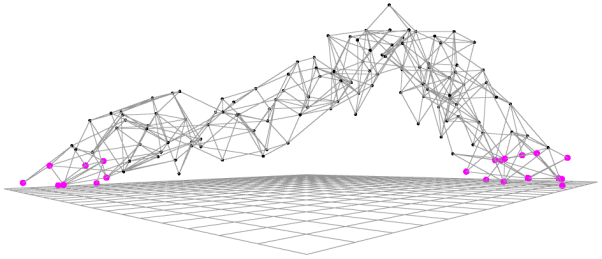
This is the critical bottleneck in the toolkit system where the level of data that will become the representation of a later constructed from increases in complexity and dimension. Here is when we begin to draw lines from points.

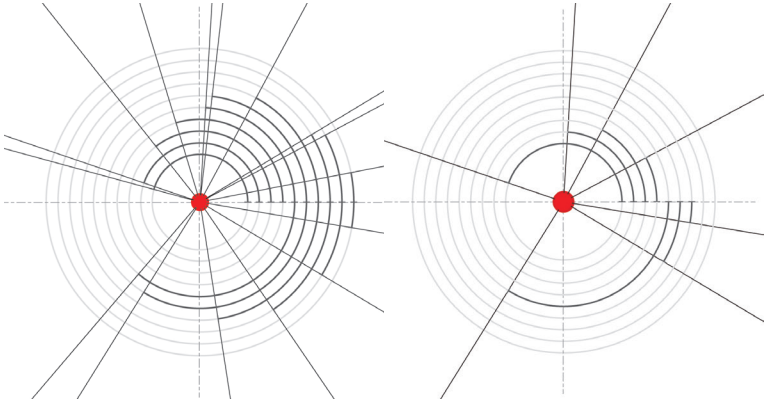
These lines we call Connections, the digital representation of a few simple properties to uniquely describe its place the the assembly. Every Connection has its own unique ID, as well as an array of two numbers: the ID's of the nodes they bridge. Length is also a property but is not computed at this stage in the toolkit.

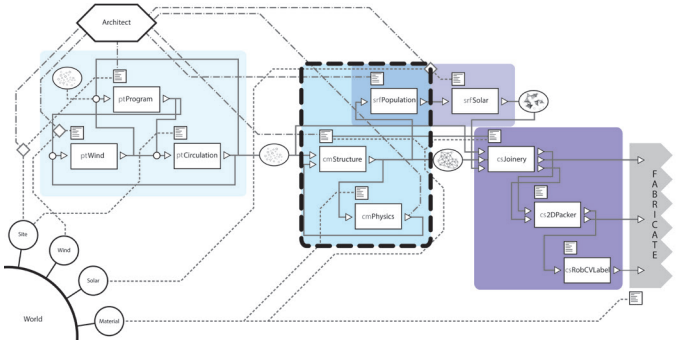
This chapter describes all of the various tools that populate, evaluate, cull, and provide feedback about the relative success and failures of a given arrangement of the ptCloud. Just because structure can be populated for most any proximity of points does not mean it will be a stable structure. A particular density of mutual connectivity of nodes needs to be achieved to allow forces to distribute more evenly. While at the sametime some angular and local densities can create impossible joinery scenarios (future collision of members and material).



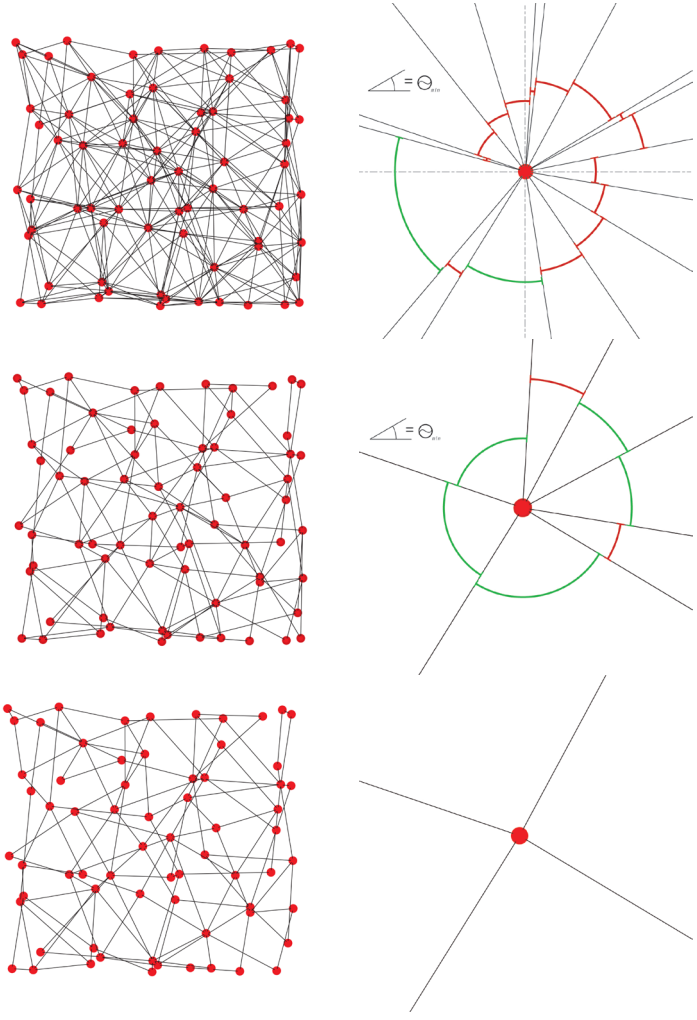
27.1 - 2
Designing an effective way to evaluate a great number of conditions







27.3 - 6
Image of the structure solver, physics and brute force connectivity.



BRUTE FORCE CONNECTIVITY

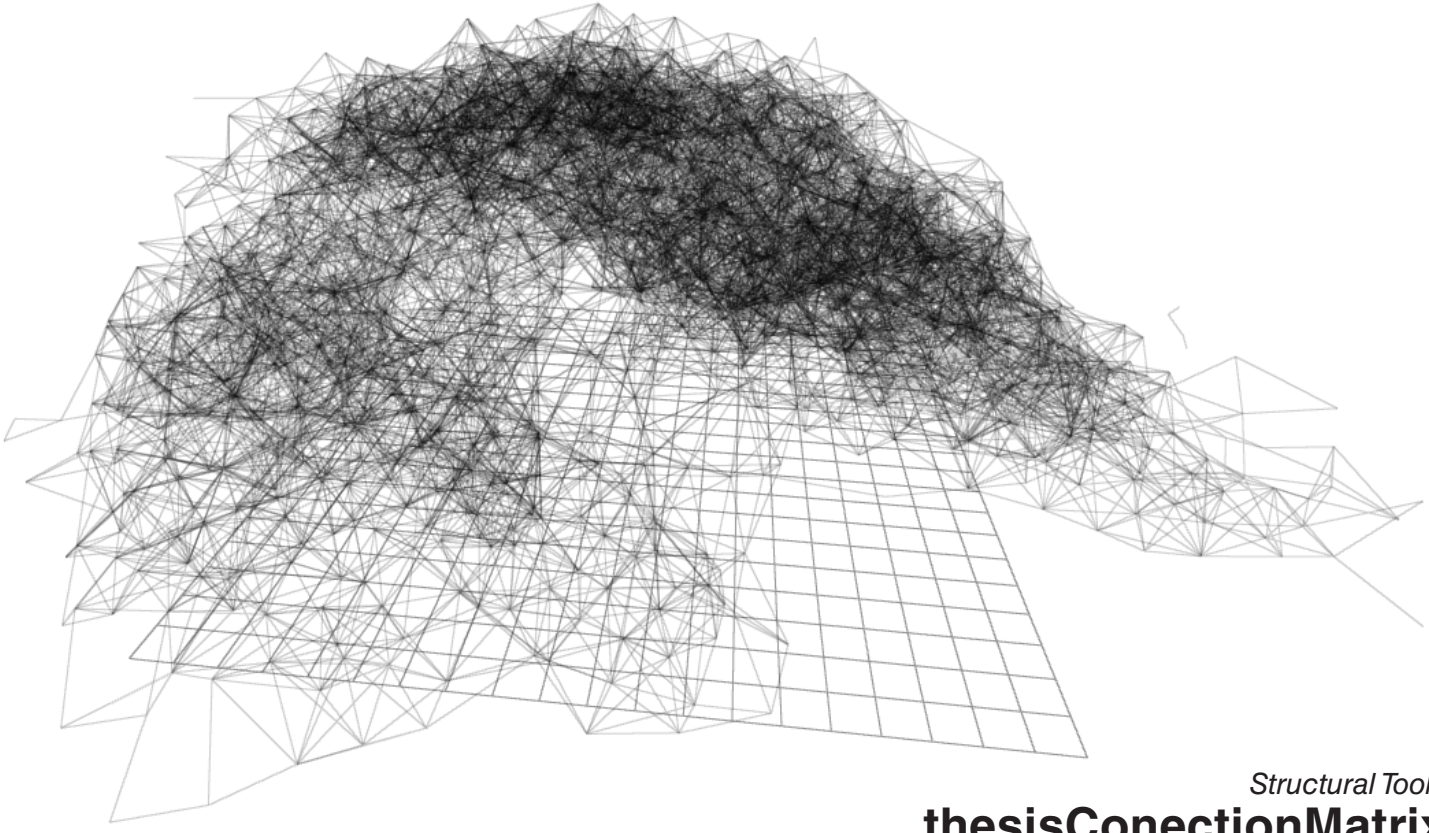
ConnectionMatrix was only a small part of a large piece of code that simply parsed the point cloud and created unique connection objects (lines) between points that were outside of a minimum radius, and within a maximum radius. This is only slightly more informed than doing a simple nested loop of one point connecting to all others. It is was important that geometric duplicates did not exist. If point A connected to point B, then point B could not reproduce a connection to point A. These objects are not equivalent, as their directions are inversed, but for our purposes they were and would produce geometric duplicates, which would violate the physical nature of the real objects that these digital objects were an abstraction of.

The bounding radius of search from one point to the next was a global parameter governing the minimum collision radius of points in PointCloud tools and the falloff of other forces. These radii parameters were pulled from our joinery design and prototypes that required a degree of separation. The lumber members spanning between, we also knew, should not be beyond a particular length. Thus the radii parameters were determined, and integrated as a parameter for the initial connection population.

This tool exists at the first crux, or bottle neck of the total toolkit system. Up to this point, the level of data being passed and manipulated were just point data. This tool introduces a higher level of order and information. Now the points in the point cloud were understood as pairs of connected points, and each point could be associated to its connecting neighbors in a topological diagram. From this tool to the next, we begin tools that analyze and introduce additional parameters to this new level of information.

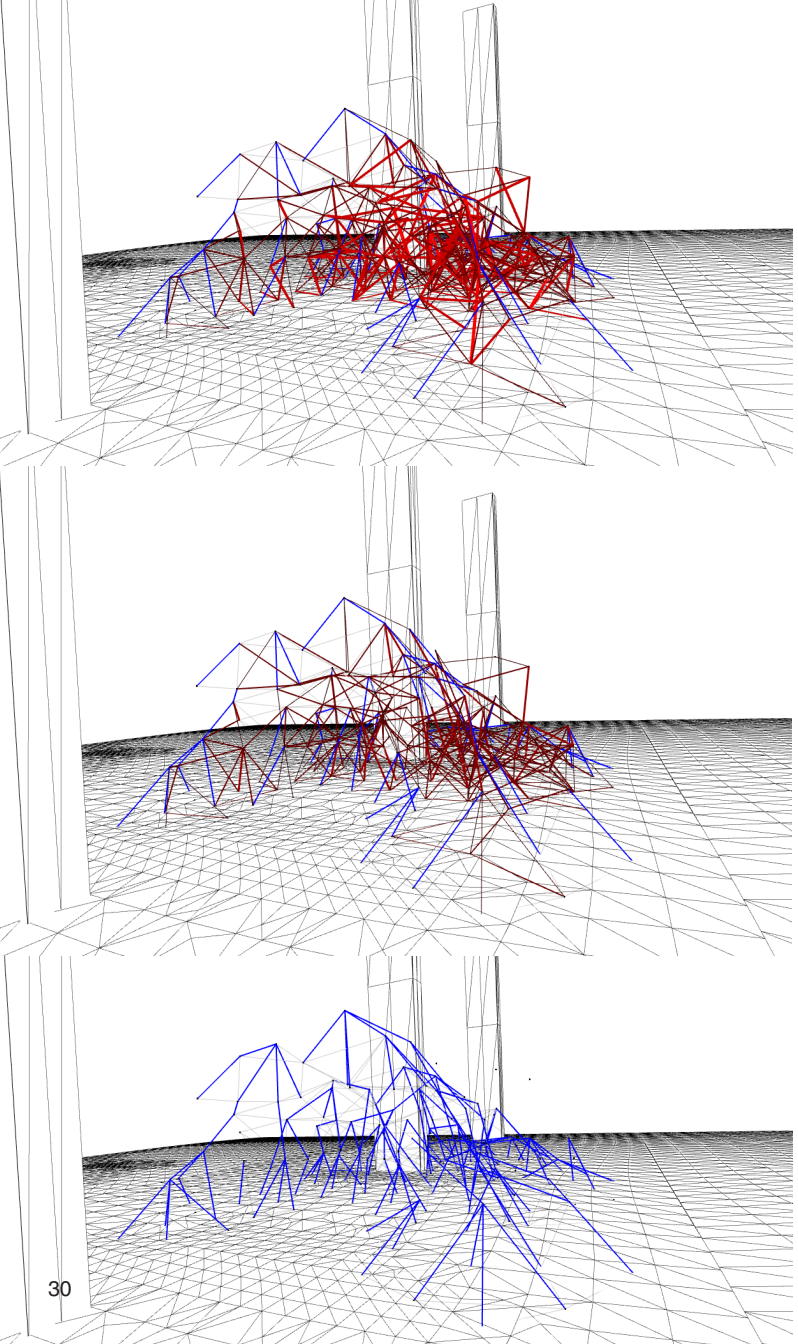
Environment: Java/Processing
Category: Structure
Input: Points
Output: Raw connectivity

28.1 - 6
Diagrams showing system-scale and local-scale connectivity rules, application and result over several iterations of the solver.



Structural Tools

thesisConectionMatrix



EVALUATING CONNECTIVITY

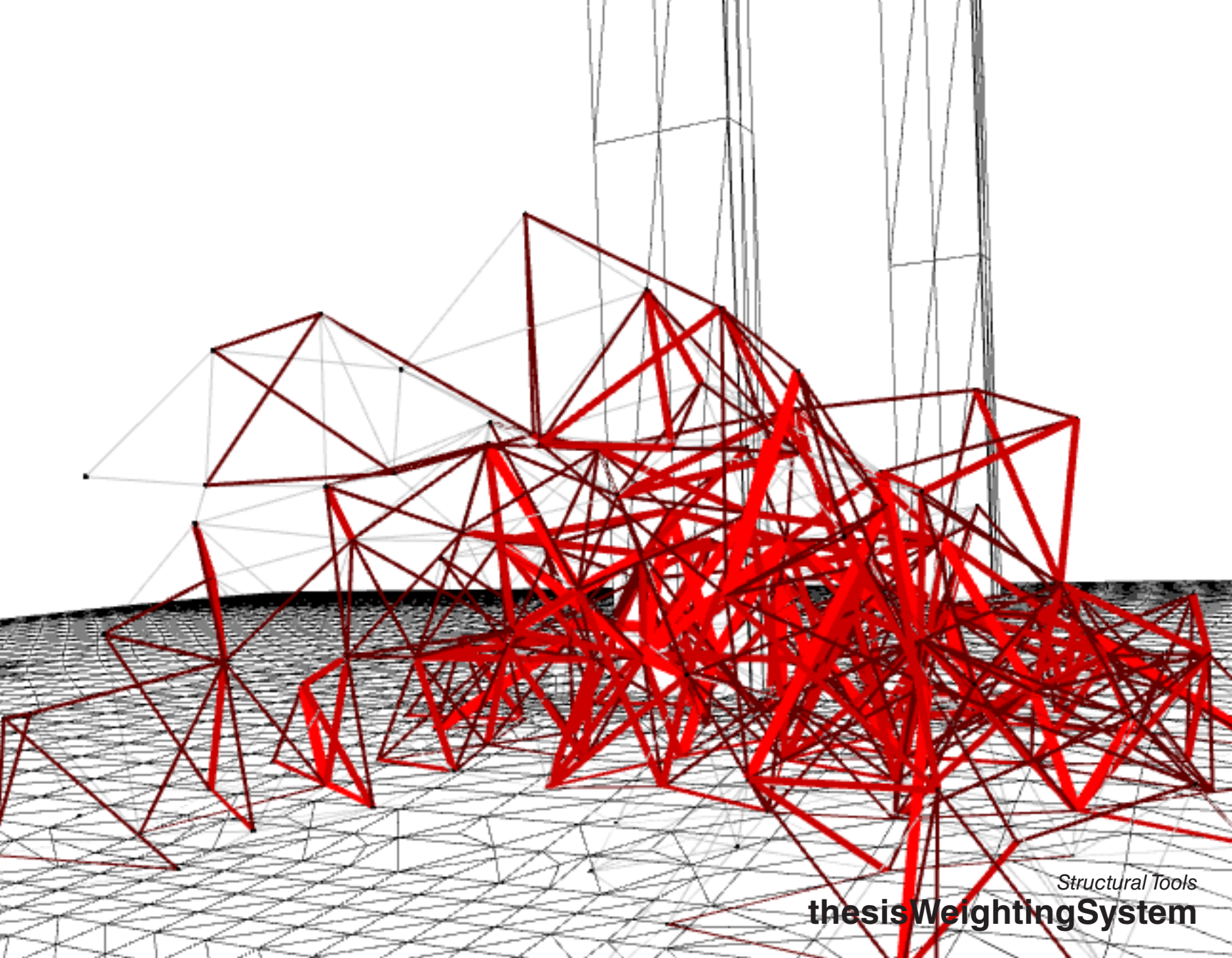
WeightingSystem takes the raw connectivity from ConnectionMatrix and begins an evaluation based on particular criteria derived from joinery tolerances and constraints. WeightingSystem was designed as an iterative solver. Performing all of the connection removal in one pass proved to remove connections that could have been kept. The rules that governed local joinery constraints conflicted directly with the needs of the overall system-level of maximizing connectivity.

One primary criteria, Planar angle proximity, determined how close two connections could be at the same node in terms of their angle of separation on the XY plane. At each node, in one pass, each connection would check the neighbor ahead and behind in angular proximity. If the angular distance was less than the tolerance, the current connection would be negatively scored. The angular distance of both of its neighbors to each other would also be checked. This would add an extra level of consideration as it was possible that the removal of one between, could fix a set of 3 that were too close. However through many designs of this script it was clear it was most effective as a recursive solver, and trying to solve for local-node criteria and maximize system connectivity in one pass was impossible.

A number of other factors also played in to negatively or positively score the connections in ConnectionMatrix. WeightingSystem would loop, remove the most negative score, and do so until all negative scores were removed, then perform one final removal pass of anything that was still outside of the tolerances of required criteria. This would occur if the positive scoring mechanisms kept certain connections alive, as they might be useful in other criteria. If by the end, they did not persist beyond its negative influence, it would be removed. This hierarchy of goals and values was a constant battle of what had to happen at a micro-joinery level, and what should be maintained at a system assembly level.

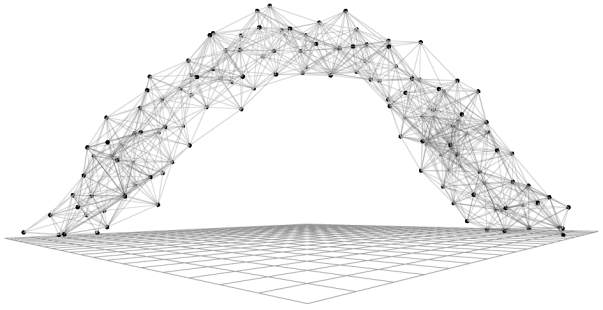
Environment: Java/Processing
Category: Structure
Input: Raw connectivity
Output: Evaluated connectivity

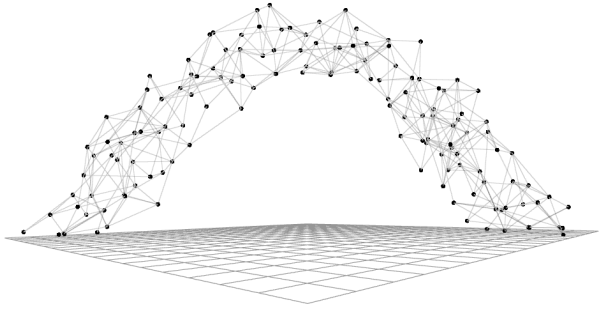
30.1 - 3
Several steps in the scoring process, rating each connection and coloring to indicate (lack of) favorability.

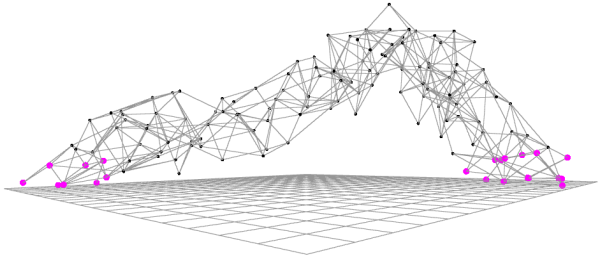


Structural Tools

thesisWeightingSystem







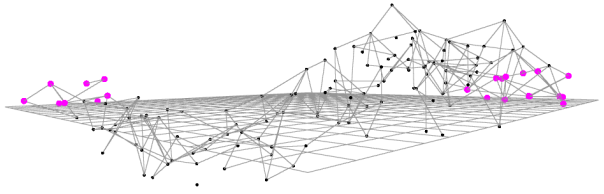
PHYSICS BASED EVALUATION

Physics was a simulation tool, that would achieve a rest-state equilibrium or indicate a clear failure, partial or whole of the assembly connectivity diagram. However the output of this tool was not something that was recursively fed back into the system, but was a feedback dump for the designer. At so few points do we, the users, ever get to really test the performance and evaluate results except by what graphic representations we have included.

