Digital Craft | In Search of a Method of Personal Expression Within the Digital

by

Wade Brown

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Architecture
in
Architecture

Waterloo, Ontario, Canada, 2018

AUTHOR'S DECLARATION

ABSTRACT

Our relationship with the digital has fundamentally changed within the past decade. A mesh of outside interests have been efficiently folding themselves into our lives. These exist as either a legion of hosted "free" web services touting the promise of a new-found collective intimacy, or a set of tightly coupled IOT(Internet of Things) applications that are slowly being pulled away from our fully capable hardware–all causing us to rely heavily on a virtual infrastructure that demands to host our work and place us at arm's length of tools that we no longer own or control.

This new bargain includes a view into our work and habits so that we can be better understood, tokenized, categorized, mapped, and finally monetized. While many today may be OK with this relationship, I'll be frank, it unsettles me. I believe something fundamental is lost in this unravelling long-distance relationship.

This thesis is a response. It pushes for a more intimate connection with technology within the backdrop of digital design and its many processes. In The Craftsman, Richard Sennett writes: "Making is Thinking," and in his text he explores the close relationship between head and hand for a small set of traditional craftsmen: a cook, a musician and a glass blower. To elevate the digital within today's architectural practice, I feel its use must also be seen as craft. But how might a relationship between head and hand manifest itself? Is there some similarity in thinking between Sennett's craftsmen and the processes of successful digital design?

I propose to investigate the mechanisms of digital *Making*, and hence digital *Thinking* through three design problems, inspired by the works of Neri Oxman, deskriptiv, Michael Hansmeyer, as well as the methods of D'Arcy Thompson, Shinichi Maruyama, Pina Bausch, and Frei Otto. By mindfully observing my exploration of these from a digital perspective, I believe it will be possible to get a sense of what makes craft possible within this realm.

ACKNOWLEDGEMENTS

While this journey was a solitary one, I must thank the following for their assistance and care:

Anne Bordeleau, my supervisor, while the material had sizeable technical breadth, you were able to help me bring it into focus for all those who might want to follow this path.

David Correa, who's frank feedback was very much appreciated, I am thankful for your perspective when grounding this work.

Rick Haldenby, for saying "yes," and for promoting this point of view for anyone who will listen. We hear you, and we will pass it on.

DEDICATION

To my wife Lisa, without your love and support none of this would be possible.

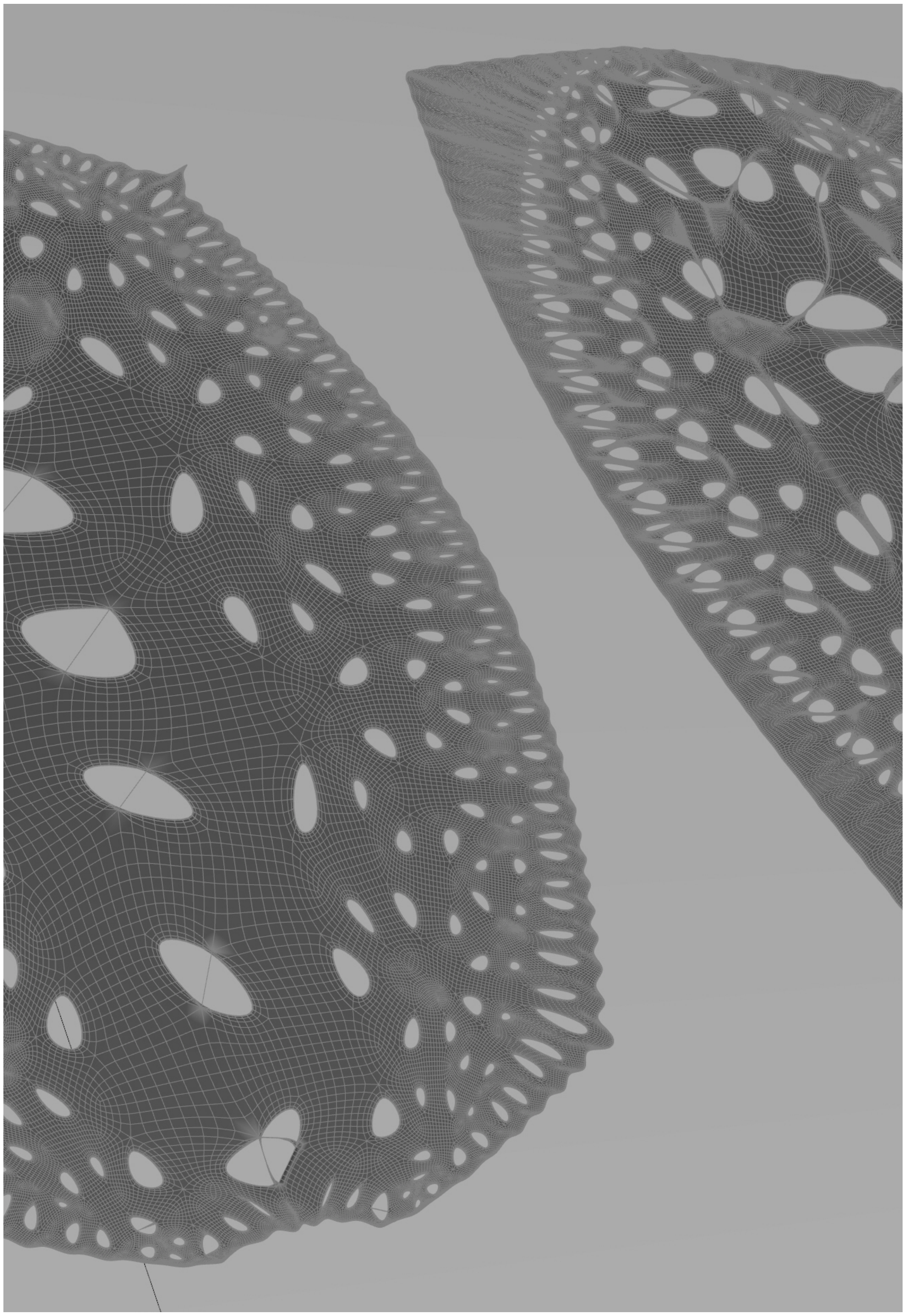


Fig 1 - Diatom Flex

Table of Contents

ii Author's Declaration
iii Abstract
iv Acknowledgements
v Dedication
vii Table of Contents
viii List of Figures

2 Introduction

8 Understanding the Organic

66 *pina* - A Choreography of Affect

108 Of Materiality

178 Conclusions

182 Bibliography

188 Appendix

List of Figures

Page	Figure	Description
ii	Fig. 1	Diatom Flex

Understanding the Organic

Page	Figure	Description
7	Fig. 2	Fractal Arches
9	Fig. 3	Alarm Clock Image
11	Fig. 4	Foraminifera Spiral (D'Arcy Thompson 1951, 857)
13	Fig. 5	Balanus Barnacle (https://meowmeow0508.files.wordpress.com/2011/06/7e3bd-2d182c368059950276c.jpg)
13	Fig. 6	Anglerfish Ovary (https://www.nigms.nih.gov/education/life-magnified/Pages/7_lt_hayden_an-glerfishova.aspx)
13	Fig. 7	Fibonacci's Mashrabiya - Oxman (http://matter.media.mit.edu/tools/details/stalasso#prettyPhoto)
15	Fig. 8	Mashrabiya Pattern
15	Fig. 9	Mashrabiya Pattern - Cell Arrangement
15	Fig. 10	Mashrabiya Pattern - Nolli Map
15	Fig. 11	Mashrabiya Pattern - Pore Arrangement
15	Fig. 12	Mashrabiya Pattern - Open Pore
15	Fig. 13	Mashrabiya Pattern - Voronoi Diagram
15	Fig. 14	Grasshopper Analysis Script
16	Fig. 15	Mashrabiya Pattern - Size & Arrangement
17	Fig. 16	Victorian Diatom Arrangement (http://studylib.net/doc/5722970/photography-through-the-microscope-da-vid-linstead)

17 Fig. 17 Campolydiscus hibernicus
(http://www.psmicrographs.co.uk/diatom-frustules--campylodiscus-sp--/science-image/80017290)

17 Fig. 18 Triangular-shaped Diatom SEM Image
(https://fineartamerica.com/featured/5-diatom-sem-steve-gschmeissner.html)

17 Fig. 19 Diatom SEM Image - Steve Gschmeissner
(http://healerdimitri.com/wp/wp-content/uploads/2015/05/42-diatom-sem-steve-gschmeissner.jpg)

17 Fig. 20 Diatom Triceratium SEM - Steve Gschmeissner
(https://fineartamerica.com/featured/26-diatom-sem-steve-gschmeissner.html)

17 Fig. 21 SEM Diatom - Thalassiosira
(http://bioinformatics.psb.ugent.be/plaza/versions/pico-plaza/organism/view/Thalassiosira+pseudonana)

19 Fig. 22 Cocconeidacean valve decomposition

21 Fig. 23 Diagram - Analysis of Pore Structure Hierarchy

23 Fig. 24 Superformula - Johan Gielis

23 Fig. 25 Superformula Grasshopper Script

24 Fig. 26 Superformula Evolution Diagram

25 Fig. 27 Superformula Iteration & Algorithmic Flex

27 Fig. 28 Fortran Punch Card Image - Joseph Huffman
(http://blog.joehuffman.org/content/binary/PunchedCard.jpg)

27 Fig. 29 Superformula Grasshopper script - extended

29 Fig. 30 Diatom Form Algorithm and Evolution Sequence

31 Fig. 31 Complexity and Symmetry Flex

31 Fig. 32 Representational Flex - Diatom Model Domain (a|b)

33 Fig. 33 Flex of Diatom Representational Algorithms

34 Fig. 34 Standard Diatom Model Iteration - Size & Iterative Depth

35 Fig. 35 Grasshopper L-System Script

35 Fig. 36 L-Systems Iterations - Perspective View

35 Fig. 37 Renders - Algorithm Results 6 Iterations

36 Fig. 38 SEM Frustule Papillae Arrangement
(Stefano, Journal of Nanoscience & Nanotech 2005 Vol 5, p22)

37 Fig. 39 L-Systems Algorithm Flex Iterations

39 Fig. 40 Subdivided Columns - Michael Hansmeyer
(http://www.michael-hansmeyer.com/projects/columns.html)

39 Fig. 41 Reaction-diffusion simulation(2D)
(https://www.chromeexperiments.com/experiment/gray-scott-simulation - Dec 22 2016)

39 Fig. 42 Reaction-diffusion form (Gray-Scott, 2 Million Voxels)
(http://www.michael-hansmeyer.com/projects/voxels.html)

41 Fig. 43 Grasshopper Script Render

41 Fig. 44 3D Reaction Diffusion Mimic Script

41 Fig. 45 Form Section

41 Fig. 46 Form Tectonics

43 Fig. 47 Scaffold - Grasshopper Script Result

43 Fig. 48 RealFlow Form Generation on Diatom Script Scaffold

45 Fig. 49 RealFlow Form Simulation

47 Fig. 50 Lighted Snapshot of Realflow Diatom Form Result

49 Fig. 51 Coulomb Field Application in Action

50 Fig. 52 Papillae & Form Development through Silica Deposit by Simulation

51 Fig. 53 Coulomb Field Application Menu System

53 Fig. 54 Coupling Experimental Setup

53 Fig. 55 First Coupling

55 Fig. 56 Successful Coupling

57 Fig. 57 Undersharing - Uniform Path Fabric - Each looking out for themselves

57 Fig. 58 Full Sharing - Random Static Charges

57 Fig. 59 Full Sharing - Organized Charges

pina - A Choreography of Affect

65 Fig. 60 Particle Swarm

67 Fig. 61 Mandelbulb Slice Voxelization

71 Fig. 62 Wandernd -"Hiking"(desktriptiv 2013)
(https://www.facebook.com/deskriptiv/photos
/a.361140117291283.85420.221944777877485/688018441270114/?-
type=3&theater)

71 Fig. 63 Schichten-"Layers"(desktriptiv 2013)
(https://www.behance.net/gallery/12957793/schichten)

71 Fig. 64 Guaddel(desktiptiv 2014)
(https://www.behance.net/gallery/13734827/Quaddel)

71 Fig. 65 Gewoge-"Waving"(desktriptiv 2013)
(https://www.behance.net/gallery/13418207/gewoge)

71 Fig. 66 2-Manifold Output SimpSymm - (deskriptiv 2014)
(https://www.behance.net/gallery/16138367/SimpSymm)

71 Fig. 67 Flow 1 (deskriptiv 2014)
(https://www.behance.net/gallery/17617243/FlieSSend)

73 Fig. 68 Eyebeam Museum
(Lynn and Rappolt 2008, p328)

73 Fig. 69 NOAH Set for the Film DIVIDE
(Lynn and Rappolt 2008) p353)

73 Fig. 70 Riemann Chair - Wade Brown & Galen Jones 2013

73 Fig. 71 Robotic Arm Cutting Blobwall Brick
(Lynn and Rappolt 2008), p276)

73 Fig. 72 Vitra Ravioli Chair
(Lynn and Rappolt 2008, p141)

75 Fig. 73 Allison's Moment - Wade Brown 2007

75 Fig. 74 Nude #1 - Maruyama 2012
(http://www.shinichimaruyama.com/ Nude#1-2012)

75 Fig. 75 Water Movie - Maruyama 2013
(http://www.shinichimaruyama.com/portfolio/permalink/385025/4ab-8f78a666a04)

75 Fig. 76 Kusho #1 - Maruyama 2013
(http://www.brucesilverstein.com/other-works/kusho/7)

75 Fig. 77 Kusho - Maruyama 2013
(http://www.brucesilverstein.com/other-works/kusho/7)

76 Fig. 78 pina - Movie poster - Wim Wenders

79 Fig. 79 Andrey Berezin & Ruth Amarante in pina - Movie scene - Wim Wenders

81 Fig. 80 Flocking - Processing Investigation

83 Fig. 81 Tensor Field - Processing Investigation

85 Fig. 82 Particle Language Investigation

87 Fig. 83 Realflow Pina Particle Sequence

87 Fig. 84 Realflow Pina Particle Perspective

89 Fig. 85 Tight-Meshed Frames of Simulation

91 Fig. 86 Loose-Mesh Frames from Simulation

93 Fig. 87 Snapshot Print Failures

93 Fig. 88 Supporting Material Removal Frustration

95 Fig. 89 Whirl 1 - 3D Print+Model

95 Fig. 90 Whirl 2 - 3D Print+Model

95 Fig. 91 Plunging Moment - 3D Print+Model

97 Fig. 92 Snapshots in Time - Spacial Moments

99 Fig. 93 VIVE/TiltBrush VR Experiences

Of Materiality

109 Fig. 94 Pneumatic Laser Output

111 Fig. 95 Makerbot - Replicator
(https://www.shoppersshop.com/2012pics/makerbot_the_replicator.jpg)

111 Fig. 96 RepWrap Mendel
(https://mybuildlog.wordpress.com/2014/02/21/reprappro-mendel-tricolor)

111 Fig. 97 Formlabs - Form 2
(https://3dprint.com/wp-content/uploads/2014/09/form2.jpg)

111 Fig. 98 Delta Rostock
(http://rwgresearch.com/wp-content/uploads/2013/03/Delta-3D-Printer-Rostock-RWGresearch-5.jpg)

111 Fig. 99 Hagia Sofia Model - Group Project 2012
(Stephane Gaulin-Brown, Patrick Verkley, Safira Lakhani, Monty DeLuna, Rachel Bruijns,W. Brown)

113 Fig. 100 Solar Sinter - Markus Kayser
(https://inhabitat.com/the-solar-powered-sinter-3d-printer-turns-desert-sand-into-glass)

113 Fig. 101 Genki - Arki
(http://genkei.jp/arki)

113 Fig. 102 Geeetech Delta Rostock G2s Printer
(https://www.geeetech.com/delta-rostock-mini-g2s-pro-diy-kit-with-autoleveling-p-936.html)

115 Fig. 103 Delta Assembly Mayhem

116 Fig. 104 Impacter Tool Provided for Assembly
(http://tv-vision.ru/magazin/product/nabor-otvyortok-impacter-me-6036)

117 Fig. 105 Geeetech Delta Rostock G2s Printer - During Assembly

119 Fig. 106 Geeetech Delta Rostock G2s Printer Moving Parts

121 Fig. 107 Simplify3D Software - Diatom RealFlow Output Test

123 Fig. 108 Repartier RealFlow Output Test

123	Fig. 109	Arduino IDE and Mega2560 Firmware Config
125	Fig. 110	First Output - From Formless to Form
127	Fig. 111	Test Print w. Stringing - Bracelet Model(Hegglin 2013)
127	Fig. 112	Collision Offset Result
129	Fig. 113	Test Piece Output
129	Fig. 114	Model Test Piece(Matsumoto 2016)
129	Fig. 115	Output Test 20% Infill
129	Fig. 116	Test Model
129	Fig. 117	Bracelet Output
129	Fig. 118	Bracelet Model (Hegglin 2013)
129	Fig. 119	Lamp Output
129	Fig. 120	Lamp Model(NervousSystem 2013)
133	Fig. 121	Roof Form Finding - Frei Otto (www.frieottofilm.com)
133	Fig. 122	Soap Bubble/Optimal Surface Calculation - Frei Otto (Photo by IL-Bach/Klenk 1987)
133	Fig. 123	Roof Optimization, Multihalle, Manheim - Frei Otto
133	Fig. 124	Wool Thread Network Optimization - Frei Otto (Image from Structure of Vagueness - L. Spuybroek)
135	Fig. 125	Simple Pneumatic Test Script - Kangaroo - Gerstheimer + Brown
135	Fig. 126	Bladder Restriction - Brown + Gerstheimer
135	Fig. 127	Bladder Creation - Gerstheimer
137	Fig. 128	Frei Otto Filament Grasshopper Script
137	Fig. 129	Filament Convergent Solution
139	Fig. 130	Kangaroo 2 - Based Patterning

139	Fig. 131	Pneumatic Volume Grasshopper Script
140	Fig. 132	Grasshopper Pneumatic Simulation
141	Fig. 133	Manual Pneumatic Chamber Creation
143	Fig. 134	UNIVERSAL Models - Laser Cutter Investigation
145	Fig. 135	Laser Cutter Operating
147	Fig. 136	Testra Driver Software Interface
147	Fig. 137	Emission - Open Laser Head w. Collimator
149	Fig. 138	Early Laser Fusion Investigations
149	Fig. 139	First successful Pressure Test
149	Fig. 140	Laser Refinement and Testing
151	Fig. 141	Various Collimated Tests - PWM & Power/Speed
151	Fig. 142	Open Beam Laser Tests (No Collimation)
153	Fig. 143	Existing Testra Controller - Transition Connectors
155	Fig. 144	Bill Gates' Famous Letter to Computer Hobbyists
157	Fig. 145	LaserWeb Interface - Pneumatic Path View
157	Fig. 146	Path G-Code Listing - Laserweb
159	Fig. 147	GRBL Controller Version - Connected Block Schematic
161	Fig. 148	Assembled GRBL-based Proof of Concept
161	Fig. 149	End-Stop Noise Filter Arduino Uno
161	Fig. 150	15A-24V Power Supply
161	Fig. 151	Raspberry Pi
161	Fig. 152	Stepper Driver
163	Fig. 153	Smoothie Solution - Connected Block Schematic
165	Fig. 154	DIN Rail System in Hammond Case

165	Fig. 155	Plasma-Cutting of Interface Ports in Case
165	Fig. 156	Signalling & Power Connection
165	Fig. 157	DIN Terminating Blocks
167	Fig. 158	Smoothie Controller - Fitting and Assembly
167	Fig. 159	Smoothie Soldering Repair
167	Fig. 160	Shrink-wrap & In-line Voltage Reduction(24V->5V)
169	Fig. 161	Pneumatic Chamber Results - Smoothie Controller
171	Fig. 162	Multiple-Line-Offset Pressure Reinforcing
173	Fig. 163	Pneumatic Structures
175	Fig. 164	Render + Pneumatic Structures

"The roots of our understanding of architecture lie in our childhood, in our youth; they lie in our biography. Students have to learn to work consciously with their personal biographical experiences of architecture."(Zumthor 2006, 57)

Introduction

In my previous career in information technology I was entangled within the predestined train-wreck described by Moore's law. My much beloved two-times improvement seen each silicon development cycle was soon to become a thing of the past. And the industry didn't disappoint; the computing market shifted to become application and network-centric as the horsepower well had run dry—drowning spectacularly in heat dissipation issues. Intel's tick/tock development cycle was re-branded to include a new "optimization" phase, but we all knew it was inserted to pacify critics of the loosening historic yearly delivery cycle. Tablets, phones, and watches boasting battery-life, connectivity, and convenience replaced the "twice as fast as last year" press releases while the silicon fab-houses worked tirelessly to pipeline and parallelize their offerings. It was the ultimate bait-and-switch. My now 8-year old Mac Book Pro is still nearly as powerful as the current release of Apple's "Less is More" interpretation of the same off-tune Miesian koan.

Within a similar time-span, a mesh of outside interests have been folding themselves into our lives. Our applications and data have been mindfully pulled from our grasp as they are led processionally towards the hosted and the virtual. Many revel in this new-found access to "free" products and services in exchange as the cloud anticipates their needs—an embodiment of the "share everything" mantra my hippy parents promoted during the 60's. While there certainly are some benefits of participating in a grand collective, I believe there is a significant cost in this distancing of ourselves from the applications and data we've become so accustomed to touching.

Architectural practice is not immune to the draw of the virtual. And in an effort to increase uptime, maintain operational consistency, reduce cost, and manage risk pertaining to intellectual property, many practices have moved their assets and tools to "the cloud." While I feel this might make basic business sense, I also feel that the loss of a low-level connection to the tools and data reduces the quality of the work after seeing this first-hand during my time at a local corporate practice.

This thesis is my response. I believe a more intimate connection to the digital is necessary for it to be more effective within an Architectural practice. In The Craftsman, Richard Sennett writes: "Making is Thinking," and in his text he explores the close relationship between head and hand for a small set of traditional craftsmen: a

"When design technique is influenced by craft, a fundamental displacement occurs. Since design customarily retreats from the material into the abstract world of drawing, while craft maintains a one-on-one relationship with matter" (Spuybroek 2016, 23) cook, a musician and a glass blower. Each succeeds in elevating their work to craft by forming unique relationships with their media and the tools of their respective trades. In this thesis I look to understand digital craft, and ask such questions as: how might a relationship between head and hand manifest itself in this context? Is there some similarity in thinking between Sennett's craftsmen and the processes of digital design? Are there differences?

I propose to investigate the mechanisms of digital *Making*, and hence digital *Thinking* through three design problems, inspired by the works of Neri Oxman, deskriptiv, Michael Hansmeyer, as well as the methods of D'Arcy Thompson, Shinichi Maruyama, Pina Bausch, and Frei Otto. By mindfully observing my exploration of these from a digital perspective and comparing my personal experiences to Sennett's craftsmen, I believe it will be possible to get a sense of what makes craft possible within the digital realm.

Within the first phase of my investigation, Geometry and the Organic, I will be mapping the formal logic of an aquatic algae cell(the diatom Coscinodiscus wailesii). Asserting there is much to learn from existing life, I will be borrowing from the methods of D'Arcy Thompson and Neri Oxman to assist me in the re-imagining of its encoding using Rhino w/Grasshopper scripting and other external digital tools.

Within the second phase, *pina* - A Choreography of Affect, I will investigate the processes involving dynamic digital form generation inspired by the work of Pina Bausch. As a choreographer of the well known Tanztheater Wuppertal, pina worked to understand the thoughts that activate and motivate her dancers. In this investigation, my impressions of a segment performed by Ruth Amarante & Andrey Berezin seen in the documentary *pina* will be the seed for an agent-based formal investigation implemented using multiple software systems.

Within the third and final phase, Of Materiality, I will look to understand how materiality can inform digital design. This phase will focus more on the creation and activation of digital *Making* hardware. Initially, a 10" Delta-Rostock design 3D PLA/PVC printer will be constructed and commissioned to bring a physical presence to digital work produced in the first phase of this thesis. In the second portion of this investigation, a proof of concept pneumatic structural design will be implemented digitally in Rhino3D/Grasshopper based upon the wool networks optimizations of Frei Otto and a polyethylene welding process involving the customization of a CO_2 laser cutter system.

Throughout each phase, I will be mindfully documenting my thought processes as

I work through to each design goal. In The Craftsman, Richard Sennett references a lesson he feels Hanna Arendt wants to pass on to him, “people who make things usually don’t understand what they are doing.”(Sennett 2008, 1) By extensively documenting the work here, I believe some patterns will emerge. It is through this reflection that I feel the mechanisms of craft within digital design will be made visible.

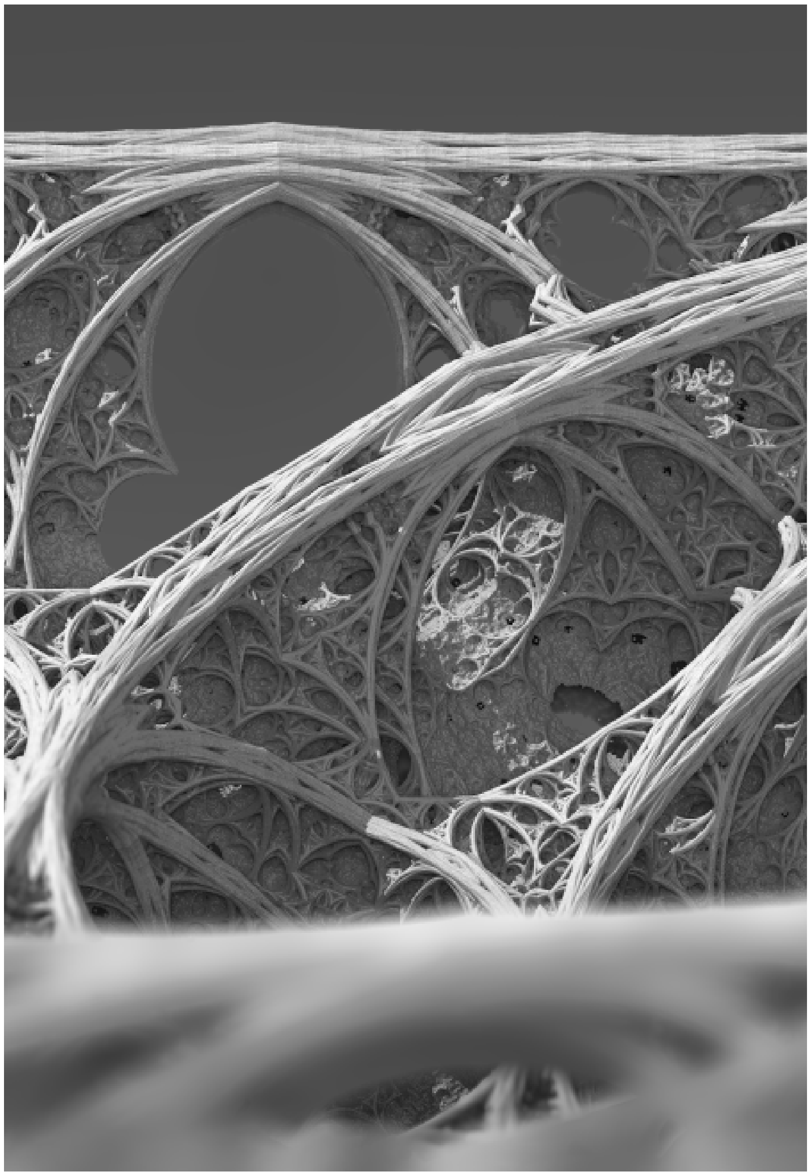


Fig 2.- Fractal Arches

Understanding the Organic

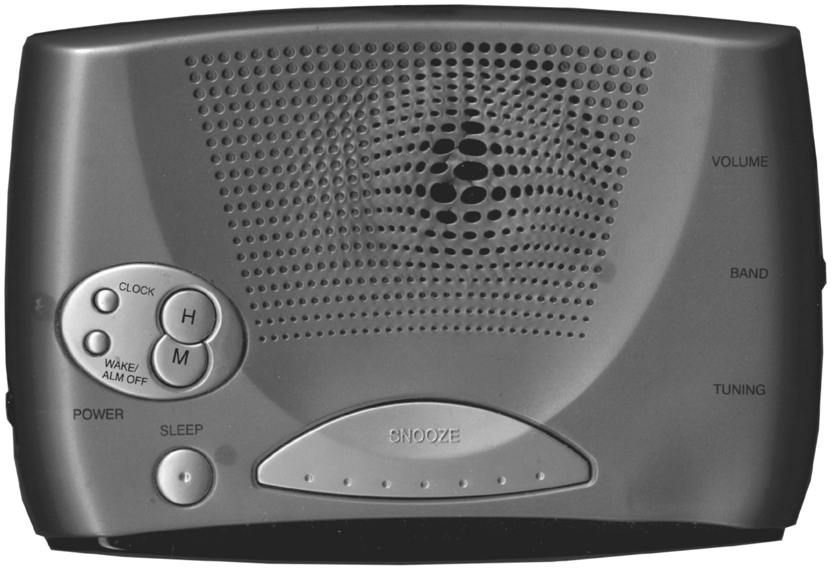


Fig. 3 - Alarm Clock Image

Introduction - An Analysis of the Organic

"Nature does nothing uselessly."
(Arist. Pol. 1, 1253a.8)

Our son had an alarm clock. One night he let the light on his night-stand get a bit too close to the clock's perforated speaker grille. The result(Fig 3 - opposite) I found remarkable. Its surface had, under the heat of the light, responded by shrinking non-uniformly based on proximity to the bulb of the lamp. Surprisingly, he rejected the clock immediately, saying that it made him feel odd; I think the term he used was "weird." There was nothing I could do to make him to continue to use it. Digging into the issue, I found that his reaction is more common than I thought - It seems that many people exhibit an irrational, almost ingrained, fear of irregularly-spaced holes known as Trypophobia. Looking back, I see that my own reaction at his age may have been similar as a fear-response at a more instinctive level, but after twenty five more years of life-experience, I've learned to revel in change. I see the event as a wonderful happen-stance and would like to investigate forms of this type.

It seems that if one is to investigate organic life within the realm of their morphology, one must first lay an offering at the altar dedicated to D'Arcy Thompson. While his treatise, On Growth and Form(Thompson 1951), is quite substantial and smells of age, it's dog-eared state gives me a sense of how valuable his work has been. This work is, quite literally, at the top most papers and books on the subject of Form and Mathematical Biology, quite an admirable status really.

The spectacular whorled foraminifera captures his ideas, and best suggests the conflict that exists between the effect of Darwinism and Thompson's growing thesis; that there is a general ideology behind the physical morphology of living things which works with adaptation, and yet exists quite separately. The foraminifera's shape is seen repeatedly in different organisms and in different environments. Their response to their environment encoded within the uniqueness of materiality and through a common generative mathematical relationship.

In searching for another life form to investigate, I propose that another aquatic or-

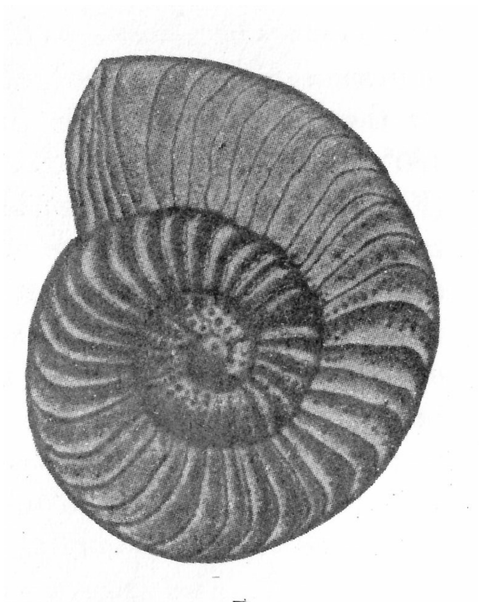

B

Fig 4.- Operculina complanata (Foraminifera Spiral)

ganism, the Diatom(algae cell), is a good choice. I've always had a fascination with electron micrograph images showing intricate micro-assemblies. The Diatom is of great interest to material scientists today as its operating (nano)scale and materiality is similar to chip manufacturers working in silicon(ZHANG 2012, 3836-3849).

Within this phase of my investigation I would like to pursue answers to two fundamental questions: How is the diatom valve(end-wall filter) organized? What digital approaches best match the form's morphology?

To answer this, I will distill some of the more basic mathematical relationships that exist within the Diatom *Coscinodiscus wailesii* by Initially performing some basic analysis on its material arrangement. It is my hope that a parametric model of its form can be developed and implemented within a set of Rhino/Grasshopper scripts once some basic organisational relationships can be distilled from some sample images. I will document each of my algorithmic and software experiments, providing a range of behaviour as the model characteristics are flexed.

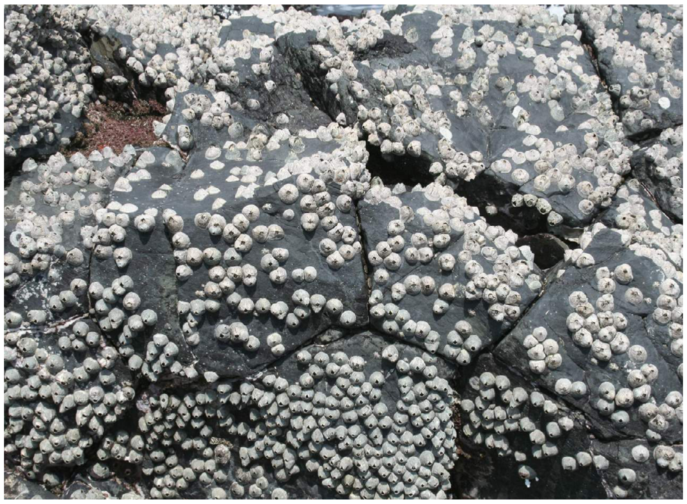


Fig 5. Balanus Barnacle

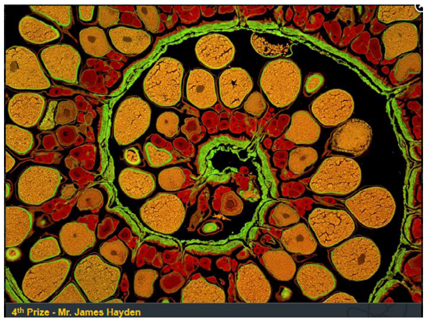


Fig 6.- Anglerfish Ovary Section

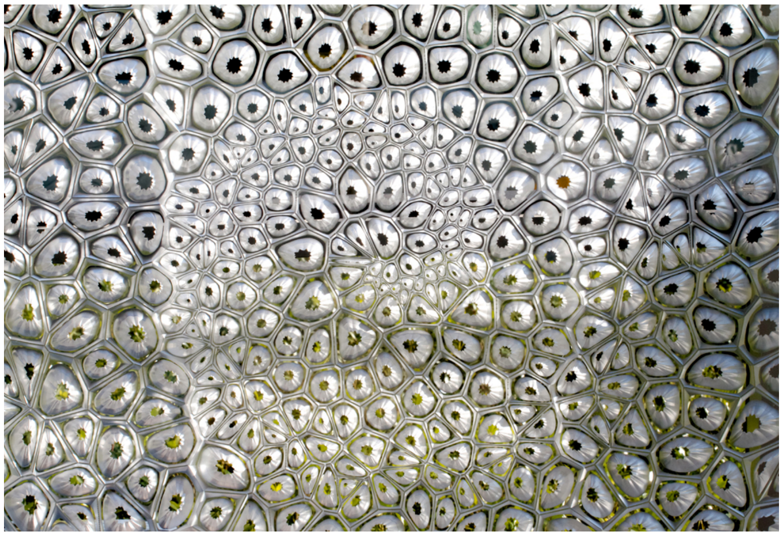


Fig 7.- Fibonacci's Mashrabiya

Precedent - Fibonacci's Mashrabiya

Neri Oxman is a design researcher within the MIT Media Lab where she leads the Mediated Matter research group(Neri Oxman). Her team utilizes organisational relationships found within living matter to fuel their development of technologies in the areas of digital fabrication, industrial design and architecture. Benefits from many productive collaborations; Christoph Bader of deskriptiv(deskriptiv 2016), Bjork, and notably with 3d Printer Manufacturer Stratasys(Stratasys 2016) are just a few. Neri Oxman's TED talk on Material Ecology(Oxman 2015) speaks to a possible future where architecture could be living and does inspire me to think of the possibilities in this area. Could designers "program" behaviour into materials? This is quite exciting to me.

The team's Fibonacci's Mashrabiya is a work that combines both the morphology of the Balanus barnacle(polyp) with the logarithmic spiral seen in an anglerfish ovary cross section in a generative sense. The goal of the traditional mashrabiya is to create a unique space protected from unwanted light, sound, and airflow. This project was set to allow for a configurable and unique way to "shape the form of light and heat moving through it"(Oxman 2015). Materially, the screen is made by applying a CNC milling process to an acrylic sheet and then later applying urethane rubber and resin composites.

To understand this further I felt it necessary to pull the pattern apart using a Rhino grasshopper script. I did a standard edge-detection and dissected the pattern into a number of layers; some were area-bounding, others more represented networks. I was particularly interested in the subdivided area of each cell, their location and relative size to one another but the networks are of note...

When the cell walls were defined by the process I applied, an interesting thing surfaced; the resultant graph looked a lot like a Voronoi network. Voronoi networks are used to determine things like "nearest hospital to a location," or "area served by a fire house" when planning a city's layout for emergency services. In a biological sense however, the "service" portion of the cell that's been created would normally be connected to some form of metabolic regulation. It could be the cell's water intake, food hole, out-hole, you name it... but the goal of the Site for the cell is for this location to be the most efficient place for distribution of materials within the cell. Since nature has had a long time working on efficiencies, it only makes sense that this network be optimized.

While it is not quite exact, for the most part Oxman's Mashrabiya does mimic the

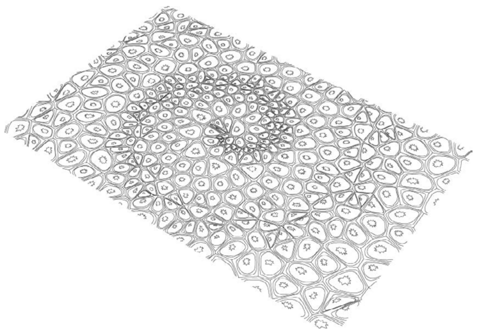


Fig. 8 - Mashrabiya Pattern

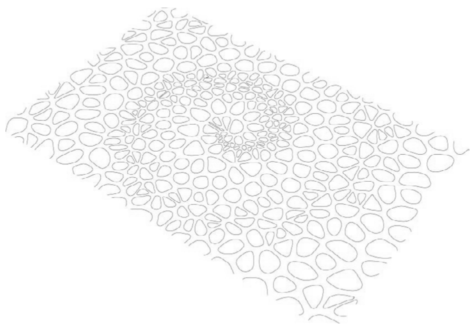


Fig 9.- Mashrabiya Pattern - Cell Arrangement

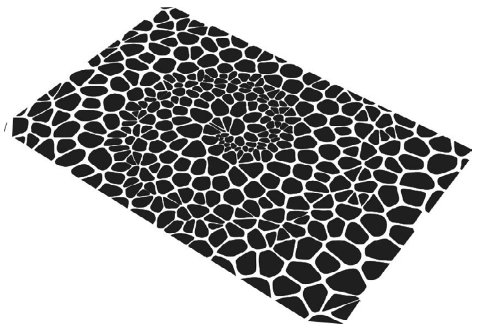


Fig 10.- Mashrabiya Pattern - Nolli

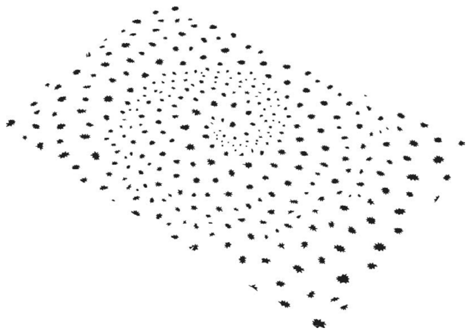


Fig 11.- Mashrabiya Pattern - Pore Arrangement

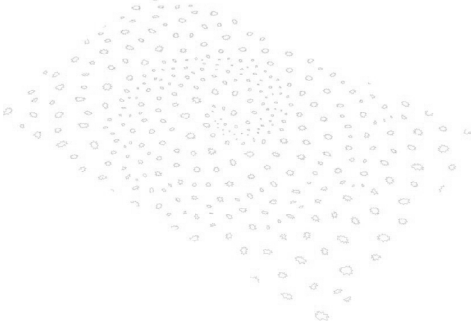


Fig 12.- Mashrabiya Pattern - Open Pore

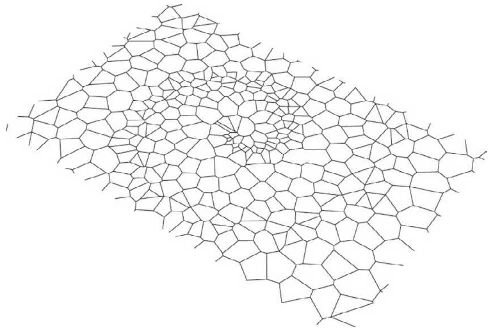


Fig 13.- Mashrabiya Pattern - Voronoi Diagram

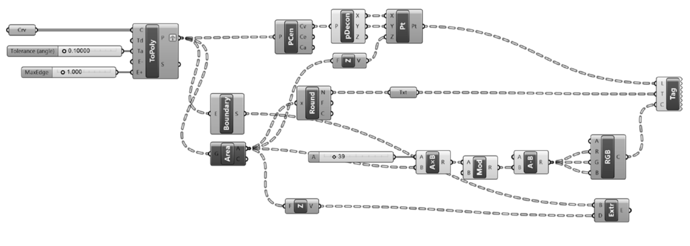


Fig 14.- Grasshopper Analysis Script

cell structure and organisation of contiguous biological entities solely at a macro-scale. To further understand the system and truly see that it is Fibonacci-Inspired, I felt it necessary to do some further analysis of cell area. Again, this was made quite quickly with another Grasshopper script result(Fig. 15 - below) and the result is significant. When I display the areas in relation to each other, it's pretty clear the system does in fact grow based on a similar system documented by D'arcy Thompson in his discussion of the Formaninifera(Thompson 1951, 850). The spine of the pattern increasing as in a logarithmic spiral and with an increase in area based roughly on the addition of areas of the cells around them.