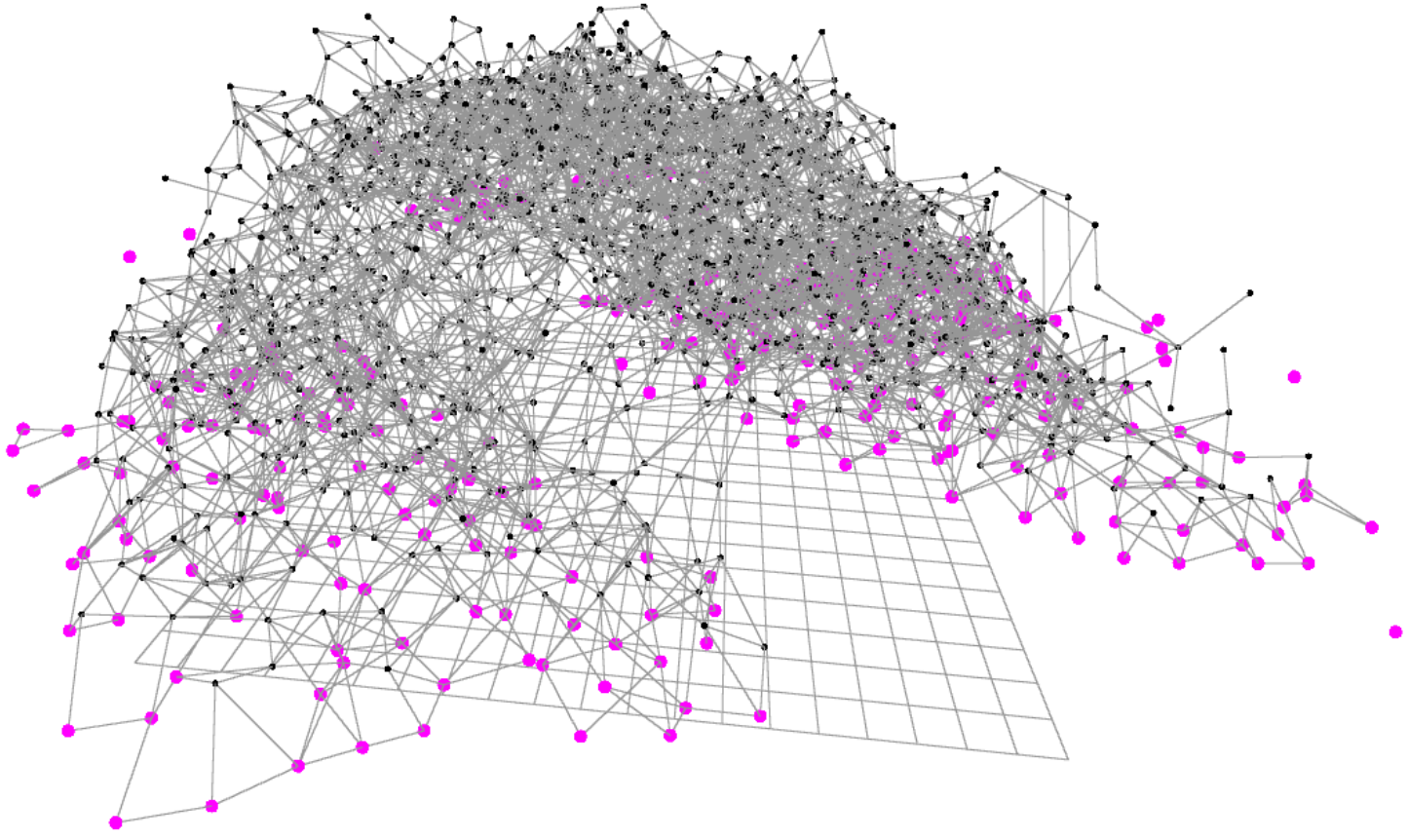
The Physics simulation would simply allow the assembly of nodes to relax. Some nodes near the ground surface would be locked as foundation conditions, assumed as bearing on the earth and immobilized. Connections could be allowed to rotate and flex within a particular tolerance during the simulation. Primarily, the simulation would clearly indicate issue of connectivity thinning, or relative distances from foundation support, dealing with moment.

However, this too was highly abstract. It was spring based. While rigid springs, connections would not behave this way as much as other missing behaviours. There was no rotational resistance, which the plate joinery would introduce. However this made it clear, our assumption that our joinery could be a pin-joint and not a moment connection was wrong. It was after trials through the observation of this tool that conclusion was borne.

Environment: Java/Processing
Category: Structure
Input: Evaluated Connectivity
Output: Visual test of stability

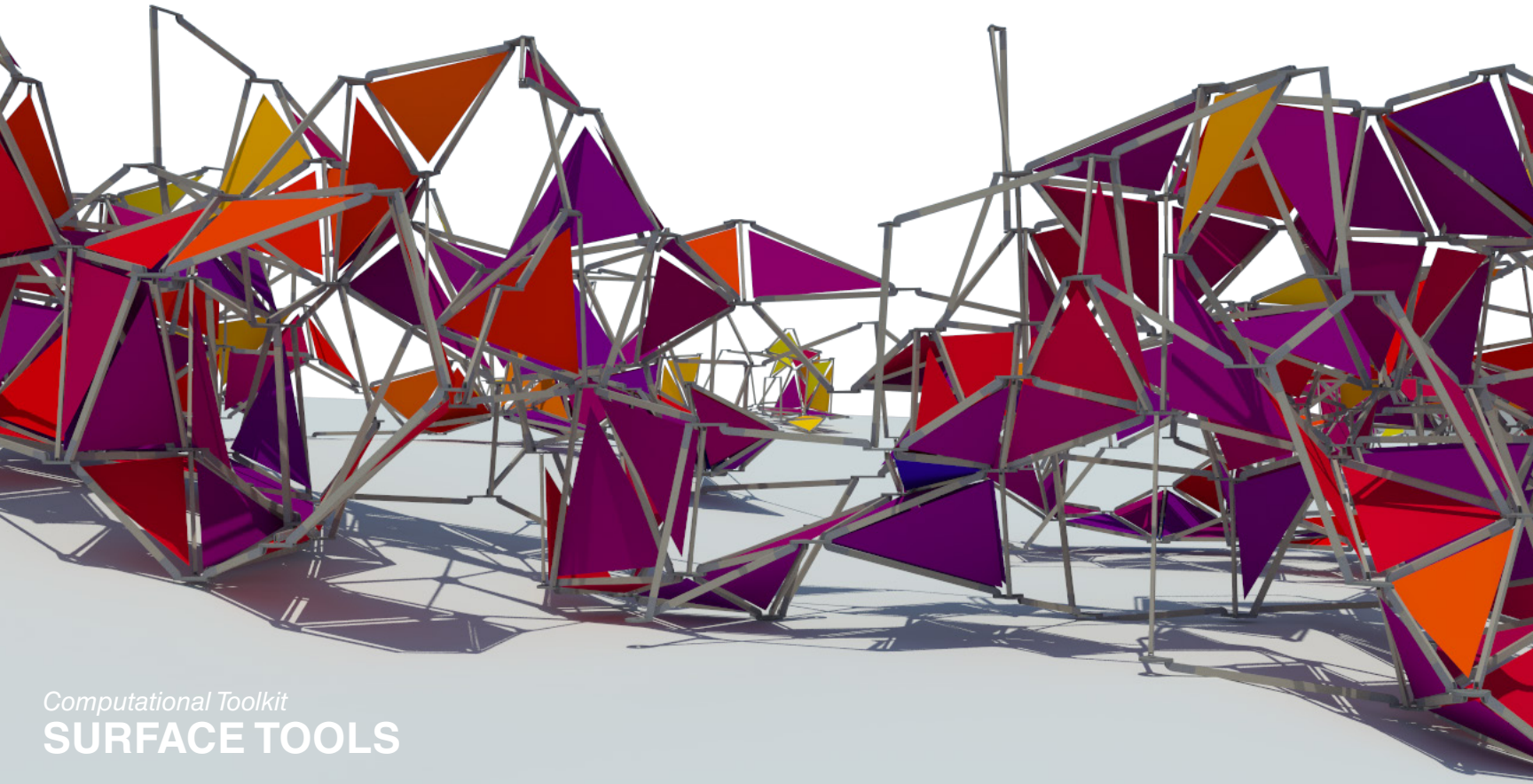


32.1. - 4
Screen captures of physics running on basic arch population of connections



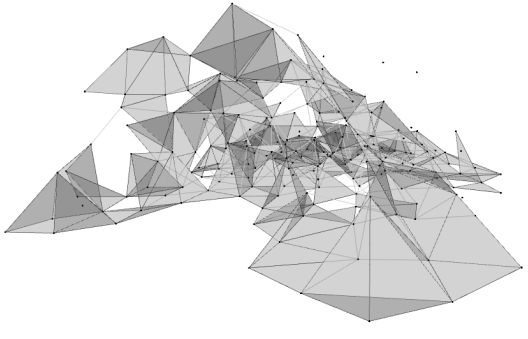
Structural Tools

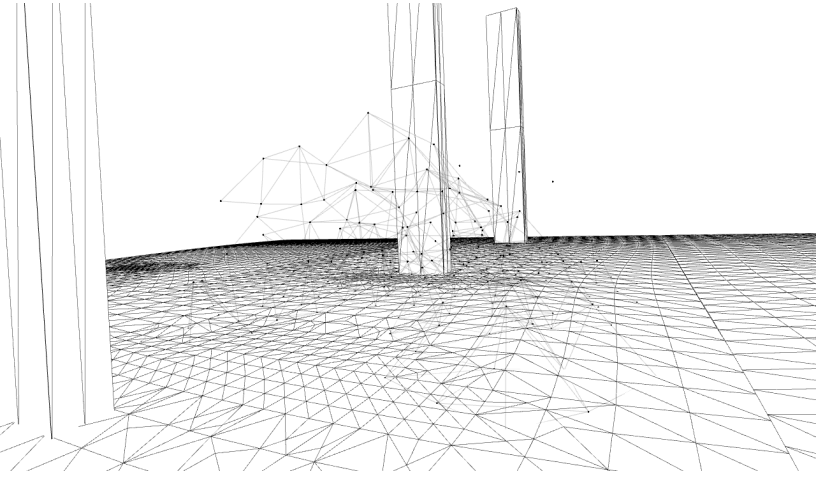
thesisPhysics



Computational Toolkit

SURFACE TOOLS





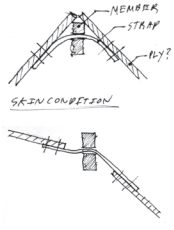
SURFACE INFORMATION

Again, the level of information increases to a higher order. Now it has become the parsing and generation of three mutually connected points of the point cloud that have successfully all produced connections in the ConnectionMatrix and found to be valid connections through the WeightingSystem.

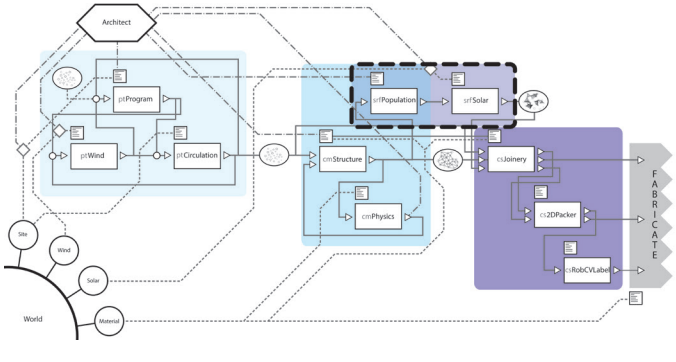
However the significance of this portion of the system is that Surface allows an area for forces to act and be acted upon. The connections primarily are a structural framework to best achieve the position of the nodes in the PointCloud.
Surface is what will block the winds, shed water, permit, bounce, block light, open and deny view, create enclosure and define space.

As the story typically goes, this is where parameters outside of our direct influence played most heavily. Our grant funds were already thin, and the addition of this surface, regardless of material would exceed limits. It also introduced fire code requirements that the rest of the specified assembly were not explicitly subject to.

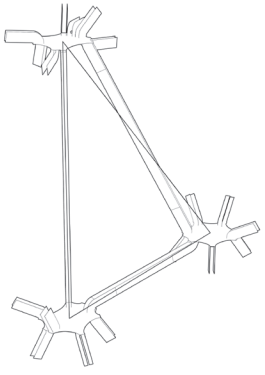
In the face of this, and complicated joinery resolutions, the Surfaces were never introduced to the full scale assembly.

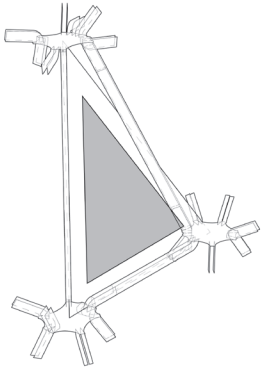


35.1 - 2
Sketches of early tectonic musings of how to mediate conflicting geometries and logics across the whole.



35.3 - 6
Shots of all steps in evaluation





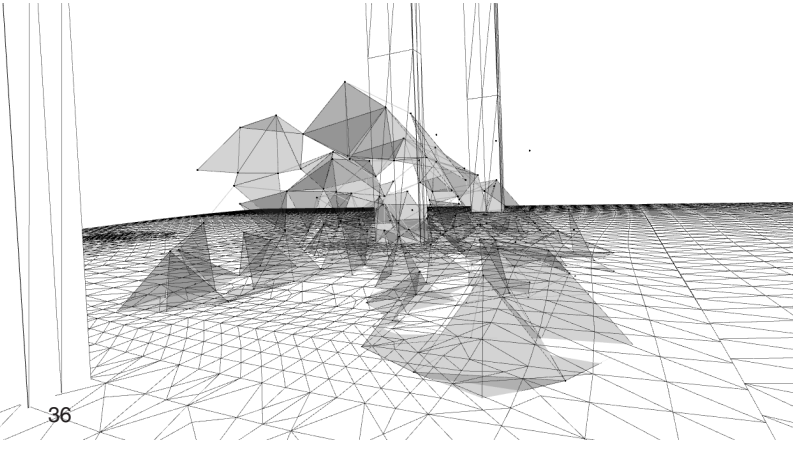
SURFACE FINDER

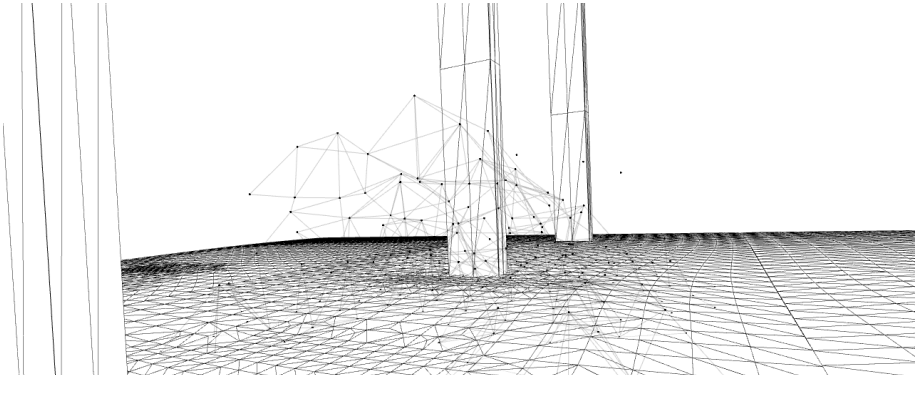
SrfPopulation was the parsing tool that walked the connections and nodes and found triplets of mutually connected nodes within the final ConnectionMatrix after running through the WeightingSystem. This produced a population of triangles within the topological framework of nodes and connections.

The next and most crucial step was checking for intersection, with other triangles but also with other connections that may pass through. An intersection of two triangles also suggests that the connections that define the edge, too, intersect.

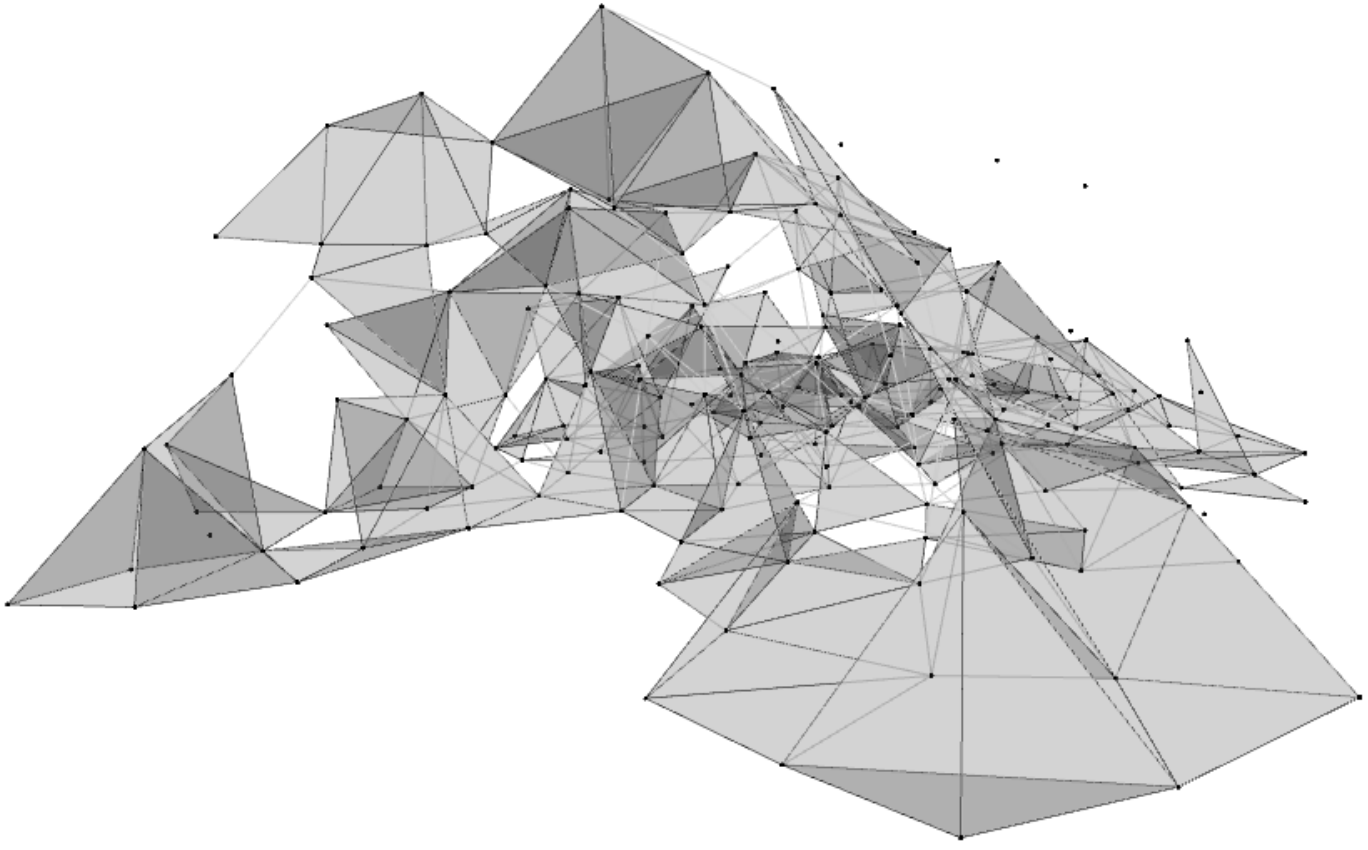
SrfPopulation performs on the system level the same way ConnectionMatrix did, by populating the existing data with higher level information within some constraints.

Environment: Java/Processing
Category: Surface
Input: Connections & Points
Output: Triangles





36.1 - 4
(Top 1-2): Attempting to understand complexity of spatial limits of panel geometry.
(3-4) Solved ConnectionMatrix shown with/out triangle population



Surface Tools

thesisTriangleFinder

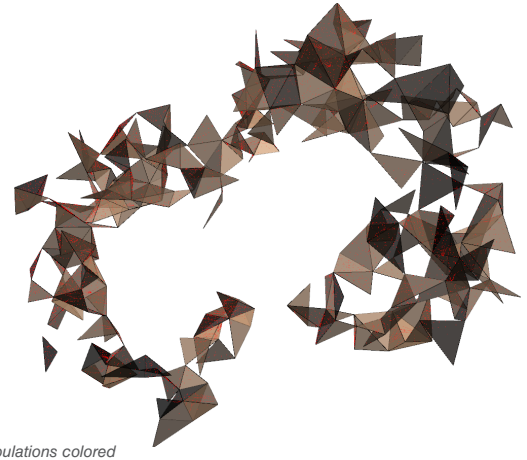
SIMPLE DATA ANALYSIS

After populating the total triangles that could exist within the assembly framework, SolarTriangles became the next tool to now introduce more environmental criteria into the system. It suggests that up until this point solar issues could not have been addressed as it is primarily an issue of enclosure. Again, this triangle object did not reach the level of material and joinery beyond our initial designs and sketches. The depiction of the colors in the render merely show a data-field condition with respect to relative solar exposures of all triangles in a given generation.

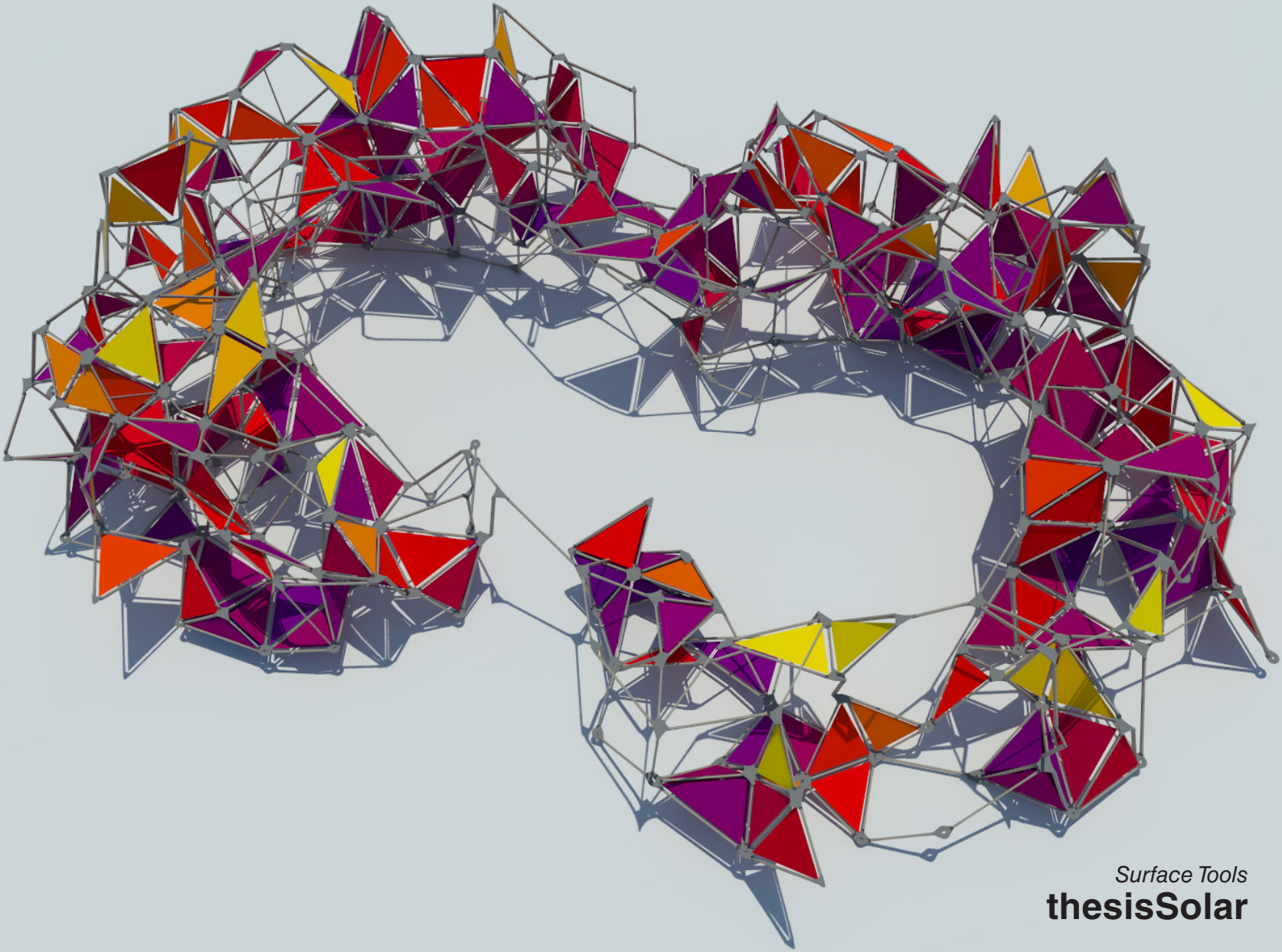
A triangle's solar value was calculated for the year long exposure angles for the site's latitude. This effectively became a list of vectors, similar to WindErode, but with three dimensional change. The value was also calculated against the triangle normal difference, which the cosine was then taken. Also calculated was local shading from other triangles for each vector.

Early iterations of this used this value to open or close internal geometries to alter the cut geometry of the face, which could block or allow light in.

Environment: Java/Processing/Grasshopper
Category: Surface
Input: Connections & Points
Output: Triangles w/ Weighted values

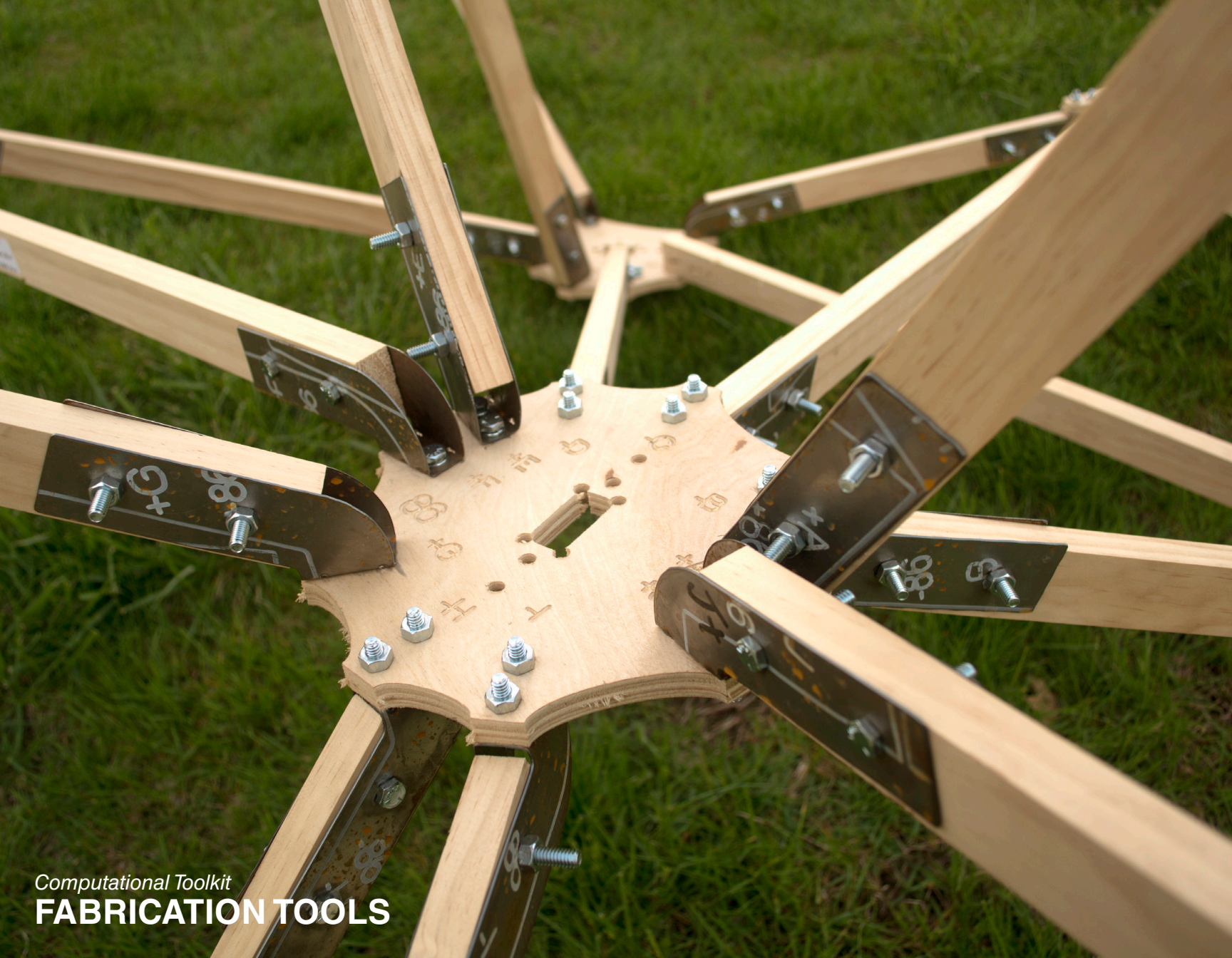


381 - 4
Renders of panel populations colored by solar vector occulsion values.



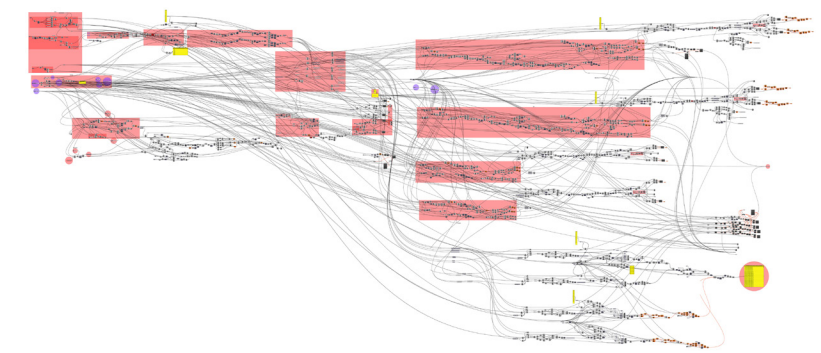
Surface Tools

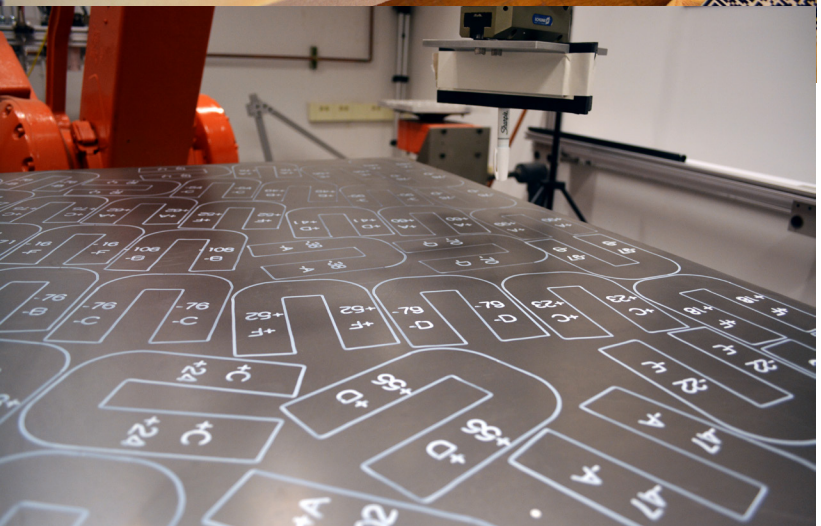
thesisSolar



Computational Toolkit

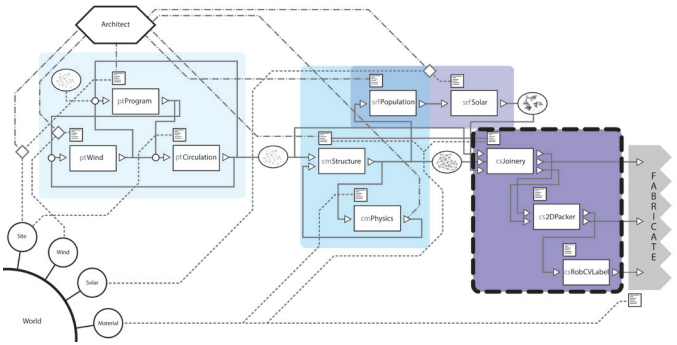
FABRICATION TOOLS



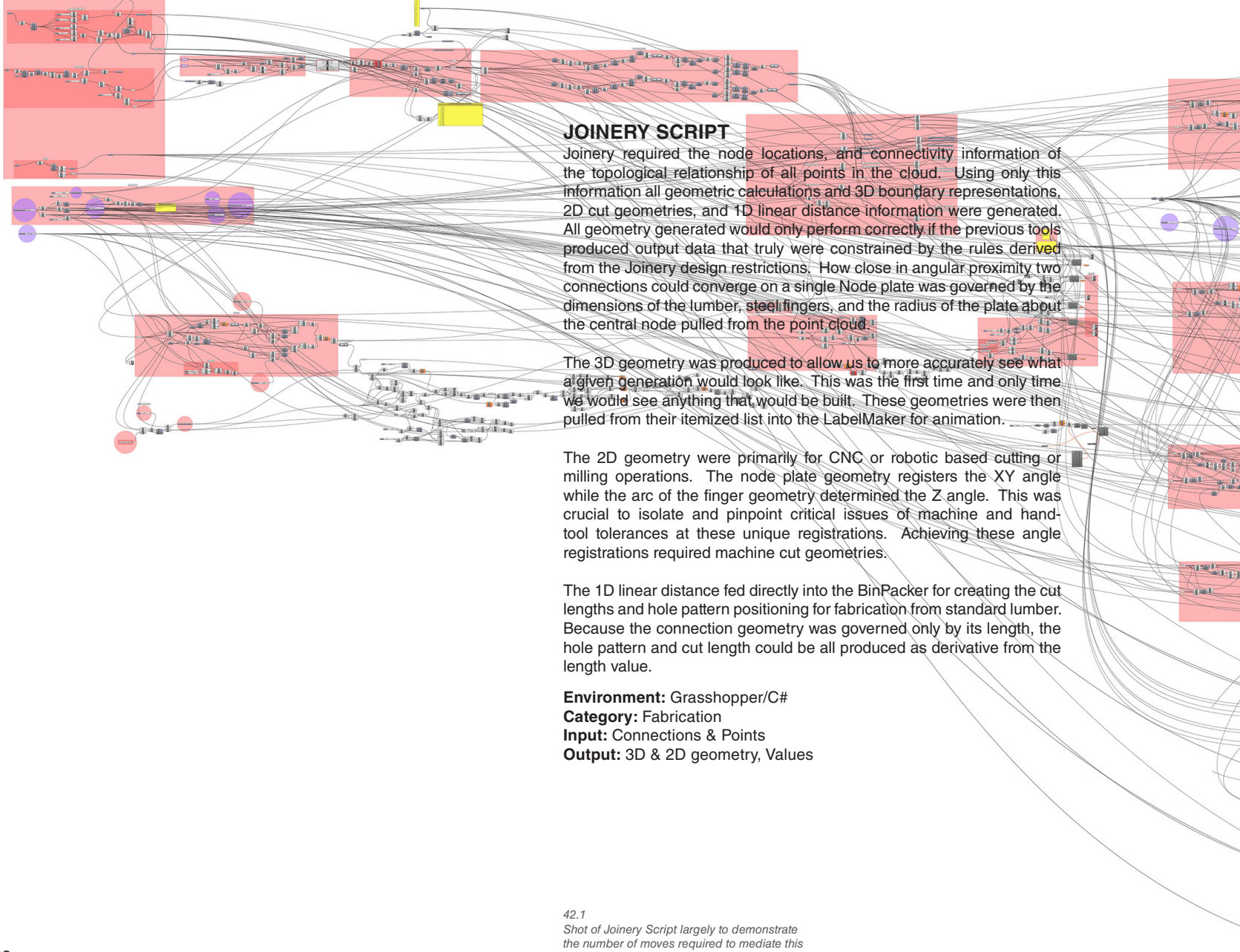


CONSTRUCTION TOOLS

At this final threshold of the toolkit system the objects, digital constructs, and spatial data now informs a process of physical manifestation. It marks the beginning of the fabrication workflow related tools, and applies no further manipulation to the generation. All tolerances, data types, data structure organizations, input variables, constraints, were derived and designed from the geometric construct tolerances of the Joinery script. Everything prior, while constrained by the rules of their physical analogs, did not behave as such. It is at this point that the translation from digital object to real objects occurs.



41.1 - 3
The digital and physical tools



JOINERY SCRIPT

Joinery required the node locations, and connectivity information of the topological relationship of all points in the cloud. Using only this information all geometric calculations and 3D boundary representations, 2D cut geometries, and 1D linear distance information were generated. All geometry generated would only perform correctly if the previous tools produced output data that truly were constrained by the rules derived from the Joinery design restrictions. How close in angular proximity two connections could converge on a single Node plate was governed by the dimensions of the lumber, steel fingers, and the radius of the plate about the central node pulled from the point cloud.

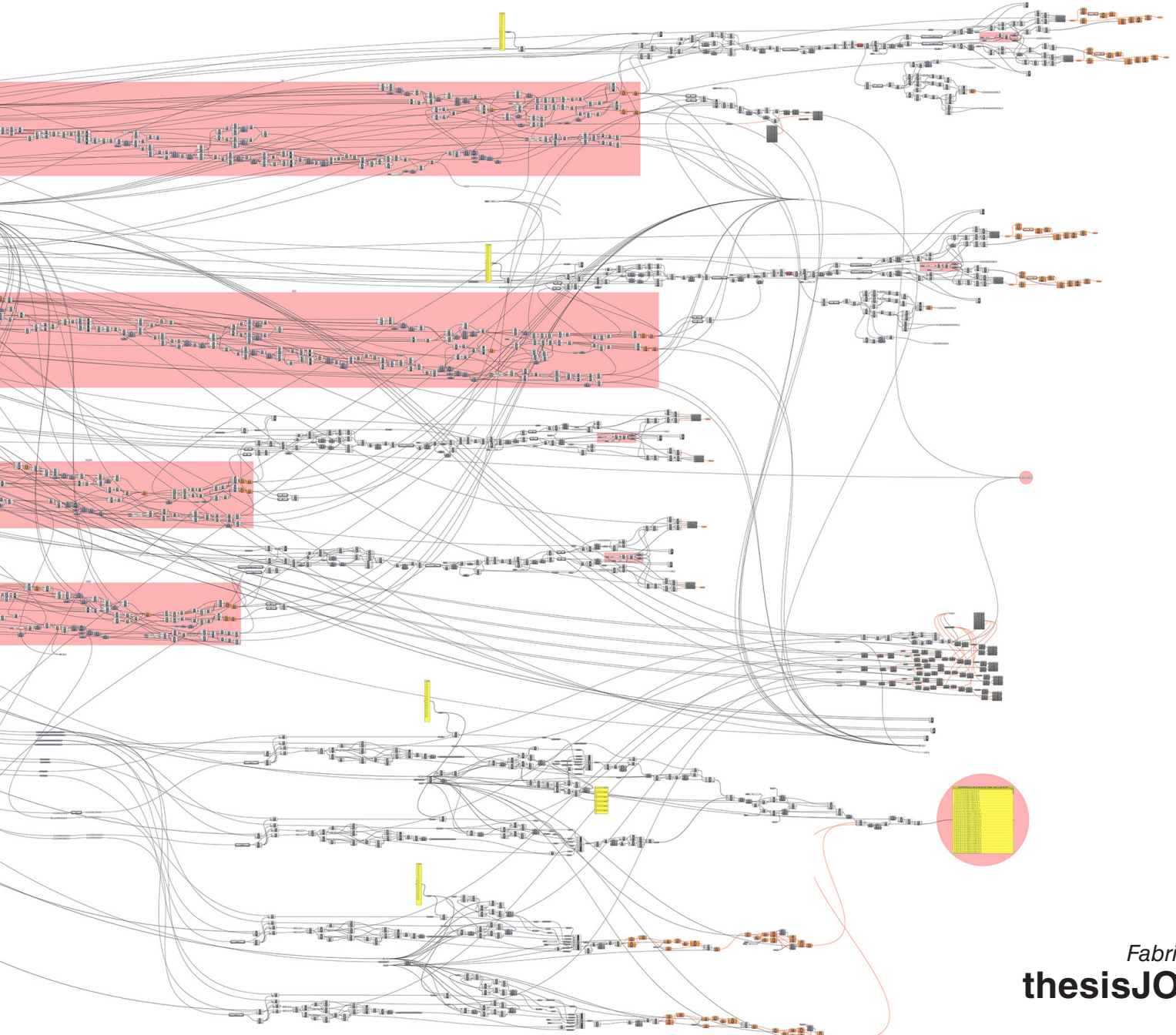
The 3D geometry was produced to allow us to more accurately see what a given generation would look like. This was the first time and only time we would see anything that would be built. These geometries were then pulled from their itemized list into the LabelMaker for animation.

The 2D geometry were primarily for CNC or robotic based cutting or milling operations. The node plate geometry registers the XY angle while the arc of the finger geometry determined the Z angle. This was crucial to isolate and pinpoint critical issues of machine and hand-tool tolerances at these unique registrations. Achieving these angle registrations required machine cut geometries.

The 1D linear distance fed directly into the BinPacker for creating the cut lengths and hole pattern positioning for fabrication from standard lumber. Because the connection geometry was governed only by its length, the hole pattern and cut length could be all produced as derivative from the length value.

Environment: Grasshopper/C#
Category: Fabrication
Input: Connections & Points
Output: 3D & 2D geometry, Values

42.1
Shot of Joinery Script largely to demonstrate the number of moves required to mediate this crucial step in the system's bottleneck.

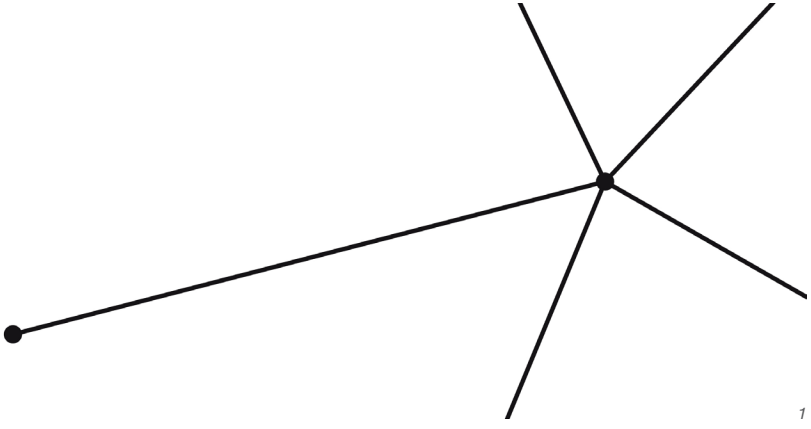


Fabrication Tools

thesisJOINERY

BASE GEOMETRY

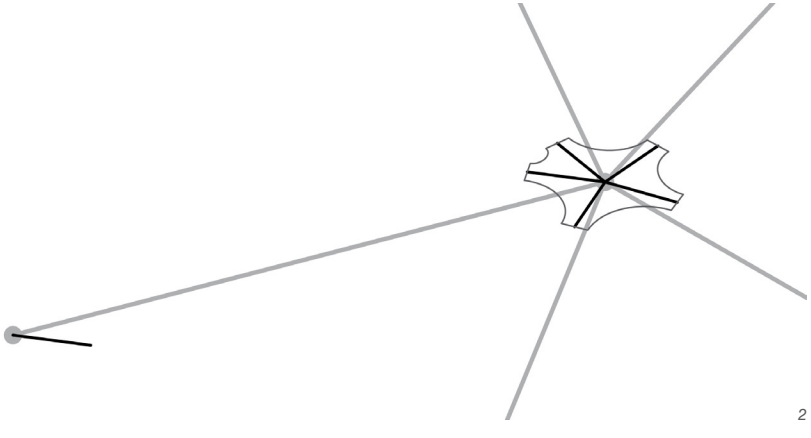
Each point in the PointCloud that survived the WeightingSystem are evaluated in respect of their connected neighbors in the topological diagram.



1

POINT OFFSET

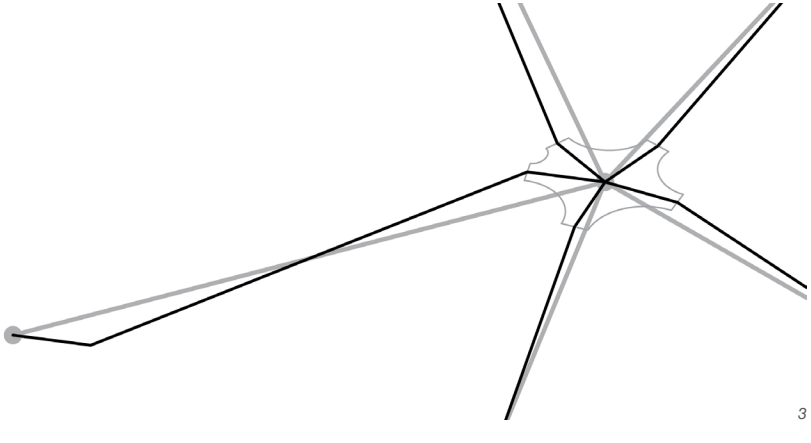
Plate geometry is generated primarily by a constant planar offset in the direction of each connection. This is the centerline of the XY component of the vector of each point to its neighbors.



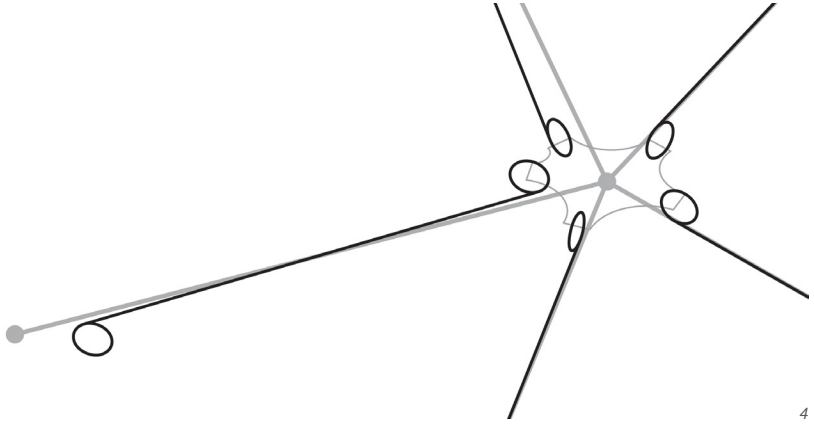
2

NEW CONNECTION AXIS

It is then noted that the directionality of the connection becomes slightly more vertical. This weakens the transfer of forces, but is necessary for the population of real dimensional objects to be assembled.



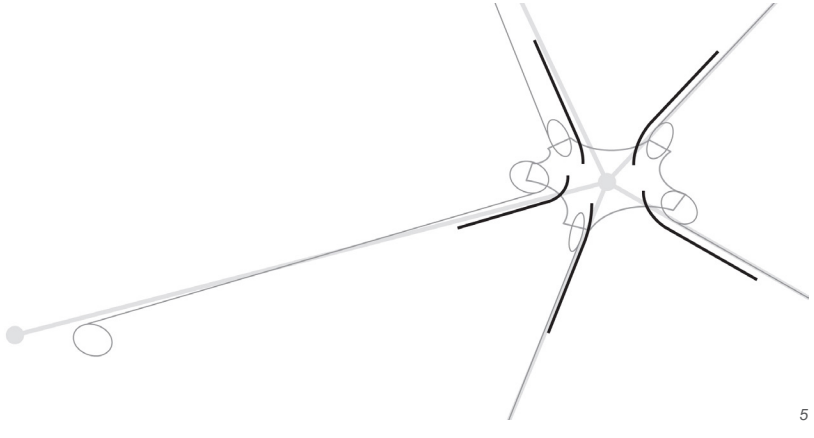
3



4

TANGENT OFFSET

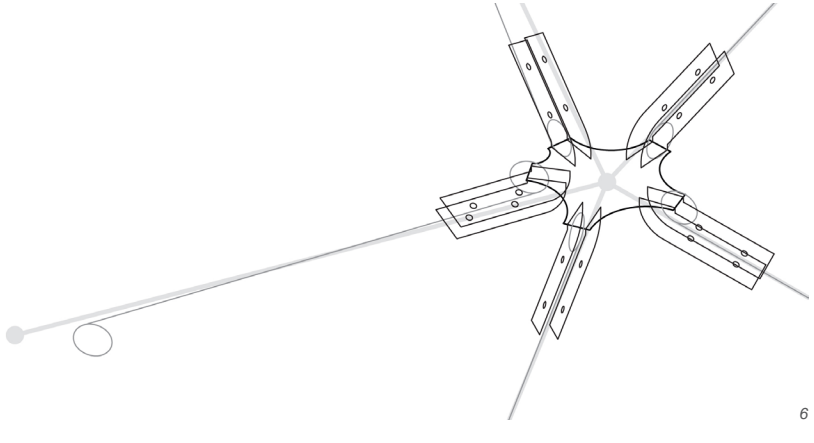
Interior tangents of two circles at the ends of each connection determines the center line of the connecting member. The radius of the circle also happens to govern the finger geometry.