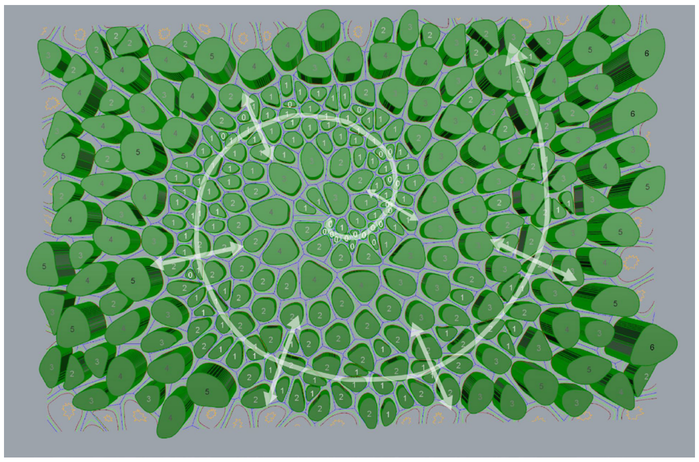


Fig. 15 - Mishrabiya Pattern - Size & Arrangement Analysis

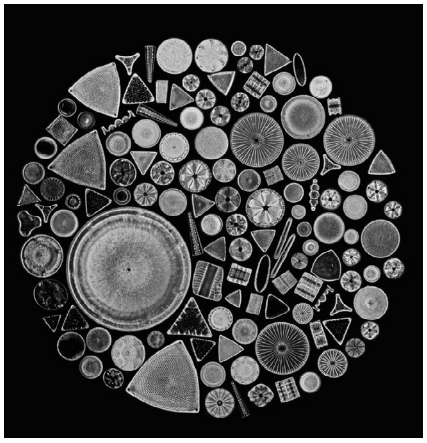


Fig. 16 - Victorian Diatom Arrangement

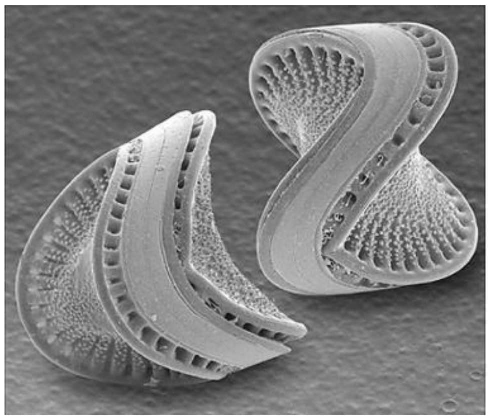


Fig.17 - Campylodiscus hibernicus

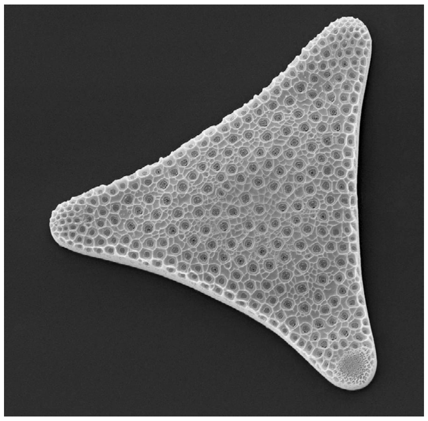


Fig. 18 - Triangular Diatom Sem Image

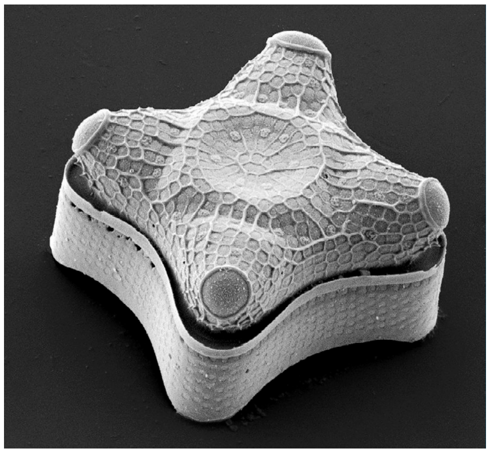


Fig. 19 Diatom Sem Image

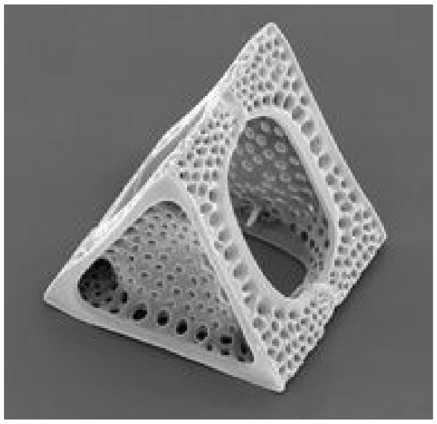


Fig. 20 - Diatom Triceratium SEM

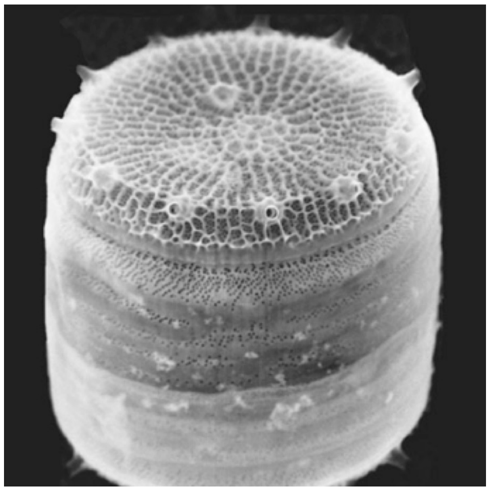


Fig. 21 - Diatom SEM Thalassiosira

The Diatom

The Diatom is a very tiny(2-500μm) single-celled organism, and a boon to researchers on a number of fronts. This form of aquatic phytoplankton(algae) is not strictly plant or animal but borrows characteristics from both, and is relatively new on the evolutionary time-scale(Stoermer and Smol, J P (John P) 1999). Diatoms are filter-feeders but do also photosynthesise, thus their environment is generally near the surface where light is more prevalent. They're size and proliferation make them the ideal place to look when trying to identify environmental effects due to pollution. The Diatom's morphology is also of acute interest to nano-technologists as they hope to understand structural formation at a scale similar to silicon microchips... in an effort to build nano-machines for a number of tangible benefits at that scale(Bradbury 2004, e306).

These organisms protect themselves by creating a silica-based porous shell shaped like a drum(De Stefano 2005, 15-24). The frustule(or valve), located at the ends of the form, come in many shapes and sizes but always provide the same function, permeability and protection for controlled fluid flow(supporting nutrient exchange). The side of the diatom is called the "girdle" region and its function is to support the rest of the structure as the organism grows and it may show some perforation to support flow.

This investigation is to focus specifically on the morphology of the frustule seen in the cocconeidacean valvocopulae(valve or end wall). Unlike our son and his Trypophobia, I am intrigued by the layered distribution of silica in this form. While I can't expect to fully understand biological processes that are not yet even understood themselves by botanists(De Stefano 2005, 15-24), I feel I can reasonably mimic what I see algorithmically. My goal is to get a "sense" of the valve's organisation and recreate a model from what I experience.

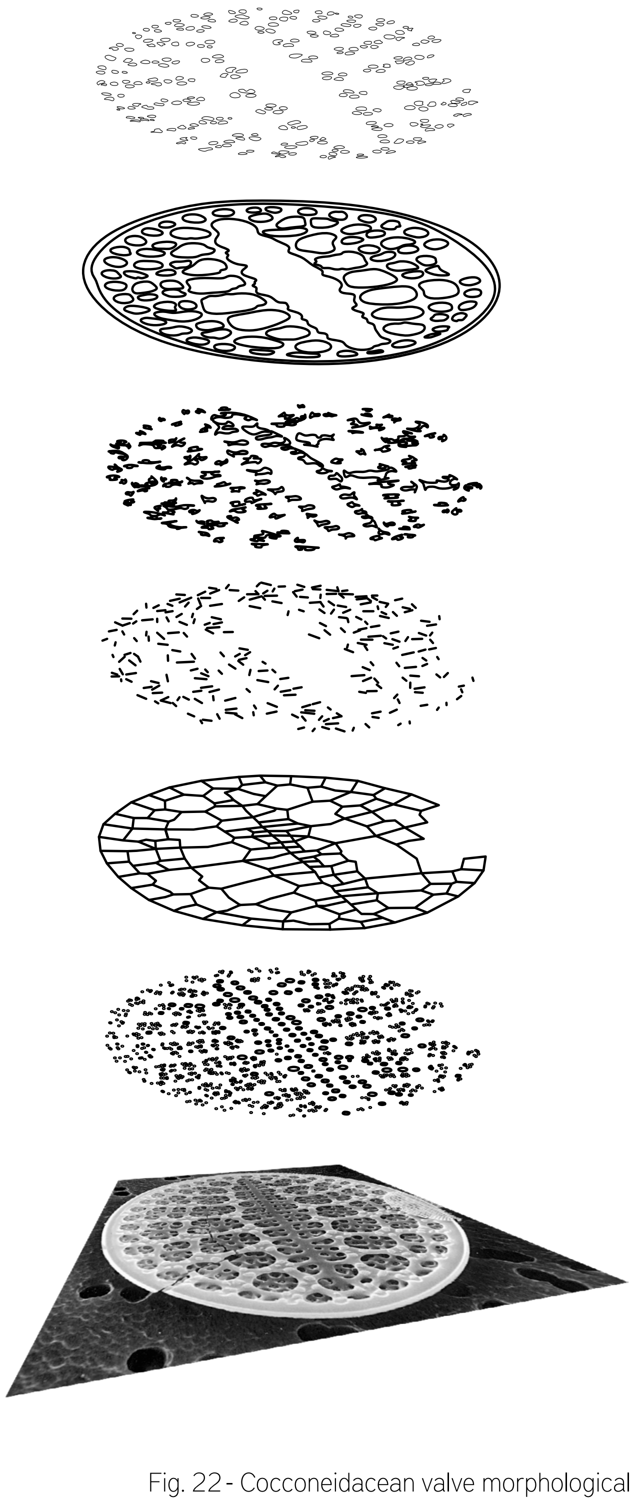


Fig. 22 - Cocconeidacean valve morphological decomposition

Analysis

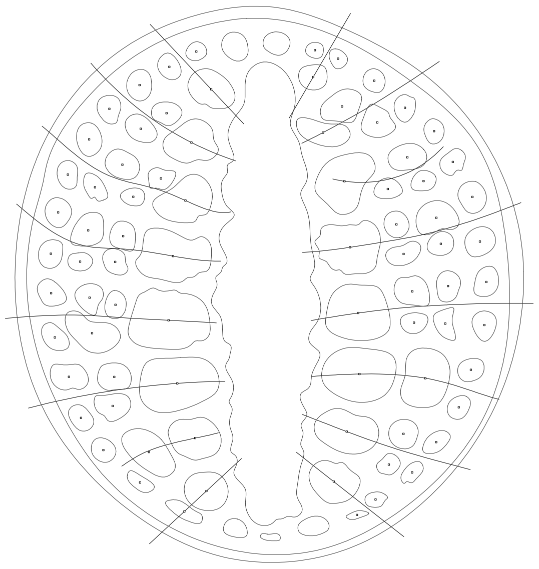
I felt, as in the Oxman Mashrabiya, that a breakdown of the valve plate detail might lead to some basic formal understanding. Because it's a three-dimensional structure, I needed to be careful when dissecting the scan; some of the lower layering was out of view and I found it easy to miss-categorize some of my observations due to complexity of the structure.

Grasshopper again became an invaluable tool. I identified and traced the solid paths of silica and reduced what I had seen into various levels of similarity. The level I was most interested in was basically the interior, or top-most visible layer. This the location of most flow and much of what was left was silica structure, even only in two-dimensions. With the holes defined, I could develop a form by plotting their position, size and frequency as long as I was able to discover their organisational relationships, as encoded by the diatom DNA and in response to this organism's local environment.

While loosely organised radially, this diatom species frustule has obvious bilateral symmetry with the mirror axis being vertical. The roughly spherical openings(pores) change in size both in radial distance from a centre and from the axis symmetry. I would classify this a pennate, but lightly so.

It's interesting to note, that when the areas of radiating pores is measured, the morphological relationship of radiating pores along a growth line(from inside to out) seems very much one of exponential decay. Visually the size of the pore decrease exponentially the further out from the centre/axes of symmetry for the diatom. Grasshopper to the rescue here; I wrote a script to evaluate the areas of the radiating pores and plotted the result in the diagram(opposite). The area histograms, while they do vary in scale with the pore arrangement, confirm my suggested observation that the sizes do decay exponentially(D'Arcy Thompson would be proud). I took what I consider the "cleanest" graphs to represent the phenomena appearing closest to the meridian of the diatom. The others, while representative, provide less pores and subsequently offer more chance for error. Overall, I believe the relationship is sound.

While this might be a stretch, another basic assumption from the morphology I have made is that as the pores reduce in size, their number increases by a factor of 2. That is, their number grows exponentially the further out from the combined central/vertical symmetric axis. It appears to me that the pores themselves multiply as a cell would the further out they appear. This sample doesn't have many radiating



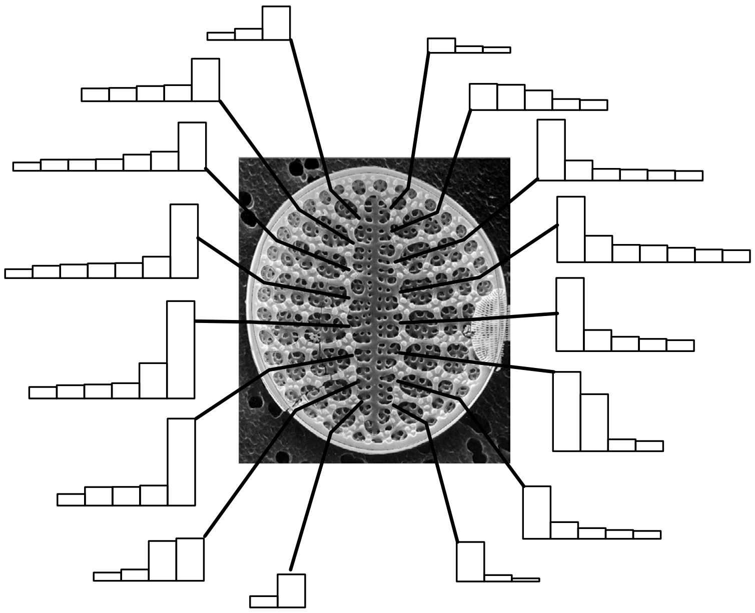


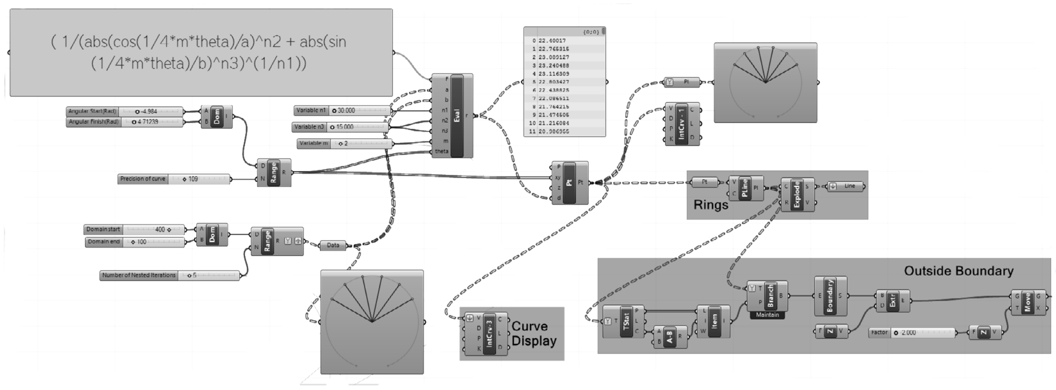
Fig. 23 - Diagram - Analysis of pore structure hierarchy

cycles(possibly 3-4) but it is my feeling that this is the case with the limited visual data available and I'm going to proceed forward with this reaching assumption as see what might come from it.

So I will be moving forward with a handful of assumptions; the organism is organised symmetrically about a centre(radial) and has some bilateral symmetry(mirrors using a vertical axis), the pores increase in number exponentially based on the distance from the centre/axis and decrease in area from the same reference. This is one slice of a very complex, but well documented, system that is "in motion." The SEM image has captured the silica structure at a point within the diatom's life cycle, so this may represent only an intermediate state of its response to environment. Also , visually, the silica aggregate seems somewhat smooth and uniform having minimal discontinuity within the valve, and with portions of the system connecting to a lower layer in the same manner.

$$r(\varphi) = \left(\left| \frac{\cos\left(\frac{m\varphi}{4}\right)}{a} \right|^{n2} + \left| \frac{\sin\left(\frac{m\varphi}{4}\right)}{b} \right|^{n3} \right)^{-\frac{1}{n1}}$$

Fig. 24 - SuperFormula
Johan Gielis



```
( 1/(abs(cos(1/4*m*theta)/a)^n2 + abs(sin
(1/4*m*theta)/b)^n3)^(1/n1))
```

```
{0;0}
0 22.40017
1 22.766316
2 23.009127
3 23.240488
4 23.116309
5 22.803427
6 22.438825
7 22.086511
8 21.764215
9 21.474606
10 21.216084
11 20.906955
```

Fig. 25 - Superformula Grasshopper Script

Encoding - Superformula & Iterations

Reaching in back through to D'Arcy Thompson again, many of forms of life he studied were at their core based on variations on processes using polar or spherical symmetries. The circle (and harmonics based on the circle) can be seen in many plants and I believe that this geometric organisational principle plays a substantial role in the morphology of the Diatom.

In an effort to find a unified Cartesian method for generating a number of basic shapes that appear in nature(circles, squares, triangles), mathematician Gabriel Lamé developed a formula(Lamé 1818) based on the Cartesian formula for the ellipse. His creation, the Superellipse, can additively generate these basic shapes through the parametrization of exponent(n) on each axis(a,b) respectively and is an extension of the basic ellipse formula in Cartesian space. This was taken further by extending Lamé's work and disengaged the axes in an effort to extend the function's abilities to generate distinct form(allowing for polar harmonics)(Gielis 2003, 333). Geilis' addition of a "mode" for the function in radial form further extended the function to discriminate based on the type of symmetry needed.

$$\left|\frac{x}{a}\right|^{2} + \left|\frac{y}{b}\right|^{2} = 1 \longrightarrow \left|\frac{x}{a}\right|^{n} + \left|\frac{y}{b}\right|^{n} = 1 \longrightarrow \left|\frac{x}{a}\right|^{m} + \left|\frac{y}{b}\right|^{n} = 1$$

Cartesian Ellipse Formula

Superellipse Formula - Lamé

SuperFormula - Gielis Cartesian Form

Fig. 26 - Superformula Evolution

To better understand Gielis' formula, I felt that a grasshopper script could be generated to produce a sense of the range of shapes produced by altering the symmetric mode(m) and by testing values for axes exponents(n2,n3). Changing parameter "n1" had little or no change on the output other than scale so I left it pretty much alone. I wanted symmetric output, so I left the axes equivalent to one another(a=b) for this initial exploration. Rhino-Grasshopper allows for a real-time instrumentation of the Formula. The script-system resolves itself whenever the parameters are changed, producing sets of solutions simultaneously.

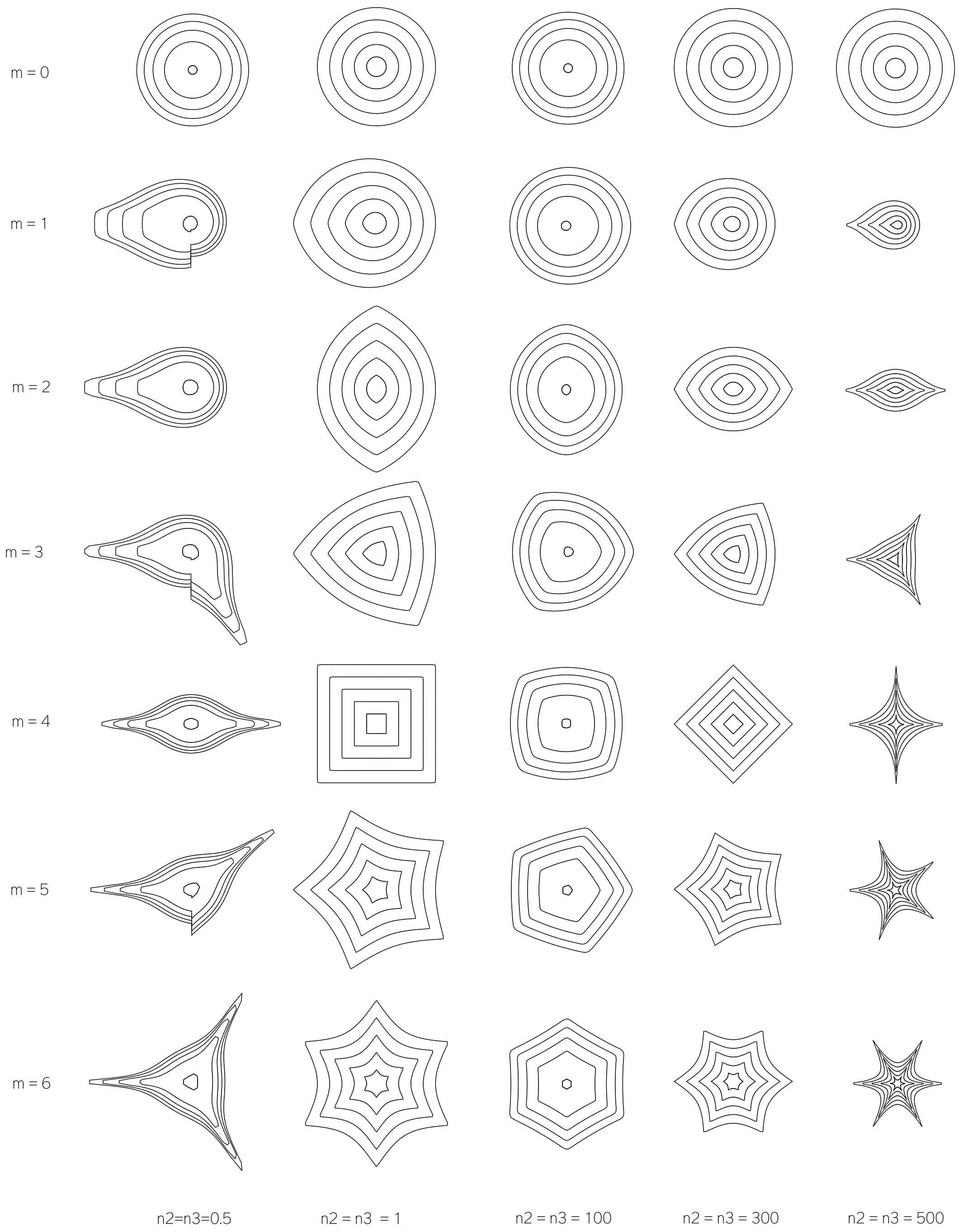


Fig. 27 - Superformula Iteration & Algorithmic Flex

As seen opposite, the output for n1/n2 equal to one or greater results in useful radially symmetric form. The system reduces to essentially a group of nested circles at all values of m=0(reducing the equation to that of a simple circle) for all angles in zero to two-Pi.

For exponent values less than zero(n2=n3=0.5) the formula produces somewhat asymmetric cases with discontinuities for odd values of m. This needs further study to truly resolve whether the cause is my implementation or the formula itself. It seems quite odd to me that the system is not differentiable at polar zero. At this point I would suspect that my script does not take into account a boundary condition correctly.

The special case at m=1. The system is symmetric about one shortened axis but the result is that it is symmetric about two, as if the axes were extended infinitely. The curves themselves extend out in the direction of the axis, resulting in an overall asymmetric shape. This case is interesting in that it is as if symmetry is “pushing” its way into existence in these trials.

In trials m=2-6 the curves exhibit the type of behaviour I can use. These cases reflect the macro-organisation of forms seen in many types of diatom. I can extrapolate to see how radial symmetry can develop using Gielis’ formula for even higher-ordered species. In the case of the Cocconeidacean valve, I think m=2 and n2=n3=1is the closest match. This configuration is a good place to begin when developing a morphological generative model for the form.

Encoding

Extending the grasshopper model with these parameters will initially allow for the formation of a scaffold that I believe best represents the regions where silica is deposited as a structural network. If I treat the output this way, I can further-develop the form to better represent the flowing mesh-based surfaces seen in the scan.

Graphic scripts in Rhino-Grasshopper are interpreted by the system at all times during the creation process. If you place a component on the work surface and connect it to sources of data or other portions of an already existing script, data will flow to the new component and the entire script will resolve to a new set of states. This development process can be relatively fast in the beginning, but as the system grows in complexity, resolution times can drop off quickly. It’s possible for the system to essentially lock-up and become unresponsive as it plugs through the work at

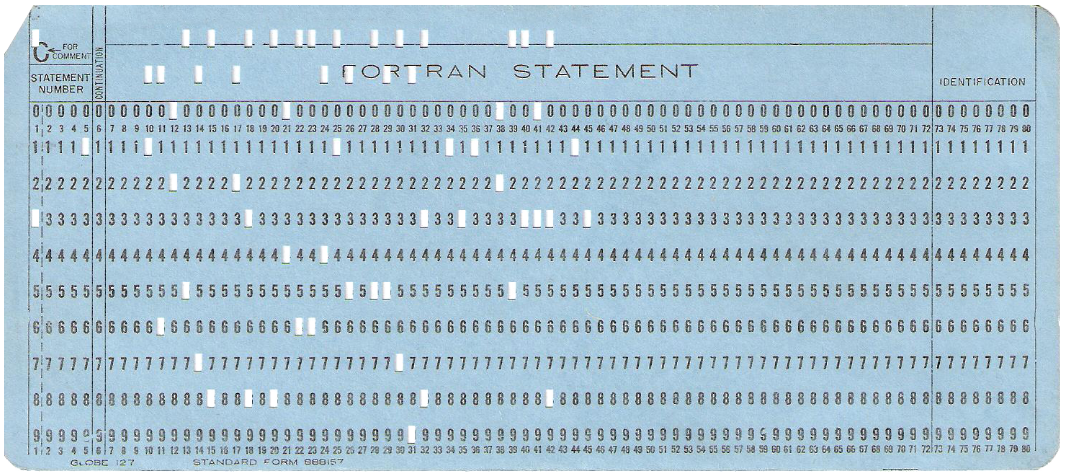


Fig. 28 - FORTRAN Punch Card - Joseph Huffman

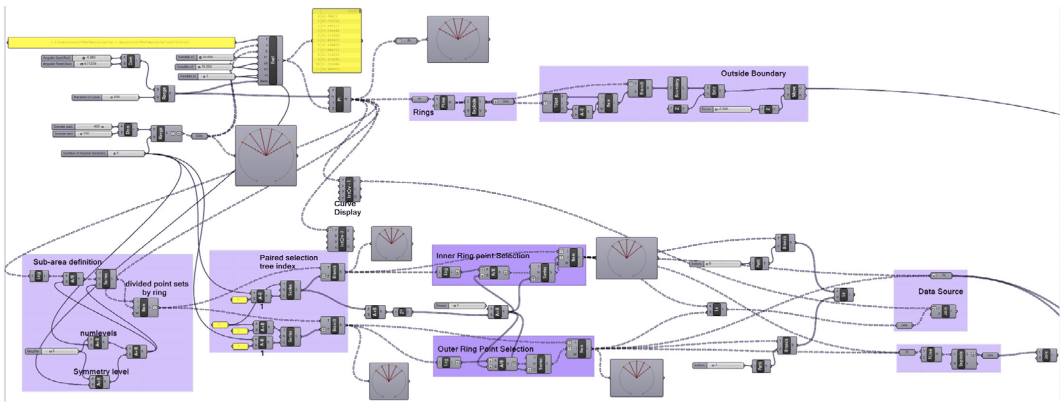

Outside Boundary
Rings
Curve Display
Sub-area definition
divided point sets by ring
numlevels
Symmetry level
Paired selection tree index
Inner Ring point Selection
Outer Ring Point Selection
Data Source

Fig. 29 - Superformula Grasshopper Script - wb

hand.

Being aware of the complexity of the developing algorithm is inherent in any software developer's toolbox, and establishing that sensitivity for a growing algorithm can be difficult. My first hands-on experience with computers was back in 1979. Our high school was a relatively new building designed in the late 60s and someone had the fore-thought to establish a good cooperative relationship with IBM at the time. They had built a section of the school to support one of their early small mainframes. The IBM model 1130 offered some text-based input at the console but mainly took its input through a card reader, and all output was printed. It had eight kilobytes of Core memory and sported a 1 Megabyte hard disk system.

I know I'm dating my self here, but I remember the excitement of sitting at the keypunch, copying my instructions and data over to cards at those consoles, one punch at a time. We were never allowed to program "on the fly," as our teacher felt that this only produced lazy, inefficient code(and programmers). Algorithms had to be documented in pseudo-code first(flow-charted), then translated to FORTRAN statements, then to cards. The process could easily take a few days in preparation for a "run."

Because the mainframe was right next door, we could walk our code over when we wanted and submit it as part of the hourly processing run; If we were lucky, we might be able to get a run or two in a day... Or if they were generous, the operators would let us run them ourselves. This was a privilege not easily obtained. Trust was hard to acquire because this was serious business... Data Processing was a career path that had a hardness to it that's difficult to describe. We sweated "up-time" even back then. The fallout of screwing around might have meant a down machine and that meant less throughput. We learned early-on the mantra of information technologists everywhere; uptime and access and throughput is everything. I can't imagine the other students, at other schools in the region, who got one run per week - board couriers moving card boxes on Sundays - endless stacks of tractor-paper - bubble cards.

Even back then, If you didn't think your algorithms through, it might take more time than was necessary to do the work at hand. Approval of the operators was paramount to access...so you thought-through and developed a feeling for the complexity of your work. Otherwise, your job would be dropped if it was thought to be "spinning" as your "cpu clicks" counter rolled over, and you might not be given the access you wanted later, just when you really needed it.

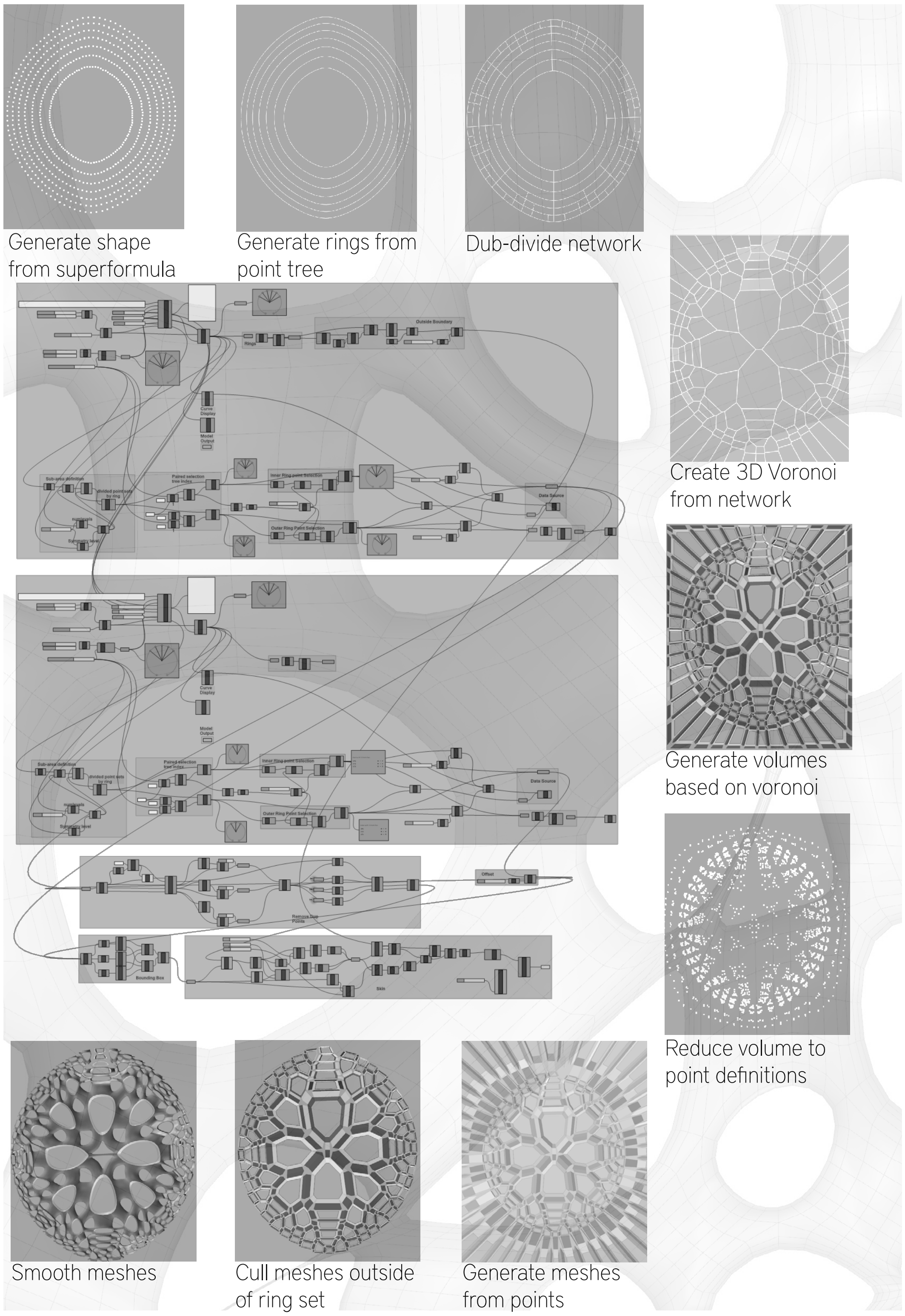

Generate shape from superformula
Generate rings from point tree
Dub-divide network
Create 3D Voronoi from network
Generate volumes based on voronoi
Reduce volume to point definitions
Smooth meshes
Cull meshes outside of ring set
Generate meshes from points

Fig. 30 - Diatom Form Algorithm and Evolution Sequence

Today, every change to the my grasshopper scripts incurs a cost in time. I get anxious when the change produces a perceived lock-up; my fingers hovering over the Ctrl-"c" keys so that the issue may not kill the whole system if my error in judgement causes all available resources to be taken. After a while I calm down, when the systems begins to finally behave predictably; Change.....wait five seconds... screen update; change.... wait 10 seconds...screen update; change....wait 20 seconds....ctrl-"c"...<esc>...<esc>...."whew," its back; won't do things that way again. Rinse and repeat.

My algorithm first begins with the Gielis framework chosen, and divides each concentric iteration into a set of resolvable points. A resolution greater than 60 seems to produce some of the best results. I quickly learned that if I chose a value over 300, the machine was never coming back from whatever hole it was dropping into, so I kept my number closer to 120 for sanity's sake.

After points comes lines. A network is produced that is further subdivided exponentially due to my assumptions during analysis. This portion of the algorithm simply adds more differentiating as the iterations extend outward. My goal was to essentially develop a loose mesh over this network and the best method I could find involves our old friend from Neri Oxman's Mashrabiya, the Voronoi network.

I believed that if I could generate a sub-network joined from the centres of each contained area from my generated scaffold network, I could build a set of areas that uniformly span my original network. Now, this is where things really get tricky... I found that if I offset my area boundaries in the negative direction by 5% or more, I would end up with a set of areas that are smaller than the first, but still centred by their original centroids. If I convert these boundaries to points, and duplicate this set of points and offset them in the Z-direction a bit, I have the framework for a tight mesh system.

At this point all I needed to do was remove any duplicate points generated in the process, generate the surface meshes using the Weaverbird plug-in, cull all outside meshes due to the Voronoi process and then smooth using the CatMulClark Weaverbird component to the get smoothness necessary.

Each iteration(and there were thousands during design) took anywhere from ten seconds to ten-plus minutes to resolve, thus turning this into a very painful process that could benefit from optimization. As an interpreted programming language, Grasshopper scripts are just that; scripts that run on top of a framework of code that is handled on-the-fly so its compiled optimization can't benefit from any look-

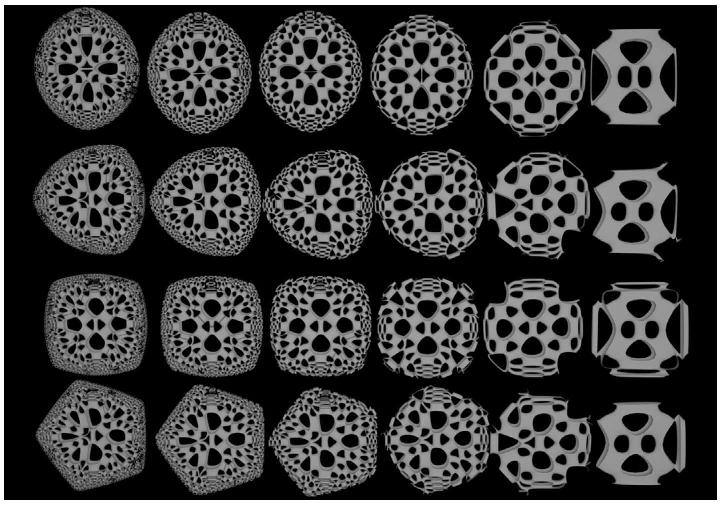


Fig. 31- Complexity and Symmetry Flex

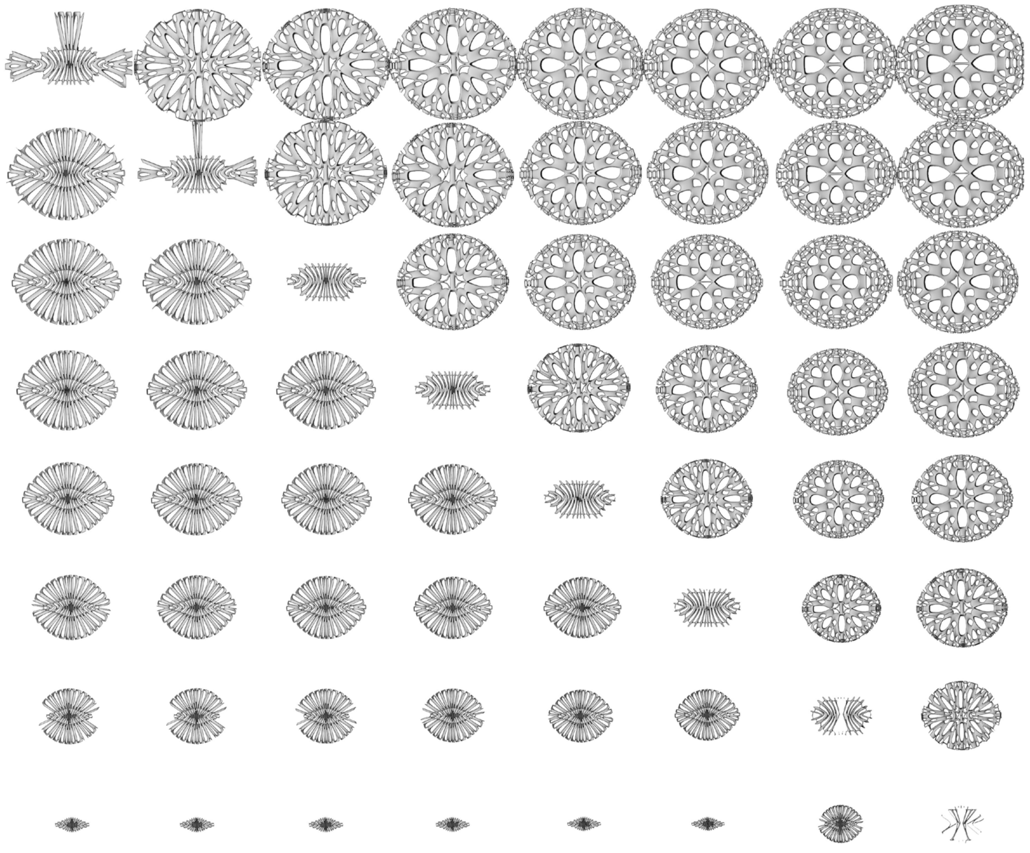


Fig. 32- Representational Flex - Diatom Model Domain(a|b)

ahead or historical demand the algorithm presents. If the algorithms had been transferred into a compiled language(C, Java...etc), I could expect two orders of magnitude of time improvement.

Further to this, when the number of iterations is increased, the resultant output behaves as expected(Complexity and Symmetry Flex - Fig. 31 opposite). The time to generate each more complex form increases exponentially to the point of me questioning whether the process will complete(ten-plus minutes). The algorithm did produce form that was expected for each symmetric mode and I'm quite satisfied with the result. While far from exact, the model's output did produce forms that were structurally similar to the valve's morphology; symmetrically arranged voids that mimic an aspect of the materiality of the silica.

If the algorithm is flexed further however, by altering both the domain of the major and minor axes(Representational Flex (a|b) - Fig 32 Opposite), some fairly nonsensical results can be created. The resultant forms seem to fold in on themselves and do not vary much other than in scale. In my view, the results aren't as important as the method. I think it's necessary to explore all the parameters without any pretence or expectation. Seeded by my impressions of the diatom's form and symmetric capability, the flexing of this generative system offers a wealth of results that may prove useful anyway.

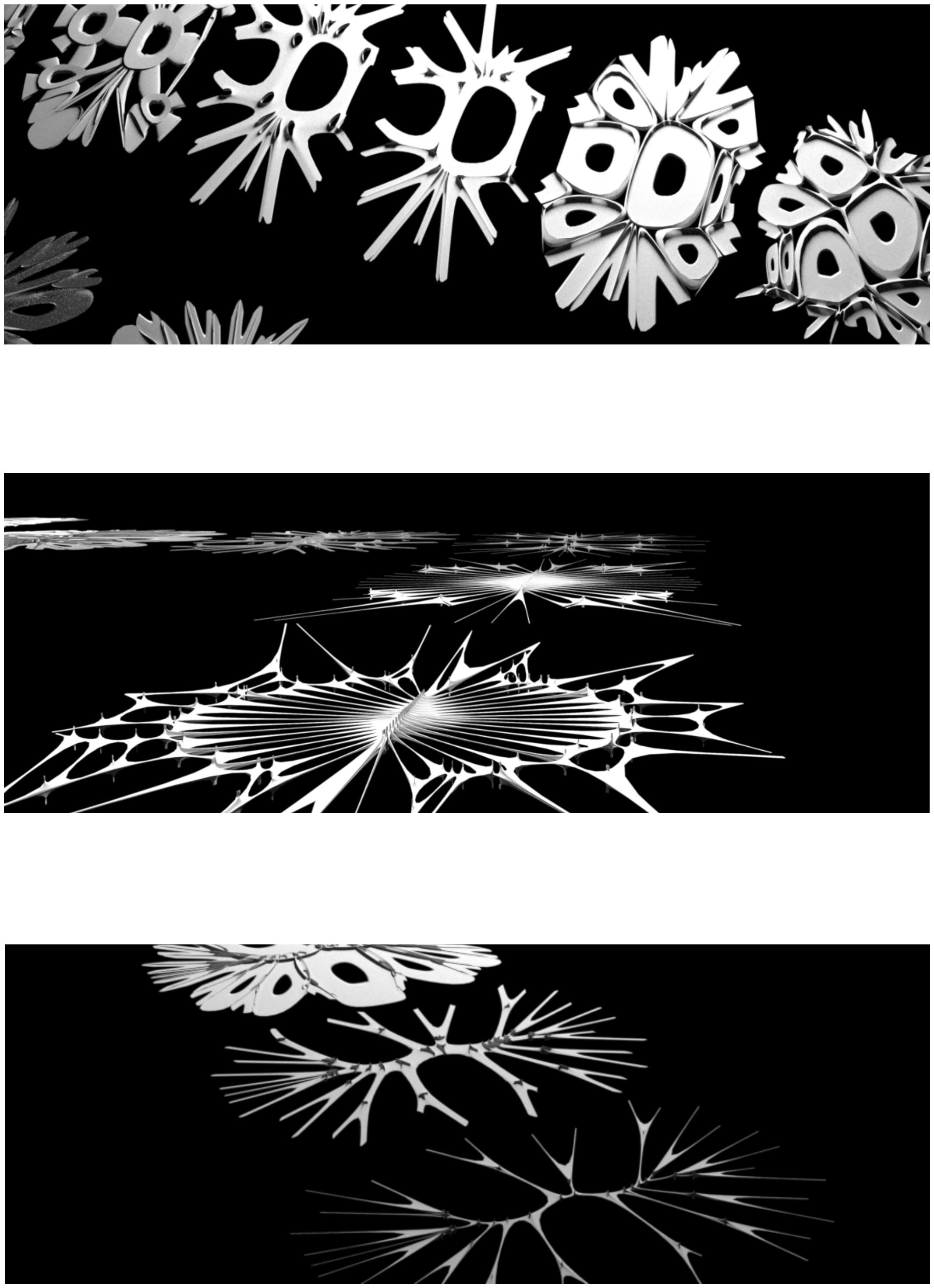


Fig. 33 - Flex of Diatom Representational Algorithms

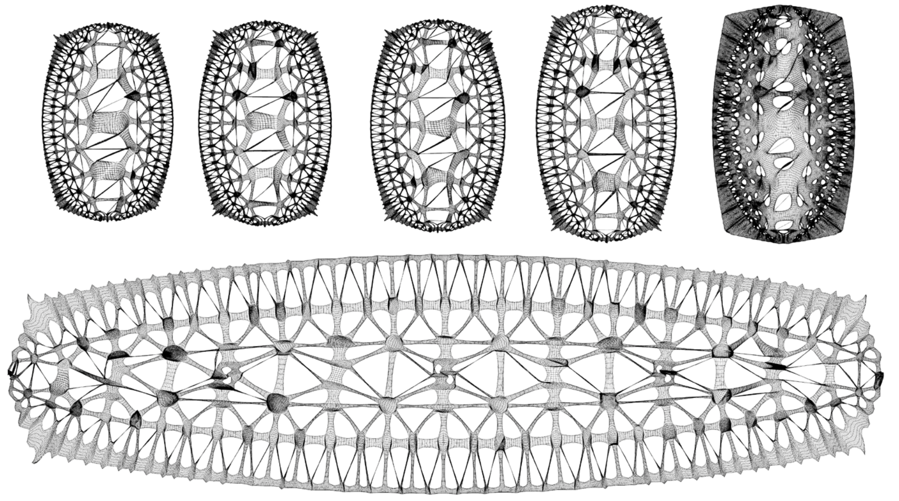


Fig. 34 - Standard Diatom Model Iterations - Size and Iterative Depth Flex

Appearance - Representation & Flex

Satisfied that I can mimic a portion of the diatom's perceived organisation, this investigation becomes more interesting when I flex its representational logic and not just the parameterisation surrounding symmetry. If you recall, I exploited a voronoi process to create the mesh that inevitably generated the form. While developing this method, many mistakes were made and this resulted in some surprising formal directions(seen opposite).

It's one thing to generate form in a vein of what you expect but quite another to discover a strange and wonderful new morphology. There are nine parameters in form generation using the Superformula, plus a further four used in final representation(mesh generation and smoothing). When I flex the latter, the existing diatom model becomes quite distorted, and in transformative ways that are reminiscent of some of D'Arcy Thompson's observations within related species and his Transformations of Related Forms.

Taken to extremes, each investigation reveals wonderful variance within an organic base set of encoding. It's as if I can almost evolve form myself in a relatively expedient way(ok, each took a while, but not as long as in true evolution). Some can be quite macabre, others I find very intriguing; I like the freedom they represent. Our son, of course, finds them all very disturbing...

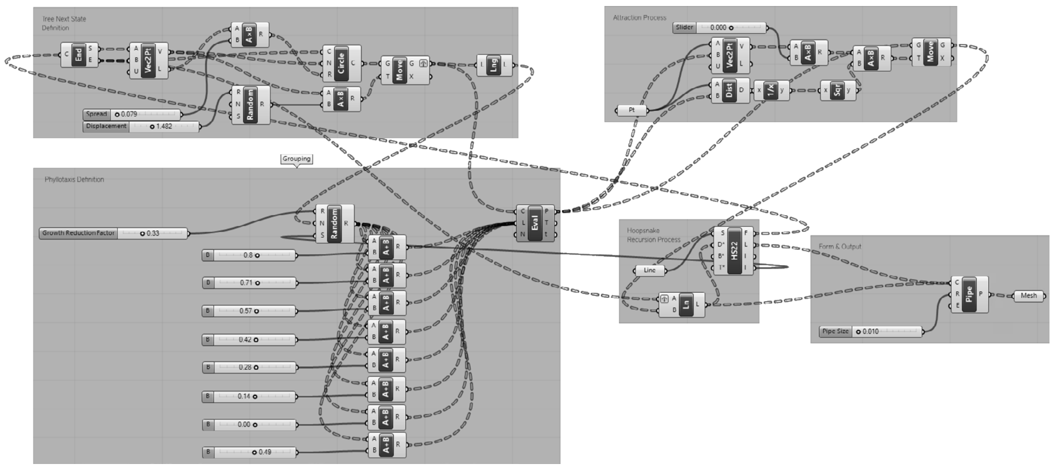


Fig. 35 - Grasshopper L-System Script

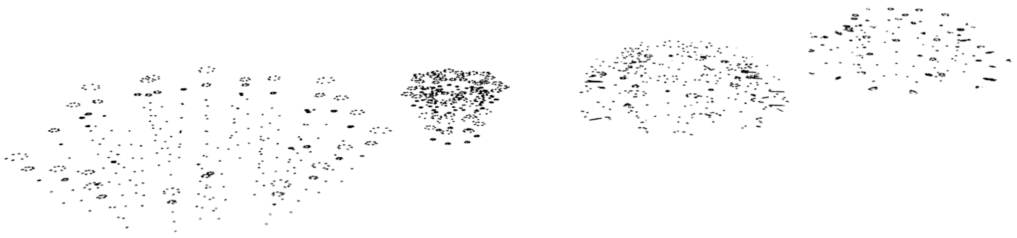


Fig. 36 - L-Systems Iterations - Perspective

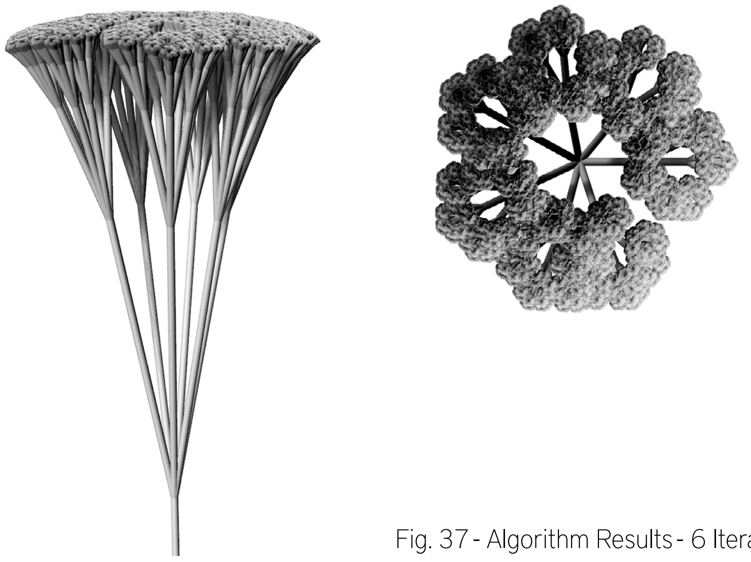


Fig. 37 - Algorithm Results - 6 Iterations Renders

Arrangement - L-System

Digging a bit deeper into the morphology of the Cocconeidacean diatom frustule, I've been thinking so far in terms of the placement of voids in a uniform silica matrix. This method has allowed me use a few boolean tricks to see the form in a subtractive way and has had positive but limited success. I'm not so sure, however, that life encodes necessarily in terms of what is *not* there but more so in terms of what *is* there(additive over subtractive).

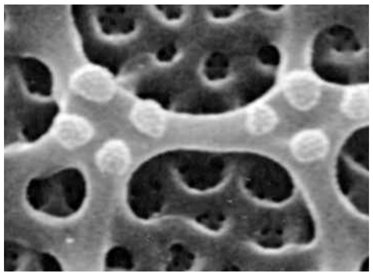
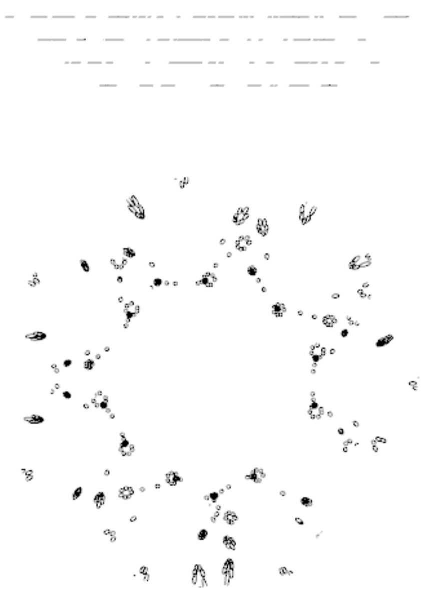


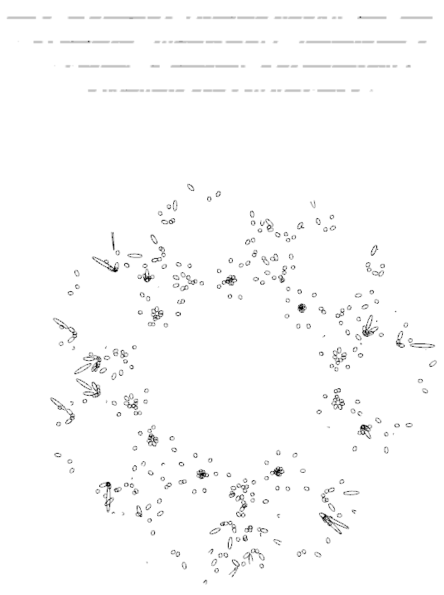
Fig. 38 - SEM Frustule Papillae arrangement

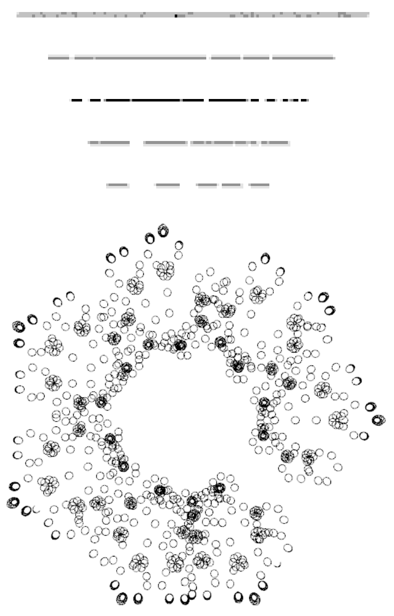
Diatom Papillae are sites of silica production in the diatom valve. In this area, metabolic processes draw silica ions from surrounding solution and activate their deposition in an encoded response to environment(Cox, Willis, and Bentley 2012, 450-459). Unsure of how this could actually be performed, I moved to construct a simple encoding experiment to see if the resultant form might resemble our frustule in some way.

My first thought was to look at the process of phyllotaxis in plants for a theoretical framework to build from. Centric diatoms must start at a "centre" and grow outward in some manner(much like many other plant-like species); possibly an L-systems approach might be of use. If I interrupt the process and take portions of it(slices) at significant moments, it might be possible to see the frustule valve iterations "grow" from these moments.

Lindenmayer-Systems, known as L-Systems, are methods of describing encoding through the use of a formal grammar involving "rule rewriting" at each growth stage and their use was first developed by an Hungarian theoretical biologist named Aristid Lindenmayer (Prusinkiewicz 1996) in 1968. Lindenmayer suggests that a plant grows using a simple recursive algorithm which is applied at each growth node as it responds to its environment. In each iteration these "loose" instructions rewrite their response to the local environment either by applying the same rules or altering them by ratcheting up/down counters controlling growth


Trial One - Slices in Elev + Plan Result


Trial Two - Slices in Elev + Plan Result


Trial Three - Slices in Elev + Plan Result

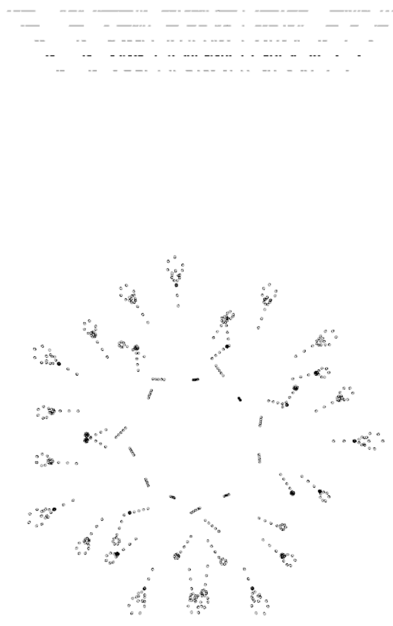

Trial Four- Slices in Elev + Plan Result

Fig. 39 - L-Systems Algorithm Flex Iterations

paths or other development characteristics. And all this is using a handful of simple instructions.

I chose to use Grasshopper again because it does support recursion through its "Hoopsnake" component add-on. It took a bit of time to get used to how this is implemented(remember Grasshopper is always trying to resolve a solution). Ideally there as to be a set of variables that are passed from one iteration to another, some are for control, others define locality of action. I wrestled with the mechanisms of how to control each iteration of execution more than the data and instructions of "making."

My algorithm is simple. For each point in a set of points:

> Find a random "next"point(within an allowable growth distance toward a "goal")
> Draw a circle with a specific radius dependent upon iteration count at this point
> Pick 8 randomized points on this circle
> Draw a line from my current iteration to each new point on the circle
> Apply this algorithm to each new point(passing "next" position to now be "current")

The Grasshopper Hoopsnake component halts execution at each iteration. I did not put a limit on how much recursion was allowed because this control was available by default. I would normally include a counter to limit this but running it in single steps was sufficient for my needs. A runaway recursive process will kill any machine as it gobbles up all available resources and cpu... and this would break most of the IT tenets I mentioned before(uptime, availability and definitely throughput). If I let it run freely, it will most definitely lock up the machine and any unsaved changes would likely be lost. I have a handle on this one.[Always follow the Tenets! Maintaining availability requires backing up repeatedly. This one is so in-grained I hardly think about it anymore.] My workflow here consists of:

> Change algorithm
> Save
> Enable solver
> Run(step..)
> Disable solver
> Bake

The results(opposite) show some promise. Diatom Valves grow from inside to out-(Seckbach 2011)and this leads to the valve permeability becoming less and less as we get close to its outside layer. It makes sense that as papillae become more dense, their activity also becomes more dense, resulting in more silica deposition.

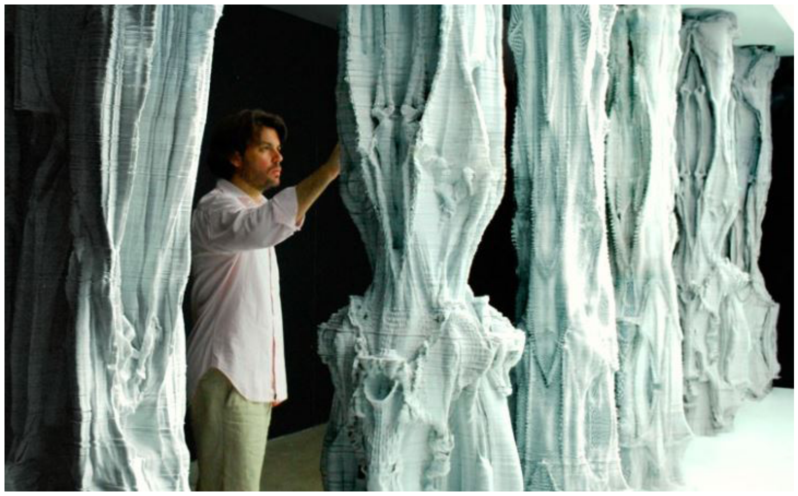


Fig. 40 - Folded Columns - Michael Hansmeyer

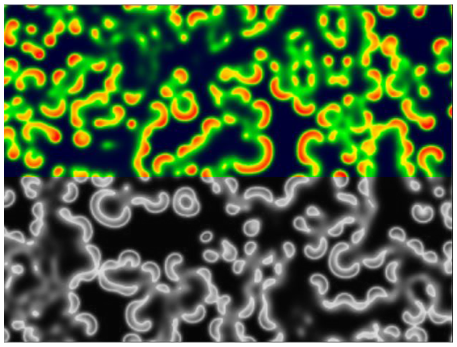


Fig. 41 Grey-Scott Reaction Diffusion Simulation(2D)

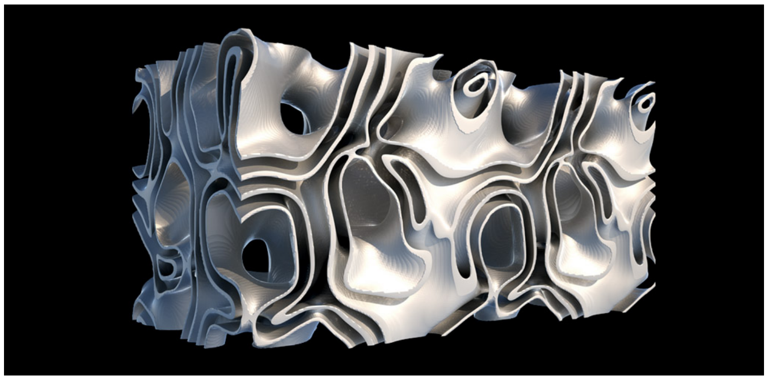


Fig. 42 - Grey-Scott Reaction Diffusion Voxelization - Michael Hansmeyer

Patterning here is defined by the use of a circle and by injecting some randomness into the algorithm at each iteration/level. This is a very rough model but can explain a portion of the morphology I see as I understand some of the processes at play.

Fluidity - Grey/Scott Reaction Diffusion

While trying to understand the somewhat alien forms seen in many diatom SEM images, it became clear that a major portion of the environment that this organism experiences was not being investigated. All diatoms exist within a freshwater or marine system, so water and nutrient flow is a major driver for the development of their morphology. What of the processes of concentration and flow?

Buried deep within my "images quod inspirare" directories is an area given to Michael Hansmeyer. His work on generative form has also taken its cues from organic life but in a more fundamental way than those of Oxman and Menges. In some of his more famous investigations, Hansmeyer applies topological "folding" to produce his macabre columns and grottoes(Fig. 40 opposite) when the folding process is taken to incorporate symmetry at the finest of scales. His digital grottoes foreshadow the birth of a new renaissance in form generation - and exciting because anything may be possible.

The flowing form generated by his Grey-Scott Reaction Diffusion investigation(opposite) is a voxelized view of the concentrations of reactants and products seen in the chemical reaction model simulation. First proposed by Alan Turing in A Diffusion Reaction Theory of Morphogenesis in Plants(Turing 1992), he suggests that as the laws of physical chemistry govern the concentrations of chemicals they then govern the resultant forms, not unlike the pure mathematical relationships seen in D'Arcy Thompson's work.

As a model for Grey-Scott Reaction diffusion reaction might embody this view in a more demonstrable sense; specifically two chemicals interacting at a rate defined by their relative concentrations, which obviously changes as the reaction produces products(and these products inhibit the reaction). This is the reason why zebras have their stripes(Liu and L. 2006, 011914) and also why diatoms of the same species have such varied forms. Their encoded response to their environment is at a macro level; leaving the reaction to sort out the details at a micro level(Cox 2010, 297-306).

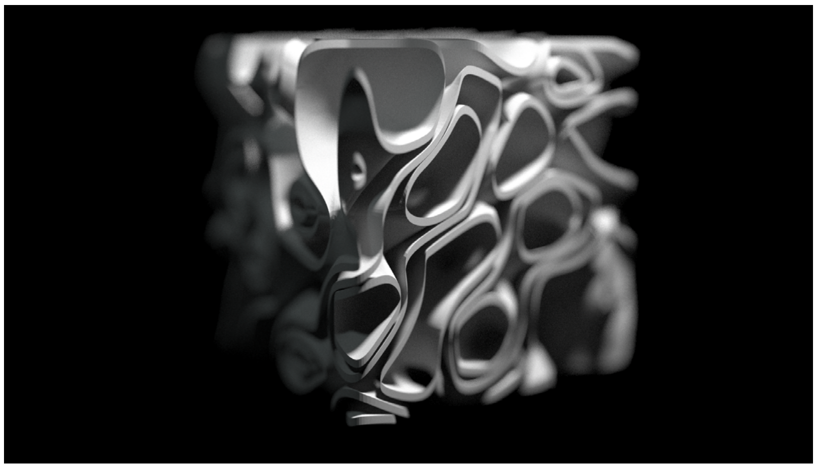
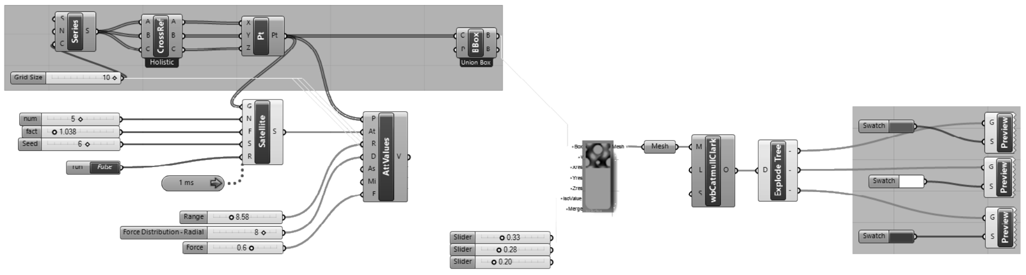


Fig 43 - Grasshopper Script Render



Grid Size 10
num 5
fact 1.038
Seed 6
run False
Range 8.58
Force Distribution - Radial 8
Force 0.6
Slider 0.33
Slider 0.28
Slider 0.20
Swatch
Swatch
Swatch

Fig. 44 - 3D Reaction Diffusion Mimic Script

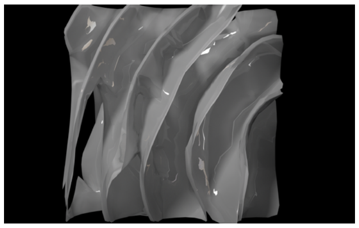


Fig. 45 - Form Section

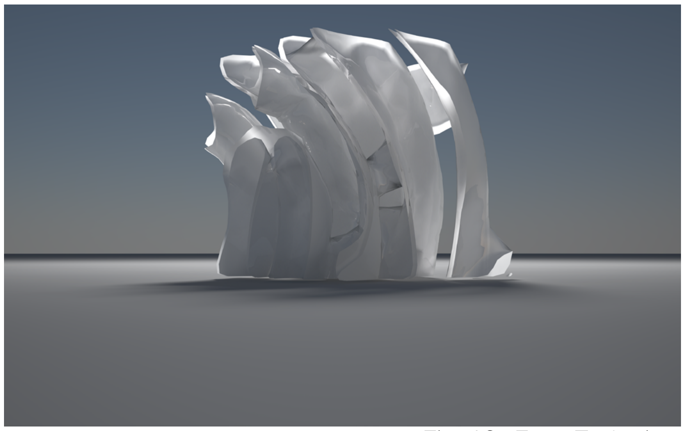


Fig. 46 - Form Tectonics

While Hansmeyer's algorithm involves a discrete 3D simulation of the Reaction-Diffusion model(plus a large voxelization), I felt that this would be a sizeable and overly complex process to duplicate. It may have been a stretch, but the form reminded me of the result of field interactions of three or more attractors within a linear field using the grasshopper "Nudibranch" plug-in(Tsiliakos and Egan 2013). A grasshopper script to model this system was easy enough to set up, and with repeated trials and adjustments to the parameterisation, a similar 3d structure was the result.

Form Algorithm:

```
Define a 3D Region
    Generate numerical series from Zero to Upperbound
    Duplicate series Two more times
    Create list of 3D points by mapping each list item to every other
Generate Three random Attractor points and locate them randomly outside the 3D Region
Measure the field strength at all points in defined region
Create potential surfaces
    For each set of strengths
        Create a set of 3D points experiencing a chosen strength
        Create and smooth iso-surface made from this set
For each iso-surface
    Extrude Surface in Z direction a defined Distance
```

The complexity of the algorithm is a significant issue(On^3+). Each iteration for a 10x10x10 (1000 point) region can take upwards of 30 seconds to resolve. Much of this comes from the iso-surface creation included within Nudibranch and the Weaverbird's CatmullClark smoothing plug-in(Piacentino 2009). Both are black boxes(equals not well documented) so it is very difficult to optimize this method.

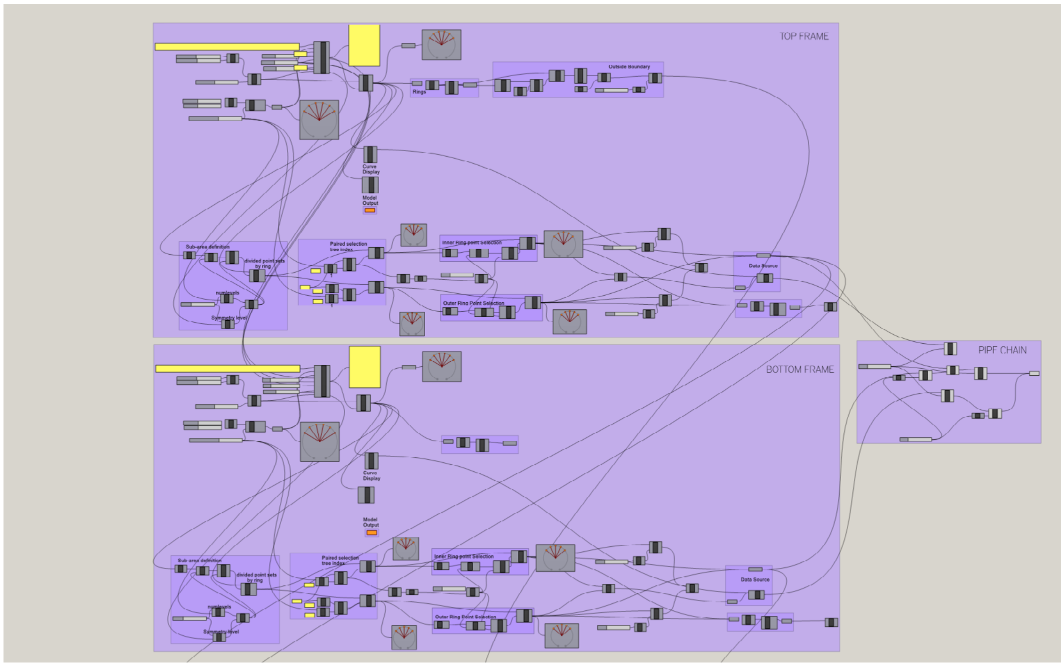


Fig. 47 - Scaffold - Grasshopper Script Result

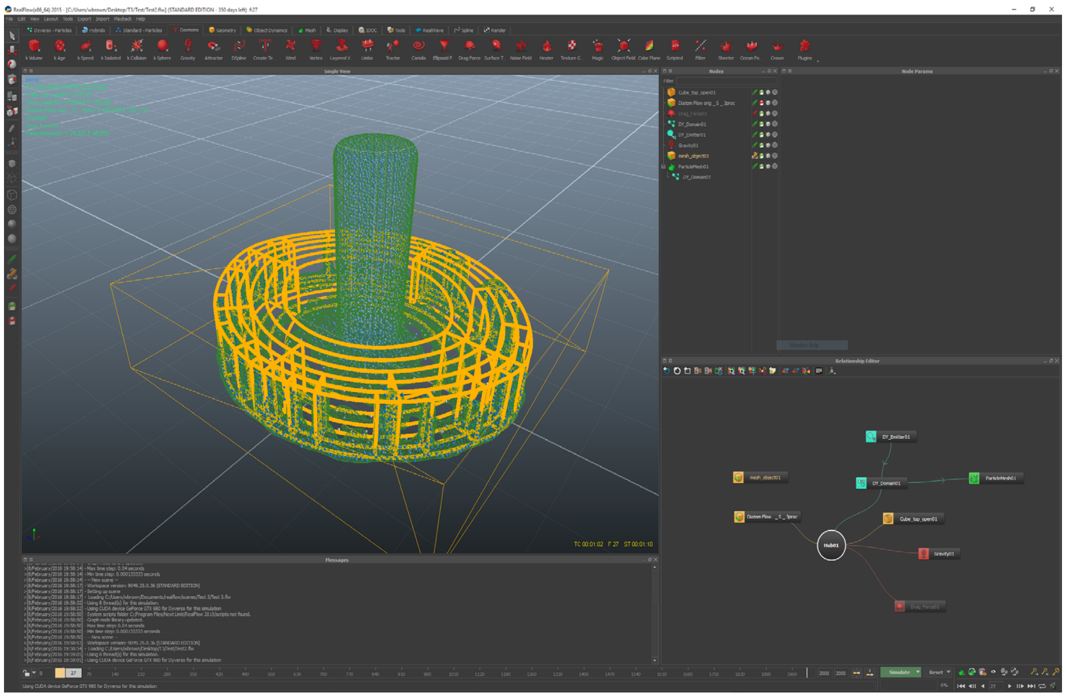


Fig. 48 - RealFlow Form Generation on Diatom Script Scaffold

Fluidity - Realflow Proof of Concept

Thinking additively, as in the L-Systems investigation of arrangement but at a larger scale, what if the formal system at play structurally in the Diatom is based on a set of pathways that form a network, or scaffolding, which produces and distributes the ions needed to bring dissolved silica from the surrounding environment?

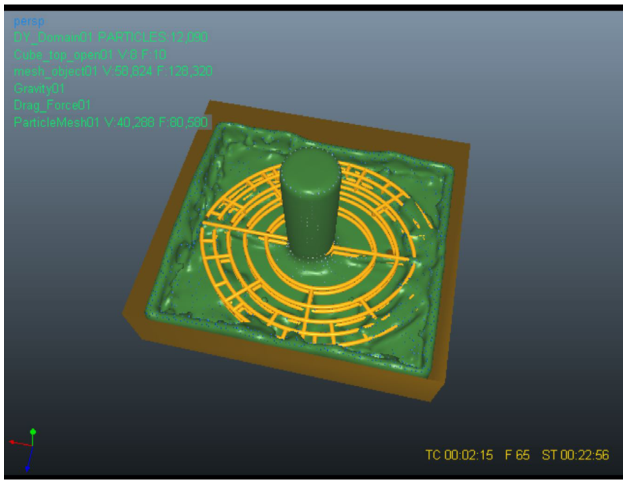
I would need to "grow" this framework using my existing grasshopper script and then give it some dimensionality. With this in place, facilitating the attraction of silica using fluid-based dynamic may produce a form likened to the diatom valve morphology.

After some research, doing this within grasshopper proved to be a dead-end. Fluid dynamics is possible within the software but it is far from interactive and even the simplest models and simulations would take more time than I have for this investigation... Enter Realflow(from NextLimit Technologies).

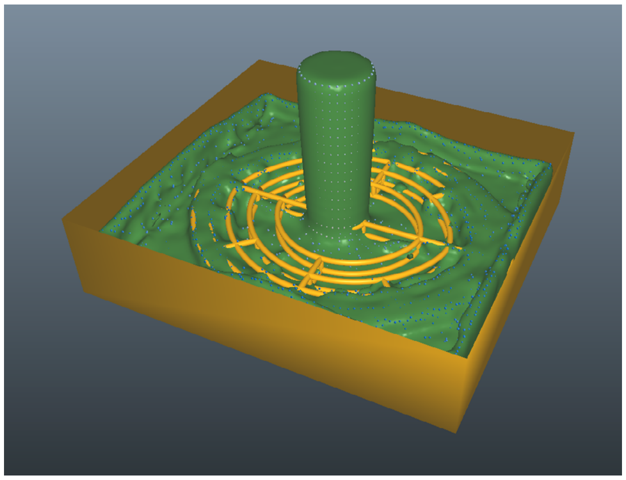
Made for the video industry, Realflow is a virtual particle simulator. It's capabilities work at many scales, and its physics engine is able to simulate with astonishing accuracy, many types of material interactions(granular, fluid, kinematic..etc.). It's workflows allow for the use of externally generated meshes(from grasshopper) and in this case, it's physics engine can utilize the GPU(a pair of Nvidia GTX980s) for hosting it's solvers. This allows for a substantial boost in performance(2x2048 cores vs. 4 cores of my Intel Core i7 CPU). And while it's not entirely correct to compare these two processors equally(Intel general-purpose cores running at 4GHz versus Nvidia vectorized cores operating at 1.3GHz), there is definite benefit in numbers; 4096 cores net about a 10x improvement in performance from within the Realflow system over execution on PC CPU alone.

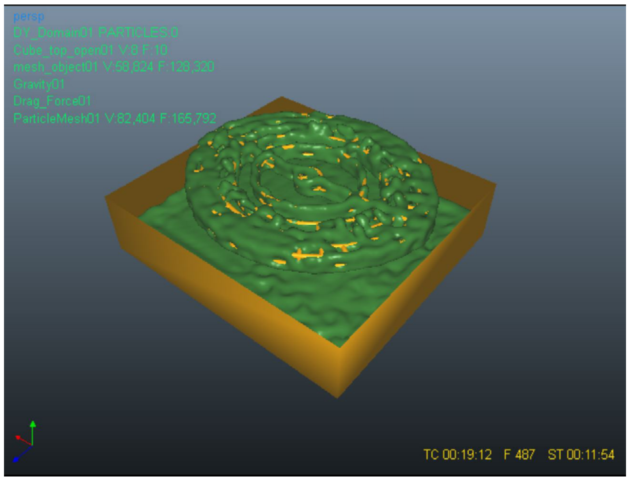
This software is formidable. Like many of these types of packages, it usually takes going through an example(or five) until the logic of its operation comes through. Luckily I'm not alone, many others have been in a similar situation and they've left a set of wonderful walk-throughs(Dieuwer 2014) on Youtube to assist. These are invaluable.

Each Realflow simulation happens within a defined hub area where all the chosen components within the hub are able to communicate with each other. The tool offers different emitter types and properties can be enabled by simply dragging and


persp
DY_Domain01 PARTICLES:12,090
Cube_top_open01 V:8 F:10
mesh_object01 V:58,824 F:128,320
Gravity01
Drag_Force01
ParticleMesh01 V:40,288 F:80,580

TC 00:02:15 F 65 ST 00:22:56




persp
DY_Domain01 PARTICLES:0
Cube_top_open01 V:8 F:10
mesh_object01 V:58,824 F:128,320
Gravity01
Drag_Force01
ParticleMesh01 V:82,404 F:165,792

TC 00:19:12 F 487 ST 00:11:54

Fig. 49 - Realflow Form Simulation

dropping them within the workspace and connecting them as I would in Grasshopper via virtual wires. The system readily tells you what nodes can receive what type of connections by enabling a wire connect or not.
In my first simulation, I imported my Diatom scaffold mesh and gave it attractive characteristics. I then created a fluid particle emitter pointed downward at the setup and configured it to produce particles with the default characteristic of water(viscosity,surface tension..etc.) at room temperature. The emitter would be configured to produce particle agents at a rate reasonable for the scope of this investigation(neither a drip nor a fire-hose) and would function for long enough to surround the imported mesh.

The kinematic engine within Realflow works with standard forces. When I first fired this up, the simulation blew agents out the bottom of the area. I had forgotten to contain the flow as I would in real life. I had to quickly model a containment box and enable collision management so that it would repel the water and not "leak."

Included within the package is a Keyframer(if you recall, this was made for the video industry). This mechanism allows the user to view the state of the simulation at any time and also allows for the choreographic control of any element in the system. It's through the Keyframer that I enabled the emitter initially and then disabled it after a few seconds so that my virtual water would not overflow the container.

Through iteration I began to see that more control was necessary. The water was moving too quickly(it "sloshed" everywhere), so I added a "drag" component to slow it down. The water seemed to not adhere to itself all that well, so I attempted to decrease it's viscosity and surface tension to help. In the end, I added a "sheeting" component to give it the consistency I felt it needed.

To give the particles form, a meshing process was added post-simulation to create the appearance of a water surface. I kept its parameters fairly close to default as altering these would send the system into very long delays during computation of each simulation frame. Here, as I saw previously in my Grasshopper Diatom form calculations, precision costs significantly- and the complexity curve is clearly not linear. I would put it closer to exponential... Alter the mesh triangle size even a small amount can send the system into itself for twenty-plus seconds per frame. When the simulation is two thousand frames(for a minute and a half of video roughly) this can push computation time into hours. At one end of the configuration lies the chunky and imprecise intersection of metaballs(a form defined by a basic spherical mesh surrounding a single point) while the other is a thin, wispy formlessness.

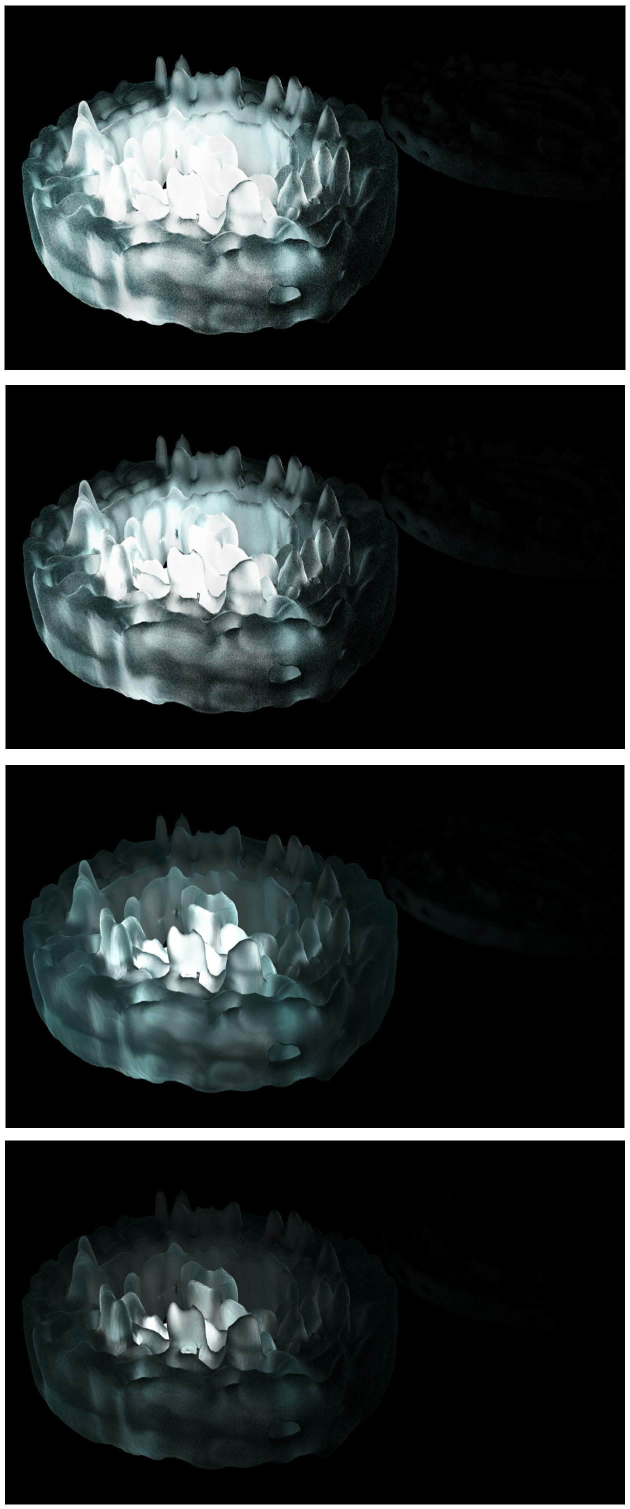
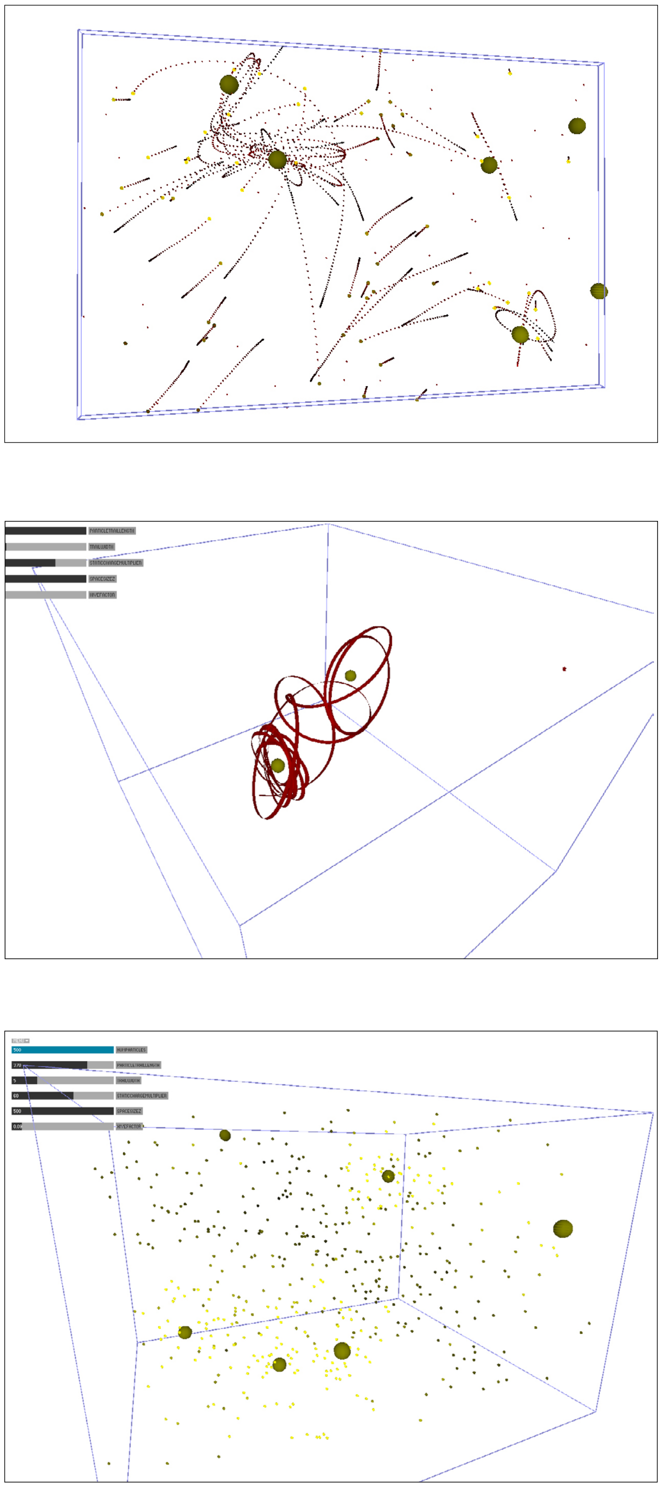


Fig. 50 - The Reveal - Lighted Snapshot of Realflow Diatom Form Result

In the end, once enough of the particles had adhered to the scaffold(and themselves), the resultant form was scripted to be lifted out of the bath of particles and allowed to settle above. This proved quite problematic as the kinetic energy of the system would not reduce despite the use of a dampening(or Drag) component. I reached out to the internet for others who had unsuccessfully used this Realflow component and found mainly successes. It seems that while this same component kept the energy of the system low enough to stop the simulation from losing its agents, it also inhibited its ability to resolve completely to a solid non-moving result. I suspect its precision became too high for the system to resolve into a specific state consistently. And while it resulted in forms close to the diatom form I had hoped to see, I believe I'd hit a limit in this approach and was left with an interesting form, but one a bit far from what I had hoped.



PARTICLETRAILLENGTH
TRAILWIDTH
STATICCHARGEMULTIPLIER
SPACESIZEZ
WAVEFACTOR

MENU
500 NUMPARTICLES
370 PARTICLETRAILLENGTH
5 TRAILWIDTH
60 STATICCHARGEMULTIPLIER
500 SPACESIZEZ
0.09 WAVEFACTOR

Fig. 51 - Coulomb Field Application in Action

Diatom - Processing Coulomb Field Application

Nanostructures within a diatom frustule can be quite complex. The valve of the Cocconeidacean Monoraphid has been amply studied and its structure heavily photographed. The image below shows the inside of a sample; its symmetry is initially bilateral, but as we saw previously, this continues to change at smaller and smaller scales as it divides outward from a central radiating point. Like a plant, there are vascular elements that grow within the form as it develops. The active external site for this growth is called the "papillae."