5

FINGER PROFILE CURVE

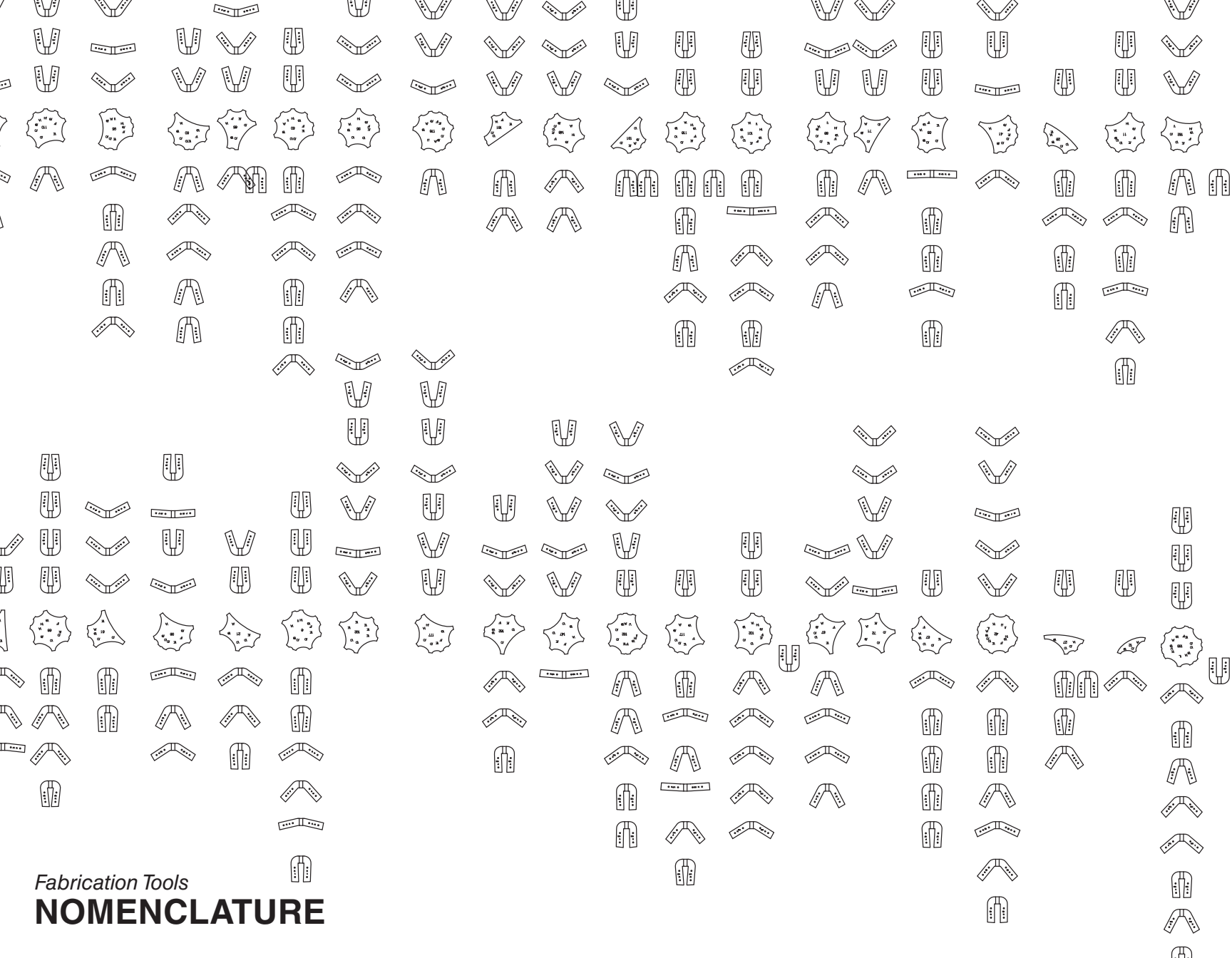
The finger profile is a simple offsetting of the centerline geometries to ensure a continuous and smooth shape that also locks in the Z angle component of the connecting geometry.



6

FINAL GEOMETRY

The full shape and hole pattern is then derived from that point, including liminal offsets, compensations for bend radii, thickness of material, etc.



Fabrication Tools

NOMENCLATURE

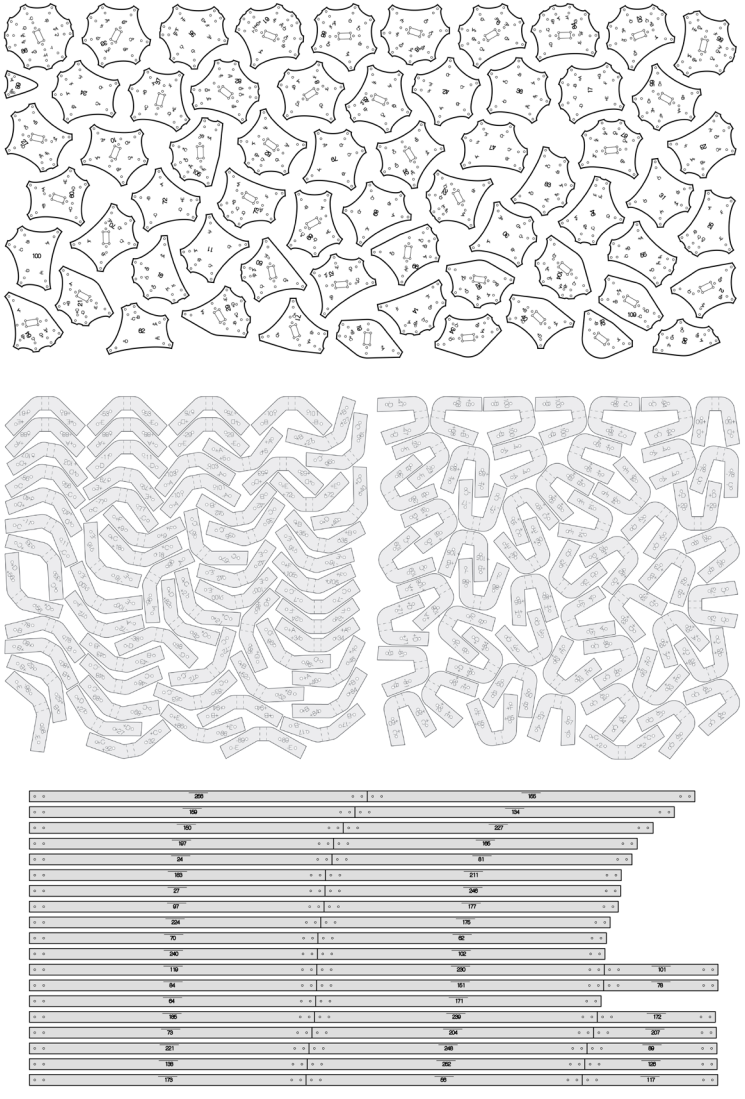
DATASTRUCTURES TO REAL STRUCTURE

The inherent organization of things via virtual data structures only benefits the user until it leaves the virtual environment. The cut material knows nothing of its origin geometry, it exists only as piece of material, with shape and dimension. Through various fabrication and labelling processes, the physical elements of the assembly became marked with their data address at all levels and of all data types. To understand the nomenclature of the assembly, one must understand the assembly.

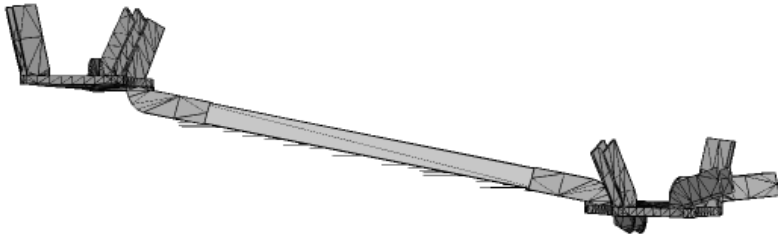
Every Point in the PointCloud that was found to be part of a valid connection became the center point of a Node Plate, cut from half-inch plywood. 20ga. cold-roll Steel Fingers were mounted offset radially from the center at unique angles at registered hole patterns in the Node Plate. The Connection Member would be cut and drilled with a hole pattern at a determined length, which would bridge between two Fingers on connected Nodes. The hole pattern and cut geometry allowed and necessitated that fingers would mount on both sides of the plate. For any given connection, one end would be held by a finger facing up, and the opposite would be the same cut geometry but mounted on the underside of the plate.

Every node was assigned a unique positive integer called the Node ID. Every finger on each node is locally assigned a character A-Z, and a sign (+/-) referring to its direction relative to the Z axis as well as its home Node ID, together forming the Finger ID. Every Connection was assigned a unique positive integer, known as Connection ID. Every Node Plate would be engraved with its number and the Letter and Sign of each finger next to the corresponding hole pattern for the fingers. Every Connection was labelled with a printed label with its Connection ID, the Node ID pair it connected, and the finger on each Node that it would connect to. In addition the printed label included a QR code elaborated upon in the ConnectionLabeler Tool Description.

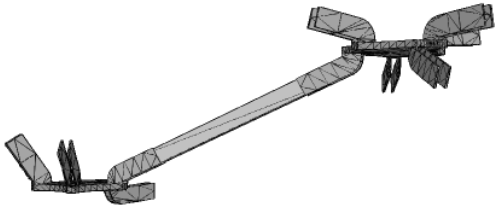
This system of creating unique identifications for all unassembled parts that contained not only its own ‘name’ but the names of the pieces it interfaces with. Thus the full ID of any given piece in the assembly could instruct the fabricator as to where and with what pieces it joins.



47.2 - 4
Only some of many packed sheets of all parts, as cut in final fabrication.



connectionID: 3
nodeA: 2 +A
nodeB: 45 -D



length: 21.7"

CONNECTION LABELING

ConnectionLabler utilized the data from each level in the system and its organization to produce a cross-referential document that contained the identifying information of itself, its connecting nodes, finger location, and 3D geometry that represented the assembly.

As described in other tools, the connection identification is the association of a pair of node ID's, the Finger ID's that connected, and the connection's unique integer identifier as well. However, because engraving this information proved to be time intensive, we opted for printing labels, and adhering them to each connection as they were machined. This provided an opportunity to jump back through the portal between the real object and the virtual.

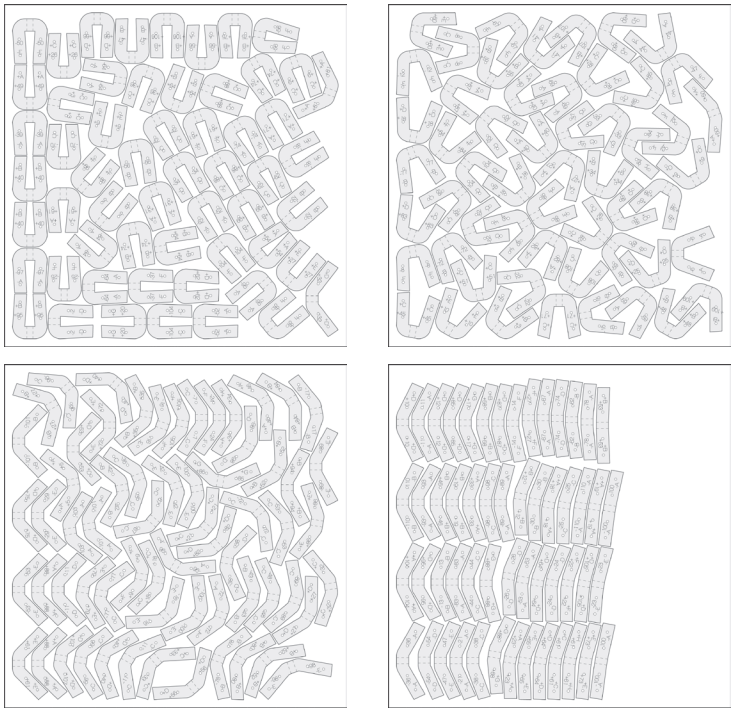
A QR code which pointed to a URL, one for each connection. At this URL a GIF animation was featured with featuring the rendered mesh representations of the connection, the Node Plates, with Fingers assembled, rotating in view, with additional identifier information, including the generated length of the connection, its ID, the Node ID's, Finger ID's.

Environment: Java/Processing
Category: Fabrication
Input: Connections
Output: Printed labels, QR Codes & GIF renders.

48.1 - 3
QR-code and the 3D gif animations that can be accessed with any network-enabled device.

Fabrication Tools

thesisLabel

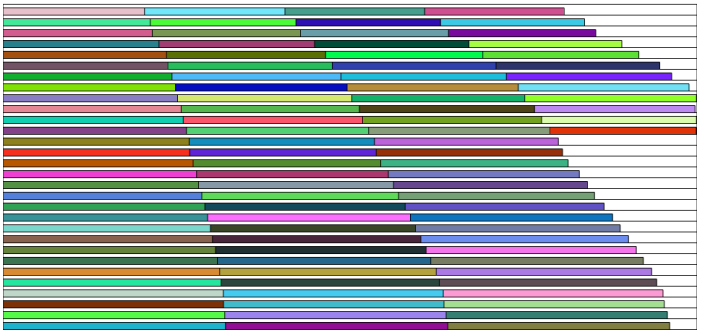


COMPONENT PACKING

BinPacker was our implementation of a well known algorithm for sorting a series of objects into set containers based on one dimension of variance. In this case the container was defined by our CNC fixture for cutting the 1x2" pine members. Since the length of each stick was known, the eight foot length became the bin size. The only variable that controlled all of the cut geometry, hole pattern, etc, was the connection length. By taking the feed of connection ID's and corresponding lengths, each member was sorted on to a stick, by working from largest to smallest, fitting any size onto remaining space on a length if possible before going to a new stick.

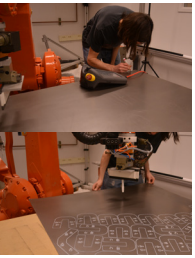
Once this was set the cut geometry and label associations using the connection ID were known and able to be produced.

Environment: Java/Processing/Grasshopper
Category: Fabrication
Input: Unsorted Connections
Output: Sorted Connections & CNC Cut Geometry



50.1 - 3
Packing algorithms galore!
Oh and the money we saved...

Fabrication Tools
thesisPacking



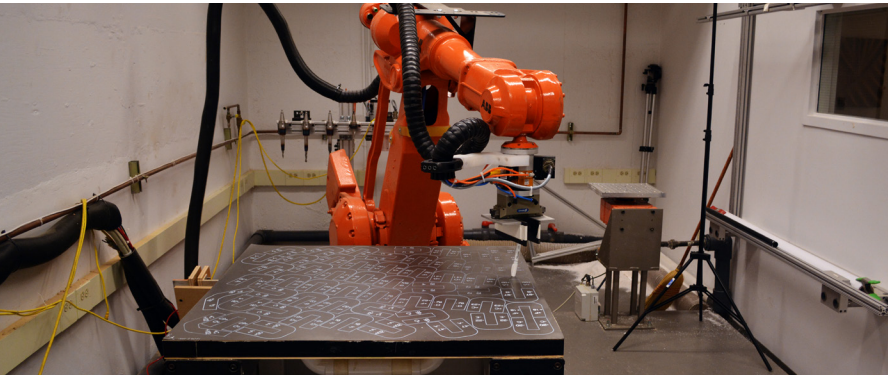
52.1 - 2
Positioning the sheet and prepping the paint-marker for another run.

SHEET MARKING

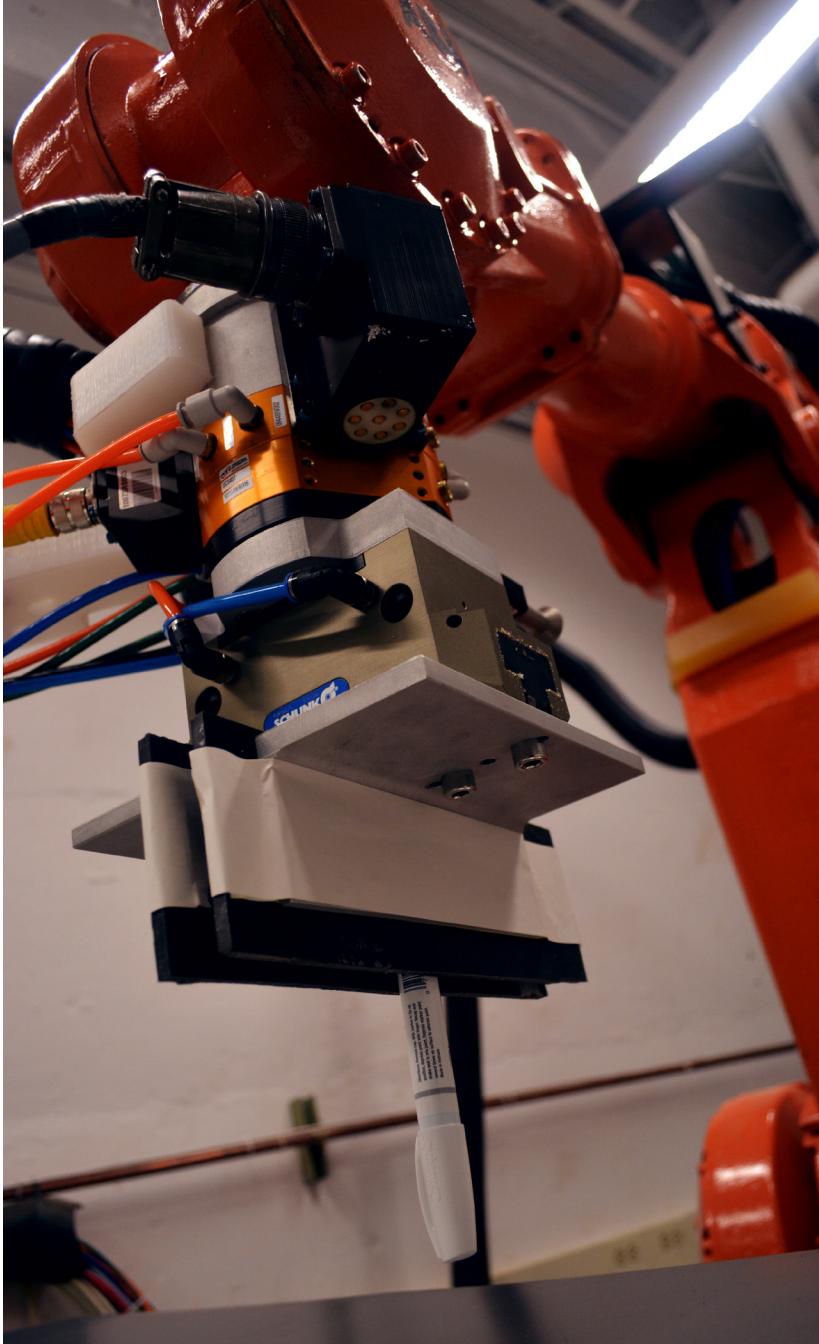
Robotic sheet marking was perhaps the most mismatched process out of the fabrication workflow. The effective operation, plasma cutting included could be done, even more efficiently with a 3-axis machine. The only freedom of using robotics afforded us was availability, and the customization of the end-effector. In this case a paint marker held by a gripper. The labeling, and accurate placement was important. As when the plasma would cut the finger geometry, the shapes varied only by the Z angle which was cut to a high tolerance. Labelling cut fingers was not an option, as it was subject to too much error.

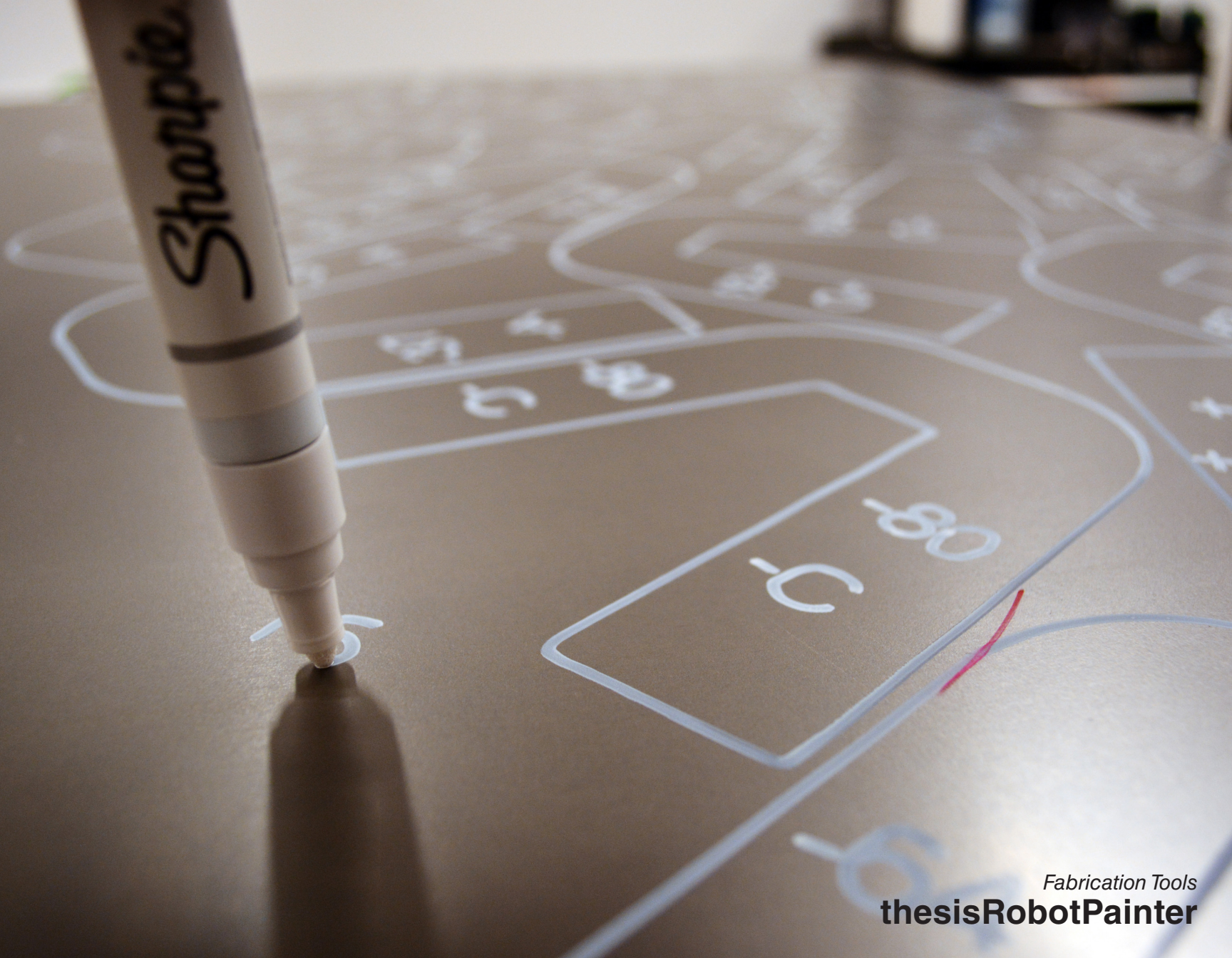
However overkill it may be for a robotic arm to be equipped with a pen for labelling purposes, the development of robotic toolpaths through HAL* served as a good way to test and debug motion issues for the later plasma cutting operations.

Environment: Grasshopper/HAL/RAPID
Category: Fabrication
Input: Cut Geometry
Output: Painted steel sheets for plasma cutting.



52.3 - 6
ABB IRB 4400 shown with paint-marker in gripper..





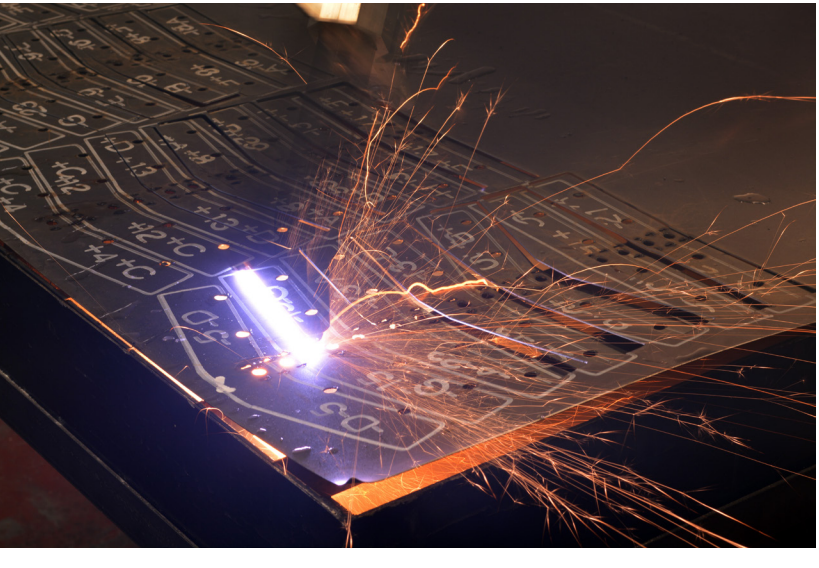
Fabrication Tools

thesisRobotPainter

PLASMA CUTTING

As with the robotic labeling, so too was plasma cutting a robotic operation due to what was available for our uses and command. Issues of toolpath and calibrating dimensional motion with triggering signals to activate and deactivate the plasma torch allowed us a level of fidelity in control CAM software typically automates. Because of this we had a number of early test cuts that exposed these issues.

Environment: Grasshopper/HAL/Rapid
Category: Fabrication
Input: Cut Geometry
Output: Robot cut files.



54.1
Node version 2 with 5 fingers.

54.2
Plasma-Cutting

54.3
Plasma-Cutting

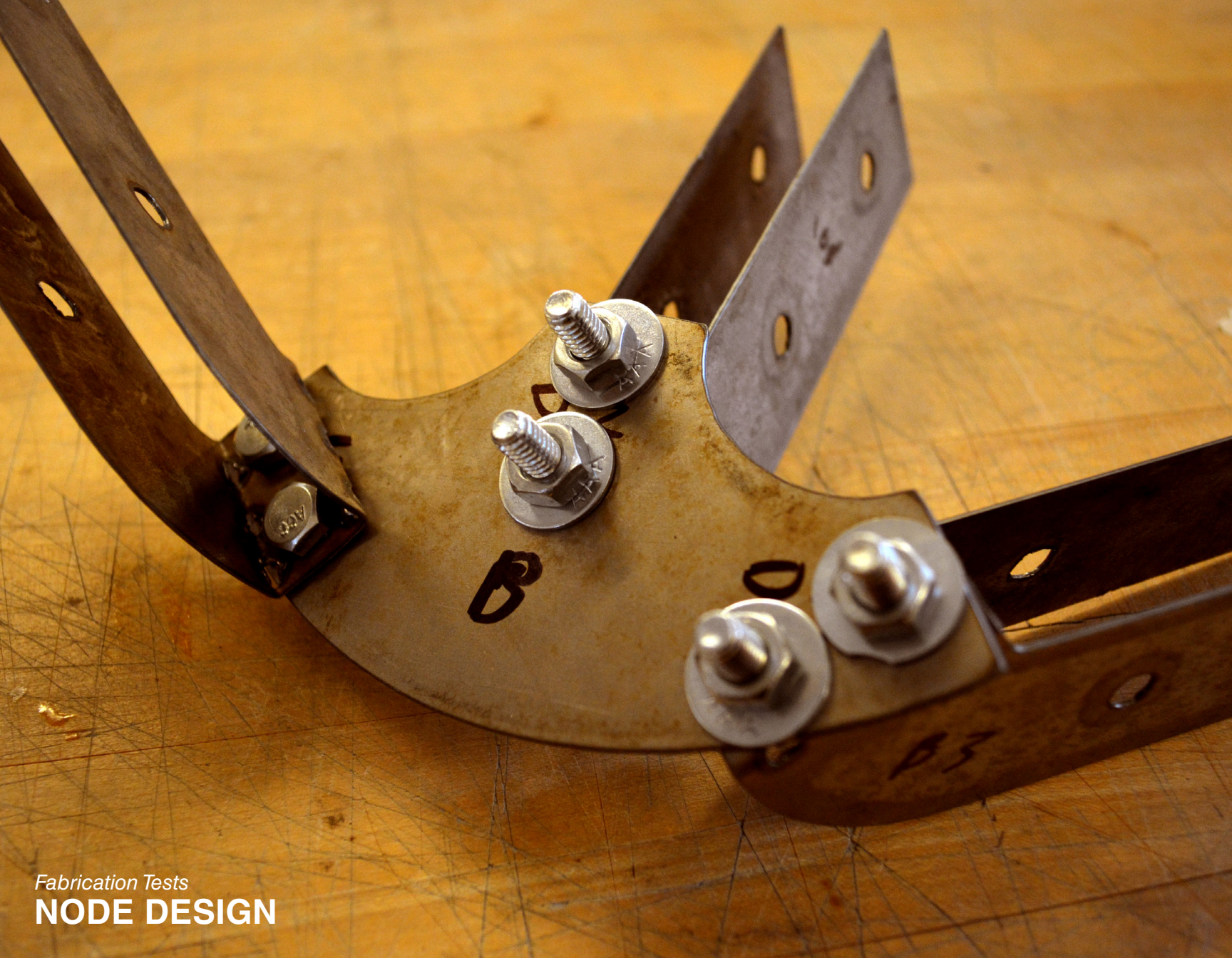
Fabrication Tools

thesisPlasma

One critique of current computational architectures is an almost joyous ignorance of methods of fabrication and formal manifestation. While rampant speculation has its place and value; methods that acknowledge tectonic relationships and real world fabrication constraints push the field of computation into a realm it is foreign to: reality. Forcing computation into a realm of reality narrows what can be done with it, shifts it from the seemingly limitless digital space into a world with gravity, and fire. Physical realities give computational systems important rules to follow and work within and against. This results in a system, while now bound by a new ruleset, is given the opportunity to push and solve much harder issues, issues of craft, intuition, safety; issues of atoms.

Our experimentations in digital and analog fabrication provided key data sets and logics for the system to act on. As designs were built, tested and analyzed this information informed us and the system simultaneously.


FABRICATION
TESTS



Fabrication Tests

NODE DESIGN

NODE DESIGN

Given our form generation was a nodal diagram, it followed logically that our structural solution should approximate this diagram. Our node design evolved alongside the structure tools and more importantly lead directly to the Joinery script itself. Node design would determine maximum spans, angular proximity on a flat node plate. This exploration was some of the more exhaustive work outside of the code itself. We understood this process would be a process of largely ruling out impossibilities, and accepting unforgiving realities.

Material, geometry, and associated fabrication methods all took center stage.

As a metric, we had our budget. Adaptable, complex 3-dimensional node joinery, perhaps outsourced fabrication immediately broke our budget. Spherical stock, regardless of material in or outsourced fabrication was inherently expensive, despite the interest and viability of robotically mapping and drilling holes for member registration and fixturing.
Welding was labor intensive, and sheet stock has insufficient section modulus to deal with bend resistance across a node. Any stock of metal that was too thick would add too much dead load, and increase member size, reducing number of possible connections (the strength of the system-scale structural viability). This is what pushed us into plywood. Ironically, a default we actively tried to move away from despite our familiarity.

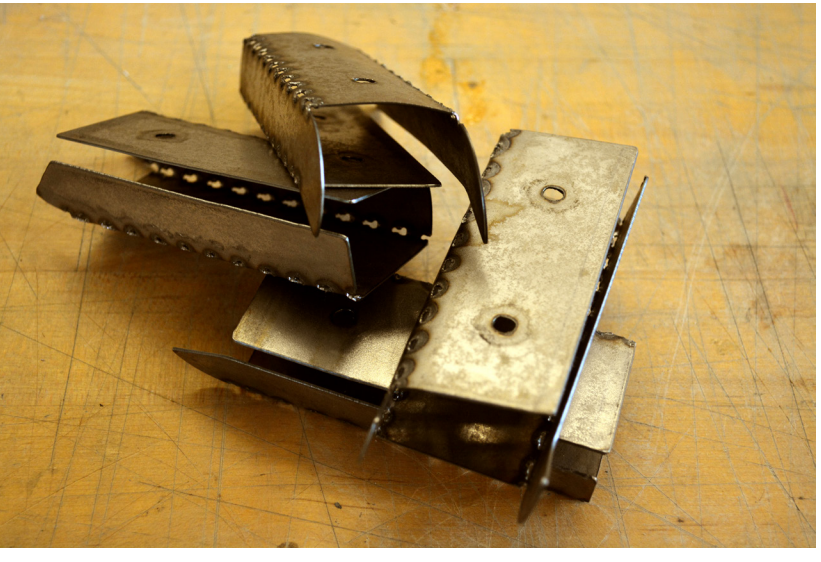
We knew early on, the further a member's centerline deviated from the node point, the greater internal forces would be imparted to the node itself. However, the closer you aim the members towards a common point, the more intersection (collision) there would be between members, therefore culling otherwise viable connections between nodes. This metric employed as a variable in the structure script helped us quickly determine optimal node size as a local structural measure and its effect on system-scale connectivity.

59.1 - 3
Various stages in the design of the node.

VERSION 2

This was the first real attempt at a working joint. A central plate with branching arms would be plasma cut with a pair of holes on each arm to accept connections. The central plate oriented to a common world plane began to isolate fabrication methods, fixed variables in geometric solving of a spatially complex problem. This datum would prove invaluable in later work.

The connection-adapter, wrapping the sheet steel around the edges at this scale proved to be problematic. The radius, and bend induced by breaking, would cause a misalignment of the two calibrated fingers that would theoretically register the elevational angle off of the plate. It was clear there were too many issues that would compromise tolerance.



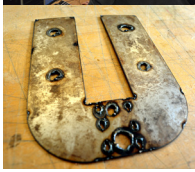
60.1
Node version 2 with 5 fingers.

60.2
Version 2 fingers, demonstrates error from bending.

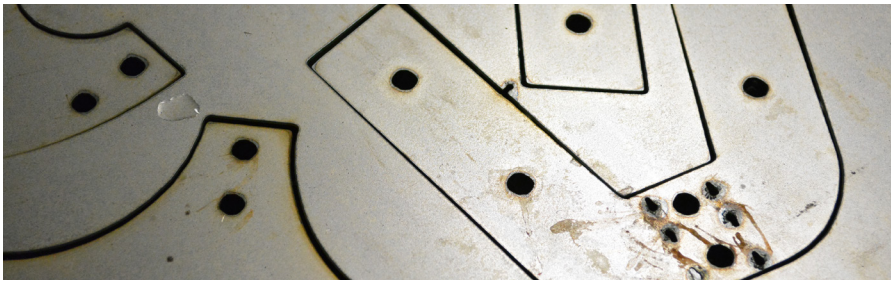
60.3
Version 2 finger, demonstrates thiness of alignment edge.

VERSION 3

Version 3 isolated the problem with adapting from a rectangular stock of wood to a fixed plane in space from all elevation angles, using the strong axis of the sheet steel. In immediate contradiction to this, the node plate was still a heavier gauge steel with the hopes fixturing with welds would make the joint more rigid. The bolting to the plate was intended to be temporary, and an on-site fixturing solution when they would welded. With the right washers, cleaner hole cuts, we realized bolting could be a workable fastening. This then ruled out the need for welding, which meant we were free from keeping everything as steel. At the time Jordan was taking a machining and welding class, which quickly excited us to the possibilities of realistically working with metal. Something that was rare to have direct control over from design through actual fabrication. Between cost, weight, fabrication, assembly, deflection and bending, we isolated the problem of the node plate and addressed it more precisely.



61.4 - 5
Initial plasmacutting tests were less than ideal.



61.3 - 6
Even more dscriptions
words are fun!
explain it lots

VERSION 3.5

It was at this point we then came back to our common material, one that in previous work of our own we have explored in various ways. We established that it was primarily an issue of a sectional modulus. The material needed to have a reasonable thickness, deal with tensile forces, resist bending, be able to be machined, drilled, and marked, and not be terribly expensive for about 40 square feet of it. Plywood fit the bill. When integrated and tested it performed remarkably well with a node-diameter at 8.34". By this time we realized the node design need not be an incredible moment connection, and the system-scale connectivity and structural redundancy would help disperse load over more members than we had been testing. This permitted us to test that assumption with our full-scale prototype.

62.1 - 4
Nodes of the prototype shown in contrast with the first.

Fabrication Tests

PLASMA CUTTING

65.1 - 3
Various stages in the design of the node.

ROBOTIC PLASMA CUTTING

Nothing about plasma cutting sheet metal requires a robotic arm, it happens where the plasma cutter is currently integrated. Having established that all fabrication would be in-house, we could control the entire process, but would also inherit the intrinsic issues with all of these processes. Naturally robotic-toolpathing solutions became quite laborious. HAL* proved to be a great way to produce RAPID code directly in our modeling environment which allowed us to quickly establish ideal workzones, issues of reach, pose and joint errors, and quickly calibrate and adjust minute geometries that would radically affect the end of the plasma cut quality.

Plasma cutting has some but a limited set of variables that affect cut quality. Unfortunately robotic motion and toolpathing can further complicate this relatively simple problem. HAL provides for an easy way to produce highly unique toolpath geometry, however with respect to tooling, it is our responsibility to integrate this functionality, effectively producing a small subset of features commonly found in most CAM softwares. Being rigorous about when the torch pierces and for how long, how to maintain an ideal linear velocity and dealing with the acceleration of the arm, balancing precision against all of this.

65.4
ABB IRB 1600 with attached plasmacutter.

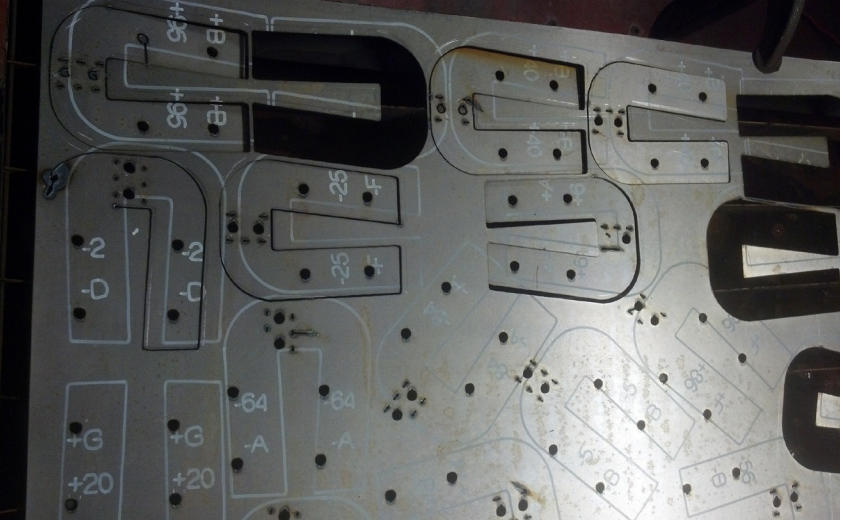
QUALITY OF THE CUT

We quickly discovered a number of issues alluded to with robotic motion. Maintaining linear velocity, even around curves or hitting corners is paramount to ensuring a consistent and minimal kerf. If the precision setting of the robot commands was too tight, the motion supervision would decelerate to hit these points in space. This is where we see the jagged edge with a lot of burn and slag on the cut. For smaller geometries, like the bolt holes, the kerf alone could radically compromise the tolerances of the hole geometry.

66.1
Dash-cuts become oblong holes.

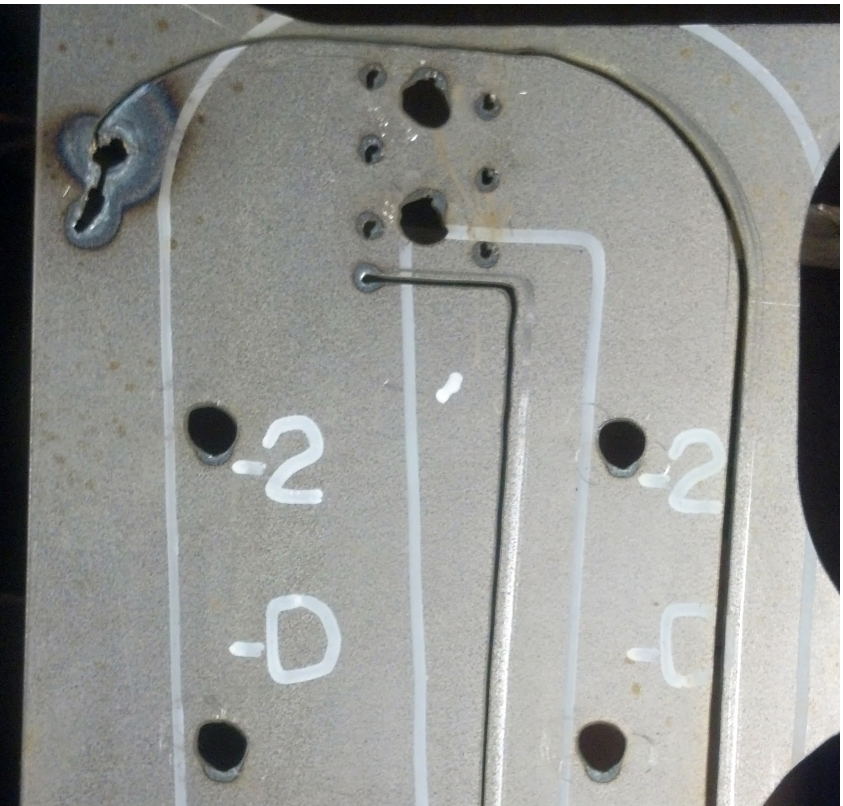
66.2
Kerf compromises hole geometries.

66.3
Slag accumulation shown in detail as a result of the points of deceleration for the robotic motion.



SHEET ISSUES

Packing and nesting was important for efficiency of material. However, using a sorting algorithm to minimize distance of transfer between cuts, reduce time, and errors in collision and torch misfires. Also, moving from one robotic workzone to another require precisely calibrating where there sheet was in the new space, and its orientation. Even with everything figured out, sometimes the sheet bows with heat, the torch loses conductivity, or errors out. With every piece being a necessity to the final assembly, it was imperative that any and all failed cuts were later re-packed and cut again.



67.1
Partial sheet tool-path halted due to error.

67.2
Motion error halts robotic motion, but tooling remains active.

67.3
Plasma deactivations due to loss of conductivity

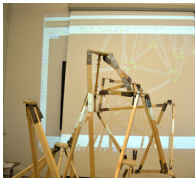
Fabrication Tests

PROTOTYPE

PROTOTYPE

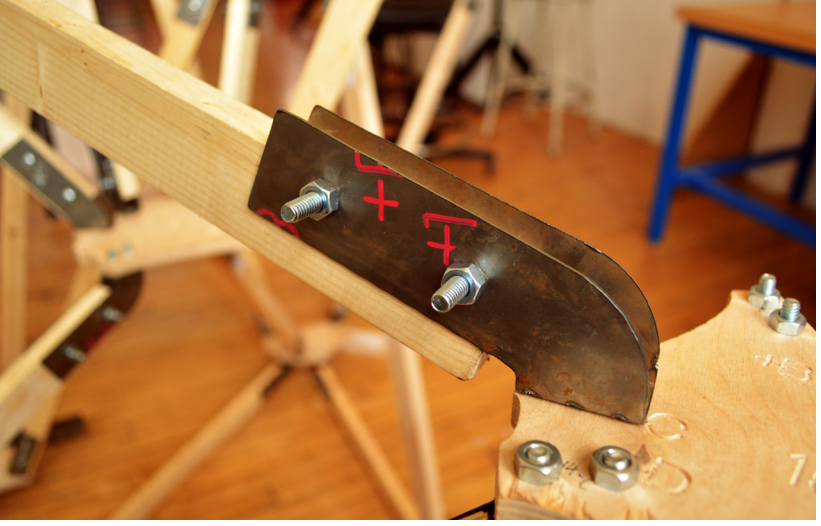
The importance of the prototype exceeded all other concerns at this stage of the delivery. This was the first full-scale, multiple interconnected assembly that was complete and fully iterated and produced through the toolkit system. This tested all things: data order, continuity, structure, nomenclature, geometry representation versus real material tolerance, fabrication file automation and execution, as well as the process of assembly. There were errors on all fronts, but even in full acknowledgement of this the prototype solved one concern for us: indeterminate structural performance. The prototype was surprisingly rigid and robust, as connectivity was high. Assembly, once errors in the nomenclature were deduced, was not too difficult, even with an ambiguity of where one starts and moves to another piece. It was not important that the prototype was going to be anything of value outside of a demonstration to ourselves to physically debug issues in the system.

If anything should be gained from this chapter it is: the decision to run a full mock-build of a representative prototype of the desired assembly single-handedly illuminated several low-level errors that would have compromised the final build.



69.1 - 2
Description please

69.3 - 6
Even more dscriptions
words are fun!
explain it lots



PROTOTYPE ISSUES

The errors were numerous, some were data-structural, fabrication tolerances, material selection and prep, internal domains and tolerances of the generated data. In the end, they all had solutions that did not compromise anything we were not already willing to work with.

One issue was the poor quality of the pine members, which as 1x2”s were extremely bowed and twisted. This caused the fixturing on the mill to improperly align holes for the fingers. Even worse, as the prototype was assembled, we were fighting the bows into where a desired straight piece would be. This put the entire assembly under a great deal of tension. We were able to source higher quality members which had minimal bow and dimensional tolerance.