The main structural material used in these organisms is silica and its interconnected system of linkages is created by these papillae. These organelles "grow" the diatom's valve shell using a reaction-diffusion process modulated by the organism's morphological encoding. The process guides the distribution and rates of ph-change within the organism to allow for silica to be deposited out of solution from the surrounding water(Cox Eileen J. 2011).

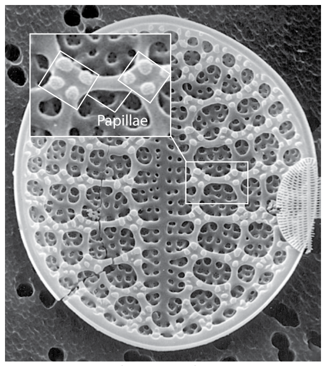
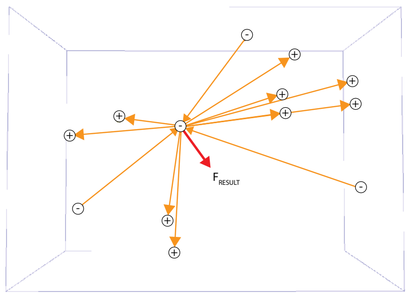


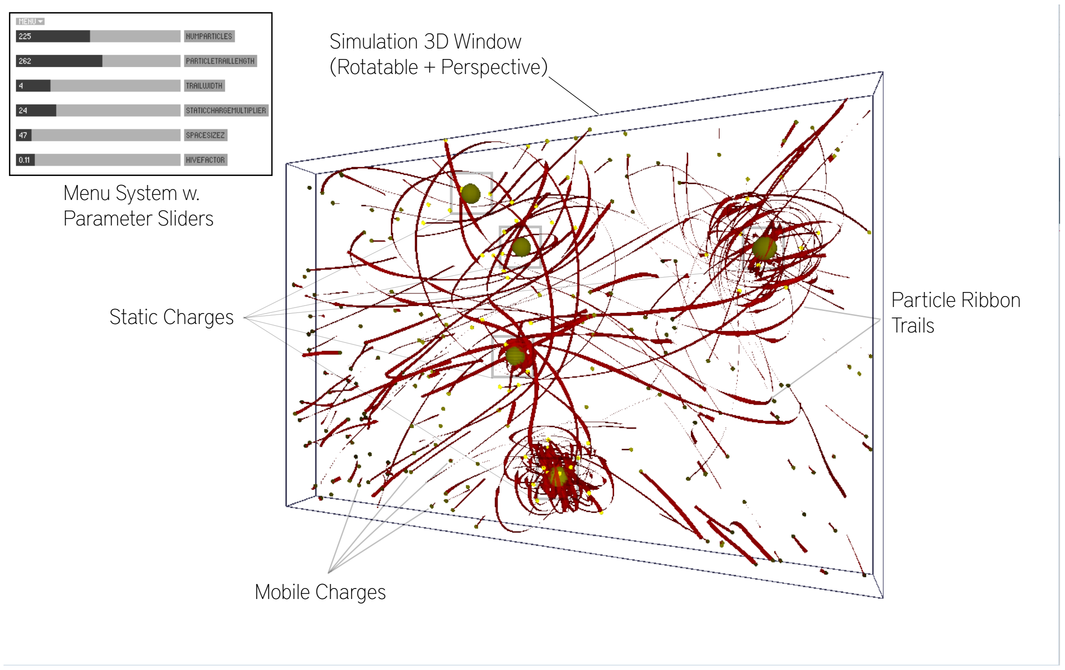
Fig 1. Valvocopula - Stefano, Journal of Nanoscience & Nanotechnology - Vol5, p22, 2005



Coulomb's Force acting from all charges

Fig. 52 -Papillae & Form Development through Silica Deposit by Simulation

To mimic this at the papillae scale(1-100µm), a software tool was developed and written in Processing to essentially set up a system of charges, both static(papillae site) and dynamic(free ions), which follow Coulomb's law of attraction. It is my hope at this stage that by altering some simple parameters such as charge strength, charge populations, and cooperative charge effects, that the paths of these mobile charges may couple. The result of which, within the processes of active papillae, would draw silica into an area for deposit(the greater the coupling, the better the



225	NUMPARTICLES
262	PARTICLETRAILLENGTH
4	TRAILWIDTH
24	STATICCHARGEMULTIPLIER
47	SPACESIZEZ
0.11	HIVEFACTOR

c	points display	**p**	output pdf of screen	**+**	add static charge
f	freeze simulation	**r**	ribbon toggle	**-**	remove static charge
i	invert background	**s**	static charge display		
m	menu	**t**	turn on point trails		
o	output mesh to dxf	**z**	Z-dimension toggle		

mouse + right button	perspective rotate
mouse wheel	zoom
mousewheel + shift	fast zoom

Fig. 53 - Coulomb Field Application Menu System

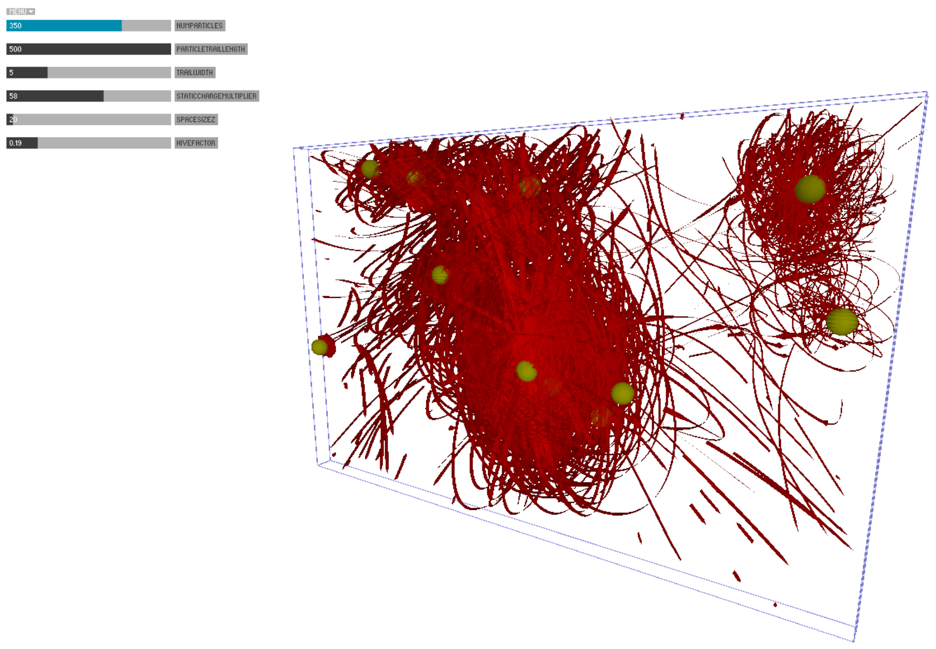
accumulation).
This real-time system utilizes an Agent-based architecture to simulate a series of individual charged particles, some fixed and some free, within a confined 3-dimensional arena. Charges are placed randomly when the software is initialized(the quantity and positions can be hard-coded). Their charge value is also randomly positive or negative and acceleration follows a simplified form of Coulomb's law. When the agents find themselves outside the arena, they are removed from the simulation and a new agent is allocated and placed within the simulation area.

Each charge affects all other charges according to Coulomb's force law. This may be attenuated through the "hive" menu slider. The number of agents, the type/length/width of trails, and the static charge field multiplier through the sliders as well. The simulation is highly parametrized and many of its values are also accessible through keyboard commands(Seen opposite).

The entire area can be rotated and field of view changed while the simulation is running. This gives the observer a chance to alter the parameters based on what they see. All views are centred at a cartesian point(0,0,0) which is located at the centre of the simulation window.

The quantity of agents within the simulation can be altered at any time through the top menu slider(0 to 1000). This proved to be the single-most sensitive control for the entire system. The algorithm is as follows:

```
For each charge
    Compute the Acceleration due to all static charges($Q_1q_2/r^2$)
    Add in the acceleration due to all other moving charges($q_1q_2/r^2$)
    Add resultant acceleration to charge's velocity
    Add resultant velocity to charge's position
    Add position to charge's trail list
        If position outside arena de-rez agent/spawn random new
Clear simulation viewspace
For each static charge
    Draw large sphere at its location
For each ion
    Draw point at its location
    For all points in trail list
        Either(add to ribbon)[r] or (add to line)[r] or (draw point)[c]
        Draw shape(ribbon/line)[t]
```



```
350   NUMPARTICLES
500   PARTICLETRAILLENGTH
5     TRAILWIDTH
58    STATICCHARGEMULTIPLIER
20    SPACESIZEZ
0.19  HIVEFACTOR
```

Fig. 54 - Coupling Experimental Setup

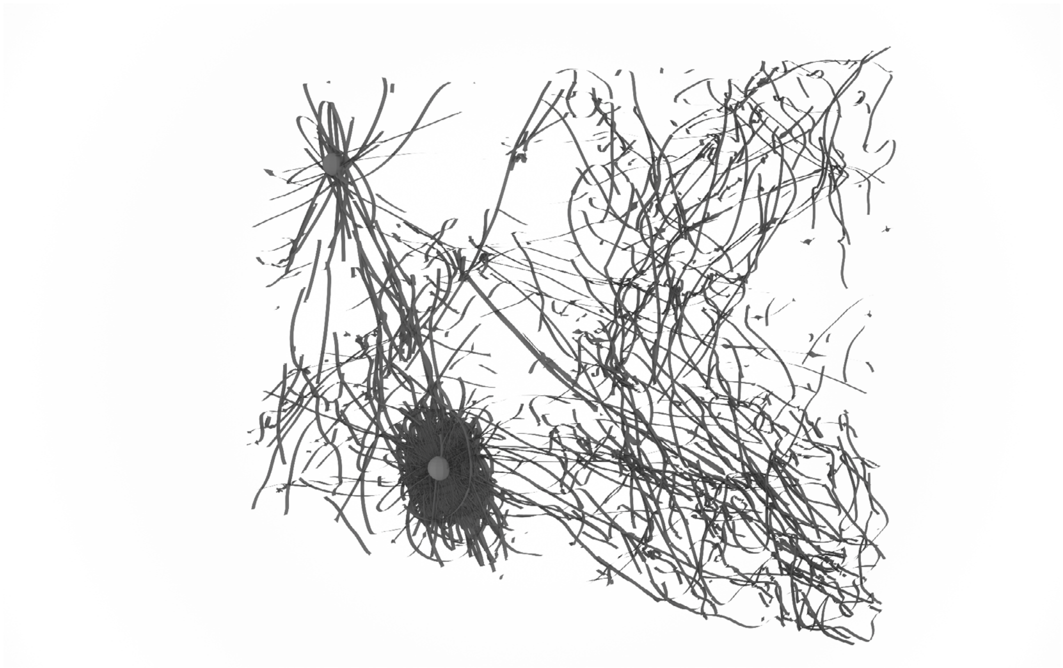


Fig. 55 - First Coupling

It doesn't take long to overwhelm the computing power of my simulation machine if the number of agents is large(>500), specially if trails are enabled. The computational time function for this algorithm is:

$$\begin{aligned} f(m,n) &= (mn + n^2) + (m + nL) \\ &= O(n^2) \end{aligned}$$

where, n = number of ions
m = number of static charges
L = length of ion trail

As each point must "touch" every other point for each iteration(n^2), this drives the computational complexity of the algorithm($O(n^2)$ using big "O" notation). This notation was invented by the German number theoretician Edmund Landau(Hardy 2008) and it refers to the tendency(or "Order") that an algorithm may exhibit as its behaviour moves to infinity. As this algorithm increases its agency, it grows at a rate similar to others in the Quadratic class.

This way of looking at complexity is invaluable when designing a generative algorithm. Just as the Systems Admin cares that your job doesn't run all day, monopolizing the machine, we would like to see a converged result in our lifetime when working on our own. This way of thinking, of assessing complexity continually, has to become second nature when programming or scripting.

$O(n^2)$ is not horrible, but still not great for large values of n. At 500 ions, that's $500^2 = 250{,}000$ calculations per iteration. Minimum. There are other concerns too. Every full iteration triggers a true *tabula rasa* when it comes to representation. This means that all geometry needs to be re-drawn(as a projection of 3-space) within the bounds of the defined 2-space window boundary so that the agents true motion can be visualized. This takes a material amount of time when the geometry is complex.

Early on this became a concern, so I incorporated a way to decrease the complexity of the trail types(dot/line/ribbon selector(+width)) sent to the graphics hardware to allow for more agents if needed during simulation settings. I found, as an optimization, that if all geometry was sent to the system as a contained Processing "Shape," output was much faster. Less calls to the graphics routines resulted in a higher frame rate and more fluid motion.

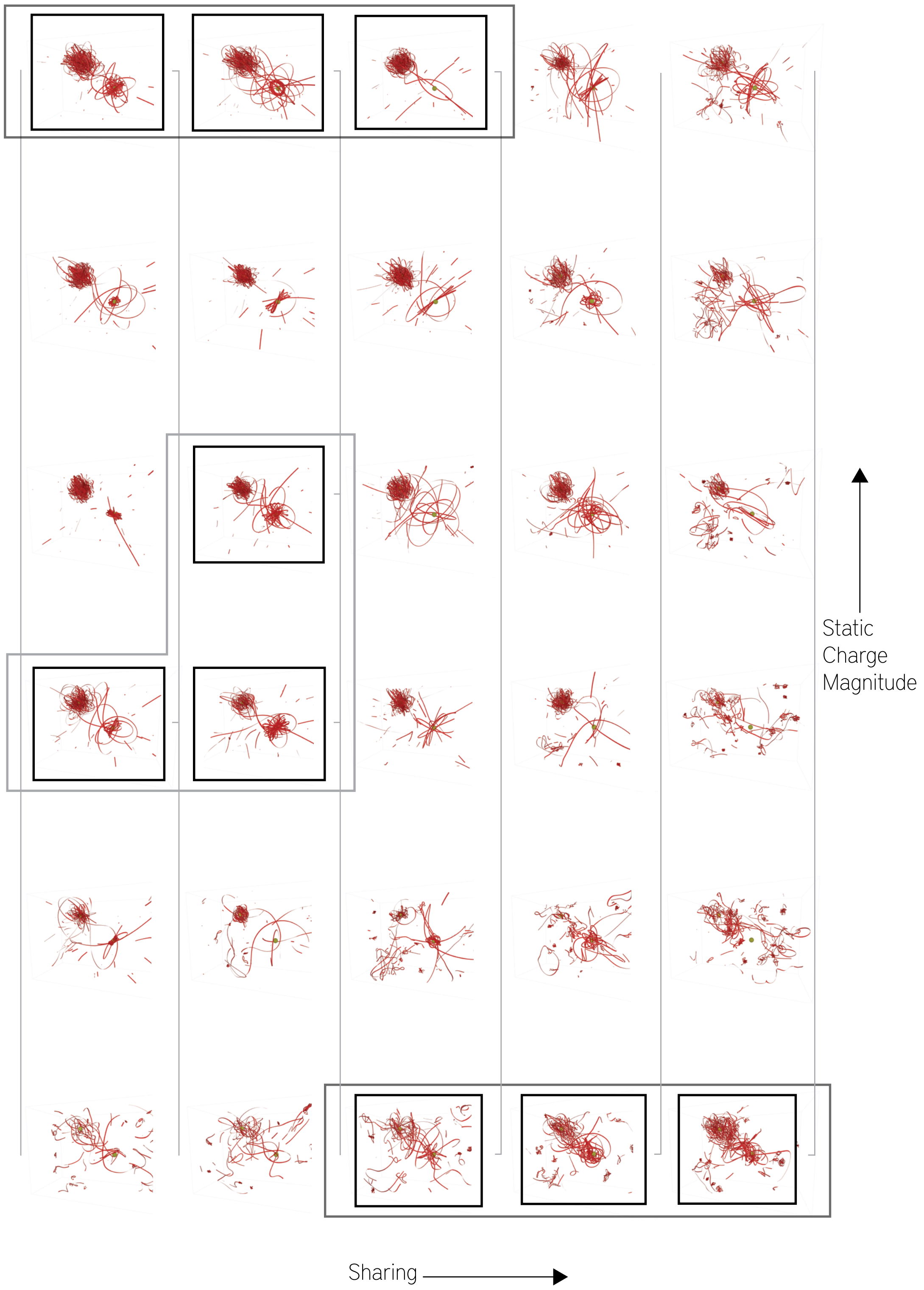


Fig. 56 - Successful Coupling

Coupling - Easing Into Materiality

My goal throughout this process was to find a way to give a sense of material "bridging" between papillae during their simulated function. I needed to "create" mass between emission sites and the best way I felt this could be done was to encourage the "ions" to dwell within a defined region between these organelles. Dwelling causes a greater chance for silica to come out of solution and deposit. The greater the deposit probability, the greater my chances for some bridging to occur.

While coupling did occur sporadically throughout many simulation runs, I felt it was important to be more intentional in my investigation. What settings produce the best bridging? And further to this, what characteristics of "coupling" were best when producing form?

I ran a set of tests to give me a sense of how this simulated system might behave. Again, this is an approximation of a simplified model to produce form using a Coulomb(charge) field as its source for affect. I would alter as my parameters, two things: the strength of the static charges in relation to the more mobile ions, and the amount of information sharing that would occur between all agents participating. The results are seen opposite. Coupling occurred in three situations:

Low Static Charge Multiple/Mid-High Sharing(empathy)

In this case all charges have the same magnitude and the system couples more easily with greater sharing. Without the greater force of larger static charges to dominate, their location and unmoving character drives the behaviour of the swarm. The resultant trail pattern creates more of a mat of trails that fill out the region between charges well. This is a very positive result.

Medium Static Charge Multiple/Low Sharing

While there is some tendency for ions to leave their associated static "home" charges, it seems to be fleeting and of low probability. The bridging behaviour I'm looking for is weak-absent but there is some coupling and more of it as sharing is increased. This behaviour does not develop further as sharing is increased. Not a great result.

High Static Charge Multiple/Low-Med Sharing

The larger magnitudes of the static charges clearly takes over here and drives this

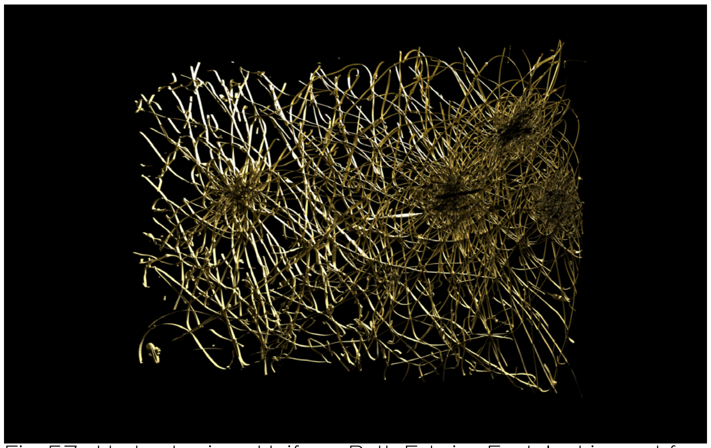


Fig. 57 - Undersharing - Uniform Path Fabric - Each looking out for themselves

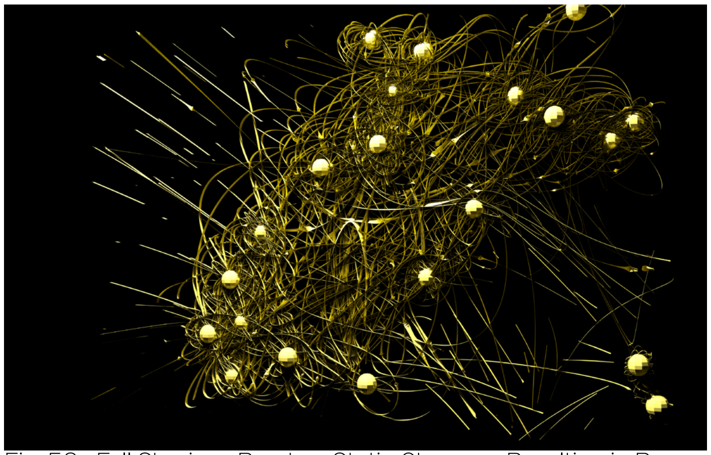


Fig. 58 - Full Sharing - Random Static Charges - Resulting in Reduction of Differentiation

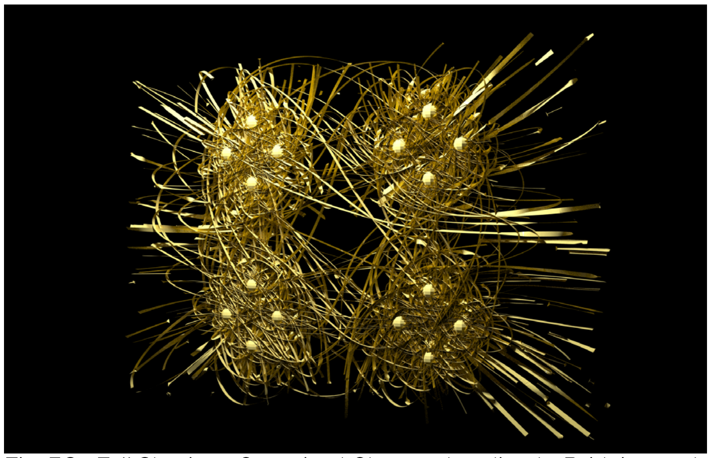


Fig. 59 - Full Sharing - Organized Charges Leading to Bridging and Path Differentiation

arrangement to hold on to charges firmly and locally. It seems that the energy embodied in the system supports some escaping to the neighbouring charge, however the ions dwell at each for a long time before they may transition. This configuration creates strong affinity and doesn't benefit from any flocking parametrization. The fabric created between the static charges is thin and not great for a bridge due to the low frequencies of transitions.

Results - Processing Coulomb Field Application

Of the three types of coupling seen, the last was best. The others were either too infrequent for any lasting form to be created or too focussed on static site dwelling to be of much use. It's clear that a delicate balance of charge magnitude and flocking behaviour is necessary within this simulation to produce a fabric of trails between any two static sites. In a true physical system, the static field decays as $1/r^2$ and there is no choice as to whether each ion's affect is shared; the system resolves all.

The driving characteristic for form seems to be the unmoving nature of the static charges. Their acceleration information(or lack thereof) is "shared" throughout the system and has an effect that defines the form ultimately. The ions may move about randomly and react to each other but it is the constant calming effect of the static charge sites that ultimately causes moving charges to dwell.

If their locations are randomized, the result(Fig. 57 -opposite) generates an intriguing form that we can make much from. This is ultimately a dynamic process and this alone has its merits. We can read much into the energy of the forms created. Ultimately, when some organisation is brought to bear, bridging at the scale needed does happen and could produce form similar to the silica bridging in the SEM image of the papillae site(Fig. 52). While there are some parallels to the form resembling something of a Eucharistic adoration , these are purely coincidental.

If this tool were to be used to further this investigation, the reaction would need to be modelled(involving a reaction-diffusion algorithm) of the silica in solution. Greater control of charge placement and mobilities within solution would be necessary. Also, the iterative approach used to solve Coulomb's force is very simplified and would need to be improved to deal with interaction when the distance is very small. The current algorithm works to ensure the system does not create energy when an interaction occurs but its controls are rudimentary and could use improvement. Round-off and imprecision within the algorithm could be handled more precisely using improved math libraries.

Results - Understanding the Organic

In this section, a set of geometric and agent-based digital methods were used to approximate the form of the Diatom *Coscinodiscus wailesii* valve area at various scales. These efforts involved encoding arrangements of various features and they approached each problem in a slightly different way: symmetric principles were leveraged as a scaffold to host a more solid framework of material and voids, L-Systems logic was used to investigate a possible arrangement mechanism for the papillae(growth) sites in the mantle, a geometric simplification of a Grey/Scott Reaction Diffusion algorithm was employed to see if there was similarity at a low level to the SEM images at a very small scale, the Realflow application was employed to see if principles of fluidity played a roll in the diatom's final overall form, and finally an agent-based method was employed to simulate processes at a much smaller scale when producing silicon bridging within the organism. The investigation followed a relatively intuitive path and was successful in approximating some of the geometric relationships inherent in the Diatom's form.

Arrangement at a macro-scale of material and void utilized a mechanism that modulated symmetry. The flex of the algorithm produced similar form but with less or more complexity, and it was easy to overwhelm the abilities of my simulation system. I was repeatedly reaching the upper-bound of its capabilities due to the complexity of my implementation. I was relatively happy with the results, and while time could be spent in optimisation of the system or altering the algorithm for better approximation to the Diatom's specific form, I felt I was able to come close enough to satisfy my curiosity.

Structure in the diatom begins with the arrangement of the papillae as they are the source of inflow of the structural material of the organism. The SEM images I referenced were of dead diatom organisms caught at a particular time in their life-cycle. As the papillae arrangement develops it can't but help to affect the overall organisation of the silica bridging. Using a plant-based L-Systems growth algorithm for site growth seemed a good method for mapping the papillae site distribution. Using a parametrised recursive algorithm, coupled with an attractor, it was possible to generate a few useful iterations. The diatom's papillae distribution may be governed by a similar mechanism. As with much in this effort, I suspect there is more at work in it's arrangement and I suspect it is likely to be seen when the system is better coupled to the organisation's environment.

The Diatom lives within an aquatic environment, so flow of resources must play a sizeable role in its form generation. One of my precedents, Michael Hansmeyer's Grey/Scott Reaction Diffusion investigation, seemed to offer a continuous sheet-based result that could appear similar at a small scale for the diatom valve form. While the true mechanisms of this physical systems require an iterative solution of the differential equations that define the reaction, I felt it must be possible to approximate its result through an analogue. The results were relatively successful, but yet again complexity reared its ugly head in even my overly simplified solution. It was again easy to overwhelm the system when it's precision was even minimally defined. While the results are encouraging, more optimization is certainly necessary if I am going to tease out a useful solution that approximates the true form.

My last attempt to generate a fluid form was a bit of a reach. I had attempted to use a number of Grasshopper plugins to "grow" the smooth silica form based upon my organisational framework but most could not work with the networked scaffold I generated from the first phase of this investigation. I tried simplification techniques(and optimizations) but the input easily overwhelmed any of the plugins and methods I made use of. At this point, having some experience with Next-Limit's RealFlow fluid simulator, I broke from the saddle of self-generated form and attempted to simulate the flow and attraction of a fluid around my framework using their system. This direction worked, producing a consistent mesh that I could finally print using the 3D printer assembled later in this section. Now the meshes had their issues, working through the parameters of the simulation it was very challenging finding just the right values that would stabilize into a usable form. The system's kinetic energy would be difficult to dampen. All the same, several manifold forms were produced and finally a consistent print was possible.

Finally, having done some programming back in the dark ages of computing was little help when it came to picking up Java. The Processing language is a bit of a walled garden for development for Java beginners. A simplified set of libraries is provided to perform common input and output without shouldering the entire burden of the Java language. It's a great stepping-stone to object-oriented language development and the interactive development environment(IDE) was simple and easy to use for a relative beginner to modern languages.

I bring this up because the object-oriented mentality is very useful when thinking of software agents. They could be implemented in most any language, but Processing and Java in particular are very well suited to the concept. I found that creating an Agent object that included it's abilities and characteristics was as easy as invoking a method in a "let there be light" way. It's very powerful. Within the Diatom Coulomb investigation it was important to have these software agents essentially create and destroy themselves as they passed out of bounds of the simulation region. Tight coupling of the environment to the graphics capabilities resulted in a very convincing and interactive simulation.

At it's heart, my goal of the investigation using agents was to create a recognisable bridging form between the non-moving papillae sites. To do this, I felt I needed to create coupling events. It was through this mechanism that consistent form would be laid down and after some time, a mass that appeared to be bridging. As I've said before, this is far from the processes that likely happen within the diatom morphology at this level, but it is a likely formal analogue.

It is notable that the "sharing" portion of the agent parametrization was applied across the entire population and it took a large number of iterations to understand how coupling might manifest. My charted results highlights coupling incidences while varying non-moving charge magnitude and sharing(empathy) shows, in my view, that the best coupling happens when a "fabric" of paths is generated between points rather than the more aggressive "thinner" connection seen when there is no sharing at all. Later, when the charges were arranged in a configuration resembling the papillae SEM arrangement and the parameters were set to those similar to the successful coupling settings, bridging did appear to happen in a predictive and positive way. Agency appears to be a very good method for generating sometimes difficult form that can't be easily defined through geometric means.

Conclusions

Getting a "feel" for any algorithm is hard. Obtaining more useful feedback than a simple "wrong" or "right" can be a godsend even in the best of times. In building an analogue of a diatom's fustule creating a system of feedback proved to be especially difficult as there are more than nine major parameters for its base implementation, plus a handful more for meshing and display. The best I could do to get a feel for its behaviour was to flex each parameter or parameter pair as a basis of behaviour and sense this affect on the system's response. The investigation shows how this was done and while the process was quite frustrating, the results were very positive.

It was possible to get a "feeling" for how the system as a whole behaved: the growing algorithm, its implementation, the software platform it was living within, and the machine it was run on. They all fed me information, whether direct or more subtly, while the system was being grown. I could do this because I was able to form a relationship with each layer. I learned what was considered "normal" at each stage and as I was able to add new elements and measure the new system's behaviour.

As a list, some of the feedback I experienced might be considered quite typical or basic: Does it crash? Does it slow the system to a crawl? Others were more complex: Does the system respond in a way that was expected? What are the limits to its behaviour? If the input values are taken out of bounds, what happens? Any good surprises? Bad ones?

To test a system, we move the inputs and hope for a real-time response. This is not always possible however, sometimes the scripting system takes tens of seconds to resolve and while very frustrating, it was very telling of the system. Coping mechanisms are developed to save work when we stress a system...constant saves, multiple backups, thinning out the test machine so as to minimize boot times when the parametrization crashes the whole system. There are so many ways of coping, or managing the situation.

The system broke often. This is actually a sizeable understatement as sometimes it seemed to be perpetually broken. I spent a significant amount of time going over added logic to understand why systems failed or how a small change could bring the house down. These failures were instrumental in affording me the ability to iteratively gain understanding at a very low level of the system. Rather than simply focussing on the nine major parameters that governed the solution for a geometric investigation, there were uncountably many internal parameters that pushed and pulled this system into existence. This internal crawling through the works was difficult but key to gaining my understanding of the behaviour of the system as a whole. Much of this knowledge became ingrained when “feeling” out the behaviours of added layers of functionality.

The question of algorithmic correctness and optimization is constantly on my mind when developing a solution such as this. Can I obtain the same results by thinking of the solution in another way? Are there other implementation choices that might improve output? It seems that every test-stress cycle triggers thoughts of “how could I do this better?” While not expecting perfection, measured improvement directed by existing behaviour should be possible.

How can I bridge the gap between idea and implementation? Like growing a pearl, the solution is added to, layer by layer. Each level being tested for stability and resiliency. At each iteration, behaviours grow out of the system. How does it effect the sound of my computer when running? I have a CPU and Disk monitor window open all the time, looking for memory consumption, cpu behaviour, heat dissipation... amongst others.

As the Grasshopper scripting environment was for the most part a closed system. I really had no idea on the specifics of each module’s implementation. I simply had to put trust into the script’s ability to function as advertised. For many of the wired functional blocks, it was relatively easy to stress their operation until they would fail. Many were quite resilient but some add-on components offered a very limited band of functionality when used. This proved to be a major factor when pushing the system forward. It required that I try many add-ons that purported to offer similar functionality(meshing being a major pain point). None offered their source for their components, so I was not able to understand how they might be failing. Being an open community, it is survival of the fittest in the truest sense of the word. Badly behaving components are not recommended for distribution on various Grasshopper script sites as people find these issues. Only the adept and communicative contributors seem to succeed. The community is strong, and it’s strength comes from being interactive with your user-base. Some build great followership, and in doing so are able to eventually market their work.

Throughout this section I became aware of a number of inputs helping me to understand the capabilities and limitations of my implementations. I reached for mental analogues often to help me build a form that approximated the diatom at several scales. It was important to watch and listen to what each software solution was telling me. Each was the manifestation of my understanding of the problem at hand and a demonstration of my ability to implement the logic necessary. Each experiment resulted in a tool being built to help answer my questions concerning the diatom's formal organisation.

I developed relationships at many levels within the work and became acutely aware of how the machine as a system responded to my changes. I instinctively grew a sense of how to minimize its loading so that the investigation could continue quickly and not be at risk to loss. I matched the development of the algorithms with a sense of delicacy when making changes, inherently "knowing" what directions would cause complexity in execution, hindering things. Each implementation was a construct made from my vision and from the resistances created by its manifestation in software and hardware. In gaining "feeling" for the development, a deeper awareness developed in my understanding. Many of my activities became automatic to me—avoiding the pitfalls while encoding my perception of the formal logic became a very fluid exercise. I was very happy to see my first bridge between head and hand built. And at my best reckoning it only took me about 2000 hours.

Fig. 60 - 2D Particle Swarm

pina - A Choreography of Affect

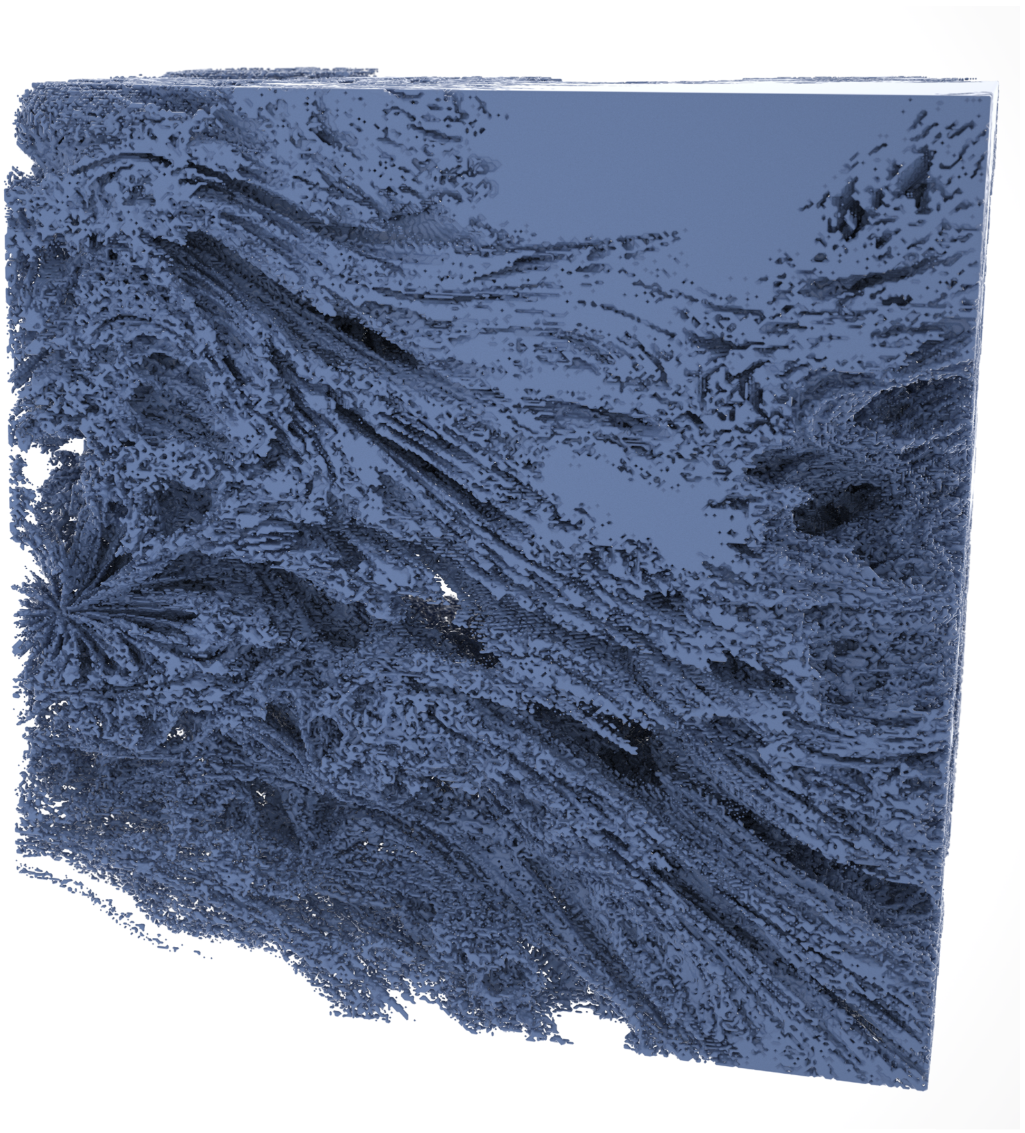


Fig. 61 - Mandelbulb Slice Voxelization

Introduction

"My Ariel, chick,
That is thy charge. Then to the elements
Be free, and fare thou well!"
Prospero, The Tempest - William Shakespeare

In my first year here at UW Architecture Don McKay boomed in one of his many, now infamous, Arch-100 lessons, "Don't spend your life form-finding, you'll blow your brains out!" This was in the context of investigations surrounding some of the more iconic architecture and furniture types. Don had been through this meat grinder himself and was astutely warning us of the well worn yet rocky shores of this pursuit; Hadid's parametric curves, Gehry's intricate flowing-yet-discontinuous facades, and the iconic Eames lounge chair being examples of when it works and is accepted. His warning was a reminder of the long list of those who didn't find acceptance, who produced and are lost in history; the assumption being that they were wasting their time looking for that "style," one that would be lauded and forever remembered.

At the time, his warnings made good sense. Sometimes though I think Don might just raise his voice to wake people from their daily slumber, to shock so as to wake. Imparting impulse, if even perceived, can have motivating effects individually and collectively. Don's impetus, I suspect, was to make us all agents of change in one way or another.

In looking for a solution at the tiniest of scales, I enlisted software agents to help me build silica bridging between ion production sites as part of the Diatom investigation in the previous section. It was a good introduction to application construction using the Java language and the solution was able to successfully show formally that bridging is possible at that scale. It was my first jump into application programming after using Rhino3D/Grasshopper scripting for much of the geometric investigation and was fairly straight-forward. I used many built-in libraries provided by the environment and it was a good proof of concept. The guiding principle for much of the entire simulation was one-dimensional, with sharing(empathy) between all the agents facilitated while they all experience the same static forces based on their positions.

While this offered a unique perspective to a problem difficult to solve through geometric means, it was also my goal in this section to try something quite a bit outside my normal experience; What if I was able to allow for greater dimensionality of interaction? I thought to investigate form generation through my interpretation of a choreographed work of modern dance.

I was recently exposed to the work of Pina Bausch and was quite taken by some of the scenes in a movie production about her by Wim Wenders in 2011 named *pina*. In the documentary, a number individuals from her dance company each performed works that they felt best conveyed Bausch's most compelling work. They also spoke about why they were so taken with her and the effect she's had on each of their lives.

The segment performed by Ruth Amarante & Andrey Berezin had the greatest effect on me. They're scene with each other in Wenders' documentary is intense and magnetic. As this thesis is about the tool and tool-making in as much as it is about digital design, I will look to find form within their work seen in *pina by* extending my work done in the Diatom-Coulomb as a proof of concept to include a larger sandbox of agents. With the number increased, I hope to see greater form definition and more dynamic results. Within this section, I will look to understand how form is created using agents, and what impact this has on tool-making to successfully manage larger populations with greater interaction.

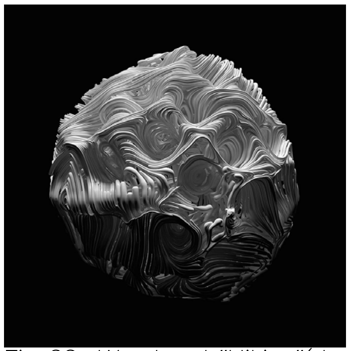


Fig. 62 - *Wandernd* -"Hiking"(desktriptiv 2013)

Fig. 63 - *Schichten*-"Layers"(desktriptiv2013)

Fig. 65 - *Gewoge*-"Waving"(desktriptiv 2013)

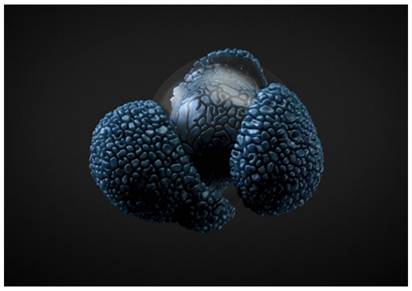


Fig. 64 - *Guaddel*(desktiptiv2014)

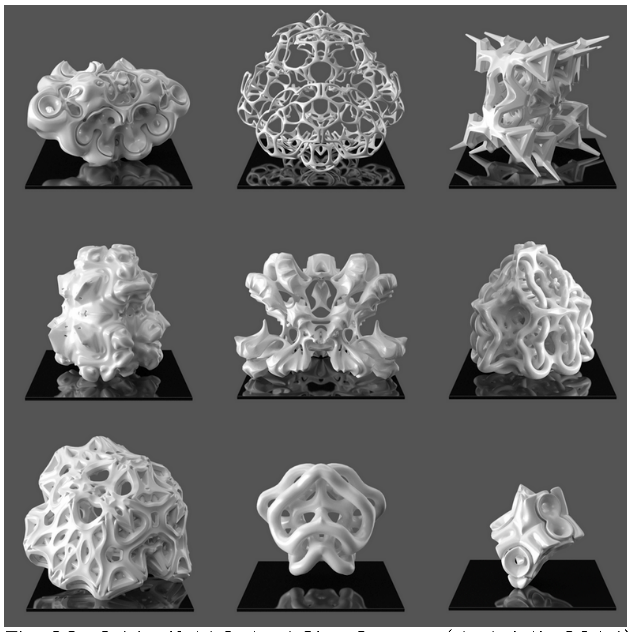


Fig. 66 - *2-Manifold Output SimpSymm* - (deskriptiv 2014)

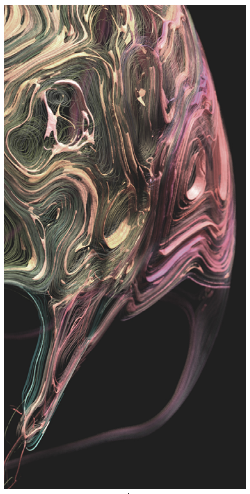


Fig. 67 - *Flow 1* (deskriptiv 2014)

Precedent - deskriptiv

Christop Bader and Dominik Kolb are two researchers(MS candidates) that now reside in the Mediated Matter group at MIT with Neri Oxman. They were previously at the university of Weingarten, in Germany, where they were completing their undergraduate and graduate degrees in computer science. Their work in application development is considered state of the art within the realm of generative form and design. They later formed a design collective, called "deskriptiv" for the marketing of their work.

Much of their efforts has only been seen through its use in form development. Some of their artwork and proof-of-concept images has been published on many sites and blogs(Behance, flickr, CA(Creative Applications), and IN(Inspiration Now)) and until recently they've published very little otherwise. This is why that it was quite a surprise to me that In December 2016 Christoph Bader and Neri Oxman were published in Computer-Aided Design, offering with what I suspect the first of many combined submissions describing Bader's previous work.