70.1
Node version 2 with 5 fingers.

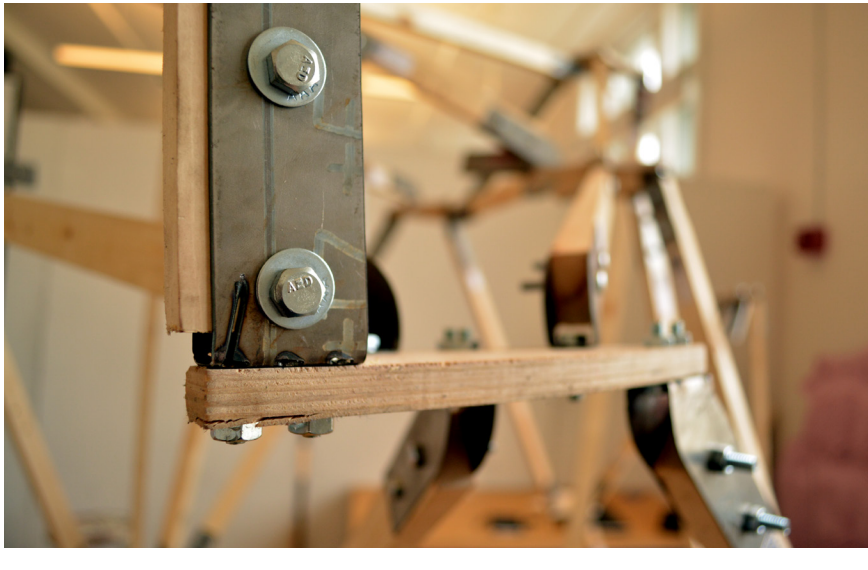
70.2
Version 2 fingers, demonstrates error from bending.

70.3
Version 2 finger, demonstrates thiness of alignment edge.

Another issue was issues with the toolpathing for the plasma cutter which produced some compromised cuts, not properly solving over a curve, or cutting on transfer motions.

Proximity of nodes was determined as a global minimum radius across all tools that would serve as a hard collision boundary of the points in the PointCloud. Unfortunately the minimum did not take into account the combined distance from the center of the node plate, offset of the radius, and the length of the steel finger.

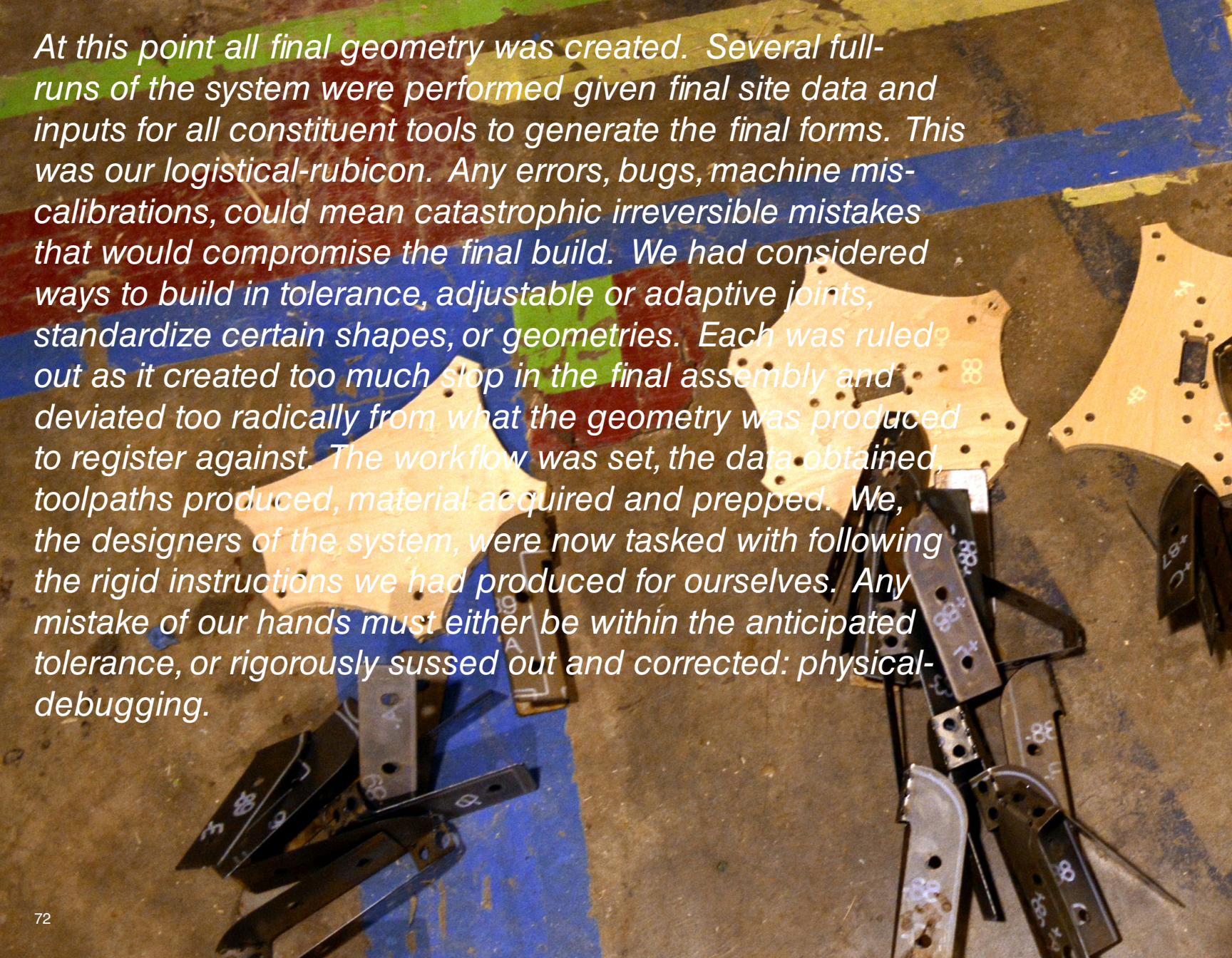
The most crucial issue, that would only appear through this complete execution is a compromising shift in the organization of the data structure between tools and environments, namely how the Joinery Script interpreted the output of the structural solver. This affected the nomenclature and labeling of pieces. Some were simply offsets of i+1 which indicated counting from 1 instead of 0. However, the rotational organization of the fingers and their labeling carried from the node, to the finger, and then to the member was compromised, as it turns out by a simple and similar shifting of data from null results internal to the script. This oversight made these indecipherable, sometimes they were correct, sometimes not and not in any conceivable pattern. A nasty bug that alone could have taken the final build and made it an impossible puzzle of uniquely mislabeled pieces.



71.1
Node version 2 with 5 fingers.

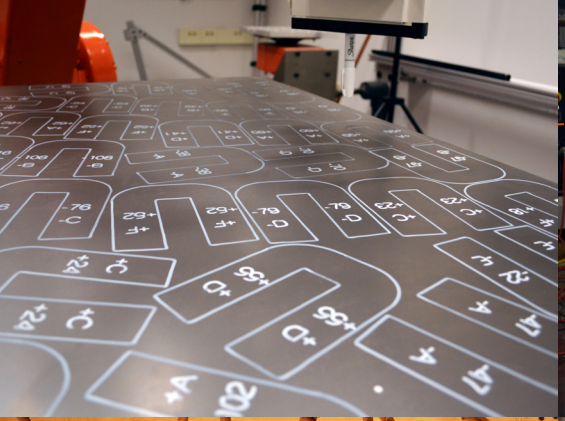
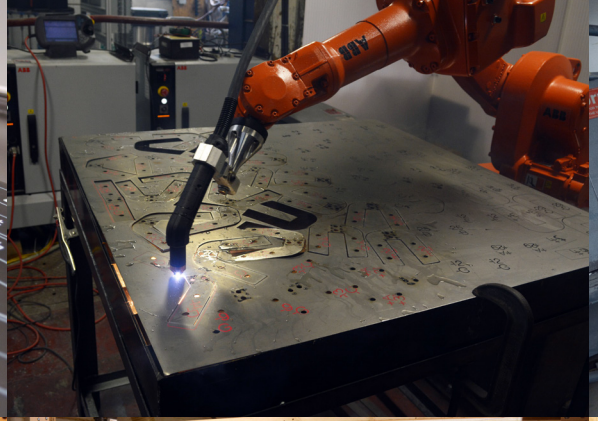
71.2
Version 2 fingers, demonstrates error from bending.

71.3
Version 2 finger, demonstrates thiness of alignment edge.



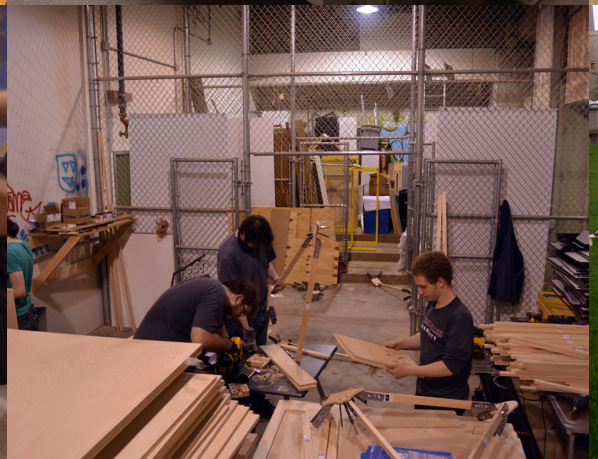
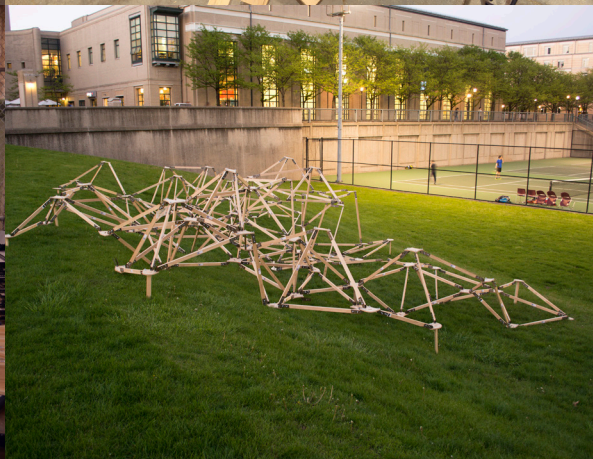
At this point all final geometry was created. Several full-runs of the system were performed given final site data and inputs for all constituent tools to generate the final forms. This was our logistical-rubicon. Any errors, bugs, machine mis-calibrations, could mean catastrophic irreversible mistakes that would compromise the final build. We had considered ways to build in tolerance, adjustable or adaptive joints, standardize certain shapes, or geometries. Each was ruled out as it created too much slop in the final assembly and deviated too radically from what the geometry was produced to register against. The workflow was set, the data obtained, toolpaths produced, material acquired and prepped. We, the designers of the system, were now tasked with following the rigid instructions we had produced for ourselves. Any mistake of our hands must either be within the anticipated tolerance, or rigorously sussed out and corrected: physical-debugging.


FINAL
FABRICATION





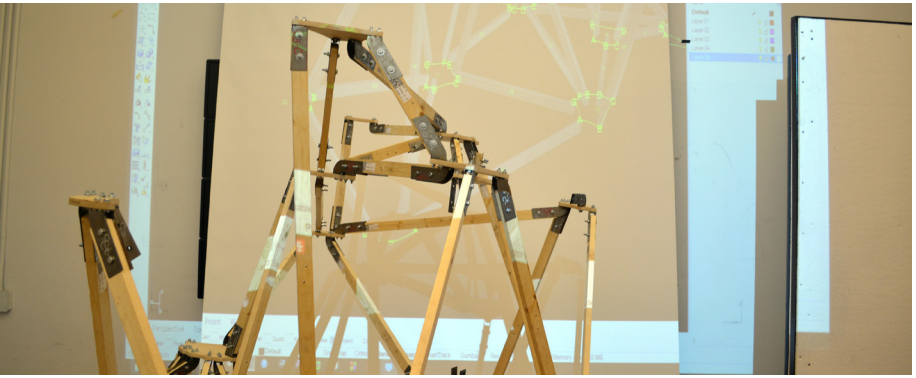
Fabrication

OVERVIEW

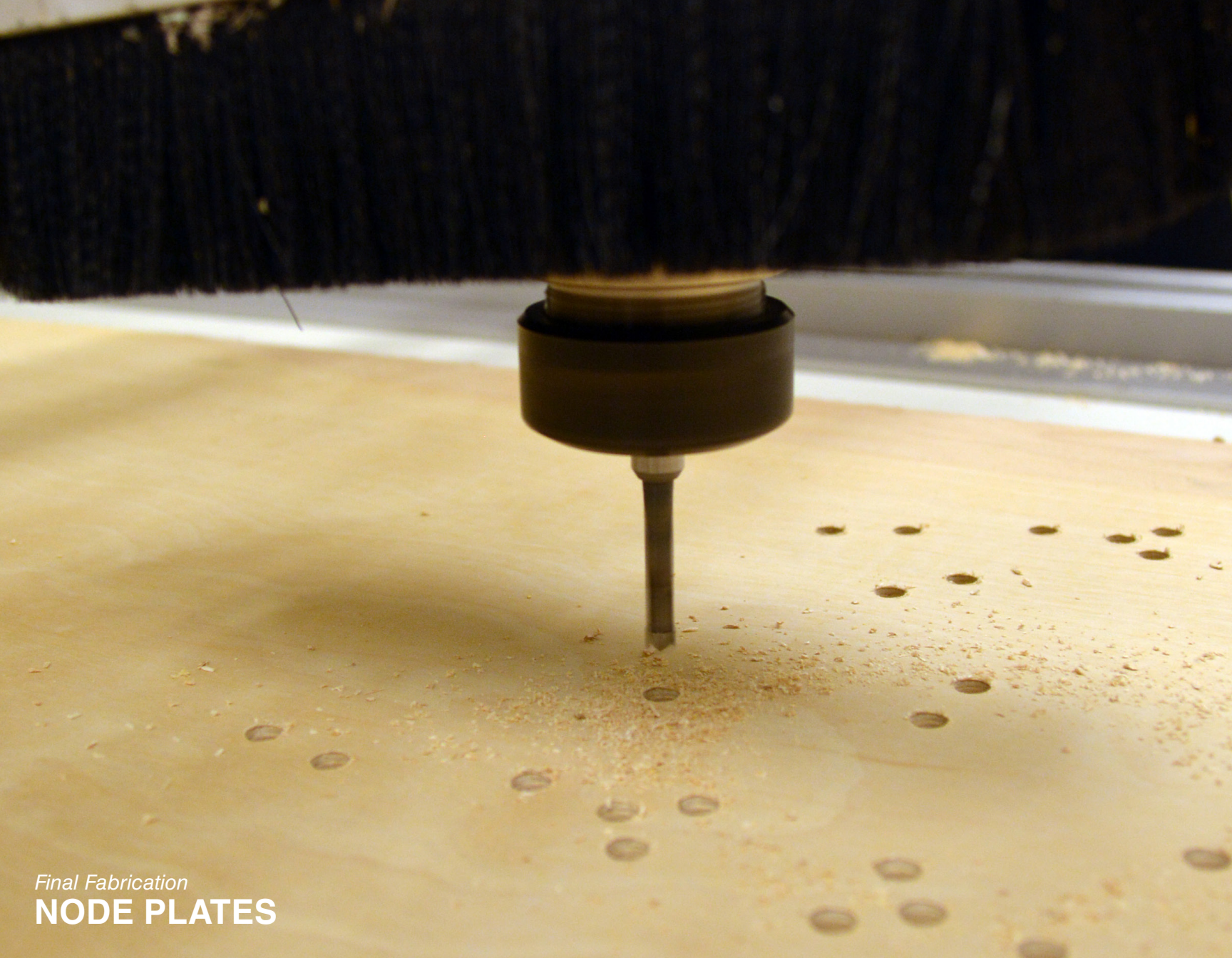
FABRICATION RUNDOWN

The entire fabrication of the built instantiation after being generated through our toolkit system, given all final site parameters and design criterion, took 10 days. Only two of which were on-site assembly. Finished under budget and on time. This process was less about testing and learning, as the ultimate goal was to demonstrate the power of the rigor and work that was performed on the front-end to allow a built form to be delivered with precision. It was an ambitious postulate, that if we designed the system well enough and embedded as much material and fabrication machine understanding, that it would perform and deliver as designed.

The final fabrication chapter is broken down into all of the discrete tasks that populated our critical path to delivery, and will walk you through all of the various things that were anticipated or emerged throughout this intense week and a half.

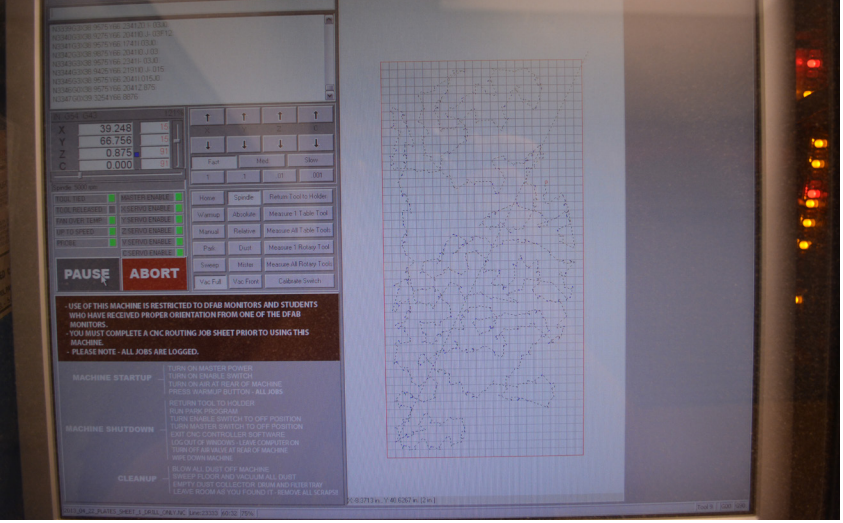


75.3 - 6
Even more dscriptions
words are fun!
explain it lots



Final Fabrication

NODE PLATES



CNC MILLING SIMPLE SHAPES

It was clear that the Node was the primary crux of the organization of our data and of the constructed assembly. The Node fabrication was also the most straightforward of all of the various machines and processes we employed. As student employees of the DFab, we were intimately familiar with the 3-axis CNC router, and how best to layout operations to mill out simple shapes with precise registration holes, and labeling. Each shape was defined by a profile, whose geometry was constructed based on the 2-dimensional layout of the incoming connections to the node. Each finger location had two holes drilled to interface to the metal fingers. Some Nodes were also considered as possible footings, and had a small rectangle milled out to allow for the node to interface with our footing solution(which later turned out to be no more than a stake).

77.1
Node version 2 with 5 fingers.

77.2
Version 2 fingers, demonstrates error from bending.

77.3
Version 2 finger, demonstrates thiness of alignment edge.

Final Fabrication

CONNECTIONS

COMPLEX FIXTURE DESIGN

Connection milling would take a new approach to how the CNC router was typically used. The bed of the CNC was designed for larger pieces of sheet material, providing a large surface area to hold down to the vacuum table. It was clear that our connection lengths and registering holes must be precisely cut, and would need a numerically controlled machine. It was then our task to create a fixture that would lay flat, and hold standard 1x2" pine members securely to deal with the vibration induced by milling. The fixture was a careful design of how to position hardware to provide the necessary clamping pressure to hold the sticks, but safely out of the way of any cutting operation, regardless of stick length or position.

79.1
Node version 2 with 5 fingers.

79.2
Version 2 fingers, demonstrates error from bending.

79.3
Version 2 finger, demonstrates thiness of alignment edge.

CONNECTION MILLING ISSUES

An unfortunate fact of this process combined with our design required some sacrifices to expedite our time on the high-demand CNC, to utilize its precision to register geometry versus cutting down on our manual labor. The bed and the stock member lengths were 8'. Balancing member sizes versus spans, our connection length domains were set to between 2'-4'. Using BinPacker to most efficiently lay out multiple members per stick, this meant that a single member would have between 2-4 connections that would need to be cut. Because of the variability of stick length, there was no way to design the fixture to secure any theoretical stick length along any part of the stick and hold it securely as it was separated from the 8' length. As a compromise, a score line was used instead to mark where to later cut with a chop saw. This reduced cut-time on the CNC, simplified fixture design, and still provided the precision we required.

Labelling is always imperative with many unique but similar parts. However etching the necessary information with an engraving bit onto each stick to properly locate it within the final assembly would account for 90% of the total cut-time of the operation. As a compromise and a unique opportunity, the QR labeler was born and utilized. While the connections were still in the fixture we labelled them by hand according to how they were laid out in the CAM software after organized via BinPacker. Later the stickers were applied.

80.1
Node version 2 with 5 fingers.

80.2
Version 2 fingers, demonstrates error from bending.

80.3
Version 2 finger, demonstrates thiness of alignment edge.

CONNECTION CUTTING

Jordan then proceeds to cut 262 unique connections.

81.1
Node version 2 with 5 fingers.

81.2
Version 2 fingers, demonstrates error from bending.

81.3
Version 2 finger, demonstrates thiness of alignment edge.

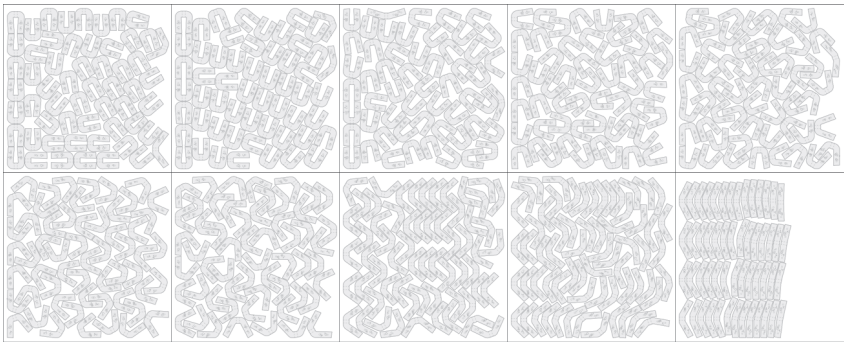
Final Fabrication

FINGER FABRICATION

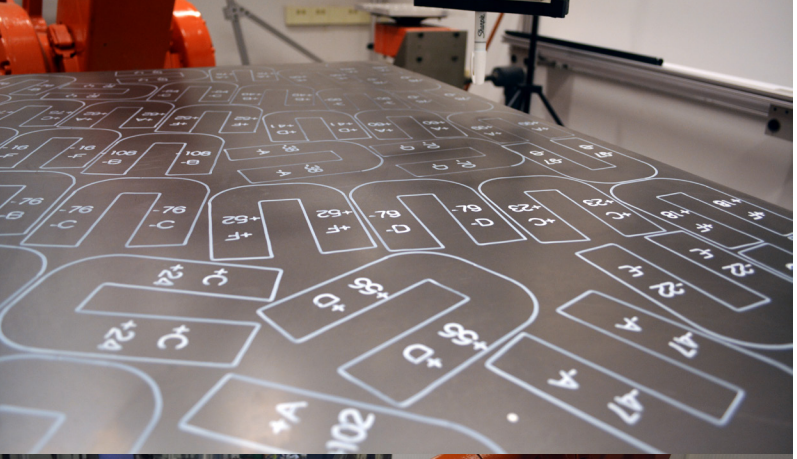
MAKING FINGERS

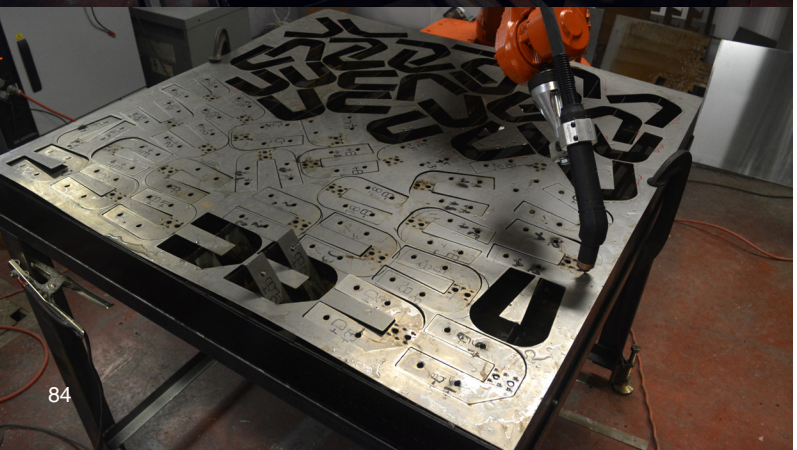
10 whole sheets of 42”x42”’ 20-gauge cold roll steel had to be laid out, packed, geometry prepped for robotic labelling, and plasma cutting for 524 unique fingers. The fingers undergo a number of steps from their initial geometry generation all the way through to their final break forming before they are ever ready for assembly. The stock size was determined by what size the plasma water-table could handle, even though the reach of the robot was limited to about half of the table. This required that for each sheet there would be two operations, between which the sheet would be rotated and re-fixed.