"Recursive symmetries for geometrically complex and materially heterogeneous additive manufacturing"(Bader and Oxman 2016, 39-47) is the long-awaited description of how form is developed geometrically utilizing recursion and various modes of symmetry in his co-written application, SimpSymm, with Dominik Kolb while at Weingarten(not sure why Kolb is not mentioned in the paper as co-developer). The magic of their implementation is their effort in creating manifold output in this project. A 2-Manifold object(surface) is one that is essentially "water-tight." This is important when translating the form into something that a 3D printer can use to fabricate as a solid object has no break in its surface definition.

When searching for exemplar generative method, the work of deskriptiv shines above all others for me. They successfully tap into an organic spirit while coming at the solution from many different poles, sometimes using software agents or exploiting recursive and symmetric method.

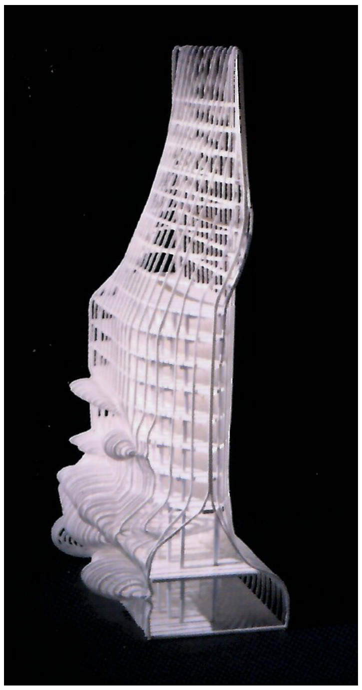


Fig. 68 - *Eyebeam Museum*
Greg Lynn FORM 2001

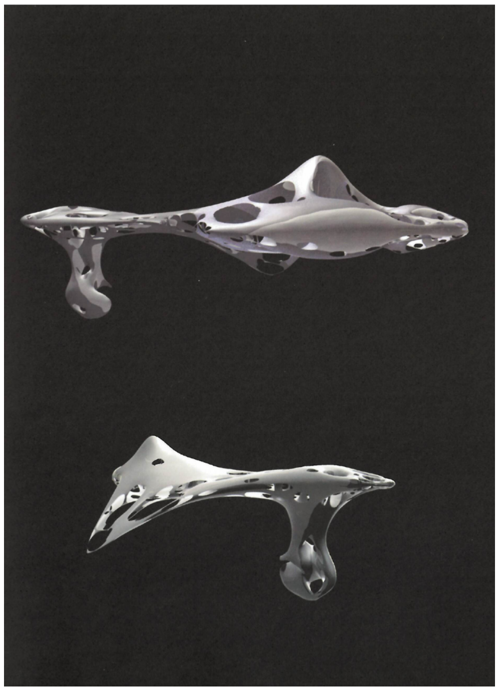


Fig. 69 - NOAH Set for the Film DIVIDE
Greg Lynn FORM 2004

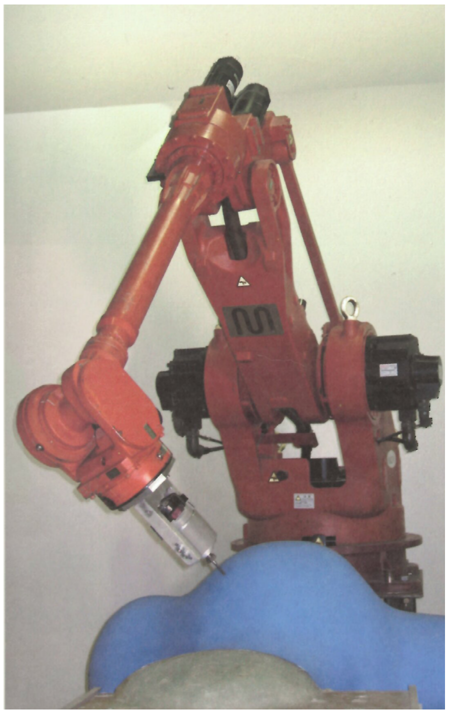


Fig. 71 - Robotic Arm Cutting Blobwall Brick
Greg Lynn FORM

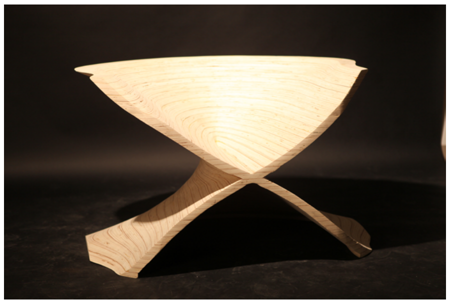


Fig. 70 - Riemann Chair - Wade Brown &
Galen *Jones 2013*

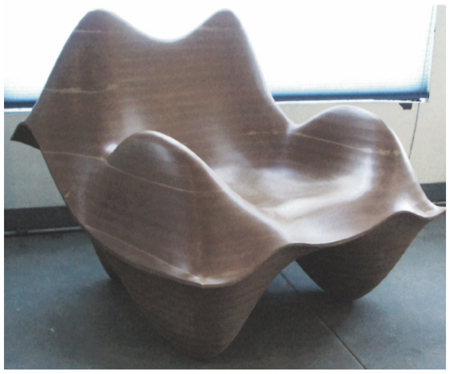


Fig. 72 - *Vitra Ravioli Chair*
Greg Lynn FORM 2003

Precedent - Greg Lynn

Greg Lynn completed his MArch at Princeton in 1988 and worked in Peter Eisenman's office(Lynn and Rappolt 2008). Lynn is considered a pioneer in the use of computation and digital tools in architecture and has an extensive catalogue of projects by his office, Greg Lynn Form. He has taught at Columbia, Yale and is currently teaching at UCLA and was the winner of the Golden Lion at the 2008 Venice Biennale for his installation.

I have to admit, when I began seeing Lynn's name as a source for inspiration in the world of Digital Architecture it surprised me. Much of his work is published in the mid-90s up to 2008 and then it stops for the most part. His formal investigations begin with the era of the b-spline and caps off at the beginning of the digital baroque. FORM was doing very important work in its day and I feel it's necessary to investigate digital's ability to connect to people. Greg Lynn would always focus on this in is projects and I think that the importance of this can't be understated.

It's easy to design virtually if there's no constraints in materiality, space, or scale(is that all?). The work can still be quite undefined in form, but if it never leaves the laboratory of the digital world, it loses its value architecturally. I think this stigma follows digital investigation wherever it goes. A digital design process is just that, a process, and not a system all to itself. It's too easy to stop designing when the form "looks" correct; and I believe many do(in error). This design process must hit the real world in some way some time, and Greg Lynn always ensured this was the case.

His work may be a bit dated (the era of splines,surfaces, folding)(Lynn and Rappolt 2008) but his methods are still valid. Today's design culture takes its queues from materiality(a.k.a Oxman Material Ecology) and this is strongly rooted in the real world. Greg Lynn's design method utilized digital algorithms for form finding but also depended on digital fabrication as part of the process. CNC, Laser cutters, and 3D printing were early at the time for small shops but were very much integral to industrial growth at the time. It's only now that these tools are becoming more available to smaller shops(or even for a home workshop). In third year of Architecture at UW we applied some of these techniques(Brown and Jones 2013), I'd like to build upon this using some of what I've seen in Lynn's work to this investigation.

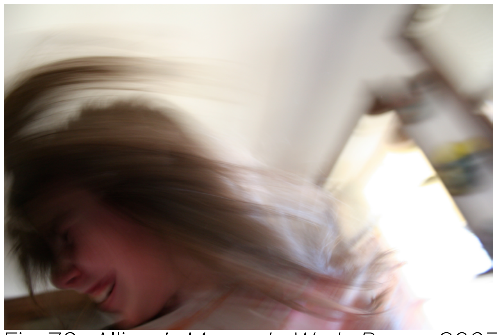


Fig. 73 - Allison's *Moment - Wade Brown 2007*

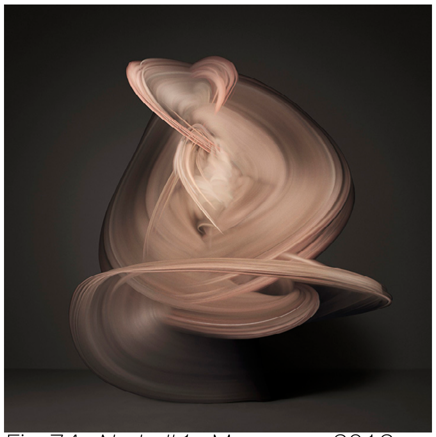


Fig. 74 - *Nude #1* - Maruyama 2012

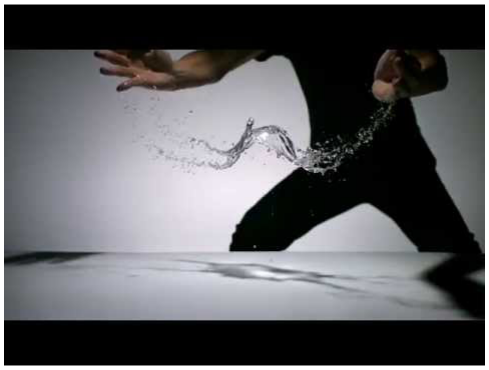


Fig. 75 - Water Movie - Maruyama 2013

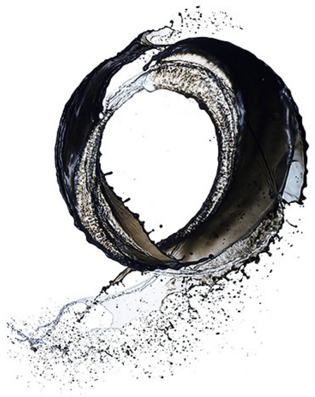


Fig. 76 - *Kusho #1* - Maruyama 2013

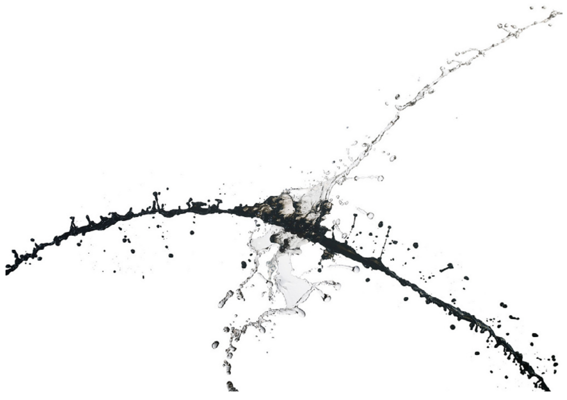


Fig. 77 - *Kusho* - Maruyama 2013

Precedent - Shinichi Maruyama

Born in Nagano Japan in 1968, Shinichi Maruyama studied at Chiba University. After graduating, he spent the next two years working for a large commercial photographic company in the world, Amana Japan, picking up method. Later, as a freelance photographer, he spent four years photographing Tibet and discovering his style expression. His interest in stop-motion photography developed when he began his time with the Hakuhodo Photo Creative in 1998(Naqvi 2010). There, he had access to high-speed strobe equipment was able to further develop his interest in capturing the moment.

As a photographer, Maruyama is concerned all about the moment. Whether it's only a fraction of a second, or a collection of seconds, his focus has been about our perception of our own existence. He plays with time on opposite ends of perception; offering a view into affect on the boundaries of perception.

For the singular moment, he says in his artist statement; "I know something fantastic is happening. "a decisive moment", but I can't fully understand the event until I look at these captured after-images."(Maruyama - opposite) He paints in ephemeral space as the sumi ink he uses in *Kusho* is thrown; each painting is unique. In a fraction of a second he connects us with the mechanisms of our perception, giving us a view into our own uniqueness. There is tremendous energy of focus in the "moment" he highlights.

For the works in *Nude*, Maruyama creates a construct of 10,000 layered moments and highlights its collection in memory. This space would otherwise be outside of our perception(almost its own heterotopia) because we get the chance to respond to the collection in its entirety at once. The choreographed motion creates a form that embodies the dance in its entirety, and while dependent upon each collected frame, can create something entirely new.

I am fascinated by these works. Maruyama looks for the moment using photography. I would like to explore this similarly, but through the use of digital tools. I believe formal investigation at this level has value in an encoded result that is dynamic, yet static and ephemeral.

Fig. 78 - pina - Movie poster - Wim Wenders

Dynamism - Affect/Object/Vitalism

"You don't start dancing. You dance."
William Forsythe

Like Le Corbusier's Modular system, where he proposed a physical generator for his projects based upon the dimensions of the human body, I decided to try something different by beginning with a piece of interpretive dance choreographed by the highly accomplished Pina Bausch. It is through her work that I hope to understand form generation using a choreography, or encoded modulation, of affect in an agent-based environment.

From her obituary in the NY Times:

> "Pina Bausch, the German choreographer who combined potent drama and dreamlike movement to create a [new] powerful form of dance theater that influenced generations of dancemakers, died on Tuesday in Wuppertal, Germany. She was 68."
>
> "Ms. Bausch was quoted as saying she was 'not interested in how people move but in what moves them.' "
>
> "I look for something else," she said. "The possibility of making them feel what each gesture means internally. Everything must come from the heart, must be lived."
>
> (Wakin 2009)

The documentary *pina* by the director, Wim Wenders, was for me a very powerful introduction to an innovative pioneer in modern dance. The story moves through notable segments by members of her dance company; each offering their take on Bausch's unique style, while at the same time offering thoughts on their time with her. I was drawn specifically to the segment focusing on Ruth Amarante and Andrey Berezin(Wenders et al. 2011), both long-standing members of Bausch's Tanztheater Wuppertal.

In this short piece(Fig. 77), Ruth is walking slowly forward, as if in a daze. She stops when she recognizes the male dancer(Andrey), and then falls forward without any effort to catch herself. He intervenes at the last moment and begins to lift her -- Slowly and carefully she then begins to walk backwards at the same time(something that must require a tremendous amount of effort and skill) until she is upright.

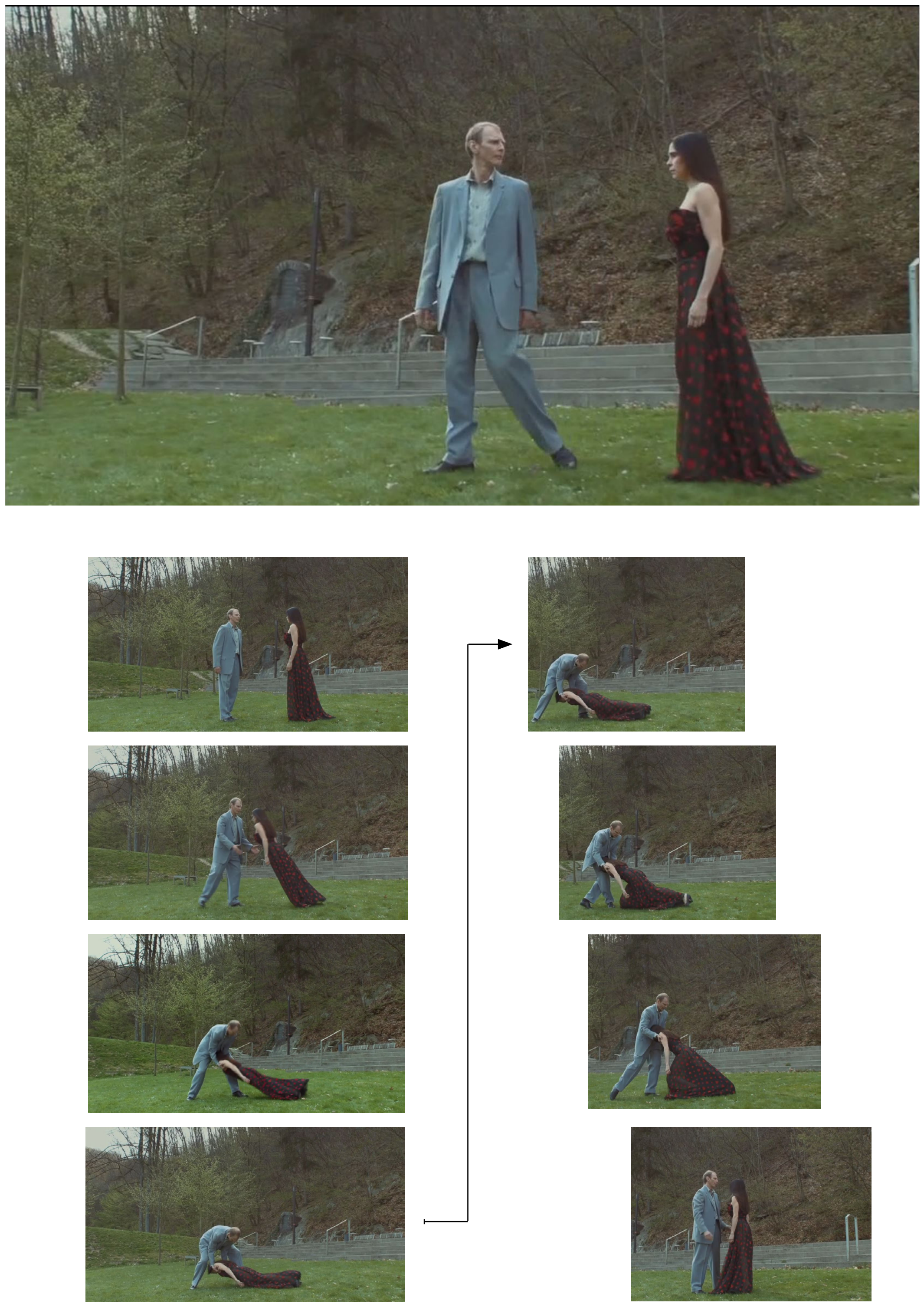


Fig. 79 - Andrey Berezin & Ruth Amarante in *pina* - Movie scene - Wim Wenders

She then resets and picks another direction..with the cycle continuing, becoming more risky each iteration, until the end of the segment. The site is mixed in grade and cover, and the music(The Here and After)(Miyake 2011) is hypnotic and quite appropriate to the piece.

I think the easy interpretation of this piece is that it is of someone lost, caught in a senseless struggle, experiencing the same situation over and over again. My interpretation of the piece is a bit different than this- I think it's more about someone pushing themselves through a difficult time. It's about courage, perseverance, and strength. I feel it signifies an internal struggle within the dancer as she tries to maintain some form of stability in her life, even while skilfully moving backwards within it.

The male dancer who catches her is part of her unconscious, and might represent a father or trusted person; I might call him a manifestation of her superego in a Freudian-sense. She gives herself entirely to this process when she stops supporting herself, almost daring harm to come. It's about boundaries, resolution through repetition and variation, and the stamina of someone in crisis. While she repeats the cycle three times within the segment, she chooses to change its parameters, almost as if she is provoking a solution... Any solution.

The Implementation

I began to think of a system that would generate software agents who's physical behaviour would emulate symbolically(with some influence of the literal situation) what I felt was happening inside the dancer's head during one cycle of the dance. Pina, in her own words was "not interested in how people move, but in what moves them"(Wakin 2009). I needed groups of agents to work together to flow and articulate themselves within the changing emotional(emotive) landscape.

This would require a system with rigid spacial controls, instrumentation surrounding the generation of fields and agent flow, a mechanism to share(empathetic activation), and a comprehensive key-framing mechanism to allow for a review of state at any time throughout the cycle. Further to this, each trial would need to be remembered offline to document the work and replayed if necessary.

I looked at these requirements ,and though that while it is certainly possible to implement within Processing, it might take considerable effort. To evaluate the work effort required, I figured that I might try to produce a simple flocking simulation to give a taste of the work needed for even a simple portion of the effort. The scientist

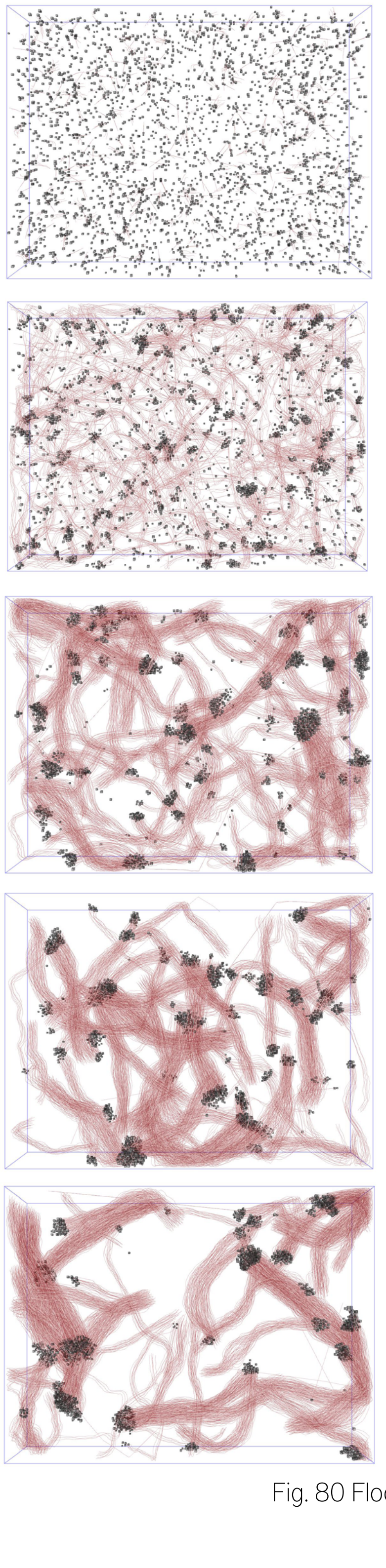


Fig. 80 Flocking - Processing Investigation

in me thought to test whether this was possible using a small sample first.

Flocking - A Processing Proof of Concept

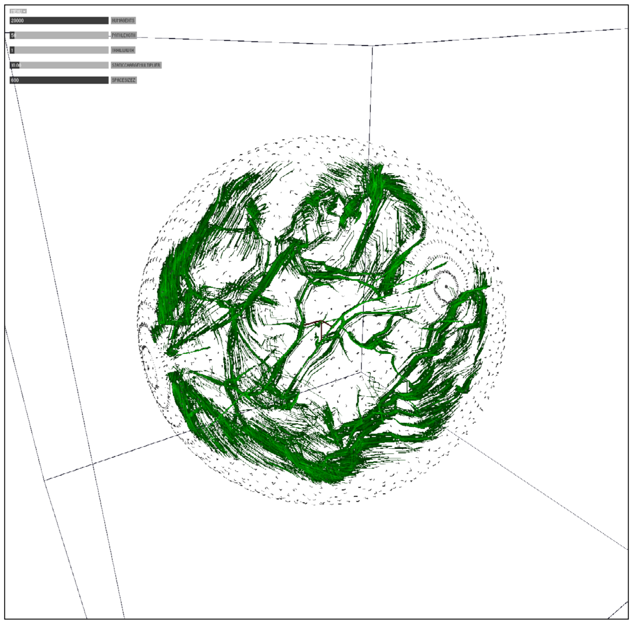
With a small sample(2000 or so agents) and by simply giving each a small amount of directed energy, I let the system follow its encoded logic. Processing has a method for managing flocking within a limited sense. I used its Boid library(within the Punktiert Physics engine) to create a population and allow a small amount of sharing within a sphere of influence, and I let it run(Fig. 78). The result took quite a while to resolve itself and while it showed promise, the simulation took seemingly forever to show its nature. Unhappy with the result, I decided to leave Punktiert's Processing's library behind and try this on my own.

Flocking - Investigation of A Flow-Field Tensor-Based System within Processing

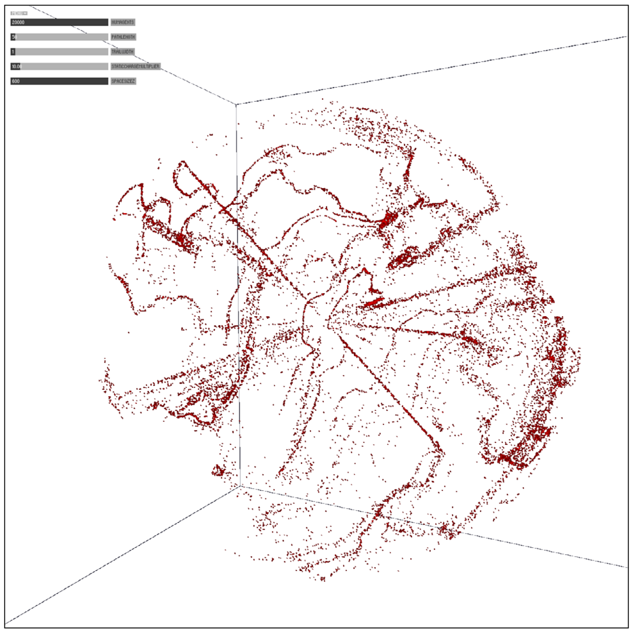
Not willing to let things go at this stage, I shifted my point of view. Rather than deal with empathy through a sharing of affect directly, I felt there must be a way to encode it within space itself. From my past exposure to the mathematics of space-time, a Tensor fits this bill exactly. It's definition is just that, a mathematical way to describe a vectorized character of space at any defined point. As agents encounter this character they can respond as their own abilities allow by altering their own states and possibly altering space itself for any agents that follow. Sounds easy right? I found it a simple extension of the flocking investigation but it became evident, as I implemented my thinking, this was a much larger problem looking for an even larger solution.

To simplify the process, I elected to implement this test using spherical coordinates as it makes it easier to generate the tensor field using harmonic functions. The area of influence initially was intended to only be a surface defined by a simple harmonic function(initially defined by a sphere). The application's results(opposite) incorporated upwards of ten thousand agents and a tensor sector resolution of 120 per direction.

Unsurprisingly, this quickly overwhelmed the Processing environment. While I was able to alter the tensor configuration through an interactive process to improve things, the system also became too complex too quickly to manage no matter the number of agents. My efforts to simplify became a bit disheartening and I decided to stop this direction of investigation.



20000 NUMAGENTS
50 PATHLENGTH
1 TRAILWIDTH
10.00 STATICCHARGEMULTIPLIER
600 SPACESIZEZ



20000 NUMAGENTS
50 PATHLENGTH
1 TRAILWIDTH
10.00 STATICCHARGEMULTIPLIER
600 SPACESIZEZ

Fig. 81 - Tensor Field - Processing Investigation

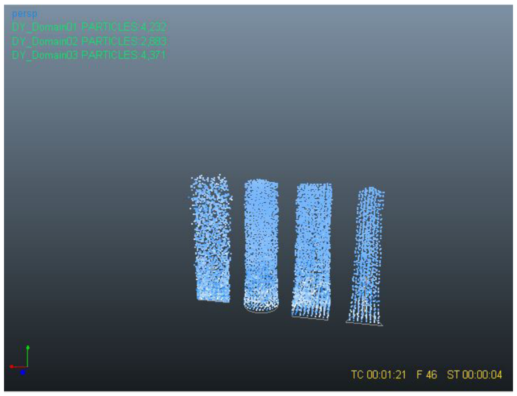
Then there's the encoding of space itself. This became a significant hurdle. The agents would certainly navigate the environment they are placed within, but how could space itself be changed to coax agent behaviour in a consistent and manageable way. At 2000 lines of code, and four classes(SVector,AffectNode,Agent, and Flowfield), my decisions in implementation became an issue for me on multiple fronts. The addition of necessary functionality surrounding key-framing and saving/replay would prove to make the code unmanageable. Each added function took greater and greater effort to implement. I had definitely bitten off too much in making this attempt.

It's here that I encountered a clear understanding of the cost of making implementation decisions and the impact of their complexity within the realm of programming. As with algorithmic complexity, this implementation was quickly heading into the area of exponential complexity($O(n^n)$). Or at least it felt this way, because I felt it necessary to re-touch each area of the code as every new function was added. Question answered; with two unsuccessful attempts, I chose to move with a commercial package to realize the Pina investigation. I hang my head, having had such great hope for the composed solution and having learned a substantial lesson.

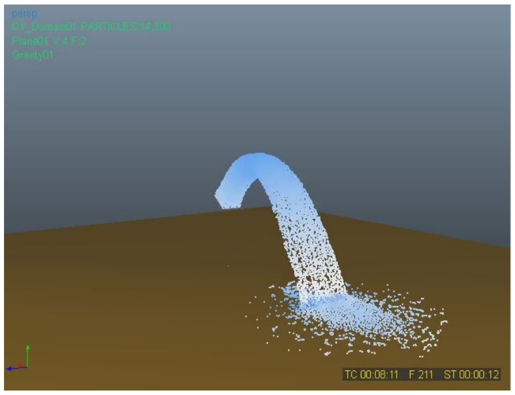
Realflow - A Commercial Agent-Based Physics Engine

While the encoding of affect within space is intriguing, there must be a better way to investigate it. From my previous work into fluid simulation using commercial packages, I elected to build my simulation within Realflow. It is a keyframe-based agent-driven system that allows for a choreography of custom particle emitters and forces. It's ease of availability to students(free 1-year student license from it's author, NextLimit) was a boon for me and it showed promise.

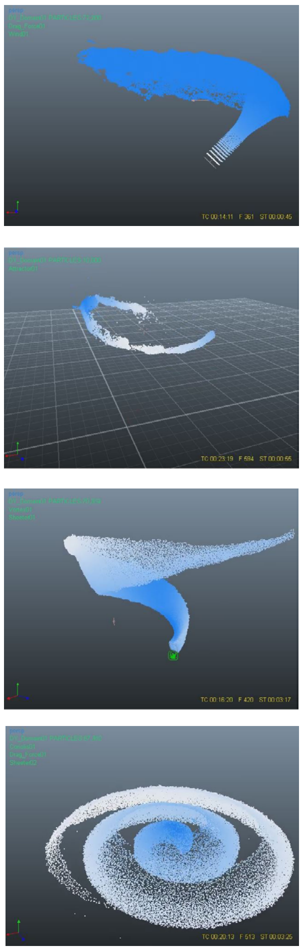
To fully utilize Realflow's abilities, I had to first work though it's capabilities and create a vocabulary of expression. I began working with it's many emitter types and physics solver engines(Standard, Dyverso, Hybrido). Based on the application, some were better for flocking while others were better for larger more macro-based situations(shattering monolithic structures..etc). One major benefit as mentioned before, Realflow's solvers can take advantage of my pc's nvidia GPUs for simulation, allowing for sessions involving tens(or hundreds) of thousands of agents in real-time. The best fit for my investigation was the Dyverso Solver for this reason. This was a boon to my work as more agents interacting quickly allowed me to make adjustments and re-run the simulations with new settings. This systems configuration


persp
DY_Domain01 PARTICLES:4,232
DY_Domain02 PARTICLES:2,883
DY_Domain03 PARTICLES:4,371
TC 00:01:21 F 46 ST 00:00:04

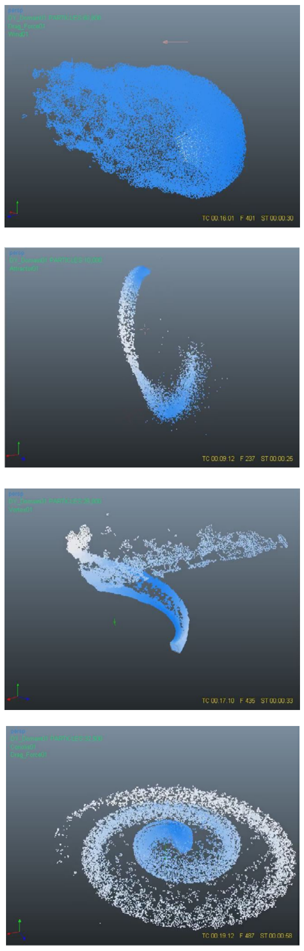
Emitter Trials - Cross-section/Randomness



Emitter Trial - Gravity Field + Collision



Field Trials with Sheeting



Field Trials without Sheeting

Fig. 82 - Particle Language Investigation

was head and shoulders above working within the Processing Java environment.

I began emitter trials with differing sectional profiles and modulated the randomness of the agents character upon being released. Realflow offers the ability to manage the density of agents within the section during each release period. They could be configured to get in each other's way more often or to "go with the flow" using a laminar component(Emitter Trials - opposite) once released.

Adding external field-based characteristics was the next logical step, including gravity-like acceleration using the square sectional emitter. This is where I began to included external mesh object collision(elastic and inelastic) by including a floor plane. I then moved through force attractors and Coriolis forces, and finally adding collision ability(elastic and inelastic) of varying degree between agents. Sheeting and grouping was added last as more flocking information sharing was explored.

All of these interactions are collected within the Realflow keyframe engine and can be interrupted, altered, and played backwards if needed. The tool writes each frame as a separate file in any number of file formats chosen to help(.abc, .sd, .bdc, .obj, .bin, .txt..etc.). The abc(Alembic) and txt(Raw) formats allow for direct access to the location, velocity, and neighbouring agents of all actors in the scene for each frame. Faster-moving agents are represented in a whiter colour that is normalized across the entire participating population. If Realflow is configured to create a particular file-type it will write them in the project directory; the more formats chosen, the more work the system does, so it makes sense to limit what is written to a bare minimum. I learned early on that the ".sd" format is sufficient for realflow to maintain the simulation, anything more should be added only if necessary. If needed, other formats can be used to create animations or simply to export the system's state into modelling software such as Autodesk Maya, 3D Studio Max, or Rhino.

Matching force-type(mapping) to my impression of the dancer's changing state took me a bit of time. As the source(diviner) for the session, I chose to represent the lead dancer's state as an emitter of particles(agents) that followed along a bent spline-like a vortex. I felt that by using a spline as an emitter, this was the best and simplest way to represented the unnatural perception of time, an expression that the dancer may be experiencing in a non-linear way, having a non-linear reaction within her environment, resulting in a lossy non-uniform and unbalanced vortex. After some trials, I was able to create a truly compelling restrained vortex emitter that seemed to keep things together while clearly under stress(like the mind of the dancer).

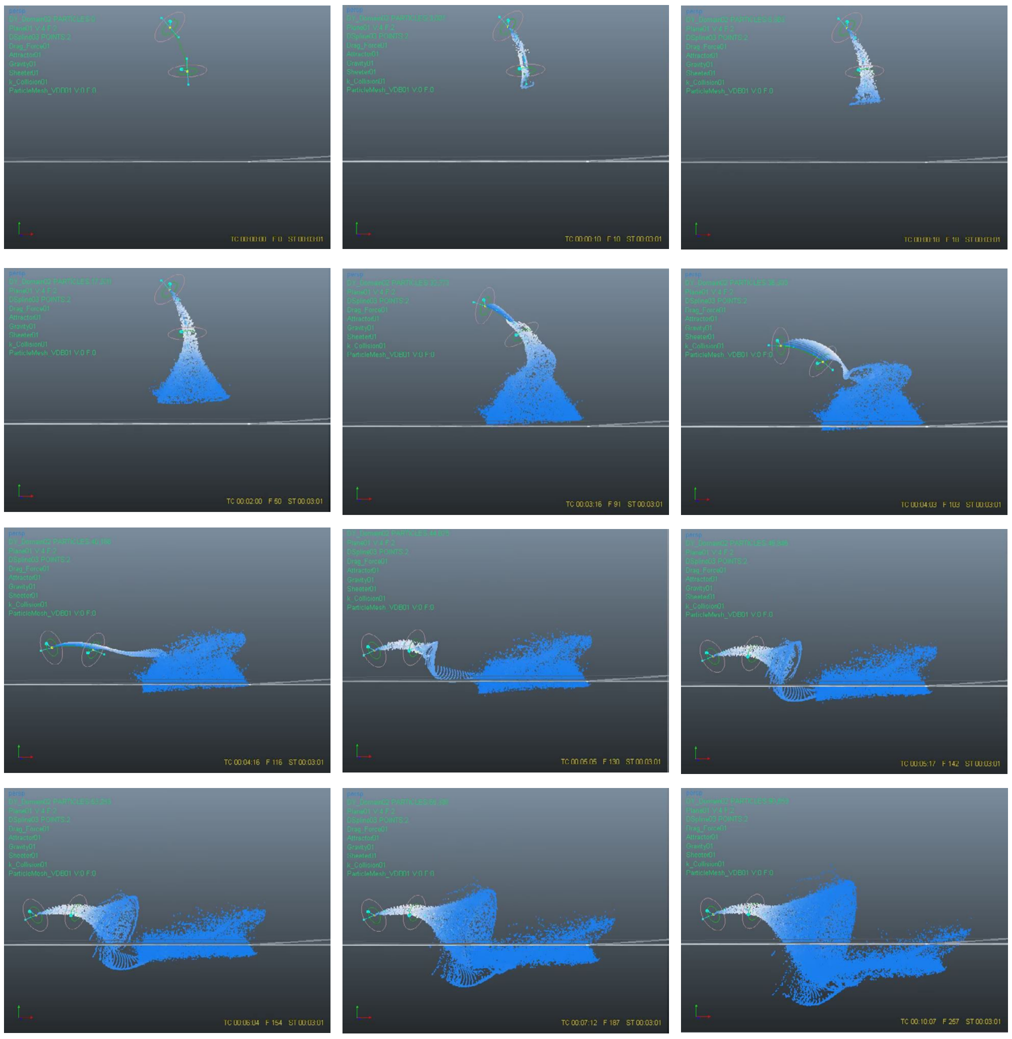


Fig. 83 - Realflow Pina Particle Sequence

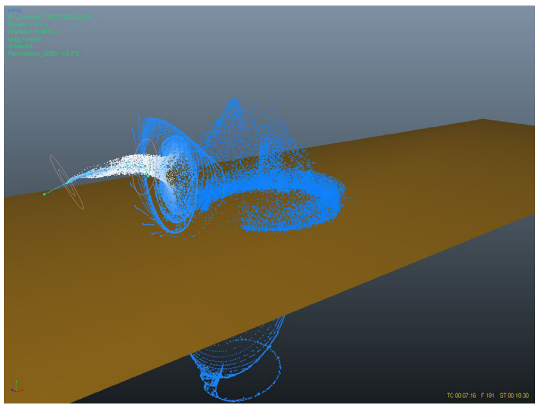


Fig. 84 - Realflow Pina Particle Perspective

I also thought if the vortex of particles was focussed downward it might appear to offer some form of perceived stability. It is through the motion and modulation of an added molasses-like dampening, threshold-based sheeting(allowing for a collective stability), gravity, inter-particle and external collision, and finally a real/imaginary-like separation of the ground-plane, where my feeling of what transpires in a cycle happens. The simulation was very computationally intensive and needed two coupled high-end GPUs(NVIDIA GTX980s) to make the workflow interactive.(Note: Since purchasing and using these premium cards for this investigation, NVIDIA has since replaced them with the new GTX1080 cards that are capable of 4X improvement in performance for the same price. This is an ongoing-risk/benefit of a relationship with outside technology.)

I split the universe here into the conscious(above) and unconscious(below). Much of what happens occurs above the plane within the conscious, but some does happen when she essentially "turns off" consciously, as she falls and is caught. The result is partially hidden below but still the processes that manage inter-particle communication allow for their interaction above and below the separating plane. The particles below continue in their own way but experience affect from all within their scope, as seen in this version of the simulation.

As the emitter matches the dancer's movements, it leaves behind active agents, all reacting to each other and the conscious/unconscious dividing surface. They blow through each other, are caught-up in each other's concerns and situations,they get pulled along, they get flung away or discarded. All the while, the emitter moves/creates/disturbs. It is the only motive source within the simulation; all other actions/reactions are secondary effects.

The resultant form at any moment is the sum of all previously captured moments, much like Maruyama's *nude* series but also incorporating the ephemeral nature of his *Kusho* and *Water dance* series. Initially, much of the collection of agents is quite consistent and the form easily understood. It doesn't take long for the mass of points to reduce into a something that I have a hard time interpreting. While this is ok, tectonically this might be a bit difficult to realize. I was hoping that by including each moment additively, something new would come through(as in *nude*) and I got this. The resultant collection of points defines a moment of complete disarray as multiple organising principles consume each other. It's marvellous and very confusing, just as I suspect the dancer's state exudes. Her outward calculated effect betrays a seething mix of opposing affective emotion.

While I create something new within this investigation, the source at some point be-

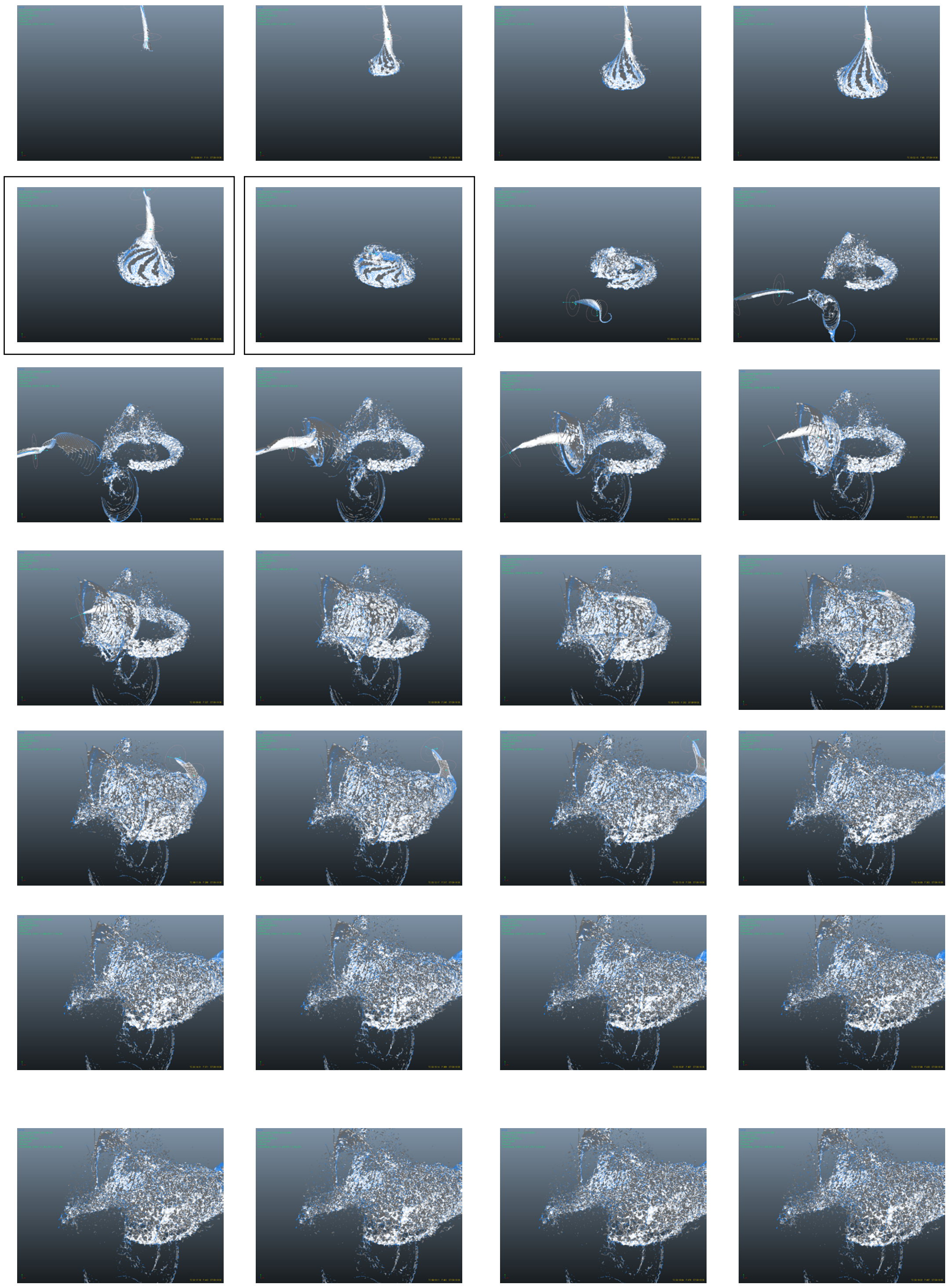


Fig. 85 - Tight-Meshed Frames of Simulation

comes secondary to the effects seen. The particles take on a life of their own, and adding more creates a floating maelstrom that masks the old. This is why I chose to look at only one cycle of the dance, because it was enough. The "old" is the affectual output of the agent stream and the "new" is the effective collective response, and while this seemingly gets a bit muddled as the segment progresses, it was what I had hoped to see by design.