However labor and time intensive these processes were, we could cut no corners. We determined where the elevation angle tolerance would be held, and this made the finger geometry crucial to the success of the assembly.



83.3 - 6
Even more dscriptions
words are fun!
explain it lots





READYING THE METAL

It was important to maintain strict organization and registration for each sheet so that the marking process would be congruent to the cutting operation. Generating and sorting geometry was already taken care of by the fabrication preparation functionality of the toolkit. The key was calibration of the physical operations of two different robots in two different spaces across campus. Fortunately both had a perfectly square 42" beds on which to operate. Great care was taken to achieve positional and rotational accuracy. Never perfect but within tolerance.

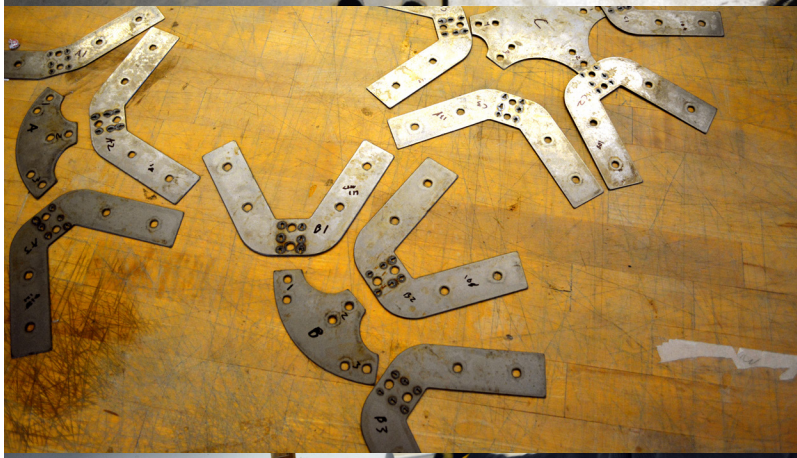
84.3 - 6
Even more dscriptions
oh ya
explain it

FINISHING THE PARTS

Each finger had plasma-cut dash marks to indicate, and bias a break. All 524 fingers had two 90 degree angles to take its horse-shoe like shape and fold them up to allow them to be parallel and match the registration holes along the connection member. The break would not allow to make such tight angles in the proximity they were positioned. This allowed for the later assembly and bolting the member to the fingers to pull them flat. This introduced some more rounding we were trying to avoid as it introduces variability at a crucial connection, but not to a critical degree.

Mike then proceeds to breakform 524 fingers.

85.3 - 6
Even more dscriptions
words are fun!
explain it lots



Final Fabrication

FINAL ASSEMBLY

BUILDING THE INSTANTIATION

After having all pieces cut, and officially being off all machines was a liberating but ominous bottleneck of our final construction process. The machines were in such high-demand at the time, we were relieved to not have to mill at odd times at night and early mornings. However, this was the only time we would encounter any errors in our fabrication. Our fabrication process was so fast tracked, we could not reliably mill out and cut all the necessary pieces to test a smaller chunk of our final construct. To find all the right geometry out of the whole sorted system and prep files just for those would introduce too much wasted time and material. It was a gamble, and we did have one snag that was emblematic of the consequences of this decision.

87.1 - 2
Description please

87.3 - 6
Even more dscriptions
words are fun!
explain it lots

Final Fabrication

NODE ASSEMBLY

111 UNIQUE NODES

As Nodes were a crucial organizational tool, they also became the focus of our first assembly efforts. 111 Nodes would take between 2-12 different fingers that would need to be attached to the marked finger location, and attached to the correct side of the plate, above or below. Of course human error occured in this process, but using simple hex bolts with nuts, reversing errors was possible and quick, compared to alternative methods of attachment.

89.1 - 2
Description please

89.3 - 6
Even more dscriptions
words are fun!
explain it lots

Final Fabrication

CONNECTION FIX

A SIMPLE ERROR

One night after finishing all the nodes, Mike began putting together the first ‘Triangle’ (discrete assembly of 3 mutually connected nodes). Upon putting the first connection into the first finger, he noticed a problem. It seemed only one hole could line up. He tried others, the same problem, no matter what connection and what finger. Jordan joined in the investigation and physical debugging. An error was found. Connection and Finger holes were to be positioned at 1” and 3” on center from each end of the member, and so to on the steel Fingers. The Finger holes measured correctly, the connection holes however were too close to the end and too close together. This only emerged as a result of a change to BinPacker subsequent to the prototype.

The fix was simple but had to happen at break-neck speed. A fixture was milled with the proper hole registration relative to the end of all connections (since that was standardized, and the lengths were still reliable). This fixture was slid on the the end of a connection, and then was hand-drilled through. It was up to only our technique to ensure that the drills were perpendicular to the connection or we would risk biasing serious torque onto the fingers and connected members. A drill press would solve this issue, but would be too slow for the number of members that had to be re-drilled.

After locating the error, the resolution and amendment was fully implemented within 8 hours time. The price we paid for the gamble we took.

91.1 - 2
Description please

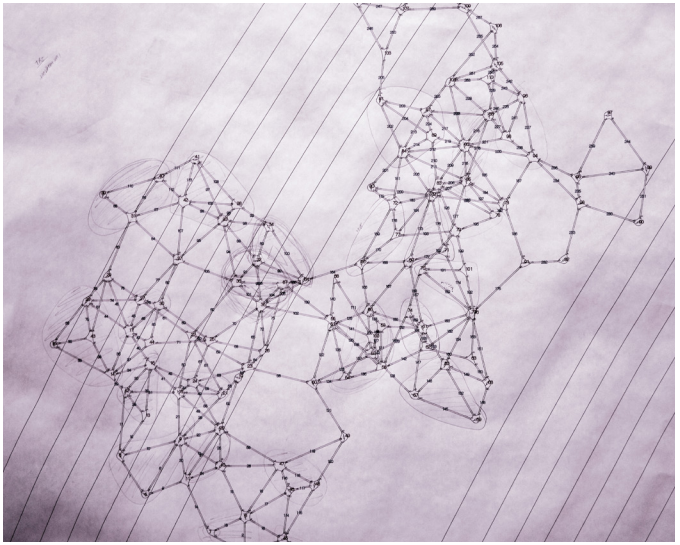
91.3 - 6
Even more dscriptions
words are fun!
explain it lots

Final Fabrication

TRIANGLE ASSEMBLY

PREFAB

After all nodes were ready, and connections fixed, we then began the process of determining how best to minimize the macro-complexity of the entire system versus the tiny components that we had ready to assemble. We needed a device to mediate these scales and not require us to be on-site to execute it. The 'Triangle' became our solution to consolidate many pre-assembled pieces into a discrete constructed chunk, that could have multiple nodes and bridging connections to other triangles and nodes and act as localized anchors from which to spawn further assembly. This eliminated the task where we started, and reduced the time-intensive task of looking through a large batch of like-pieces for only one particular piece. This way many simultaneous self-contained mini-assemblies could take place without delay. Triangles were picked out of the single plan drawing that was ever plotted for this construct. Each triangle was circled, and marked when assembled. Volunteers joined and left as they pleased, able to quickly and efficiently contribute to the effort with little downtime nor confusion.



93.3 - 6
Even more dscriptions
words are fun!
explain it lots

Final Fabrication

ON SITE ASSEMBLY

FINAL CONSTRUCTION

On-site assembly occurred within two days. We picked one end to start from, tried our best to calibrate relative to the slopes and terracing of the site, where this node was placed, and continued out from there. Many volunteers joined, and were able to do so with ease due to the clear construction logic, organization, and nomenclature of the pieces. It was a matter of matching a connection between the two nodes it was marked to bridge, attaching to the fingers it was due to join at. The geometry and physical assembly made it little question of where something should go. If it was incorrect, it could easily be determined, and confirmed with the mismatch of the ID's of the erroneously placed components. We spent more time teaching volunteers how to use a socket wrench than we did correcting their mistakes. Of the few that occurred were corrected by other volunteers.

The second day, the resemblance of the total assembly to the digital representation was clear. There was also one minute adjustment to be made before it was completed. It seemed that the start point was correct, but further down it seemed out of sync with the slope and curvature of the site. Many colleagues were gathered for a concentrated effort to lift and pivot the assembly a little more to the west. This was in a way the largest example of the difficulty of calibrating the physical world with the digital representation of it. The final instantiation occupies two terraces and the slopes between, allowing for a specific registration and would force our hand to ensure it met these slopes within the tolerance needed. With a simple rotation, the assembly was placed on the ground again. It was clear from the way it was flexing and what nodes were gouging the earth previously, that we had found its correct orientation.

An observation was made after this orientation fix. All of the labelling on the node plates were populated on the node representations while in their intended positions in the 3D environment. The site model data, and all 3D environments in our code were Y-positive North, the labelling text was also generated all oriented to Y-positive. Upon looking at each plate, they were all oriented the same direction: North, real, magnetic-pole-of-the-actual-earth North. Of all the stress of what might fail, what tolerances would break, what dangerous macro-level mysteries awaited us as we put this monster together in plainview of the entire campus, nothing was more satisfying than seeing something so ridiculously precise actuated in an immense physical thing.

95.3 - 6
Even more dscriptions
words are fun!
explain it lots

Final Fabrication

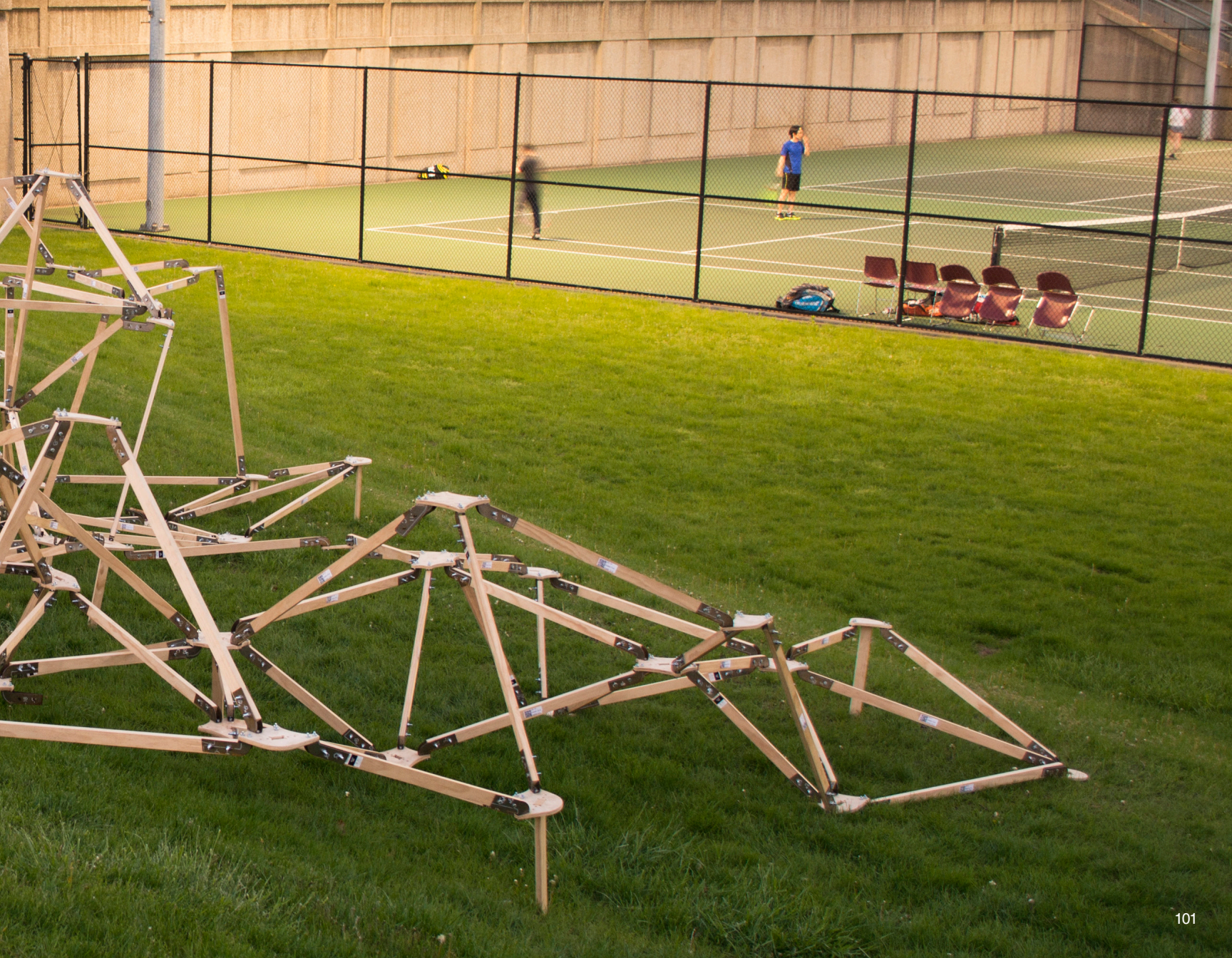
DAY 1 PROGRESS

Final Fabrication

BUILT INSTANTIATION

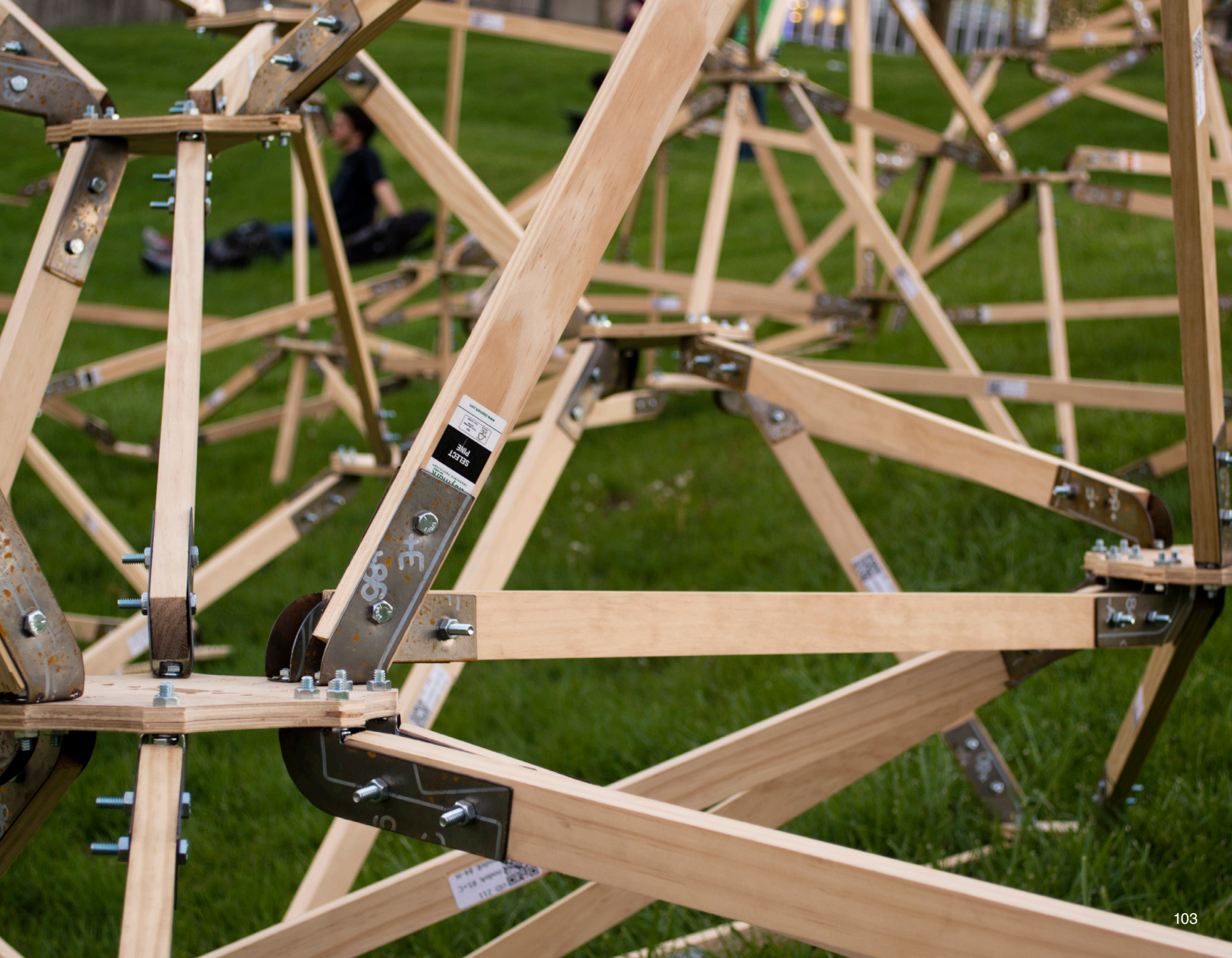
Final Fabrication

BUILT INSTANTIATION



Final Fabrication

BUILT INSTANTIATION


SELECT
PINE

Final Fabrication

BUILT INSTANTIATION

THOUGHTS OVERVIEW

Instantiation as Architecture:
The built instantiation is experienced independently of the logics of its generation. This disconnect leaves the instance in an ambiguous state between architecture and object. Altering qualities and factors in its siting, scale and tectonic execution could reposition it in a more architectural state.

A Priori vs Systems:
Systematic computation preserves the connection between intent and execution. Erroneous results force refinement of the system, and not the resultant form. Whereas traditional design practice amalgamates conflicting logics into a single object, obscuring the connection of intent to execution.

Craft and Authorship in Computation:
Craft in computation is based on knowledge of low level functionality in involved processes. This knowledge and the computational logic employed in the execution of the system transfers the authorship of the designer to any resultant instantiations if the design intent and logic are congruous.

Construction Process:
Systematic computation engaged with digital fabrication allows for the explicit linking of digital and physical information. Construction logics and nomenclatures emerge based on this relationship, controlling the degree of on site decision making.

First Generation System:
The toolkit was designed as an open framework that can accept and interfaces with a multiplicity of tools, types and functionalities allowing for the incorporation of additional data sets, conditional gates, analytic and evaluative subsystems. It can also be understood as a component of a larger framework that operates across different scales, engaging with higher level logics.

Systematic and Non-systematic Computation:
Non-systematic computation exists as the realization of a priori forms through the use of discrete computational functions without consideration for the direct interplay and communication between them. Systematic computation arises through the direct articulation and acknowledgement of the relationship between discrete computational functions and subsystems.

EXPANDED NOTIONS

The statements, premises, and implications of these arguments either generated further discussions in review or were our own form of internal testing to prove the validity of a logic we were to adhere to in the work and way we framed our thoughts on the work.

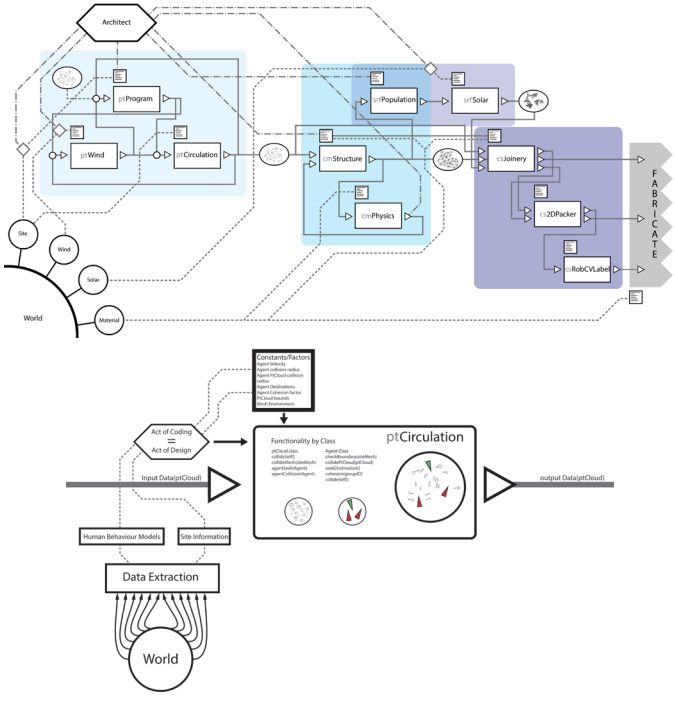
Computational Toolkit

The idea of the Toolkit, or the characterization of the script or program being a 'tool', came from an analogous discussion of separation of designer/maker, design process/act of making, and design/made object. This allows us to remove design process from the mind of the designer, and examine it as its own standalone element, that it too, can be designed, created, controlled and understood. If this is true, it stands to reason that analysis of the decision-making process could extract, in theory, the design intuition of the designer, and begin to understand the rules and logics employed. If then we translate these decisions and rules to code, as a bound tool, within a larger system of tools linked together, we could externalize and codify the design process.

In terms of our work, the tool as a singular object or script allowed us to think of the performance and functionality of a given tool only in relation to the logic and data it had to take into account, and accomplish a particular goal. The design of the tool involved taking a known variable, or data set, one of architectural significance in a traditional design process. The tool, based on its functionality would use the input data to perform a particular task, ideally yielding a result congruent to the logics employed but also our external goals for what it is meant to achieve. If the output did not accomplish this, we redesigned the tool, but not the object yielded.

This compartmentalization allowed us to focus and accomplish particular goals and subdivide the task, but also allowed us to construct and understand the interplay of these tools as an open framework of discrete tools and subsystems. This is where systematic computation at the macro-scale occurs, each tool could be used in arbitrary ways or ways that are incongruent to the logics it was constructed with. However, if they are to be linked in a larger system that has loops, conditional gates, and particular orders of operations, then the entire toolkit could behave as one macro scale system of tools and subsystem loops and operations.

As a functional aside, it was interesting when we noticed that a single script could actually be subdivided into many discrete tools and subsystems. Our choice to include them into a singular script had primarily to do with fidelity and control of data structures. However the autonomous piece of code was no longer understood as just that but became just another subdivision in a larger hierarchy of systems and subsystems.



107.1 - 2
Toolkit system diagram and Tool Zoom-in Detail

108.1 - 3
These three images are too considered 'instatiations' produced by the system of tools described earlier.

Instantiation as Architecture

The built instantiation is experienced independently of the logics of its generation. This disconnect leaves the instance in an ambiguous state between architecture and object. Altering qualities and factors in its siting, scale and tectonic execution could reposition it in a more architectural state.

Being a thesis about Architecture, it was our intention to show evidence of a process that could produce 'architecture'. There are fundamental qualities of material population and perceivable boundary to define space, therefore implying an architecture. Primarily this was the result of not finding an economical skin solution, and one that was part of our system of generation and control. The object itself is a skeleton, a literal framework, for something to occupy, fill, and populate. The spatial location and distribution was calibrated to allow for these spaces to emerge as the clustering became dense enough.

Another observation was simply what it means to see the object devoid of all of our work behind the scenes. As a form and series of material qualities, what is it? We fully accept its reality and the interpretations that others have, and therefore willingly abandon our own. It was not a concern of ours, speaking honestly. Ideally, it would be architecture, a spatial construct, and the leap from the instantiation to the discussion about what the process means for architectural practice would not be as big of a leap.

As a self-critique, our argument to fully incorporate material and fabrication processes as a generative and limiting data set to the architectural design system ended up demanding so much it limited our ability to execute a fully viable 'architecture' given our resources. If we took a shortcut, it is no stretch of the imagination the instantiation could have been wrapped up, into a pretty finished thing.

Our dedication to the merits of what it means to be honest about the work held us to what the system allowed to be yielded. The work is what it is, what could be done with what we created, the time, labor, material all hard limits that clearly defined the scope. We make no claims that our toolkit can make things that it did not.

Fabrication

The incorporating fabrication machines, technologies, processes, assembly and construction logics into a systematic computational workflow served as evidence to our critique of contemporary work.

Computation in architecture, not to mention its conflicting operation in regards to a traditional design process, typically involves research in micro-scale material investigation and sciences, often fail to use the sciences of the final scale, scope and materials and their respective fabrication processes to inform the generative cycle in the computational front-end of the execution. There are notable counter-examples, yet due to the constraints and fidelity of computational software, and technology are still short of a fully functional architecture. Those that do intend and eventually manage to produce a viable architecture, do so at the compromise of not only the initial generative logics, but often do them in an abstract space with no information of the material reality it is be exposed to. This is partially a case of difficulty, fidelity of information and the complexity-space of the problems being engineered out of the process in favor of traditional means of execution.

Material, machine, and other parameters, tolerances and data sets were specifically included in a number of the tools functions, constraints. The extraction of this information was an observational process and by no means automated. On many parameters we needed to determine within what tolerance we could abstract the data. Fluctuations in material thicknesses, tolerance of machines to cut precise geometries, kerf of subtractive processes, the repeatability and precision, were all weighed and measured. Some could operate as constants. However, in the case of a terribly bowed 1x2" pine stick, this blew far beyond what we could account for, and automate in any efficient manner. It was our goal to accept these truths and push the software to account for the variability of real data. Instead, to simplify the matter, we just selected better grade lumber, and measured bow and torque of material to ensure it would be straight enough to mill correctly and not compromise the dimensionality of the assembly.

A Priori vs Systems

Systematic computation preserves the connection between intent and execution. Erroneous results force refinement of the system, and not the resultant form. Whereas traditional design practice amalgamates conflicting logics into a single object, obscuring the connection of intent to execution.

This was the primary question that our thesis tackles directly, a methodology of design as an alternative to current and a long established history of practice, and not just in architecture. This dichotomy that was implicitly outlined by merely adopting and developing our method highlighted a crucial argument. The two 'opposing' design methods, after defining systematic and nonsystematic computation, are a-priori and systematic design.

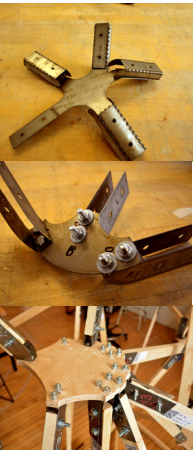
A-priori is a platonic concept, meaning an idea or concept is an absolute truth that is universal and independent of the physical realm. Consider the brass sphere. There is an absolute geometric truth of the sphere, then there is the instantiation of the brass sphere as made by the craftsman, artist. No matter the material or tool, the made sphere is and will never be the absolute sphere. By a-priori design, it is meant that the designer has the absolute concept, and the goal of the designer is to most directly realize that concept through means of representation of that abstract idea. The architecture is pre-conceived, and exists solely in the mind of the designer before anything or anyone else receives a translated form of it.

The other, systematic design, involves the shift of the act of design from the object to the process. One designs a system that yields the final object, not the object itself. This can be thought of as a meta-design, if the low level is the object itself. Our postulate is that the system is itself is designed. That means the feedback loop and iterative conventional means of design happen not on the object, but the on the process that yielded the object. If the final form is erroneous, the form is not edited, the logics employed are edited.

For some this creates a problem, one that we see as a unique feature, and advantage to this method. The designer could in theory act completely ignorant of the final forms, and only operate on low level performative feedbacks, a metric-space. Let us suppose a building that creates coverage and spans a space. If these are the only criteria on which to evaluate the success of the built form, then success criteria can be simple boolean values, did it span and cover the space: yes or no. The designer can work until results on all criterion yield true values. Obviously this is simplified, and to bring another point.

Architecture is a massively complex co-dependent system of variables and data sets that are constantly at conflict with each other. Complexity does not arise from quantity of elements but by the number of connections and dependencies within the system. Given this the example outlined before does not discuss even the most basic questions of the myriad ways one can create span and coverage. This is a point of contention, as it is clear with current development and level of functionality of software and fidelity of computing to move between real and virtual realms, that for architecture to be fully generated in a systematic computational model is currently unachievable. However, as with the development of all computer languages, libraries, and programs, we are on the path to building to that level of functionality, control, and fidelity. In theory, if one could design a system and span the low-level to high level decision-making logic frameworks, building in metrics for internal automated iterative-analysis loops, systematic computation could completely supplant the traditional design method.

Counter-arguments to computation on any level or method of usage in architecture enter at this juncture of the distinction of a-priori vs system design. There are a few, involving flawed assumptions borne in ignorance, fear, and controversy, others that take case against the level of authorship and separation that this new method begins to suggest. The traditional means of design, and object-oriented design, that is taught, practiced, and universally accepted as the primary or only way of working in architecture is superior because it is not fraught with issues of code and abstraction, or is to a lesser degree. It also creates a more direct causal relationship between designer and design, maker and object. This causal relationship is directly tied to level of ownership, authorship and responsibility. We make the argument that through designing a process that yields these same designs, that the level of authorship is the same! This is not how most



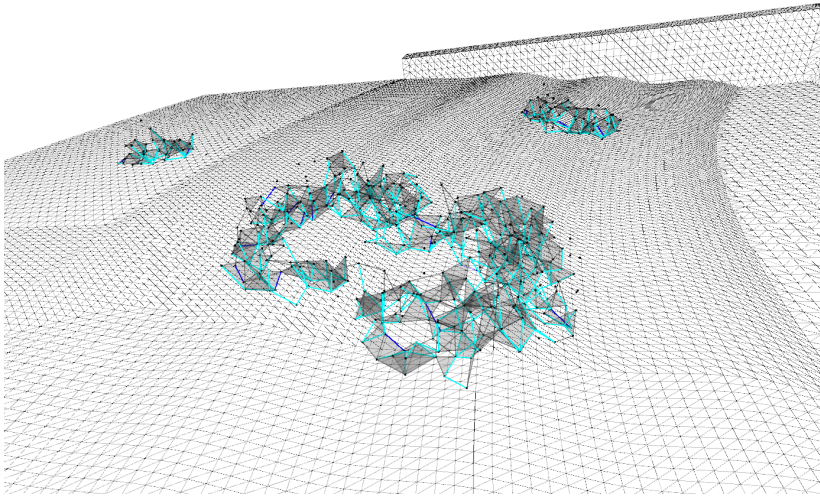
110.1 - 3
Multiple joinery iterations shown with varying implementations of material and positioning strategies. A design process that had to be understood as a singleton within a wider array of unknown forms.

designers nor society currently interprets the problem of the “black box” of computational tools due to systematic complexity obscuring the legibility of a linear series of causal relationships.

The argument that the anti-computational crowd is making: the output of the code is not the property nor within the control, and therefore authorship, of the designer. This is not the case. As we expand further in the subject of craft in computation, code that is designed by the designer, should be within the control and authorship of the designer. The output of this code is and should be also within the control and authorship of the designer at the same level of fidelity and magnitude exerted in the design of the code itself. To say that somehow the unintended consequences of code are no one’s fault suggests that the computer is an autonomous entity capable of uninitiated actions. This is also not the case. Computers do what they are told through instruction and perform action and make decisions only based on data that they are given. The perception that this is not true can be created, but it is just that: a perception.

Other counters descend from this point in terms of the level of understanding suggested by the assumptions they are based on. There is a hyperbolic dual to the one described above about the authorship issue. Because so much of code, coding, coding environments, operating systems are the collective effort and collaboration of years of development and dependency, the one who codes on the functional level owes all of their work to those that came before. This is an originalist argument. Someone before the designer in question created an algorithm and that algorithm is somewhere in the system of the designer’s code. Therefore the work is not the designer’s but property of the first one to create that algorithm. This argument is usually fielded in the subject of geometric or spatial maths, because they are more directly legible in the work. If one uses Gregory Voronoy’s algorithm for spatial subdivision, your work is therefore authored by him, more so than it is your own. However, one can take this further by the same logic. Who created the algorithm that allowed the parsing of collected points in space, calculation of the distance of two points, the computational model of 3D space in which this data is stored, how data is managed, structures of logic gates and binary pattern controls? Euclid? Des Cartes? Ada Lovelace? Where does one draw the line and take command of the code and therefore the result? Yes, all of these things must be used in the final execution, and no single designer can rework years of computer science just to unequivocally say they own the work. Still the designer, if they are to adhere to good craft in code, would understand and master the concepts of these lower level functions, and leverage them when it suits the designed purpose. One can not operate in complete ignorance of the functionality of an algorithm, employ it, and claim they are in control of the code and therefore the result. It does take a trained eye to critique a design that is a slave to higher level functions not fully being leveraged to the goals of the designer. Anytime there is an intent but could not be followed through because the script did not achieve it is evidence of this pitfall.

Beyond that arguments cycle around adherence to tradition, the unquantifiable elements of design thought and human experience, more conspiracy of misunderstood computer functions, nostalgic tendencies and fear of change.



Craft and Authorship in Computation

Craft in computation is based on knowledge of low level functionality in involved processes. This knowledge and the computational logic employed in the execution of the system transfers the authorship of the designer to any resultant instantiations if the design intent and logic are congruous.

The idea of a craft is well known and understood within architecture, and there are long histories of the craftsmen and architect once being the same, being strong collaborators or one moving from one practice to another as a way to inform each respective task. Computation as a medium, a thing to be tuned, wielded, and designed, then too, must also have a property of its creation and design that is subject to the same idea of craft.

For a moment we must speak as individuals, and effectively novices, within a larger field of study that is not our primary field and expertise, but a medium we have had to master and understand to properly utilize it in our work. Computation and the best practice of coding usually involves creating the most simple, and therefore least taxing, code to perform a given task, and do so given all possible scenarios, inputs and usages without error, correctness regardless of input. This is part of our work, and in this regard there is very bad code that we have written, and there are solutions we have found to some parts to speed a process, or simplify a calculation.

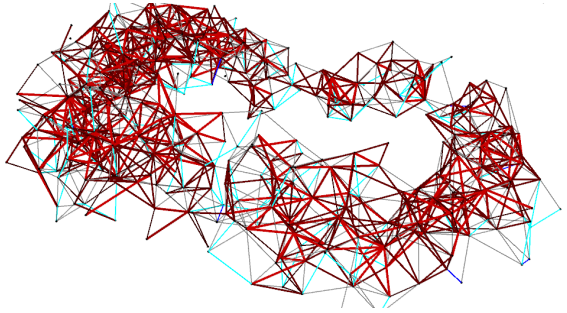
The primary purpose of describing this subject is in relation to its implementation in design.

Craft in computational design concerns the connection of design intent to execution, through control and therefore authorship. A designer has a goal, but perhaps not a form, and intends for code to achieve these goals through the performance of the result output. If the form does not perform as intended, that suggests there is missing or erroneous logic in the code. If the intent is to have a particular performative criteria met, and the logic does not imply that the result will accomplish this, meaning it may or may not by chance, then it is invalid. If the code is based on false data, it is unsound. A solution can be solved given that the intent and logic arguments are valid, and based on sound data. Failure to do so merely requires further work with the fidelity of the translation of the design logic to the code itself. Valid code is congruent to design logic and intent.

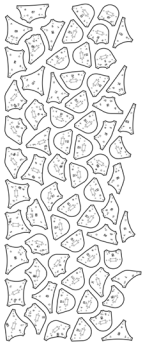
Code is an abstraction and translation of design logic. One must explicitly and accurately mediate this translation. This brings up the issue of control. Because computers exist in a relatively barren space, devoid of many assumptions and data sets we can recall without conscious effort to base decisions upon, and functionality that is incredibly low-level compared to the high level operations of our conscious and subconscious design thought. It is the task of the designer to explicitly include everything from the lowest level to accomplish the successful transfer of design process to codified logic.

To have this control involves the low-level understanding and operable capability in code. There are consequences to all functions, not knowing the full extent of these consequences, let alone the conceptual understanding of the maths and theorems of how they come about, is to relinquish control of this operation and jeopardize the congruity of intent to code and therefore the results it may yield.

Authorship also follows this same argument of control and operability in code. Authorship however implies there is not merely the capability but the intentional manipulation of control over something to exert some expression through this medium.



111.1
Those colors must mean something...



112.1 - 3
Artifacts of the functionality of the toolkit system designed to assist with the construction process

Construction Process
Systematic computation engaged with digital fabrication allows for the explicit linking of digital and physical information. Construction logics and nomenclatures emerge based on this relationship, controlling the degree of on site decision making.

This topic evolved from the discussion of the drawingless-construction, the digital information linkages to on-site parts, and implications and realities in practice. The QR-coding of parts is by no means new. Barcodes for parts on site for complex construction processes are imperative to avoid human error, in sorting, labeling or placement. Other highly digital-dependent construction processes go so far as having a superimposed geometric system to always cross reference.

The animations of the digital geometries that represented the pieces in a partially assembled state with additional information to help correlate and correct on-site errors was certainly a feature of utilizing a fully computational workflow. However its actual usage in construction was superseded by the logic of the nomenclature system.

The nomenclature borrowed heavily from the data structures that governed the continuity of objects and their calculations, manipulations and relationships. The naming had no spatial information, only relational information to other pieces and parts. A Node shows what fingers it needed, what side, and on what slot. A connection member indicated on what node, and what finger it attached and bridged between.

The unexpected consequence of this was an incredibly efficient and clear construction process. In this system's design, we only considered our ability to read the naming of parts. As things progressed, it was clear we would need help from volunteers during construction. To our surprise, all volunteers picked up the system after one explanation. We spent more time teaching people how to use a socket-wrench then we did teaching the nomenclature. Errors that occurred were quickly determined, and fixed by other volunteers.

First Generation System
The toolkit was designed as an open framework that can accept and interfaces with a multiplicity of tools, types and functionalities allowing for the incorporation of additional data sets, conditional gates, analytic and evaluative subsystems. It can also be understood as a component of a larger framework that operates across different scales, engaging with higher level logics.

This subject is where speculation enters the discussion about what our thesis implies. Under what possible conditions could such a process really do the things we set out to achieve, and more? We've established, internally, that because of the legwork of building many levels of the system, we had to operate at a very low level to ensure correctness in execution. Simulations were abstract, fidelity was low, geometries were simple and opportunity for formal variety and other functionality of the system was highly limited due to the scope it was targeted to achieve.

The one critical element that is missing from the system is an analytic, and learning, tool. As a designer creates, they learn about the results of their intentions, their assumptions(input), its success(metric-base performative evaluation, scoring). To an extent, we served as this tool, the gatekeepers, the input source, evaluating things that were beyond our scope to construct a metric to reliably sieve out failures.

It was also speculated that as one develops this toolkit, similar to a library of functions, that this could be understood as one very small subset of operations the larger scale architecture-generation system would operate upon. Our focus was on what could be done, and what we could do to demonstrate the scope of effort and resources required to produce what we could show. Taking this and previous discussions into account, we accept this speculation as a possibility.

Systematic and Nonsystematic Computation
Non-systematic computation exists as the realization of a priori forms through the use of discrete computational functions without consideration for the direct interplay and communication between them. systematic computation arises through the direct articulation and acknowledgement of the relationship between discrete computational functions and subsystems.

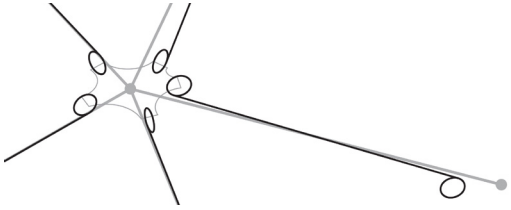
The distinction of the usage of computational tools is primarily the subject matter of the defining of systematic vs non systematic computation. Systematic computation involves a process that in a way that is creating a larger tool that governs and controls sub-tools and systems, and that at no point is there deviation from that system within it loop and process. This can be done as an analog process, but is subject to human error in deviation to a particular rule set that is intended to define and control the system.

Non-systematic computation is the traditional and accepted model for designers to interface and use these new tools, digital but not considered as an aid to the act of designing, merely leveraged to represent. Prepackaged software is designed for this usage primarily, but are increasingly providing ways to create control systems and leverage the functions and environment the software provides. Typically, these softwares are used to create a representation of what the designer has already decided upon, and the tool serves as a way to translate this abstract human thought and conception to 3D data that can then be transferred to other media: drawings, models, etc. The software and tools contained do not or are not supposed to inform or prevent action or will of the designer to be impeded. The relationship of the designer to the tool is less of an introspective process, but one of master and slave.

This is not an entirely accurate characterization as one creates an object, the act of creation and the perception of all the stages of that creation informs the act of creation itself. Juhani Pallasma(4,5) refers to this as a haptic feedback loop, that designing is never a linear system, but involves loops between the tool, object, and designer. Because there is an inherent lack of physicality to the digital tools, therefore the haptic connection is lost. However, other means of perception and understanding can still create a feedback loop.

It is then reasonable to suppose that as the designer, non-systematically leveraging a software is also being influenced by the nuances of the interface, views of the creation, now for the first time being put to numbers and visualized. The feedback loop and analysis begins already at this stage. This sets up an important assumption, digital tools and their use are just as informative throughout the design process as physical tools, just through different modalities and levels of cognition.

If this is true, then let is be examined in the context of non-systematic versus systematic computation and design processes. In both there is the same feedback loop but on two different levels. Traditional design practice and iteration informs the designer about the design, but must then deduce flaws in the logic of its creation, rather directly understand and address them, underlying form instead of perceived form(6). Whereas if a designer develops a system for the design process, and receives erroneous results, this points to flaws in the system.



113.1 - 2
Outward appearence; underlying form.

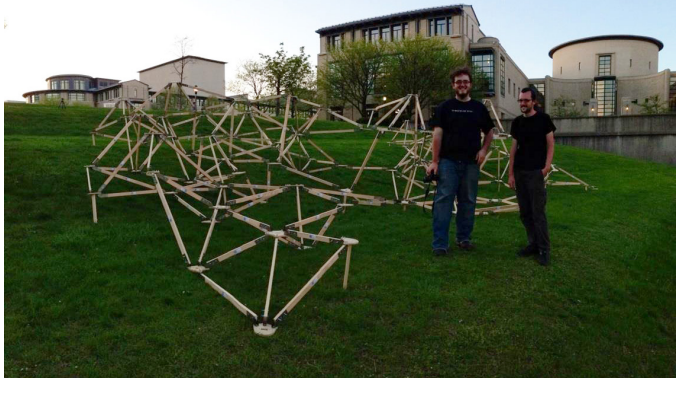
CONCLUSION

We have little to present as a concluding statement that would not otherwise reiterate arguments already stated. We accept the interpretations of what we have done in the manner they have been presented, we hope, in the most honest and true form. Much work with in architecture deals with a time-scale and scope that often extends into possible futures than the realities of the present. To this end, we hold back from including speculation as content or evidence for the thesis. Rather it is our desire that our thesis statement(s) imply these futures or realities.

Some points of clarification however may be in order. The work depicted within the scope of this book was completed by two individuals, Jordan Parsons, and Michael Jeffers within the timespan of one semester, Spring of the year 2013. Built work received most welcome assistance from a few selfless friends. This partnership was indeed a case of group work that exceeds the capabilities of either individual. This served several ulterior motives as the concluding work for our 5-year profession Bachelor's of Architecture degree. These motives became formative arguments outlining the thesis itself.

We both sought to reconcile the logic and power of code, with the potential for deconstructing a systems approach to a design objective. Success criteria was intentionally limited to low level metrics to avoid passing high level judgements on results, a tempting pitfall. This was a standard we held as we knew the way we worked served as evidence for the statements we made, the methodology.

It was an ambition to provide a body of evidence to support these statements. This is why honesty, abstaining from speculative leaps, was imperative. We did not want to rest our confidence on conjectures that would more convincingly depict the power of the process we are describing. We had to do it ourselves, from the bottom-up, to show others and ourselves that it could be done. There were naysayers. This is for them. We can only hope this serves as further evidence of the potential of this particular method for approaching design, as a system itself.



114.1 - 3
Image credits: 1-2: [EN]Coding Architecture Conference 2013, 3: Phyllis Kim. Depicting the thesis team, early in the process (1, 2) presenting their work, pictured with Madeline Gannon, Niel Leach, Wes Mcgee;. Then after completing the final build(3), notably sunburnt.

To conclude we would like to focus on all the people and experiences that led to this apt summation of two individuals' work.
We would like to thank the academy…

Carnegie Mellon University's School of Architecture, those that taught and supported us both in our 5 year journey, as well as many other professors, staff, colleagues, across the university. Our advisors for testing us, guiding us, and most of all trusting us. Dale Clifford, P. Zach Ali, Eric Brockmeyer, Madeline Gannon, Golan Levin, Zack Jacobson-Weaver, among others contributed greatly to the intellectual development through their advising. We also want to make mention of all of our friends and colleagues who were there for the midnight crits, and most of all the final construction phases. There are too many to name, but we are going to try anyway:
Henry, Sara, Ying, Dan, Becki, Sharon, Chasen, Jake, Harris, Sam, Steve, Matt, Alex, and a whole army of young architecture students from all years joined in near the end and at a point record-keeping and cognitive processes were limited.

115.1 - 3
Friends and fans abound!

WORKS CITED

(0)Herbert, Frank. Dune. Philadelphia: Chilton, 1965. Print.

(1)Kolarevic, Branko. Architecture in the Digital Age: Design and Manufacturing. New York, NY: Spon, 2003. Print.

(2)Kolarevic, Branko, and Kevin R. Klinger. Manufacturing Material Effects: Rethinking Design and Making in Architecture. New York: Routledge, 2008. Print.

(3)Landa, Manuel. A Thousand Years of Nonlinear History. New York: Zone, 1997. Print.

(4)Pallasmaa, Juhani. The Eyes of the Skin: Architecture and the Senses. Chichester: Wiley-Academy ;, 2005. Print.

(5)Pallasmaa, Juhani. The Thinking Hand: Existential and Embodied Wisdom in Architecture. Chichester, U.K.: Wiley, 2010. Print.

(6)Pirsig, Robert M. Zen and the Art of Motorcycle Maintenance: An Inquiry into Values,. New York: Morrow, 1974. Print.

(7)Schumacher, Patrik. The Autopoiesis of Architecture. Chichester: J. Wiley & Sons, 2011. Print.

(8)Terzidis, Kostas. Algorithmic Architecture. Oxford: Architectural, 2006. Print.

TECHNICAL RESOURCES

HAL Robot Programming & Control – Thibault Schwartz

Grasshopper – David Rutten

Rhinoceros – Robert McNeel & Associates

Java SE 7 – Oracle

Processing 1.5.1 – Casey Reas, Ben Fry & co.

C# – Microsoft

ABB IRB 4400

ABB IRB 1600

CNT Motion Systems 950

7/16” Rachet

1/4”-20 Bolts, lock-washers, nuts

1”x2”x8’ Select-Pine lumber