Realizing The Tectonic

This digital investigation has been very virtual up till now. At some points within any digital design session I feel it's important to either produce perspective renders involving useful materiality characteristics(and lighting situation), or to physical models. Both require some further processing as part of the tool chain. It's important to close the loop in my design process and engage as many senses as possible.

Up till now I've been working with groups of agents as defined points, all visually creating a field of changing form, that seems consistent solely based on the their sheer number. We can see this effect in flocking birds in the fall and the undulating perceived form that their movement provides. Unfortunately points alone do not make a form; they can only abstractly represent what that form may be. If we had infinite points, the form may ultimately be well-defined, but we simply don't. It's at this time in the process that I have to describe the concept of "meshing. "

To define a form, one way to look at it night be to create a set of rules to describe what is "inside" and what is outside" for the set of regions within the domain where this form resides. Points help, but we get into the issue of resolution and having enough of them again. If I take each point being an indicator of what can be considered "inside" then I might be able to use its location to host a sphere(an object with a simple geometrically-defined "inside" and "outside") of sufficiently small radius so as to not conflict with its neighbouring points(and their spheres). The result might approximate a form but it would still be made up of discrete objects. This representation is known as a "metaball" representation and while this may look more closely like a form, it's not there yet.

What if we take the radii of the sphere's used and increase them until their perimeters overlap? Visually, the spheres collectively might appear to be an object, but geometrically the collection of spheres have a mix of "inside" and "outside" regions

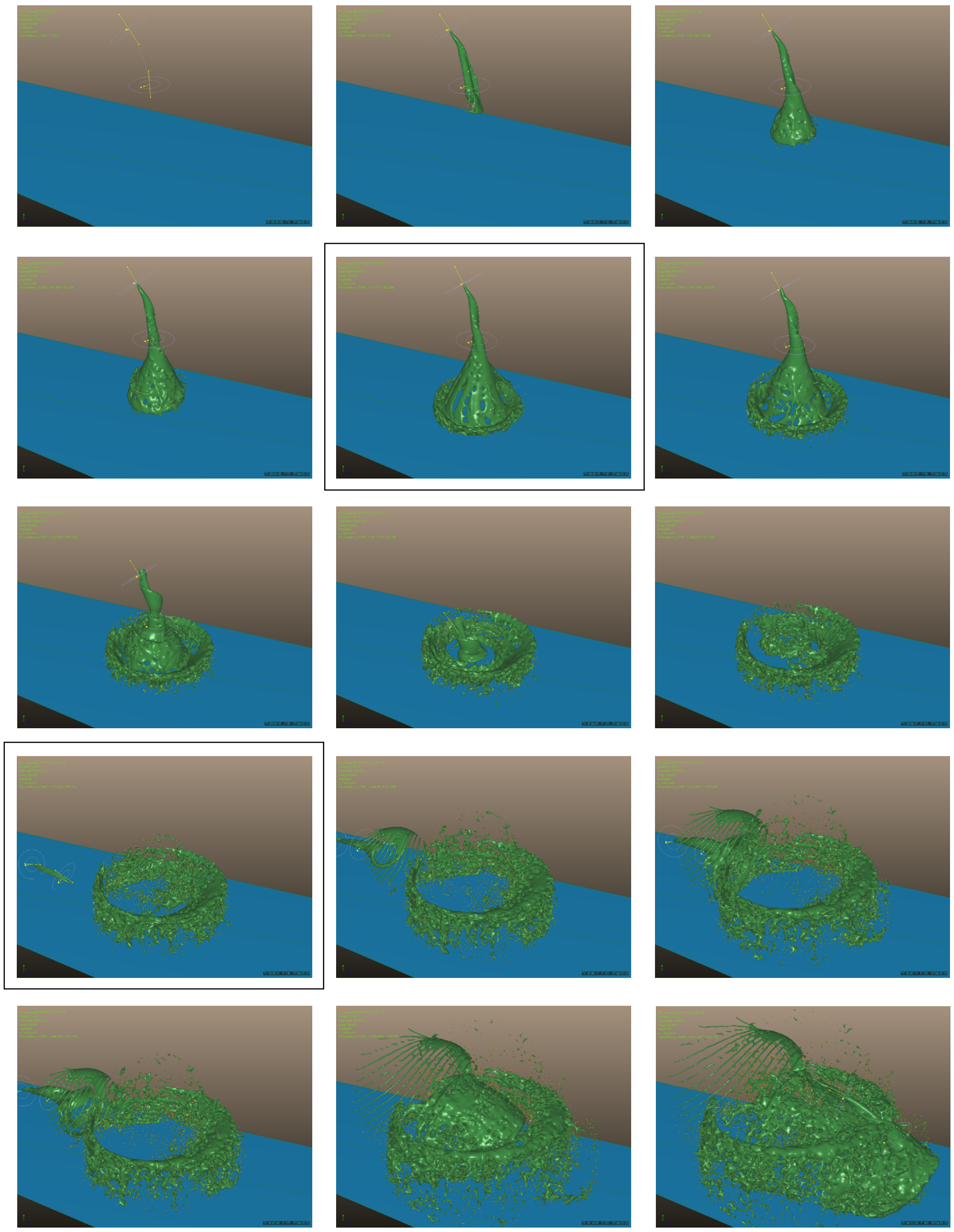


Fig. 86 - Loose-Mesh Frames from Simulation

within the set of intersecting surfaces, some regions being more "inside" than others. I would say that at this stage, if the inside surfaces are removed and the resultant surface definitions being divided into either smaller rectangles or triangles, we then have what might be called a surface "mesh."

We could even further approximate a better coverage of the spheres by stretching this surface to best cover all the points by individually altering each points' sphere radii. The resultant surface could range from one that is more pointed and exact through to one that is round-ish and "blobby," yet cover the set of points completely. This is a rough description of the meshing functionality that exists within the Realflow system. There are many other methods, some faster than others, but for this investigation I feel it important to speak to what Realflow is capable of.

Meshing is very important within the process of realization of form digitally. Without it, my simulations involving software agents would find it challenging to find physical form, be it within a detailed visual render or when output to a 3d printer. Both require that an "object" have form and this form usually must be defined by a surface mesh or more current NURBS definitions as being manifold(a.k.a "watertight").

Using the meshing processing within Realflow is relatively easy; one need only add the particle meshing module to the simulation graph and enable it's output file type to be processed(usually .bin or .obj) during a run. Because this can be very computationally intensive, I used this functionality sparingly and only when I need to look to produce for render or 3d print. The surface tension and viscosity of the perceived surface mesh can be altered to best create the coverage necessary. I used these parameters to do a best fit; the results were quite good and I was able to produce meshes that represented the agent population.

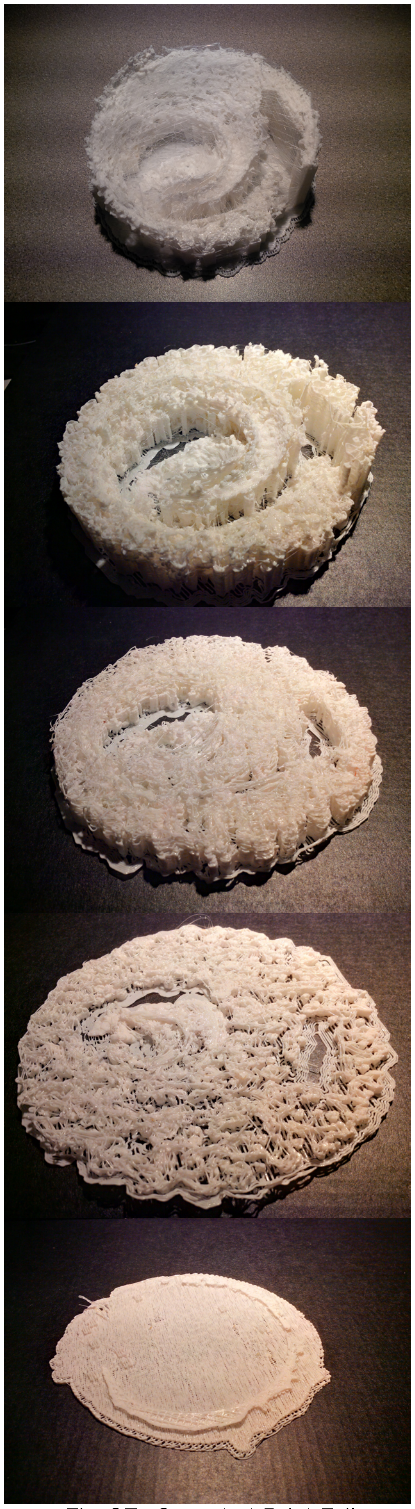


Fig. 87 - Snapshot Print Failures

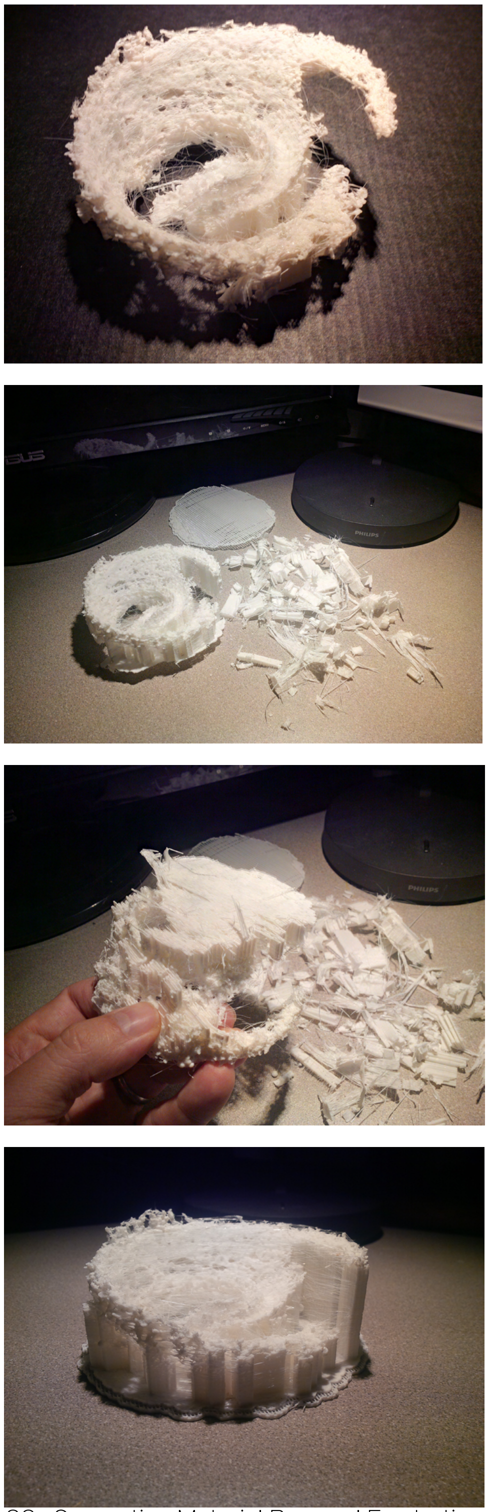


Fig. 88 - Supporting Material Removal Frustration

3D Printing

To successfully 3D fabricate any output, I needed to begin to understand the strengths and weaknesses of the printer I've assembled as a part of this investigation. Through iteration in the past few months I generated a number of wins and spectacular failures. The printer I've built melts a corn-based PVA filament and deposits it in a precise fashion on a roughly 8" circular bed. It does so layer by layer(in ~0.1mm increments) until a recognizable object is produced. An average print job takes 16-24 hours and requires considerable supervision. I've taken four separate snapshots of the above simulation for this submission and it has taken 3 continuous weeks to produce what you see here. Each is a moment that I feel is significant; each I've felt needs be seen in greater detail by printing or render.

While I am building in what is essentially an organic material, I recognise that this process is highly modulated by complexity and materiality. The melting point of the base material needs to be modulated throughout the process to ensure the print can complete. If the material is too cold initially it won't adhere to the base of the printer table. If it's too hot the material will leave wispy trails, clog up the nozzle, and expand physically, impeding the mechanical movement of the print head.

Learning from mistakes has been a large part of my process in capturing these wonderful snapshots of the falling and swirling forms successfully. One particular moment(opposite) "sings" to me and it's been a wonderful learning/frustrating process. The iterative output seen on the left was caused by a number of failures, both in the mechanical systems and in my chosen mesh parameters. Because of the intricacy and discontinuity of this generated form my printer tried to shake itself apart throughout much of this phase. I was constantly armed with a wrench and some loc-tite, hoping to catch issues on the fly and to rescue it from walking off its table.

The base design of the printer included the ability to incorporate two extruder heads. Each can ideally work within the same print to offer two material colours, or in my case, the promise of PVA support material. This material, when soaked in water, will dissolve, leaving the PLA plastic material behind. Ideally, this would afford the production of parts that require internal support during a printing cycle and would open up the machine's capabilities dramatically. Unfortunately, The second print head alignment became an issue within the print of the vortex(opposite) as it would collide with PLA that had expanded near the end of a print line run. The

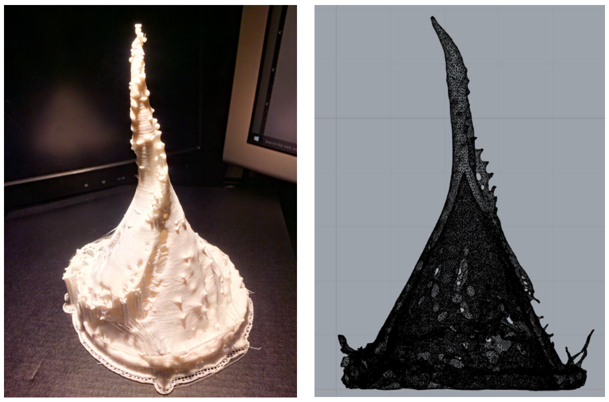


Fig. 89 - Whirl 1 - 3D Print+Model

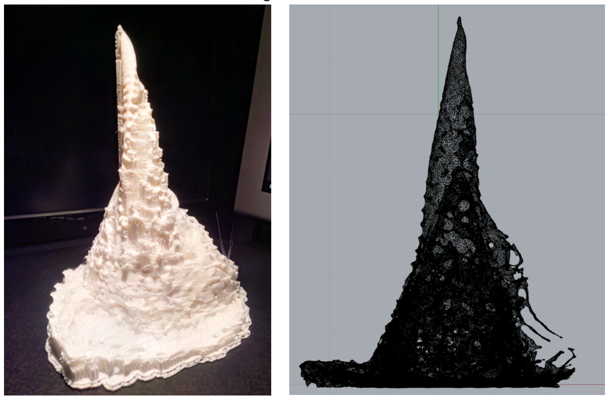


Fig. 90 - Whirl 2 - 3D Print+Model

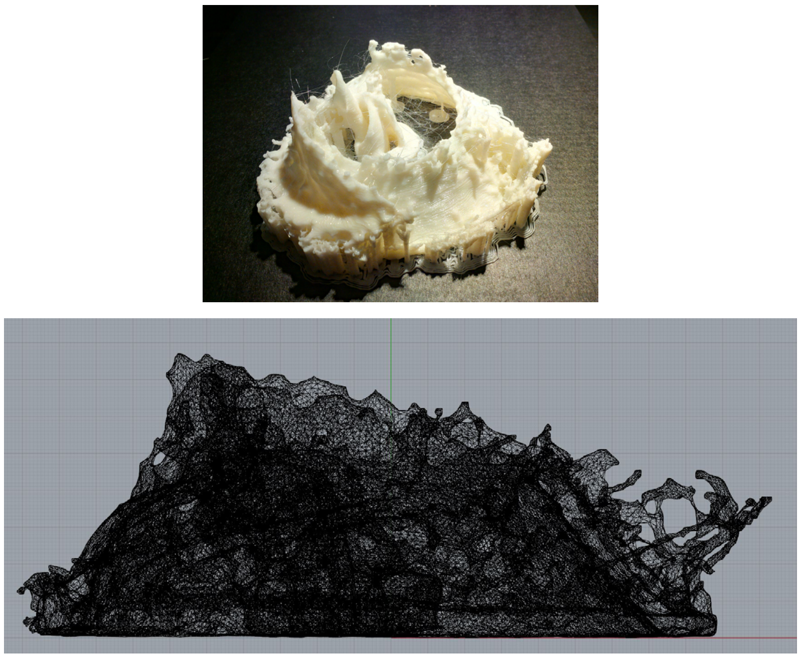


Fig. 91 - Plunging Moment - 3D Print+Model

effect being that the entire print head would be jarred and slip from its position or the entire piece itself would dislodge from the build platform entirely. After a handful of attempts, it became necessary to remove the head so that prints would complete. I have to leave the 2-head print system alignment for another day when the output time line is less critical. It's commissioning is no simple task.

This decision required that I use PLA as support material and that implied that there would be a lengthy process of support removal. Although the support process is governed by the slic3r software used within the Repartier host, there are some configuration parameters that allow for a rectilinear grid, a honeycomb pattern, or simple pillars to be used. All will do the job and all can be somewhat difficult to remove if not done while the print is still warm. PLA can become as hard as concrete when given a chance to cool down. The successful print seen in the images completed in the early hours of a night pass. I had to use a pair of pliers and cutters to remove the majority of support material. It seems timing post-print has quickly become another parameter to having a successful plot.

In addition to the mesh generation of Realflow, it became necessary to simplify and correct mesh-related issues outside of its generation. There were many instances where there were errors during generation, leaving holes and naked edges that would give Repartier issues when slicing. Rhino's mesh tools leave something to be desired, and as a NURBS system this was not wholly unexpected. I learned quickly to use the tool MeshLab for these types of issues. Built for the 3D scanning industry, it is a marvel at mesh repair and for re-meshing a model in preparation for 3D printing. It's simplification algorithms make it easier on slicing algorithms to determine the mesh surface clearly and this can be a boon on more complex models, like the ones produced within this investigation(.ie those not modelled from primitives).

While far from perfect, and only inches in size, the 3D printed models were able to give me some ideas concerning the form and its characteristics. There really is nothing like a physical model to help convey real-life properties of a wholly virtual set of ideas. The toolchain is a bit exhaustive and a little specialized, but after a few iterations, I was able to forget about the tool and delve into the form and its concerns. The tools became an enabler and faded more into the background of this investigation.

Toolchain:

VLC/AdobePremiere-->Realflow(Encoding Paths/Emitters)-->Rhino/MeshLab--
--> Repartier Host(Slic3r)-->Delta 3D Printer

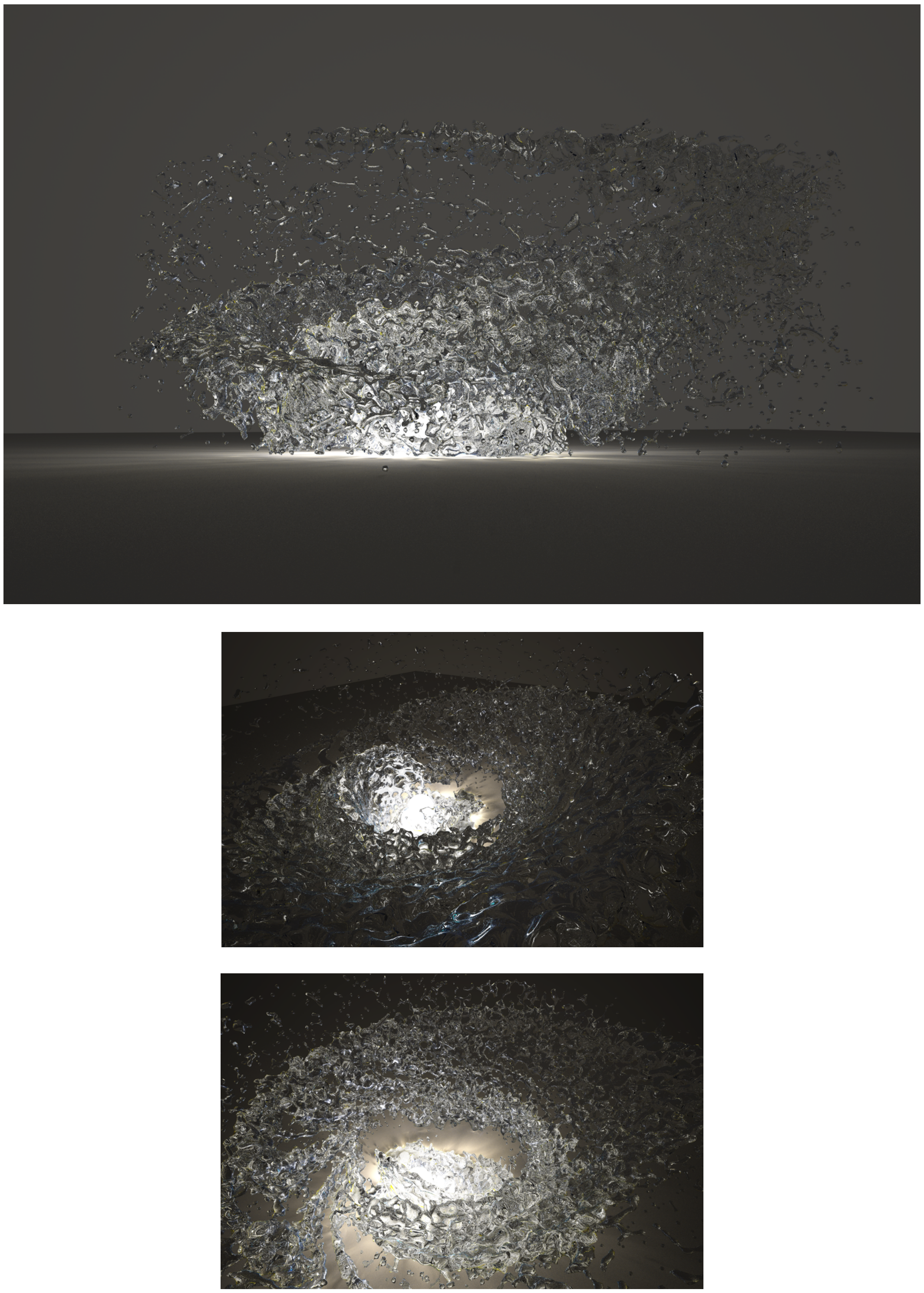


Fig. 92 - Snapshots in Time - Spacial Moments

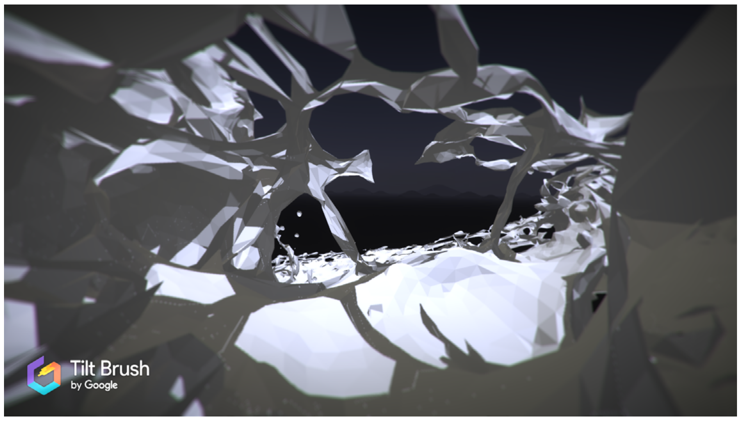
Renders & VR-Based Visualization

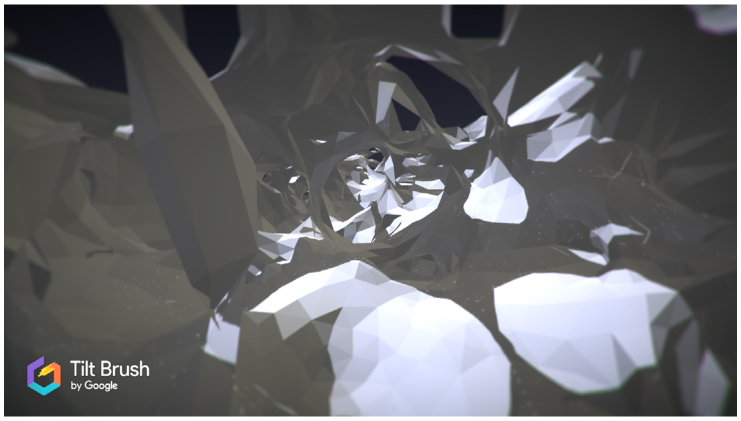
The 3D perspective is a powerful tool for communication. I chose to produce a render image of the vortex created by agents as they spin and bounce off of the ground plane when the dancer passes from a conscious state to occupying the unconscious. It's a reverberation of her mental state reflecting off of a place she can't see directly. Adding a central light may have been a stretch, because the majority of the action happening here is at the vortex's edges; there the emotional energy is at its highest. She holds this together however, and that in my mind requires a level of control that comes from a central place. The render was meant to show this control, and it's stasis - a measure of demonstrated ability for all to see.

All of this is manufactured. It starts with my interpretation of the mental state of the dancer and ends here with the visual constructions seen opposite. Clearly the views show a situation that is ephemeral and I feel what we see is meant to capture, additively, the result of her mentally moving through a process of crisis management. I chose a transparent material, not to re-enforce the fluidity of the situation, but more to allow each piece to offer up its place in this collective torment.

I'm not proposing that a building be built to embody the characteristics of the investigation. In an Architectural program that can be the default question of all formal investigation. I started this portion of my work wanting to pull on the human element to seed a generative system of investigation using digital methods. I will use what I find here in my future career as an Architect to find a way to express the human element within my designs. In this exercise, I'm left with a simple gesture and sometimes that is simply enough.

I can't build this thing because of its temporary nature. Also, because much of the form is not connected. Without physical connection it might be hard to build. While this didn't stop Diller Scofidio and Blur(Diller Scofidio + Renfro 2002)





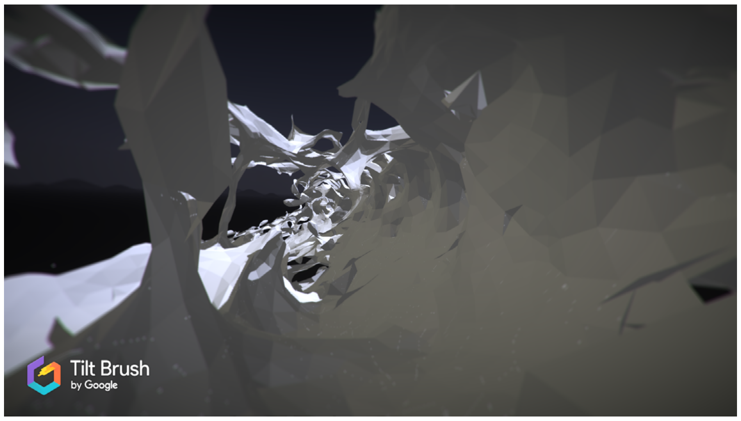


Fig. 93 - VIVE/TiltBrush VR Experiences

VR-Based Visualization

OK. I've been dying to take some of the model work in this investigation and actually climb into it. I can't easily build it. It's not very easy to print a 3d model of it. Perspective views are good and can be insightful. But nothing compares to actually being inside.

I had been using a toolchain for VR-based visualization that was fairly time consuming when trying to activate a design from Autodesk's Revit:

Revit-->3d StudioMax(grouping/materiality/scale)-->Unreal 2 Game engine(Collision/Hosting)

This would work for most any model but it could take an hour or more to go through these steps. 3D StudioMax was necessary to produce the ".vbd" file that unreal needed for it to recognize the mesh as an "asset" that could be included in a game instance. I would normally start a project by using a based template that included a first-person avatar and I would build upon this. Time consuming, but at least the output could be simply an portable executable that would allow anyone with an Oculus or HTC Vive to walk through the design.

Once Google got into this business with their "Cardboard" product things have begun to get quite exciting. As a showcase product, they produced Tilt Brush. It's a fully interactive 3D immersive VR drawing program. My first experience with it with my Vive was transformative; placed in a dark-ish open space, I was able to pull from a pallet of tools to draw in ink,paper, vibrant neon light...you name it. Their addition of procedural brushes reminded me of some advanced Photoshop tools. This real was a game-changer for me.

It wasn't until late fall when the folks at Google added the ability to import external meshes into the environment. It was limited to a maximum size, but I knew I could use Meshlab to reduce the size of my meshes to fit, or at least a portion. I took one of the more cone-shaped tower models that I was able to output through to the 3D printer and with just a little bit of simplification, was able to import it into Tilt Brush. Finally, I was able to see what the space inside was like. At maximum size, the perceivable height was probably 20 or so meters. I was reminded of some investigations by Greg Lynn in his NOAH work for the film DIVIDE(Lynn 2004).

Since Google's Tilt Brush, a number of interactive tools have come onto the scene. Kodon is billed as a 3D sculptural tool where an artist and add and remove material from a primitive-created mesh or from a small imported mesh. It's still alpha soft-ware, so there are lots of bugs, but it shows tremendous potential. These tools are getting close to allowing a person to interact directly within a virtual space on their project. I see a native Revit tool that will allow an architect to design within this space not far off. There's already a direct interface "service" called "Iris Prospect" that ties directly into Revit and it allows for some parameter alteration of the model in place. Similar tools are floating to the surface for Max and Rhino. Very exciting.

Results

In this section, it was my intention to investigate form creation through the mechanism of software agency. Initially, I had hoped to apply this method to the problem of papillae bridge formation within a diatom fustule(valve), then extend this to an entirely new problem involving a formal interpretation of a moment or two within a modern dance segment choreographed by Pina Bausch.

Concerning the second portion of my Agency investigation, I had a very difficult time looking for form within the *pina* dance segment I loved so much. It's not that the segment wasn't full of inspiration, or wasn't delightful in its own way, but deriving form from an imperception(its affect) is new territory for me as. Affect is that which exists before a response, this includes awareness. As Pina Bausch looked to understand what moved her dancers, I decided to engage at this level and interpret this myself, hoping to pass something on to my own.

Form generated from my perception of the mental state of a dancer requires some form of language, and like Pina, would need some method to guide and record my agent's performance. I began a couple experiments to test the waters: One investigating the mechanisms of flocking, and the other of altering space itself.

The flocking simulation was fairly straightforward and Processing's Punkiert library afforded "Boid" classes that would help to manage a population of agents. The physics engine made small work of the effort however these libraries were easy to overwhelm. It didn't take much for the resultant simulation to become chunky. Boids within Processing worked in a limitied sense and really only with small populations.

I chose to extend the test and write my own system to manage flocking behaviour. This effort turned into a significant work. To simplify the investigation, I elected to make an assumption: Space imbues affect. If I could encode affect within space itself, maybe I could entice the agent's environment to guide their behaviour. It was stretching a few things, but I think it is essentially sound. I then made a another simplifying assumption as well; working within spherical coordinate would simplify everything. From defining an "Affect Space" to easier agent status calculations, this method might make the test a bit easier to implement.

Affect Space is essentially an implementation of a Tensor field(a vector quality defined within space) that I would drop agents into. It took quite an effort to implement a system that would manage a population of agents that could exist within its system, and while the results showed promise there were a few hurdles that proved to be insurmountable.

The system became very difficult to control, even with a small number of agents. I had implemented the system in such a way to cause agents to essentially take themselves out of play if their position was outside the region defined. This process would then cause the system to "re-spawn" a new agent and place them randomly within the defined affect space, a strategy used in the Diatom-Coulomb simulation. This became an issue when inconsistencies within the tenor field would affect the agents in too great a manner.

Issue in defining "Affect" within the tensor field became the governing problem. As I had for the purposes of the investigation hard-coded the harmonic functions defining this quality. The system became simply too difficult to manage. It was with great sadness that I decided the experiment had run its course. While the system was significantly better able to handle more agents than the built in processing libraries, there still weren't enough and they became too difficult to handle. I made the decision to apply RealFlow to the same problem.

NextLimit's Realflow proved to be a dream, offering all that I had hoped when it came to managing agents. Their key-framing environment would allow me to first define a "language" based upon affectual components, and it would allow me to modulate these affects within an environment that could host ten-times as many agents. RealFlow as it seems was written to use the GPU(Graphics Processing Unit) of my PC to accelerate calculations. As GPUs are made of specialized hardware designed to manage a tremendous number of parallel calculations, it was a boon to my work that I could essentially make use of the equivalent throughput of a supercomputer to manage the agents I create. This sped up simulation time and precision considerably.

After working through a few exercises to become familiar with NextFlow's programming(which utilizes a graphical interface), I looked to produce a language of affect that I could draw on during this sections effort. Working through all of RealFlow's force sets, I chose to choreograph physically the path the dancer, Ruth Amarante, takes while she emotionally emits particles caught up within the vortex of her thoughts. The result proved to be interactive and quite engaging. Realflow would allow me imbue spirit through modulation of affect. The resultant form, made from 200,000+ agents is able to self-interact and resulted in a number of iterative animations. When a parametric meshing process was applied to the agents, a moving, self-interacting form was produced. It was significant moments from this that I chose to pull from when looking for form.

Much of the output of this work resulted in form that was discontinuous and very ephemeral. It put my 3D printing process through great pains to reproduce. The printer had to be re-tightened and calibrated between every print because the vibration was so bad. There were very few continuous sections within the optimized mesh and this caused the printer to have to raise and lower itself repeatedly for every layer to capture all the detail.

The forms could also be experienced through render and VR-based visualization. I was able to optimize the mesh from several moments and import them into Google's TiltBrush and Unreal-based VR environments.

Conclusions

This section of my thesis was thoroughly the most engaging for me. To delve into software development and alternate inspirations for digital form was really rewarding. I had hoped to get some sense of what was involved in the software agency seen in deskriptiv's work and I got this in spades. It is really difficult to build a software system, let alone build one that needs to be so interactive and dynamic.

This investigation highlights an important part of digital tool making, the trial. I built two software tools(Boid and TensorFlow) to see whether it made sense to build this functionality from scratch. Both were learning investigations and neither ended up being used for the final form development because of complexity and implementation issues. If time permits, this type of low-level investigation is absolutely necessary in distilling the true nature of the problem. While there may not always be time in a architectural practices to follow paths this way, I believe its instrumental(literally) in gaining understanding at a digital level. I wouldn't label them as failures in any way, more but forks in the road. No paths are bad paths. It's simply important to best fit the right tool for the right job. I learned here that these tools need a tremendous amount of improvement to perform as I had hoped, more improvement than I was able to provide at this stage of my technical competency.

RealFlow's system was the best way to handle the larger number of agents within a physics-based environment. It's GPU-accelerated physics solver allowed me to reach a level of formal precision that I felt necessary while imbuing the intended spirit within the form. There were issues of control and feedback, but I was able to adjust.

It was far more rewarding to be the author of all controlling logic of the tool during the Boid and TensorFlow tests. In building each tool, I was able to work within a similar framework as I did for the Diatom geometric formal investigation, but with one difference. In each I had greater understanding of the low-level operation of most of the formal algorithms. The Boid investigation utilized some libraries within Java and the results pointed me to a more direct, home-grown implementation being necessary with TensorFlow. With the later, I built all functionality from the ground up, only depending on basic graphic libraries for its function. The reward was instant, I was able to increase my agent-count dramatically when having greater control. And while ultimately I had to put its development on hold, it points the direction for future development of that approach, which I feel is ultimately sound. It's important to take risks and reap the benefits of lessons learned; this being a great example of this.

Of Materiality

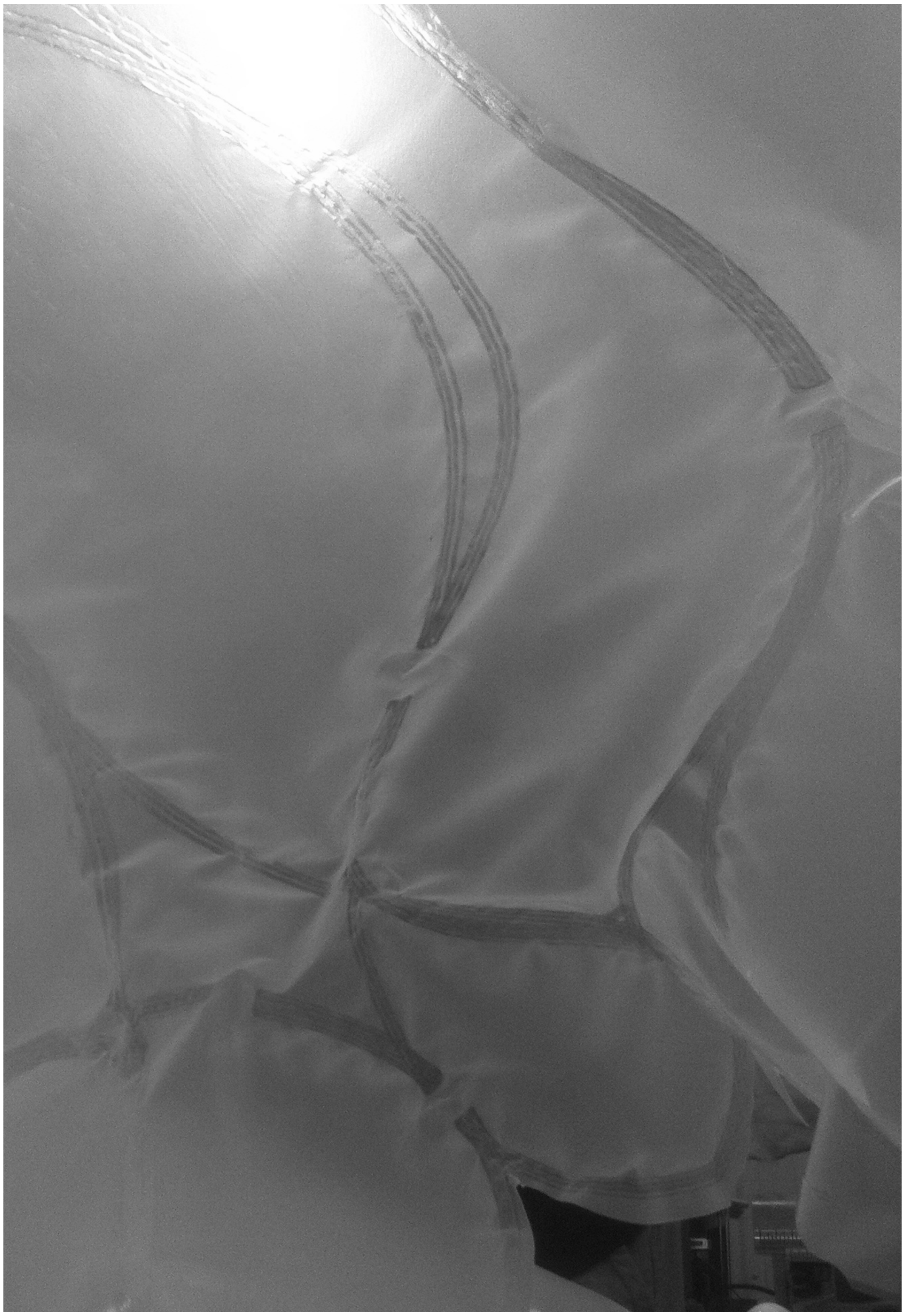


Fig. 94 - Pneumatic Laser Output

Introduction - Material Investigation

To design in a material is to form a relationship with it. Too often we as architects and designers spend much of our time working within a virtuality, a place *derived* from a reality but with its very own and separate character—a tabula rasa. A case could be made that we all see ourselves within a space derived from our own very personal perception of what is real(and what we see as true), but very seldom can this model push back and inform; after all, our perception can only be a subset of the actual true reality, and how often do we surprise ourselves within a model we live within.

An object's materiality presents the real essence of its physical character. Given the chance, its organisation and elemental composition create an analog computer that can resolve any potential physical situation. As in Newton's laws -- "Action/Reaction, Opposites are Equal," Richard Sennett's Craftsmen develop their craft as their material forms them while they form their materials. His chef works tirelessly to be able to cleave a single grain of rice(Sennett 2008,167), meanwhile the rice grain works on the cleaver, while the cleaver works on the chef. No matter the tool, the material of the crafts imparts its lessons on the craftsman.

In this section, I will be searching for an understanding of how materiality informs design from within a digital framework. This section is in two parts: the first documents the selection, assembly, and commissioning of a Delta-based 3D printer, the second documents a pneumatic formal investigation as part of a group project in Digital Fabrication during the summer of 2016, I documented my experience working with 6mm polyethylene as it was formed into iterative models of a self-supporting pneumatic structure. As with previous sections, a digital investigation requires that we build tools using software and hardware. This section does both, and asks: what does materiality contribute to the design process? How does it affect the relationship between head and hand of the digital craftsman?

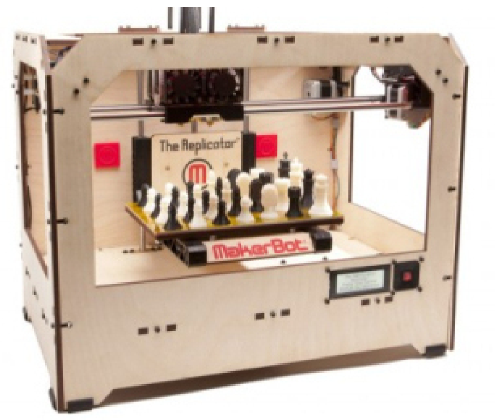


Fig. 95 - Makerbot - Replicator

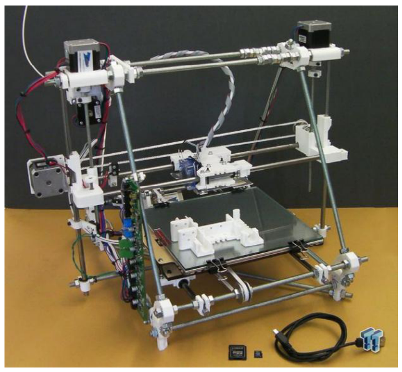


Fig. 96 - RepWrap Mendel

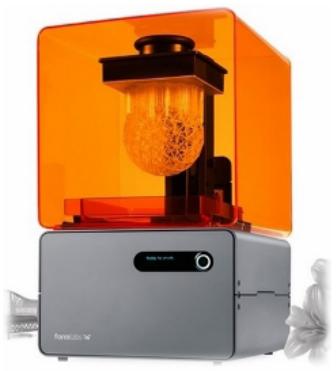


Fig. 97 - Formlabs - Form 2

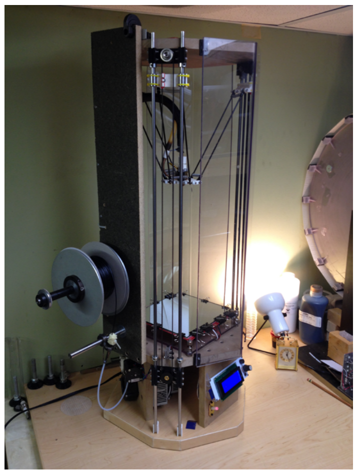


Fig. 98 - Delta Rostock

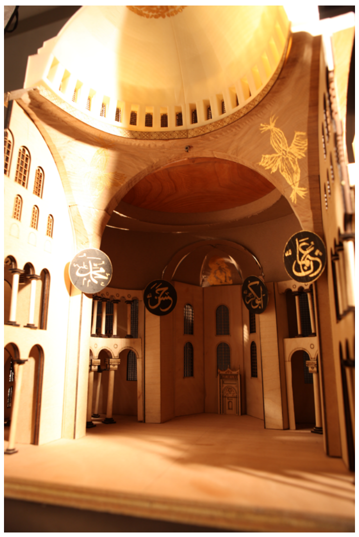


Fig. 99 - Hagia Sofia Model - Group Project 2012

Making

My first hands-on exposure to 3D printing was at a friend's place. It was the first iteration of Makerbot's *Thing-O-Matic* back in 2010. Their printer was 100% open source and was hand assembled by him. It had its issues(bed levelling, extruder clogging, speed, model detachment, horrible software...) but this machine could extrude virtually any model that could exist within a four inch cubic volume. These additive machines seemed quite simple really(they are essentially a computer-controlled glue gun). It wasn't until my third wide-eyed visit that I asked the question, "why is this thing on your stove in the kitchen?" He said the PVC filament stunk to high heaven and it was the only way it could function in the house(with the vent running).

The *Thing-O-Matic* was one of many printers that floated on a collective dream, "a printer that could print itself"(RepWrap *Mendel*). This was never actually realized, but the dream does continue. While these machines are mainly made of plastic, there are metal parts(extruders & mechatronics) and the electronics(control) that are simply too complex and varied in materiality and scale. But it is a nice dream.

For this project, I wanted to learn from my friend's lessons in building and maintaining the *Thing-O-Matic*. I had also purchased a Makerbot *Replicator-2* myself as well back in 2012. I had high hopes for it, but in the end it was as unusable a printer for me as the *Thing-O-Matic* was for my friend. They both were able to produce but not without many hours of supervision and countless failed prints. I ended up giving my printer to him for parts. The hobby market was simply not ready for prime time...

With two attempts making, it would have been easy to toss in the towel. There are better commercial options out there, but I couldn't help but think that they were heavily overpriced. The Stratasys *Dimension* printer that the UW School of Architecture uses in their shops is dependable but very expensive(capital and operating). To print the 8" dome for a model of the Hagia Sofia needed for a group project in 2012, we had an outlay of $800. Broken into halves due to size, each half of the dome took roughly 13 hours to print. The end result came out perfectly, but the cost was staggering. New, these "closed system" printers cost approximately $35K($12K used) and have a sizeable on-going cost as there is a filament dissolving bath system to maintain and the inflated charges of proprietary filament cartridges.
After four years, I decided to look again to the market to see if things had im-

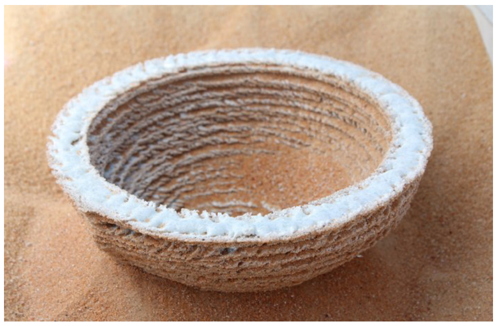


Fig. 100 - Solar Sinter - Markus

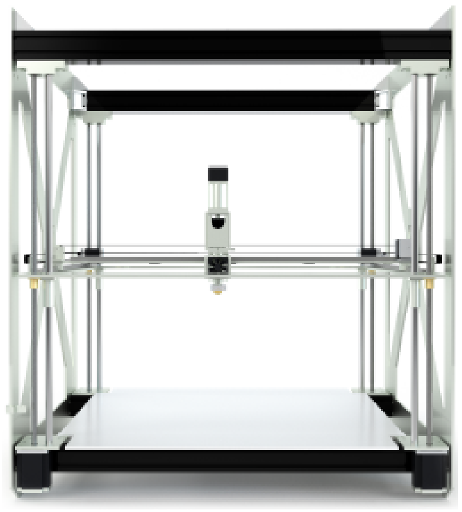


Fig. 101 - Genki -Arki

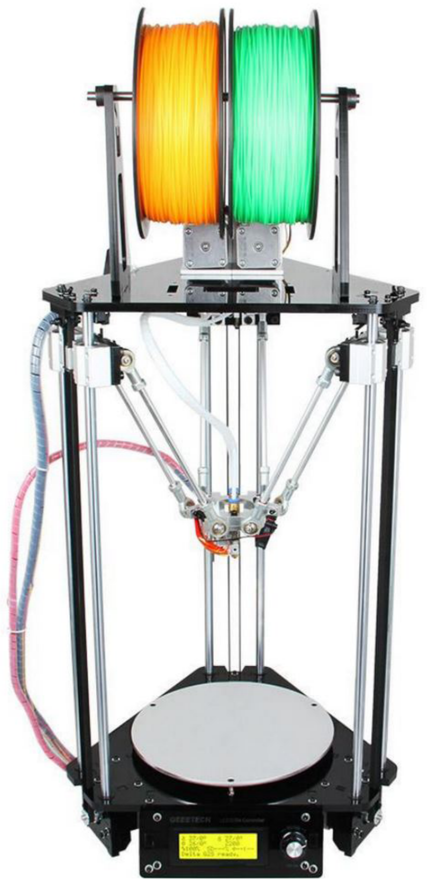


Fig. 102 - Geeetech Delta Rostock G2s Printer

proved for the home user and this project. Makerbot had since been purchased by Stratasys and more than doubled their prices(and now offering "closed" multiple offerings), Kickstarter began funding a host of new 3D additive systems(Formlabs' *Form 2* resin system, which "pulls" the new piece from a laser solidified polymer bath, being a notable addition), the Delta Rostock design seems to have come into its own with many variations(many still open source), and scale has been pushed with the printer from Genki(with a $5000 price tag but a cubic meter build volume from Japan). Markus Kayser's *Solar Cutter* has also evolved to become his *Solar Sinter* printing in silica using the sun's energy. These systems have seen a coming of age.

Choosing

Looking for a more cost-effective option than anything Makerbot, I was intrigued by the Delta design(pictured opposite). Its inverted-tripod arrangement for control uses the physics of the lever to move the extruder heads very quickly, and the use of a Bowden extruder(filament feeding stepper-motor separated from hot extruding end) dramatically reduces the mass of the actual extruder platform.

Reduced mass equals less momentum(Newton's second law - F=m*a -> F=m*dv/dt -> F*dt = m*dv) and this results in less of an impulse(F*dt) needed to be produced by the steppers to move the platform.

Although a more standard (a.k.a Makerbot) Cartesian gantry system is easier to setup and calibrate, the steppers in that system will always have the mass of both gantries to deal with, resulting in more energy needed, and thus more time per move. As this design is quite innovative, and for reasons stated above, I opted to investigate a kit from Digitmakers in Richmond Hill.

I read the reviews.

I saw the annotated walk-throughs(Painless360 2015).

I saw that there was lots of availability(Digimakers).

I liked that the price was right($338US/$499Can).

I liked that it was based on the Arduino Mega2560(open-source processor with open-source Java Integrated Development Environment(IDE)).

I was sold.

Fig. 103 - Delta Assembly Mayhem

The Build

This is a kit. I gave serious thought to stretching an existing design(realizing a not-so-uncommon dream to have the cubic-meter build volume of the Arki), but the time simply wasn't there within this investigation, nor was this its focus. I had to keep reminding myself that this was about the path not the size of the hill.

Instructions didn't accompany the package. Neither was there a direct support line to Geeetech themselves, rather just a pointer to shared forums. Both seem to be the trend lately. Following a link off of the sales page, it took a few minutes for me to find myself in front of the assembly pdf. It included many photos and instructions that referred both to their website and Youtube walkthroughs to enhance the process. It's clear that this is a continuously evolving design. For the most part, there were only a few issues. Funny, I remember when a booklet and phone number were standard when buying most sizeable products; I guess this doesn't scale, at least not at this price-point.

Officially the cheapest tool in the world

Fig. 104 - Impacter Tool Provided for Assembly

All small components came in numbered bags. I spent much of my time just trying to keep things organised as the bags were ordered by part type, not stage of assembly. I could see that if there wasn't some form of organisation, this would quickly reduce to chaos… and based on the dependability of the instructions, the probability of damage, because of mating incorrect components, was high enough. This had to be done slowly, and carefully.

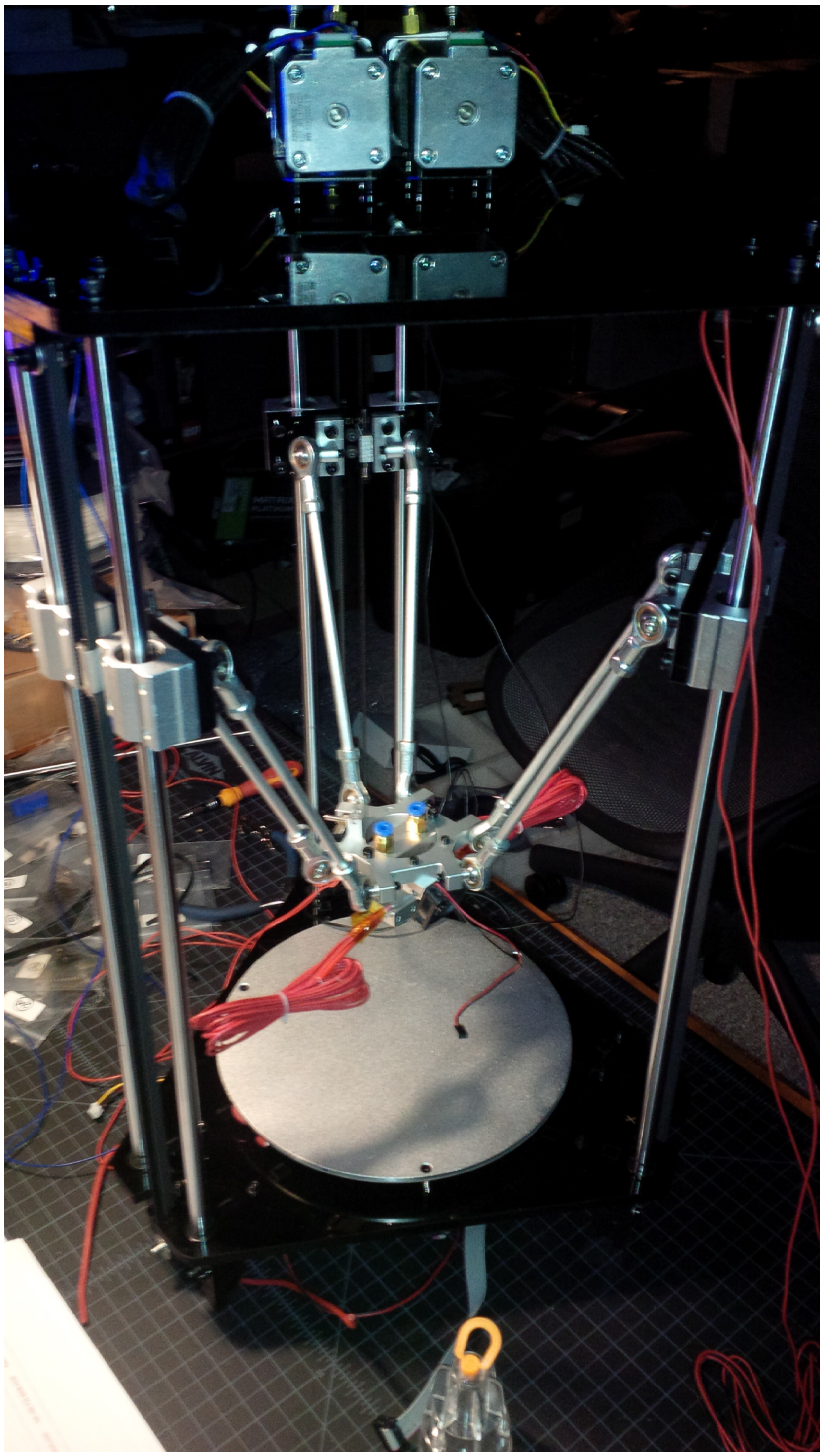


Fig. 105 - Geeetech Delta Rostock G2s Printer - During Assembly

The kit came with a multi driver, and is in my experience, the cheapest tool in the world. If there was one single-most effective way to further the cause of chaos, this tool is it. I've scoured markets far and wide to ensure that I have the right tools for most jobs(finding the best in as far as the electronic markets in Akihabara, Tokyo) and I can spot the truly evil ones. This one truly leads the pack...

A bad tool, is almost guaranteed to:

break - causing damage to yourself or the assembly
bend - ruining the part(negating and future chance at adjustment/disassembly)
slip - damaging your work surface, yourself, or the assembly

Despite good intentions, and even the best instructions, things don't always go together as planned. I can see that when creating a bill of materials, capturing the sequencing correctly can be a challenge, specially when the assembler applies their experience and know-how of the way things "should" go together when doing their part. How an assembler thinks should be taken into account...

My issue during this process is that bad tools ruin the machine during assembly. In this specific case, the tools supplied have to be used because the fasteners seem non-standard and only able to marginally mate at the best of times with what's provided. This thing is a jigsaw with bad-fitting interchangeable pieces, and I am assembling it with boxing gloves on. Frustrating at best. Fitting requires assembly and partial disassembly, over and over; meanwhile, the provided tool is working hard to distance its mate as much as possible through wear. At limit, I had to find fasteners (proper ones) and customize them to meet the immediate need. I consider this a partial fail of the system(and assembler); I'm part of this, and sometimes I find it difficult to get my head around the thinking of other designers.

Things that impress me about this physical design:

Metal bushings and linear bearings have been used
There's room for minor adjustment in many areas, but not too much
The result is rigid and very stable
Connectors are idiot-proof and are keyed uniquely based on function
Fastener torque settings are unnecessary - snug is usually good enough
Many parts could be replaced with other off-the-shelf common units

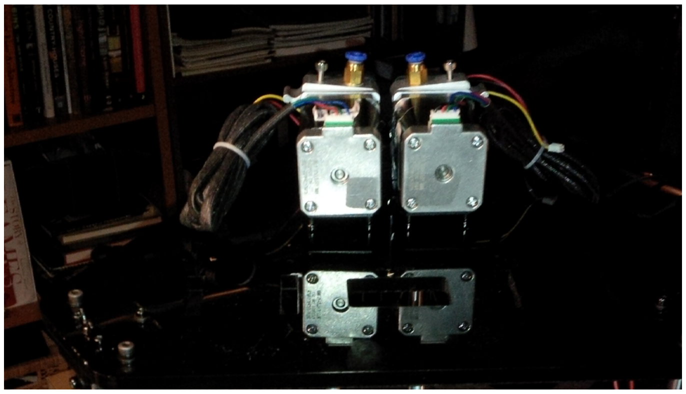
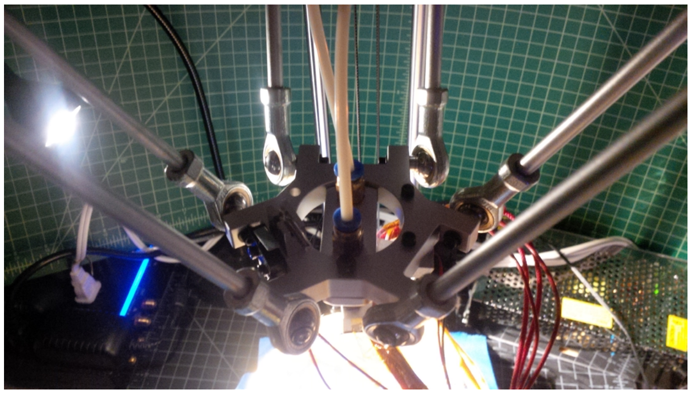
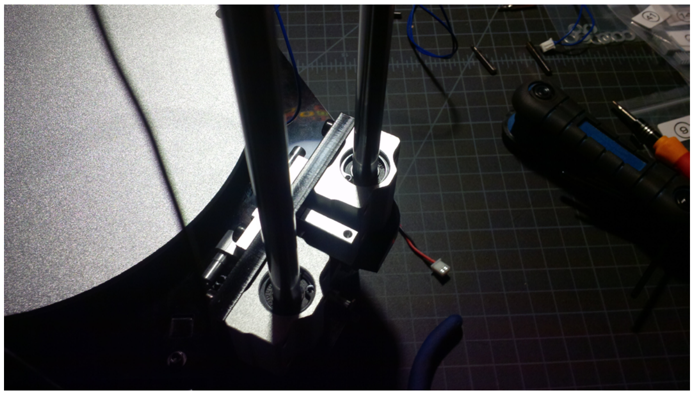




Fig. 106 - Geeetech Delta Rostock G2s Printer Moving Parts

Things that do not:

Some critical areas have too much play
Wire management has not been thought through
The dual-head design is very difficult to adjust and calibrate
Use of some custom fasteners is problematic and unnecessary
Bed levelling is mechanically quite difficult

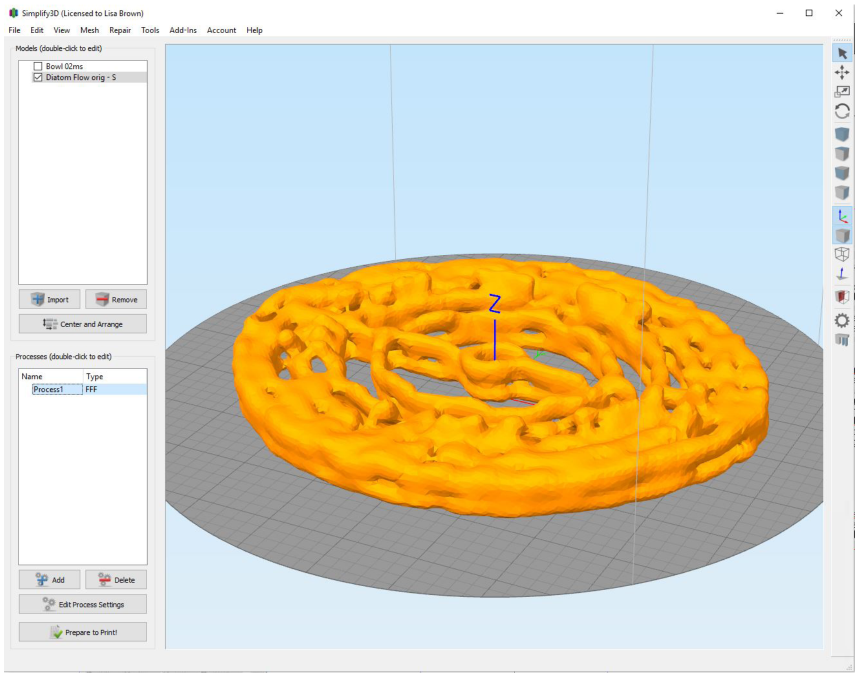
Despite the issues, the printer's build went well. No reviewer vocalized similar experiences. I suspect they simply didn't vocalize these annoyances and considered them par for the course on such a cheap printer; I'm not usually so forgiving and would normally put my own build video together and share it so others might benefit.

There were three outstanding design features that struck me as quite exceptionally implemented in this design: the extruder system, the print head platform system and the axis actuation system. Each I consider very well though-out and I think it necessary to discuss their merits here.

The filament extruder is doubled up to allow for either dissolvable support filament delivery or to simply have another colour or type of material within the same print session. A standard extrude consists of a stepper-driven worm-gear, a stiff flexible pipe for the filament to travel through, a hot end(electrically heated nozzle), a temperature thermistor(to measure the heat level of the nozzle), and a small fan to keep the cold-end of the nozzle cool.

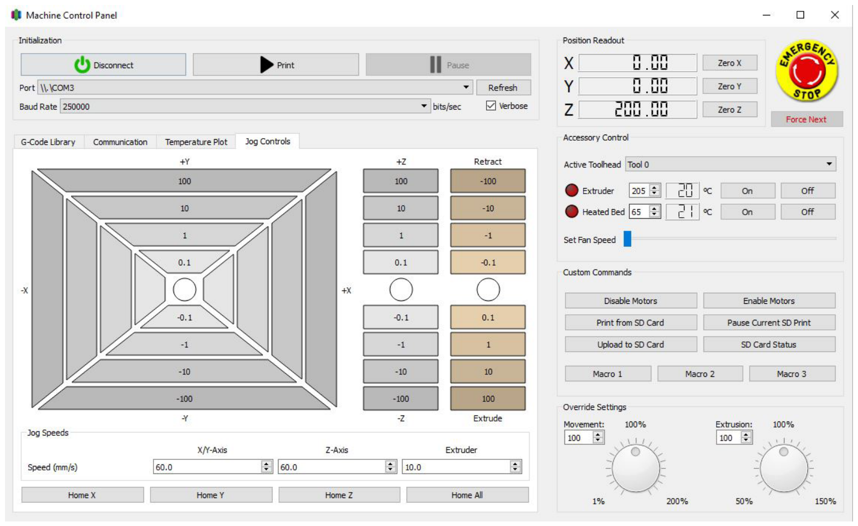
This extruder system separates the feeder from the hot nozzle to reduce mass at the platform and it is the stiff plastic filament guide-pipe that makes this possible; it contains the applied pressure from the feeding stepper so that the filament can be delivered in a very precise manner(during a print, the filament is pushed and pulled back when needed and is at all times metered out to meet demand). This configuration is known in the industry as a *Bowden* extruder and Geeetech has chosen to use all metal parts on the hot and cold ends to ensure dependability.

The print head platform is all aluminium, lightweight, and particularly rigid, with a somewhat isolated area at its centre for a separate hot extruder end. The platform was initially made of 3D printed plastic in previous iterations(from the Repwrap printer printing the printer days), because of the heat levels at the centre and the forces exerted on it, this proved to be quite problematic over time. With six ball-hinge connections all pulling on this part at great rates, it didn't take long for the


Simplify3D (Licensed to Lisa Brown)
File Edit View Mesh Repair Tools Add-Ins Account Help
Models (double-click to edit)
Import
Remove
Center and Arrange
Processes (double-click to edit)
Add
Delete
Edit Process Settings
Prepare to Print!
Z

Bowl 02ms
☑ Diatom Flow orig - S

Name	Type
Process1	FFF



Port \\.\COM3
Baud Rate 250000 bits/sec ☑ Verbose

X 0.00
Y 0.00
Z 200.00

Active Toolhead Tool 0
Extruder 205 20 ºC
Heated Bed 65 21 ºC
Set Fan Speed

Jog Speeds

	X/Y-Axis	Z-Axis	Extruder
Speed (mm/s)	60.0	60.0	10.0

Movement: 100% 100
Extrusion: 100% 100

Fig. 107 - Simplify3D Software - Diatom RealFlow Output Test

plastic version to be literally pulled apart even under nominal loads. The aluminum version added very little difference in weight, could easily deal with the stresses, and as an added bonus wicks, away any extra heat transferred from the extruder through the fasteners and radiatively.

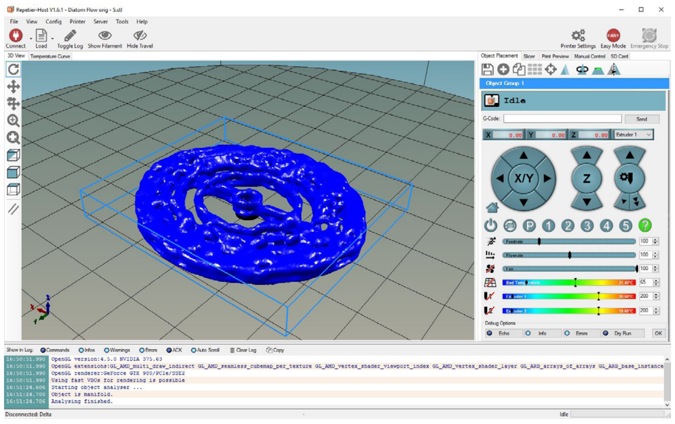
The platform is moved by aluminum rods that are connected to a pair of linear bushings which slide on steel rods. These bushings are moved by a drive-belt that is held in place by a couple aluminum pulleys, one of which is driven by a stepper. As the belt moves the bushing assembly, this changes the resultant distance of the control arms to the platform. Modulate these from three equiangular directions, and the platform can be put at any place within the build platform and at any height with alarming speed. An all-metal choice here was the best. With even small variances in the assembly's configuration, the positioning of the platform can be very precise. These parts also experience minimal wear with the vibration caused by the constant repetition of printing, layer by layer.

Printing - Software

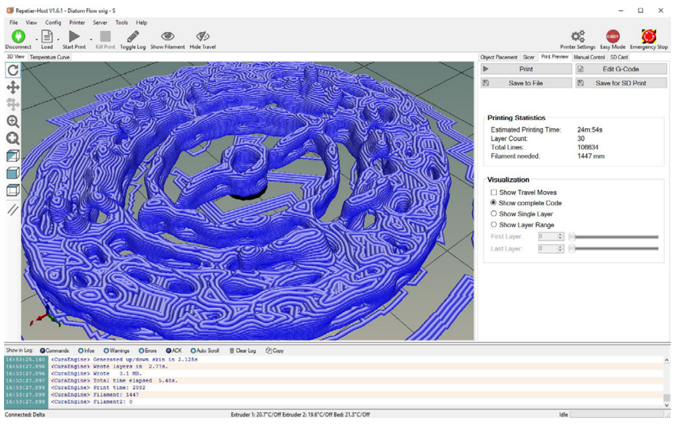
Something that is always forgotten with many hardware systems is the software that is needed to make it go. Often, this is the part of using the product that makes or breaks the whole experience and it's hardly given even a second thought when sizing the solution. Most don't think much about the printer driver for your standard paper-based printer, but a bad driver can render the device useless if it doesn't support your workflow. In this case, the printer came with nothing. Yes, you heard that right; this printer leaves you to your own devices - opting instead to point you at a few public-domain options... The rest is up to you. In a sense, this was perfect for this investigation because it didn't tie me to any one software manufacturer and I could choose the best fit, not tied to any one architecture. For a more inexperienced user, this could spell disaster.

Functionally, this software needs to:

- Take an air-tight 3D model(usually in STL or OBJ format)
- Determine if any of portions of the model need support
- Adding removable "feet" to the model (as necessary)
- Slice the result into layers
  - Distil paths for the print head to follow per layer
- Convert the these paths into "G-code" used to move the print heads
- While printing it must


Repetier-Host V1.6.1 - Diatom Flow orig - S.stl
File View Config Printer Server Tools Help
Connect Load Toggle Log Show Filament Hide Travel
Printer Settings Easy Mode Emergency Stop
3D View Temperature Curve
Object Placement Slicer Print Preview Manual Control SD Card
Object Group 1
Idle
G-Code: Send
X/Y Z
Flowrate Fan Bed Temperature Extruder 1 Extruder 2
Debug Options Echo Info Errors Dry Run OK
Show in Log: Commands Infos Warnings Errors ACK Auto Scroll Clear Log Copy

```
16:50:51.990 OpenGL version:4.5.0 NVIDIA 375.63
16:50:51.990 OpenGL extensions:GL_AMD_multi_draw_indirect GL_AMD_seamless_cubemap_per_texture GL_AMD_vertex_shader_viewport_index GL_AMD_vertex_shader_layer GL_ARB_arrays_of_arrays GL_ARB_base_instance
16:50:51.990 OpenGL renderer:GeForce GTX 980/PCIe/SSE2
16:50:51.990 Using fast VBOs for rendering is possible
16:51:24.606 Starting object analyser ...
16:51:24.700 Object is manifold.
16:51:24.706 Analysing finished.
Disconnected: Delta                                   Idle
```


Repetier-Host V1.6.1 - Diatom Flow orig - S
File View Config Printer Server Tools Help
Disconnect Load Start Print Kill Print Toggle Log Show Filament Hide Travel
Printer Settings Easy Mode Emergency Stop
3D View Temperature Curve
Object Placement Slicer Print Preview Manual Control SD Card
Print Edit G-Code
Save to File Save for SD Print
Show in Log: Commands Infos Warnings Errors ACK Auto Scroll Clear Log Copy

Printing Statistics

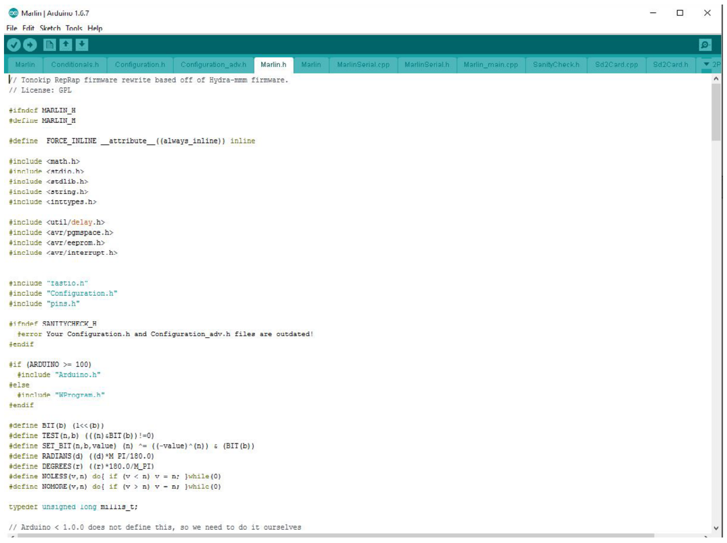
Estimated Printing Time: 24m:54s
Layer Count: 30
Total Lines: 106634
Filament needed: 1447 mm

Visualization

☐ Show Travel Moves
● Show complete Code
○ Show Single Layer
○ Show Layer Range
First Layer: 0
Last Layer: 0

```
16:53:25.140 <CuraEngine> Generated up/down skin in 2.125s
16:53:27.896 <CuraEngine> Wrote layers in  2.77s.
16:53:27.896 <CuraEngine> Wrote   3.1 MB.
16:53:27.897 <CuraEngine> Total time elapsed  5.48s.
16:53:27.899 <CuraEngine> Print time: 2082
16:53:27.899 <CuraEngine> Filament: 1447
16:53:27.899 <CuraEngine> Filament2: 0
Connected: Delta          Extruder 1: 20.7°C/Off Extruder 2: 19.6°C/Off Bed: 21.3°C/Off          Idle
```

Fig. 108 Repartier RealFlow Output Test


Marlin | Arduino 1.6.7
File Edit Sketch Tools Help
Marlin Conditionals.h Configuration.h Configuration_adv.h Marlin.h Marlin MarlinSerial.cpp MarlinSerial.h Marlin_main.cpp SanityCheck.h Sd2Card.cpp Sd2Card.h

```
// Tonokip RepRap firmware rewrite based off of Hydra-mmm firmware.
// License: GPL

#ifndef MARLIN_H
#define MARLIN_H

#define  FORCE_INLINE __attribute__((always_inline)) inline

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>

#include <util/delay.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>


#include "fastio.h"
#include "Configuration.h"
#include "pins.h"

#ifndef SANITYCHECK_H
  #error Your Configuration.h and Configuration_adv.h files are outdated!
#endif

#if (ARDUINO >= 100)
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

#define BIT(b) (1<<(b))
#define TEST(n,b) (((n)&BIT(b))!=0)
#define SET_BIT(n,b,value) (n) ^= ((-value)^(n)) & (BIT(b))
#define RADIANS(d) ((d)*M_PI/180.0)
#define DEGREES(r) ((r)*180.0/M_PI)
#define NOLESS(v,n) do{ if (v < n) v = n; }while(0)
#define NOMORE(v,n) do{ if (v > n) v = n; }while(0)

typedef unsigned long millis_t;

// Arduino < 1.0.0 does not define this, so we need to do it ourselves
```

Fig. 109 - Arduino IDE and Mega2560 Firmware Config

Manage rate of G-code execution
Manage temperatures of the extruder ends
Manage temperature of the print bed
Manage flow rates of extruder heads

This is quite a lot to do well! Some of the more popular software options available are: Printrun, Repartier, ReplicatorG, and Simplify3D. Windows 10 comes with its own host software built-in, but it's very limited in its support of only smaller commercial cartesian gantry printers. I opted to give Simplify3D and Repartier both a try. Simplify3D has great user reviews but costs($150US/$200CAN), and as a more hands-off solution, I am interested to see if it's mac-like "just works" reputation holds. Conversely, Repartier is fully open-source and depends on the Cura & Slic3r slicing engines(both very configurable). It supports most open 3D printers on the market and is very configurable(possibly being a bit complex to use).

Simplify3D Host

I purchased this directly off of their web site. I was able to download it and install within minutes. The software is subscription-based, which means a login is necessary for each time its run. The site says this is to ensure quality levels and that all updates are installed as they are offered. I'm not that naive. It's hard to make money in an industry where others are giving away their software...this is a form of DRM plain and simple. Installing the software on another machine, with the same credentials, results in a block. Strike one for me unfortunately...I like to run what I purchase on my desktop and laptop sometimes. This conflicts with that and pushes me into the direction of open-source pretty quickly.

Functionally the software is pretty boilerplate, it supports a large number of printers. Mine wasn't in the list but their printer configurator builds a profile with relative ease after a few questions. It's object slicer seems to do a reasonable good job. One small hiccup though; it caused my delta printer to home quite hard(thought I'd damaged it after a few very hard "bangs" making its attempts). One email to support(yes, I actually got a person!) and things were squared away quickly. Nice software!

Repartier Host

Repartier host software is also downloaded from the repartier.com website but is completely free. No login required. No DRM. It installs easily and includes a few

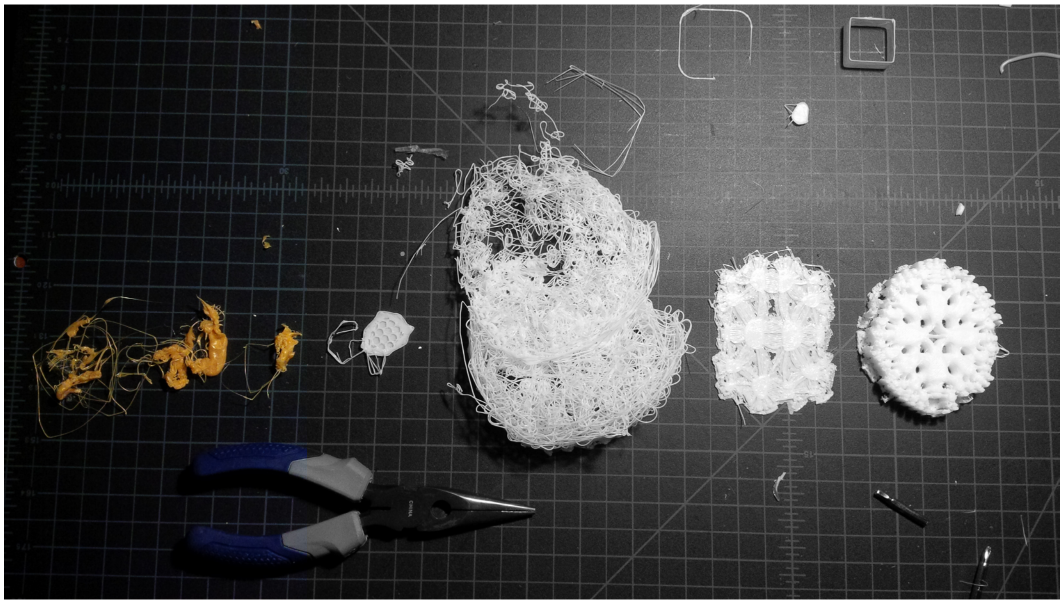


Fig. 110 - First Output - From Formless to Form

slicing options, setting itself up to allow for remote control(and viewing if your pc sports a webcam) if wanted. The software is quite similar to Simplify3D's functionally but allows for much finer control of the printer and slicing algorithms(Slic3r and Cura).

This software is open source, and this can be a blessing and a curse; while macro-control parameters are configurable within the software(like Simplif3D), many of the more basic controls(like printer characteristics/capabilities/rates) are only changeable via a Java IDE(Interactive Development Environment) and live inside the printer's firmware. Once changes at this level are made, the result must be compiled and uploaded to the printer's controller(an Arduino Mega2560) over USB.

This sounds like a lot. It is. But in the end, it gives you much more control over the printer. I found after flipping back an forth between each program, that the Repartier host was the best for use with the Delta Rostock G2s. Speed was the same for both and at the limits of the controller and mechanical setup of the its implementation. I was able to take advantage of this added control to print some models much more easily than with the Simplify host.

Calibration - Machine Commissioning

I found this to be a relatively powerful phase of the project. It is alchemy. Like the grotto, form first begins with formlessness - or in this case form(extruded spool of polylactic acid(PLA) filament) begets formlessness(molten PLA) which begets form(machine output plus its impressions upon me).

The machine struggles to do what I ask as it generates its first proto-formations, like a *prima materia* searching for its cause. As one issue presents itself(is the extruder feed-worm-gear pressure enough?), another pops up(fillament nozzle temperature - too high? too low?). It is the very antithesis of chaos, all the while the universe working against it; always trying to render it to entropy in its goal to democratize the energy involved as much as possible.

The machine encodes, reacts, and the material responds in its own unique way. I'm seeing crap come out of the nozzle. It's initially not adhering to the build platform, and without a basis to push against, it looks like the organisation I was looking for can't find itself. The output just flails around, being dragged by the movement of the nozzle. Constraints are an important part of form making here. Once the extruded

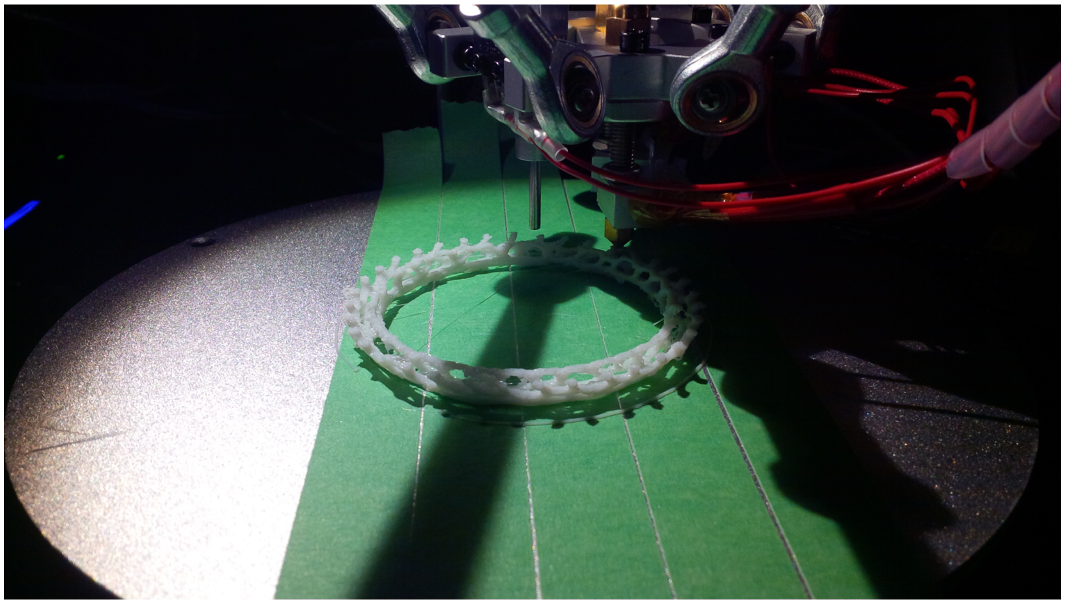


Fig. 111 - Test Print w. Stringing - Bracelet Model(Hegglin 2013)

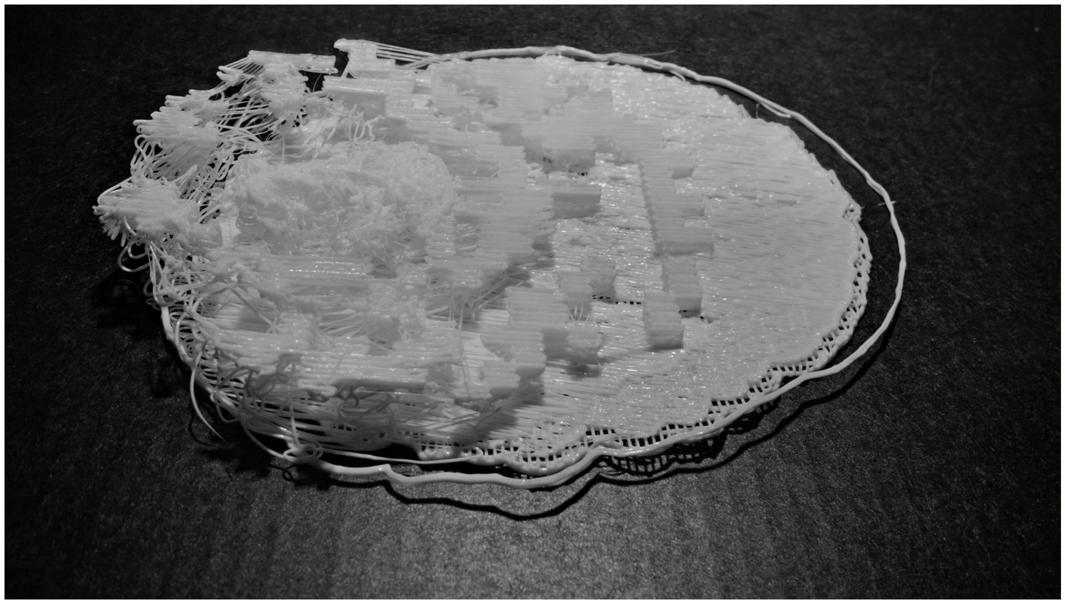


Fig. 112 - Collision Offset Result

proto-form *connects* with the surface, work can begin satisfying the *telos* of this effort.

This is a learning process, and one that asks to look at the intersection of a number of influences:

Operational Parameters

Model	- Slicing Parameters(infill rate)
Basis	- Where is zero? Auto Zero Process ok/Not-ok?
Head	- G-code feed rate?
	- Z-hop amount(backfeed rate)
	- Head alignment(
Bed	- Best temperature(60C)
	- Levelling
	- Best adhesion(What type? Kapton tape? Painter's tape? Glass?)
Filament	- Type?(PLA/ABS/PET/Nylon)
	- Best temperature?
	- Rates of flow?

To calibrate, the goal is to convince the machine to do as asked. It's good to start small and build upon a foundation of successes(and failures). It's very scientific. In this case, I need to first get the extruder extruding in a deterministic way. The system is resisted by the variability of the extruding process, caused by both the rates of the feeding stepper and by the heat level of the extruder head. If the head is too cold, the filament feeding system will grind away at the unmoving filament and wear at it, making a success less likely because there will be less to grip on when the resistance decreases. So a higher nozzle temperature is better than lower because the it's easier to reduce this temp than to cut and reload the feeder due to a thinning filament source.

There is a caveat on the high-temp side as well however, if the temperature is too high, the filament will begin to burn. And a burnt filament means: turning the whole thing off, opening a window, explaining to my wife that I'm not burning the house down as the smoke works its way upstairs through the cold air return of the furnace and finally, waiting the 10 minutes for the nozzle to cool so that I can handle it and use a file/drill/needle to push the burnt slag out of the head before I can begin again. This portion is an exercise in patience as balance is found.

Results? Optimum nozzle temperature range is 95C to 105C. Common practice from the manufacturer is that PLA works best at 95C. This is crap, at the very least

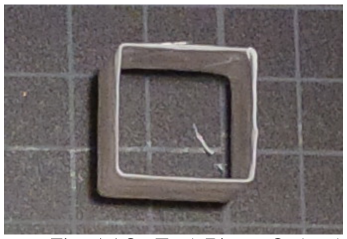


Fig. 113 - Test Piece Output

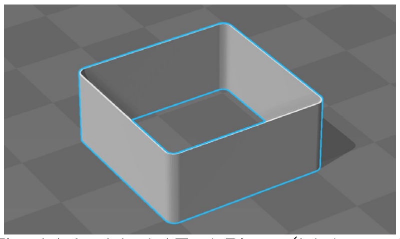


Fig. 114 - Model Test Piece (Matsumoto 2016)

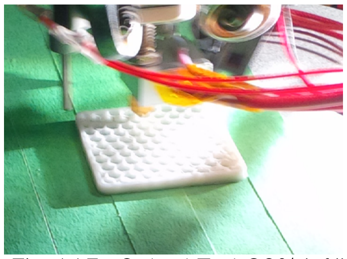


Fig. 115 - Output Test 20% Infill

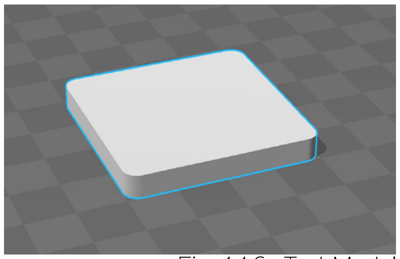


Fig. 116 - Test Model

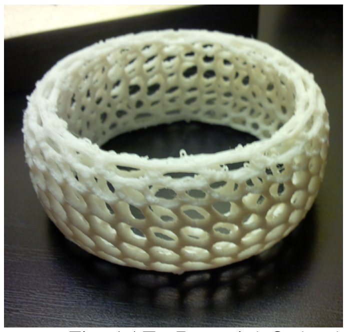


Fig. 117 - Bracelet Output

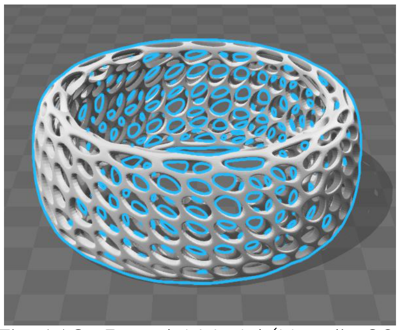


Fig. 118 - Bracelet Model (Hegglin 2013)

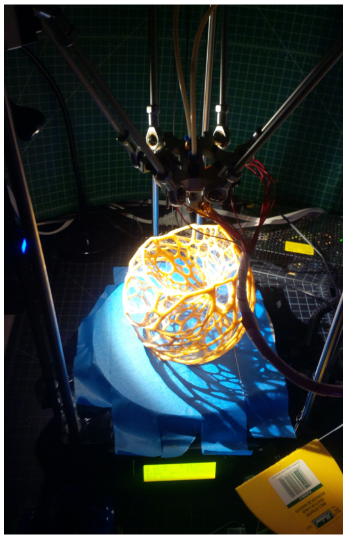


Fig. 119 - Lamp Output

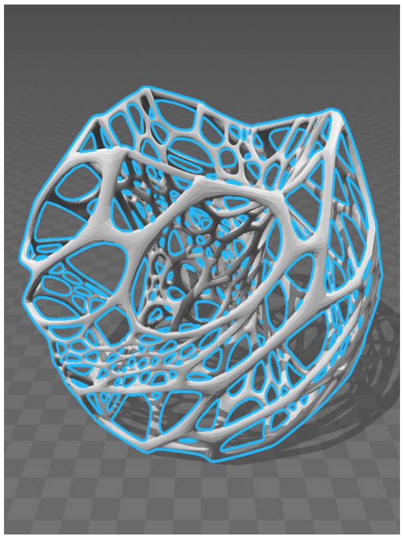


Fig. 120- Lamp Model(NervousSystem 2013)

for my setup. I have to depend on temperature thermistors and their analog to digital conversion to be in calibration on their own, so I take my numbers with a grain of salt. Within the realm of this investigation, absolute numbers are not necessary; workable parameters are however. 95C equals a slower slaggy extrusion, where 105C equals thin, sticky and usable with the possibility of some even thinner wisps of PLA following the extrusion after each run(removable after processing by hand).

This next leads us to head movement and its issues. As described before, there are two heads in the system. Both move in unison as they are both attached to the same head platform. If one of the heads is off by even a portion of a millimetre it can catch on the work being produced by the other head and jar either the work or the control systems mechanical calibration. Both are disastrous(see opposite - Offset Results); if enough sheer force is exerted on the PLA extruded form, it may detach from the build surface and the process has to be restarted from the beginning, but if the form resists more strongly, a collision will cause the steppers to lose track of the location of the extruder platform(as they essentially slip a gear), rendering all future positioning requests moot because its lost its place in the process.

Either way, both require a restart with the latter needing a full homing recalibration of the entire system(10+ minutes minimum), and both result in a disfigured form. If the collision happens close to object completion, it could mean that a multi-hour reprint will be necessary. Again, precious time is eaten up by this situation...and its management is an exercise in patience.

In the end, I found that I could configure the heads to be raised a few millimetres during transition to a new extrusion location, so that the heads essentially rise above the work when not extruding. They are then lowered when necessary. This reduced the time the heads were close to the work and so reduced incidents of collision. This didn't remove them altogether, because the process of laying down traces of PLA isn't always consistent(the material expands inconsistently creating "bumps").

This issue has been dramatically reduced using this strategy but it did have a cost. The speed of the platform can be substantial. As I highlighted earlier, the mass of the platform is low and this allows for a very nimble operation. The system as a whole depends upon a number of moving parts, each fastened together using standard and non-standard fastener types. The "bouncing" produced by my configuration of the head platform path and the speed of the device's g-code execution can cause the whole machine to literally shake apart. I've had the head system detach itself from its arms solely because the fasteners used vibrate themselves out of their mates. When I felt that the system was close to its final configuration(ad-

justment-wise) I took the step to use lock-tite(blue) to hold it together under normal use.

In an effort to test the system, I opted to print a few items:

a single-walled square(Matsumoto 2010)
a quarter-inch partially-filled square two inches on side(wb)
a complex small three dimensional mesh form(Hegglin 2013)
a larger (4″x4″x8″) complex mesh form(RosenKrantz and Louis-Rosenberg 2012)

Starting simple and working to more complex forms, the machine and software were tweaked iteratively. Things improved until the machine was printing reliably. One issue that plagues the system now is that the system leaves small hairs on edges throughout the print. Called "stringing" or "oozing" successive retraction adjustment for the filament extruder is apparently the solution. This solution did not work for me during this investigation, but it was realtively simple to remove them manually.

To realize any digital form, I felt it was necessary to produce models. Throughout the efforts here, virtual form was created and I've documented the work with many rendered images. As with any iterative design process, the more senses you engage, the tighter the feedback is in the work; and the more feedback, the more accurate can be a decision/response as part of the design. Digital design can produce iteration very easily, but as Lars Spuybroek states in his essay, *The Digital nature in Gothic*, (concerning visual media)"There you see everything and believe nothing."(Spuybroek 2011, 39) It was important to build a 3D printer to understand the digital way of making. And for all its flaws, the printer commissioning was a success. I was able to gain considerable understanding as to how this technology works and why it can be problematic in its current implementation. I also learned much about how important the right tool is for the right job.

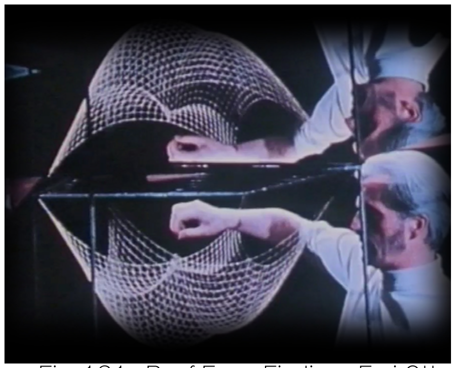


Fig. 121 - Roof Form Finding - Frei Otto

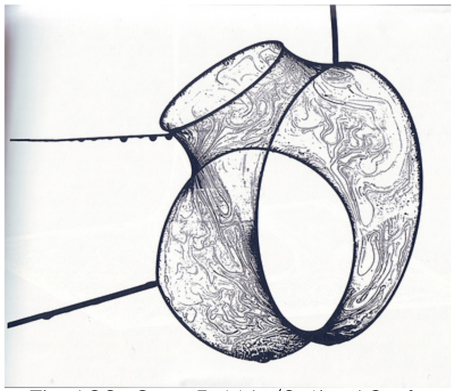


Fig. 122 - Soap Bubble/Optimal Surface Calculation - Frei Otto

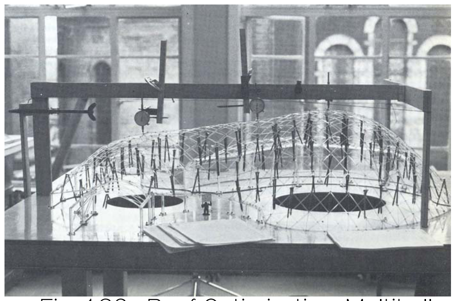


Fig. 123 - Roof Optimization, Multihalle, Manheim, Frei Otto

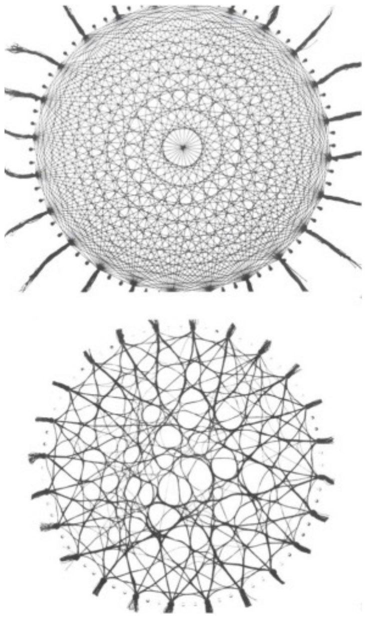


Fig. 124- Wool Thread Network Optimization - Frei Otto

Precedent - Frei Otto

Analog computation is not a new concept. Antoni Gaudi is renowned for his use of chains and weights when form-finding the catenary vaulting and roof of the famed Sagrada Familia. Frei Otto continued this method of investigation, looking for optimal structure in his now infamous soap bubble experiments. His impetus, by his own admission, came from a position of "serving the poor" by thinking of ways of doing more with less. Having been interred during World War II an allied camp in France, he was exposed to tent structures throughout his experience.

Soap bubbles resolve themselves through dynamic forces(gravity/surface tension) to a system of equilibrium. They offer a minimal surface as a solution when the work is done.

In wool fibre experiments that he did with Marek Kolodziejczyk at his 33Institute for Lightweight Structures (ILEK), Stuttgart, 1991, similar forces are realized in two dimensions to take a sub-optimal path network(produced by stringing wool fibres across a circle Fig. 123 opposite) and resolve the best network using surface tension again as the fibres are coated in water. As the system works to find a solution, the existing tension of the threads is used to modulate the result. While mathematics has evolved to solve this type of problem today without physical models, this method was able to find an optimal solution from a set of possible solutions on its own. Each time it is run, a different optimal solution prevails...

I find his influence substantial as Otto hacks physical systems to help him find his solution. As an architect and structural engineer, he builds working analogous tools to resolve larger physical systems. I would consider him an exemplar designer and architect in his methods.

Understood his models were just that, and could not always gather and deal with all issues. In his wooden grid structures, he built-in added springs to cover those variable he couldn't model precisely. The result was a structure that was resilient and performed well within expected behaviour(deflection under load).

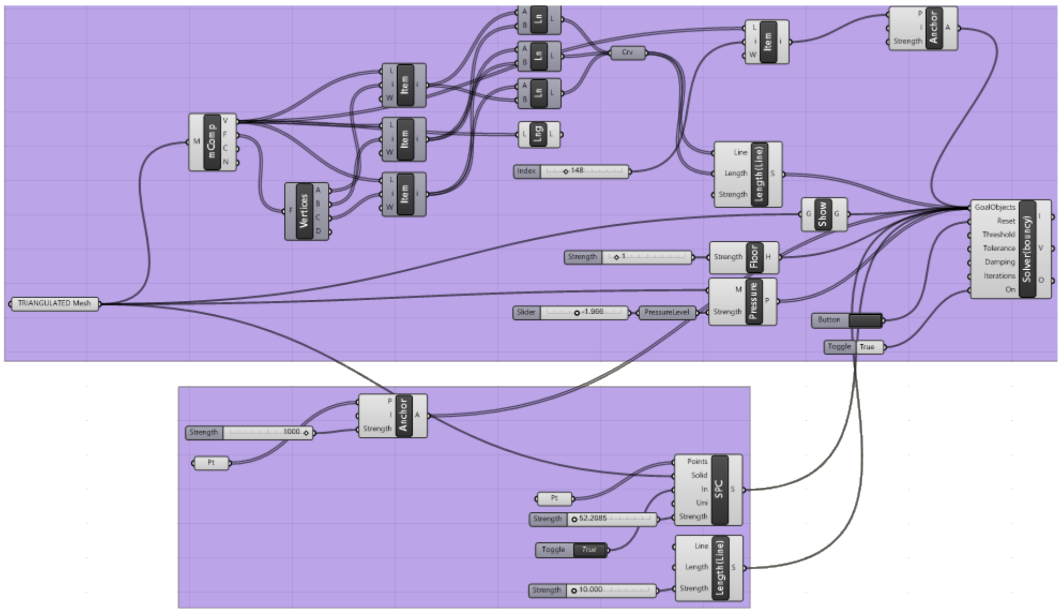
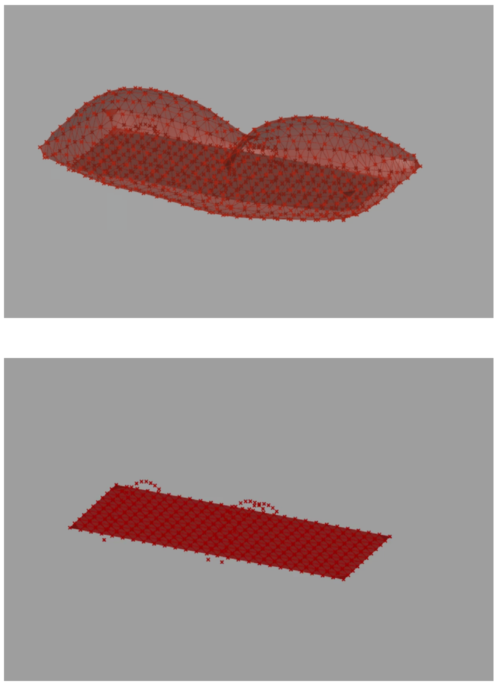


Fig. 125 - Simple Pneumatic Test Script - Kangaroo - Gerstheimer + Brown



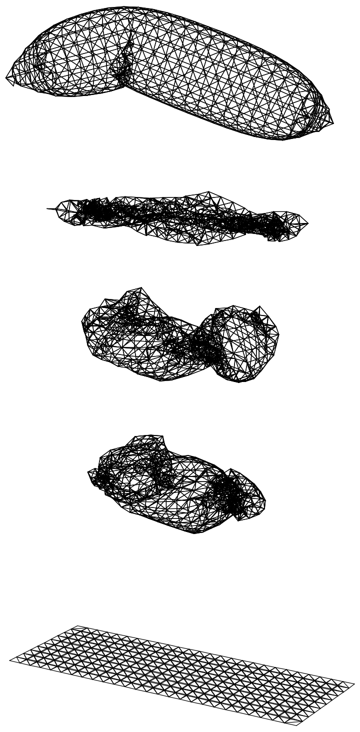


Fig. 126 - Bladder Restriction - Brown + Gerstheimer

Fig. 127 - Bladder Creation
Gerstheimer

Case Study – Pneumatic Structure Proof of Concept (Group Project)

For our term project in a Computer Fabrication elective(Arch 684) taken this past summer, Geoff Gerstheimer, Mark Longo, and myself elected to design and build a pneumatic structure through generative means. The work was split as follows: Presentation(Mark), Generative algorithms(Myself and Geoff), Manual Modelling(Geoff and Myself), Laser Fabrication(myself). For the purposes of this investigation, I 'll speak to the work I touched and will give credit to Geoff, where necessary.

Our investigation began by looking into Rhino Grasshopper components that could be used for building a pneumatic envelope. Geoff was initially able to create and animate our first air bladder by initially lofting a solid box manually and then by using the Grasshopper Weaverbird(Giulio Piacentino 2009) add-on library to create a triangulated mesh from it with varying degree.

The resultant form was fed into a newer add-on component that performs physical simulation using various force types called Kangaroo(Piker 2015). The version we used is the new updated version two of Kangaroo that deals with force elements in a much more fundamental way that version one. Because of its newness, this required a lot of trial and error to understand the add-on's new methods for simulation. Fields are now replaced by components that modulate the change expected on the objects used, switching from an "affect" point of view to one of "effect." This threw us for a loop in the beginning because it felt counter-intuitive, but after a few trials the simulation seemed easier to manage this way.

We chose initially to use a "Pressure" component and an "Angle Change" constraint on the edges of the mesh to first cause the envelope to expand and hold form based on the mesh's resistance to alter its internal angles. Kangaroo offers a few different types of solvers(Normal/With Momentum(bouncy)/Zombie(keeps all data till end)) and we chose to go with the "Bouncy" solver as it seemed to be most life-like in its results.

This simulation seemed to nail the effect we were hoping to see, but to ensure the system behaved best we also chose to add a mesh edge "Stretch" constraint(applying Hooke's law to the mesh edge system) to ensure that the envelope held its form while under external load. It seemed that the system was more resilient with two constraints during testing. Without the second constraint, the bladder could not expand under pressure to relieve applied forces.

So a large egg roll-like bladder doesn't make for an interesting space; at least I

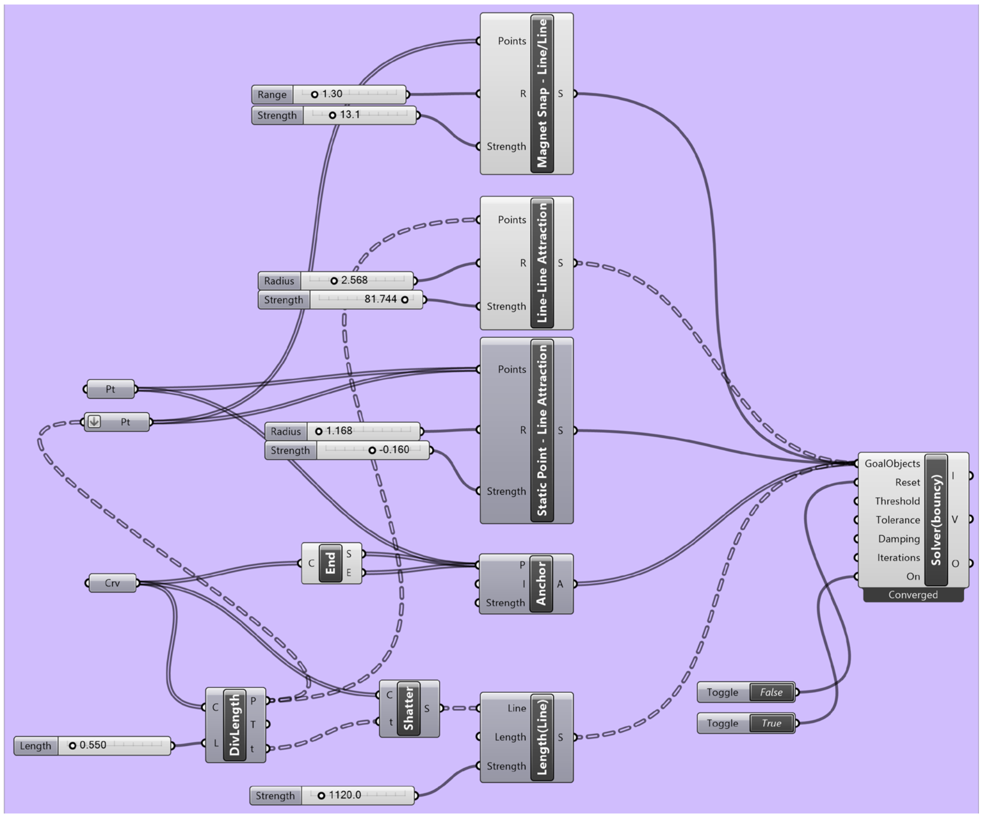


Fig. 128 - Frei Otto Filament Grasshopper Script

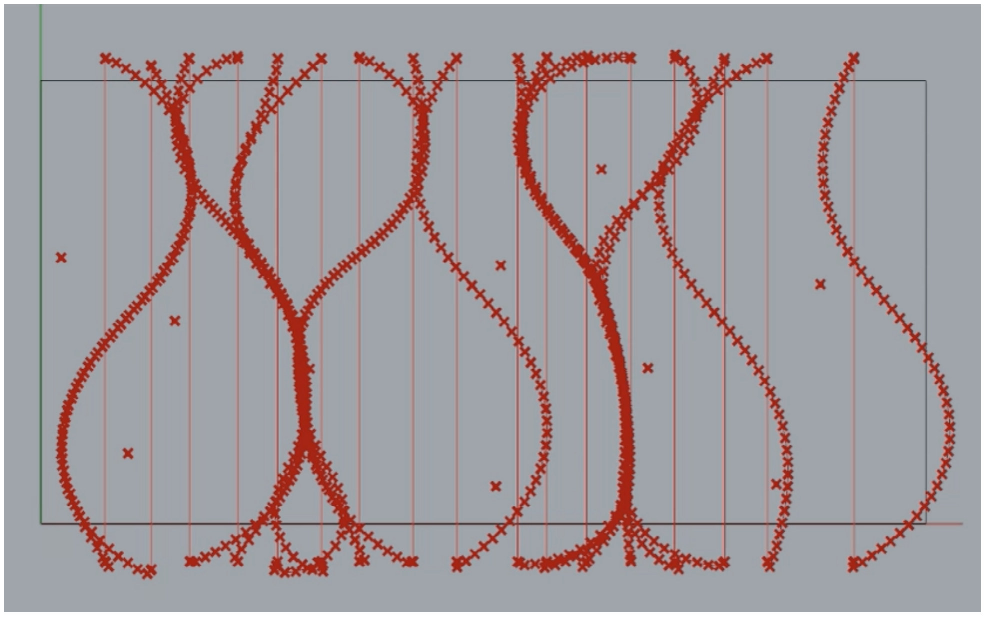


Fig. 129 - Filament Convergent Solution

don't believe so. To convince the form to arc over any occupants it needs to be divided and articulated in some manner. The problem could be assessed from a couple of directions; I could optimally arrange the enclosed pneumatic areas, or I could optimize the separators(networks of lines) that divide these regions. An enclosed chamber resists bending, so an offset arrangement of chambers might be able to hold itself up and resist its own weight when inflated while creating an interesting space underneath. The method we chose to accomplish this was inspired by Frei Otto and his wool fibre experiments. Why not find an optimal dividing network of lines and let the chambers sort themselves out?

Otto's investigation began with a circle and an n^2 network mapping of all points to all other points. I elected to simplify our investigation so that we might be able to create a single arched area that occupies the length of the material space. A set of lines was drawn across the space and my algorithm was applied to the grouping.

Using Grasshopper and Kangaroo again, I affixed the endpoints of all lines, divided them into a fixed number of segments, and applied similar rules to that of the pneumatic chamber work previously(Hooke's law & edge angle constraint). To entice a more life-like behaviour, I added a line-line attraction constraint and then added a further line-point attraction condition to refine the system.

There was limited success of the script in this state. Every time I ran the system, a different solution was found but some appeared overly simple and not very interesting. I decided to add more lines, increase the precision of the simulation, and also add attraction/repulsion points to allow me to "seed" the system and guide towards certain, more interesting, solutions.

Complexity issues reared their ugly heads again. It was easy to overload the system to the point of inactivity. I learned quickly(mainly due to the interactive nature of Grasshopper and its scripting system) that operating with 17 line segments and a precision of 50(lines divided into 50 representing points=49 line segments) was pretty much the interactive limit of this method.

Interestingly, with this configuration the system would come up with a different solution through most runs. Many were similar, but not exact. I can only attribute this to changing round-off within the Kangaroo solver as I would normally expect the same solution each time. All the same, what we found provided at least a class of solution that met our needs; this not being much different from Frei Otto's own results. There were solutions that were optimal from a network solution standpoint but not usable for a pneumatic solution. Some did not create enough chambering to be

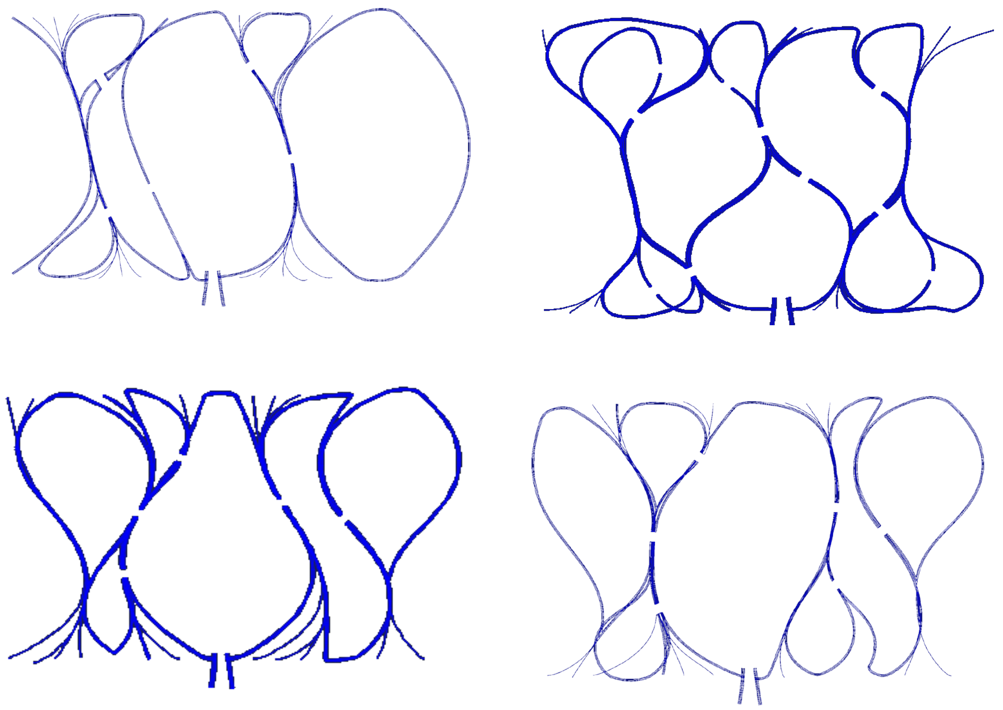


Fig. 130 - Kangaroo 2 - Based Patterning

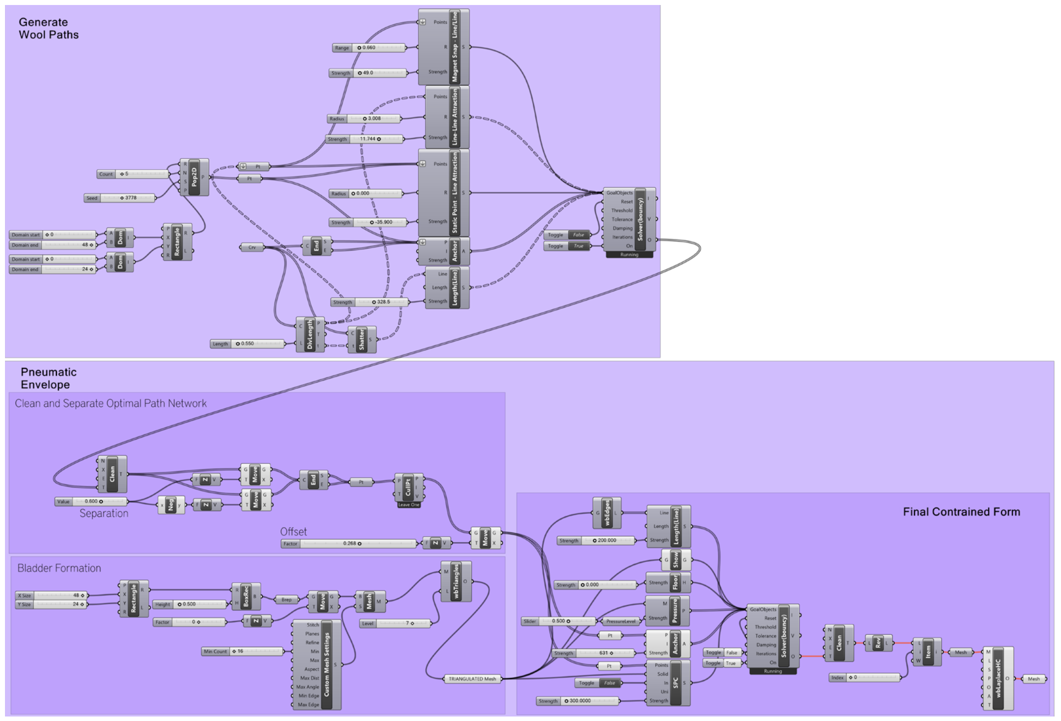


Fig. 131 - Pneumatic Volume Grasshopper Script

useful for this investigation and I culled these from our set of useful arrangements. Others created chambers that spanned side-side and if they were too large, they too were culled. We needed solutions that had smaller and offset results for this to work(common sense). Air delivery path openings were added manually post-simulation to ensure the system was inflatable. The realities of a physical system began to enter the work at this point.

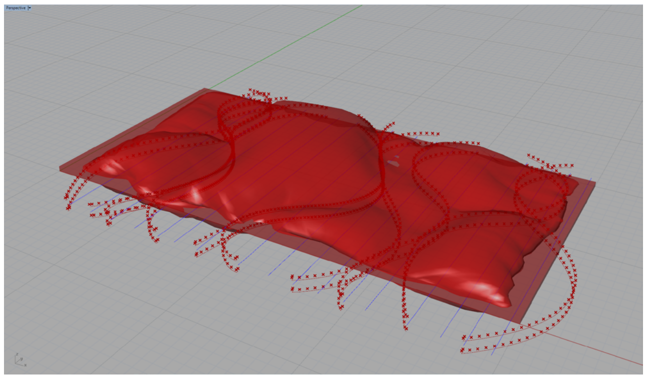


Fig. 132 - Grasshopper Pneumatic Simulation

Once this was completed, I elected to combine our pneumatic simulation work with the path-based optimization results. It is here that things got interesting. I had hit the upper-bound in interactiveness with the path investigation alone. Adding pneumatics into the fray proved to be an issue. An executive decision had to be made; rather than simulate the entire system real-time(Which would have been more correct), I chose to take the sets of paths found in the last part of my work as static input to the system for the next phase.

To allow them to be dynamic(and effected by each other while path-finding) would have bogged the system down too much to be of any use. I pushed forward to see if it was possible to simulate the creation of a divided pneumatic system with the paths set provided. The resultant grasshopper script(opposite - lower) and input(opposite upper) were combined. The simulation was able to resolve a pneumatic volume if the paths were used as an additional repulsive constraint on the initial pneumatic system used at the beginning of this investigation. The results(above) were positive, and quite reassuring.

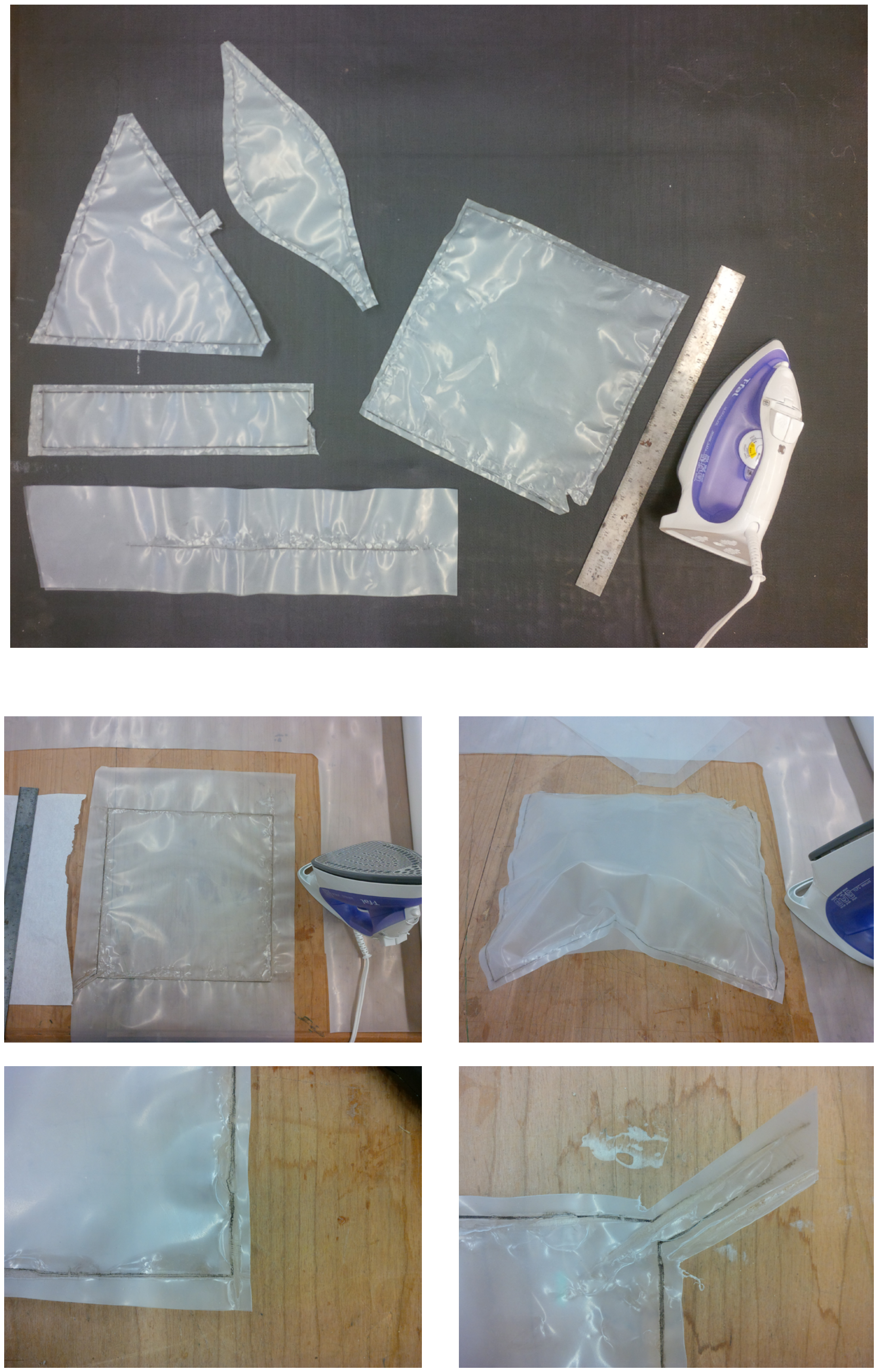


Fig. 133 - Manual Pneumatic Chamber Creation

Making

We chose to test our efforts using a readily available 6mm polyethylene plastic sheeting(moisture barrier) used in building construction everywhere. I also felt, from my PVC 3D printing trials, that the polyethylene would off less smell when melting and less smell meant we could work with it more closely within the shop.

My first trials began simply. I bought an iron, and some parchment paper, and a metal ruler. The silicone in the parchment paper protected the wooden workbench below from the melted poly during these tests. The setting on the iron was at its highest(cotton) and steam was not used. I felt that if I was able to run the iron along the straight edge provided by the steel ruler, I might be able to modulate and adjust enough for any curve to get a bond to happen between two pieces of poly.

It took a few attempts to get it right, but I was able to get a line to appear fairly quickly. When I pulled on the test strip, the poly was well melted and did not rip or tear when stressed manually and began to feel quite optimistic about this process. Moving forward, I thought it a good test to try to create a few enclosed geometric shapes and see if they could hold any pressure. We had a smaller aquarium air pump that could be used and a large(185psi) air compressor for extreme tests.

After manually melting each of the primitive shapes, I attempted to push air into them using a small cut nozzle area that I had melted as an inlet for each shape(pictured opposite - bottom right). My compressor had a large-ish manually controlled nozzle that I felt would slip into an entrance this size. This proved to be problematic at the best of times. Any thin entranceway would constrict immediately when any high-pressure air was added(thank you Bernoulli!). So I had to remove the manual raceway if this was going to have a chance of working.

Once this issue was managed, the bladders were able to accept and maintain a shape as long as very small pressure was used. They were riddled with leaks and any area with a sharp change in curvature(a.k.a. any corner) easily popped under minor stress.

The idea seemed sound, but there was very little consistency and the use of corners was a definite nono. I needed to find a better process to melt the poly and also one that would allow much faster production and more versatile shape generation. The breaks in bonding also kept us from finding what stretching the poly could actually take under pressure. Enter the laser cutter...

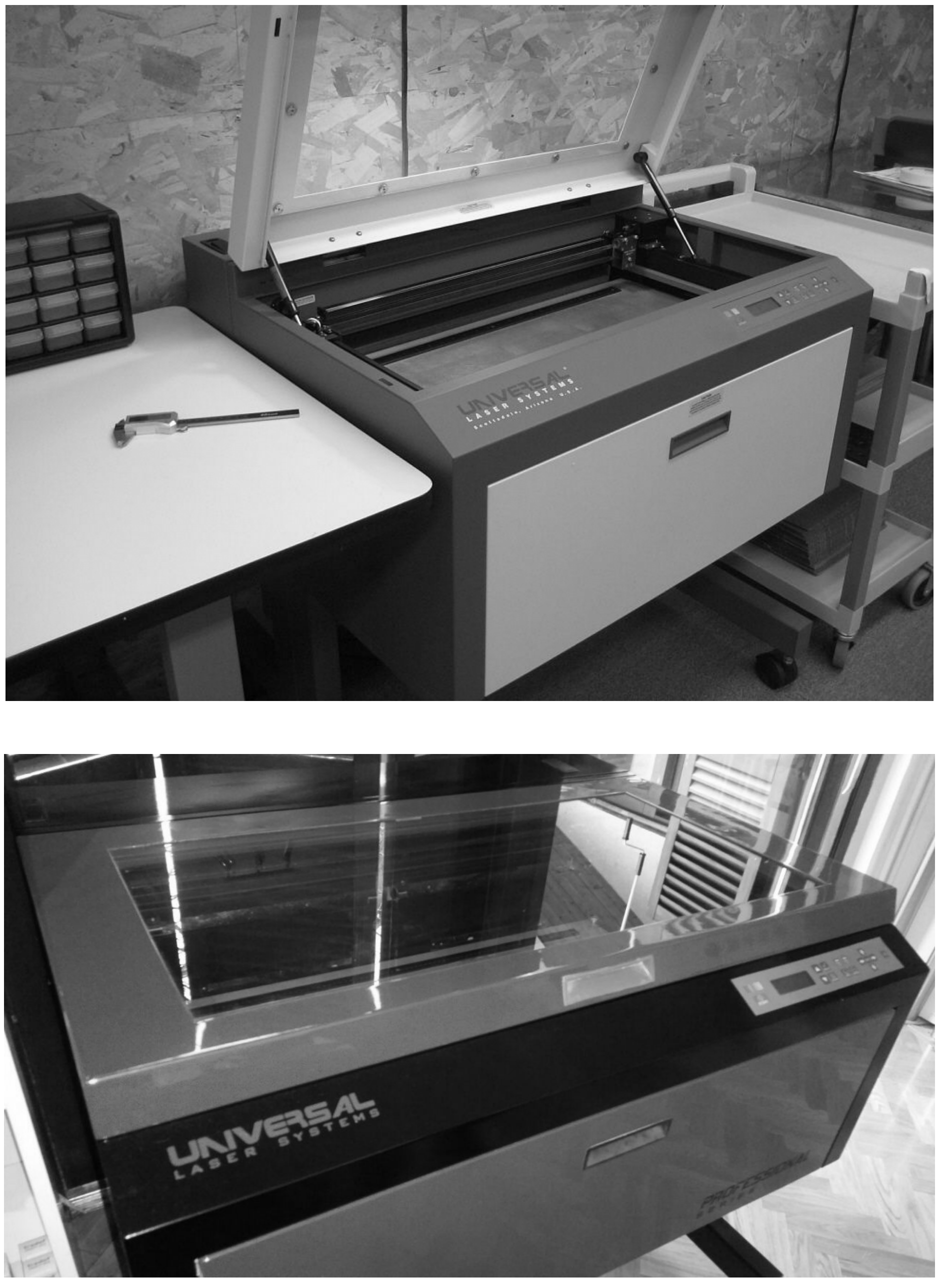


Fig. 134 - UNIVERSAL Models - Laser Cutter Investigation

A Tool and its Hack - Emission Technologies

Welcome to the 1970s.

In 1st year undergrad it became clear, amongst the band-aided masses of our sleep-deprived class, that a laser cutter was “the tool” for model making. Crafting models by hand was imprecise and despite taking all precautions, the chances of cutting yourself was quite high. After model #3 I knew I needed to find a better way. If I looked left and then right in the studio, most had experienced a negative run-in with their Exacto in one way or another.

I found that the quality of my manual cutting work was ok, but it was clear that I was more often making copies of the same floor plan or wall section over and over, and that their manual variation was creating issues. I’d seen that the school had a couple well-loved laser machines, and while they were certainly available, access was limited generally to business hours and this was not at all convenient for those of us who didn’t skip daily classes or book laser time two weeks ahead “in case” it was necessary as so many did. I knew I had to investigate getting one of these tools for my home shop.

I was one of those kids lucky enough to go through the Toronto Science Centre in its heyday. Today it’s but a shell of its former glory(although Moriyama’s presence is still felt in many areas) as budgets shrank and it seemed to me at least, imagination took a long vacation. In 1974 it was *the place* to go for any wide-eyed technically-minded child and school boards bussed them in by the thousands each year. The displays were mainly hands-on and their presentations were enthralling. Other than the obvious A/C high-voltage Tesla display, and the newness of being able to actually touch a computer, I can’t tell you how amazing it was to see their large CO_2 laser cut through a real brick.

This was the stuff of science fiction, and Star Wars wouldn’t be out for another 3 years. I was taken. This same year, the father of my then best friend came in to show off a HeNe laser in our grade four class. He was a visiting Physics prof at UW and it was “career day.” You have to understand the time. There were no tiny diode laser pointers, hand-held calculators or Internet yet..most people were just making the change to cheaper colour TVs from their old trusty black and white sets. This was real in your face science fact. To have one of these to experiment with was quickly added to my now ten year old bucket list. I think everyone in class added it to theirs’ as well.

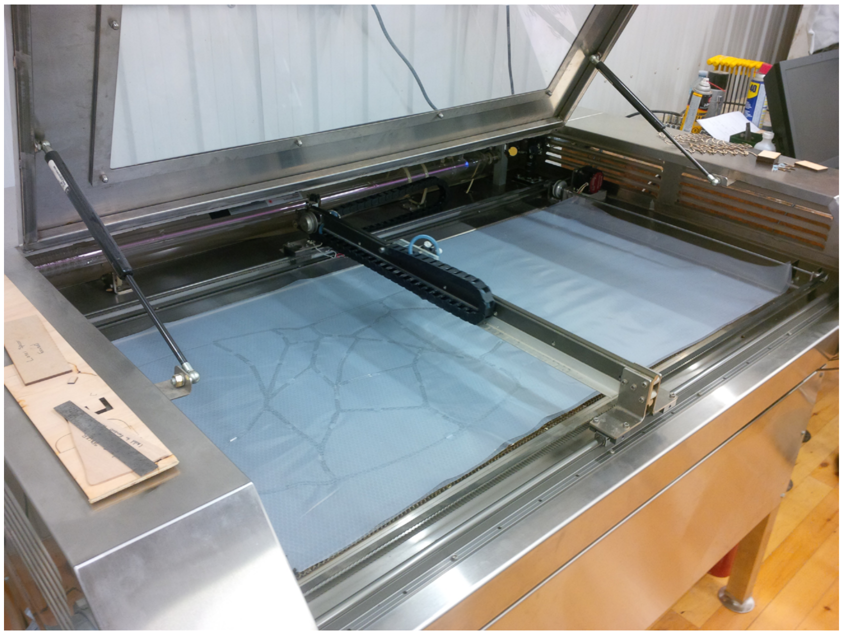


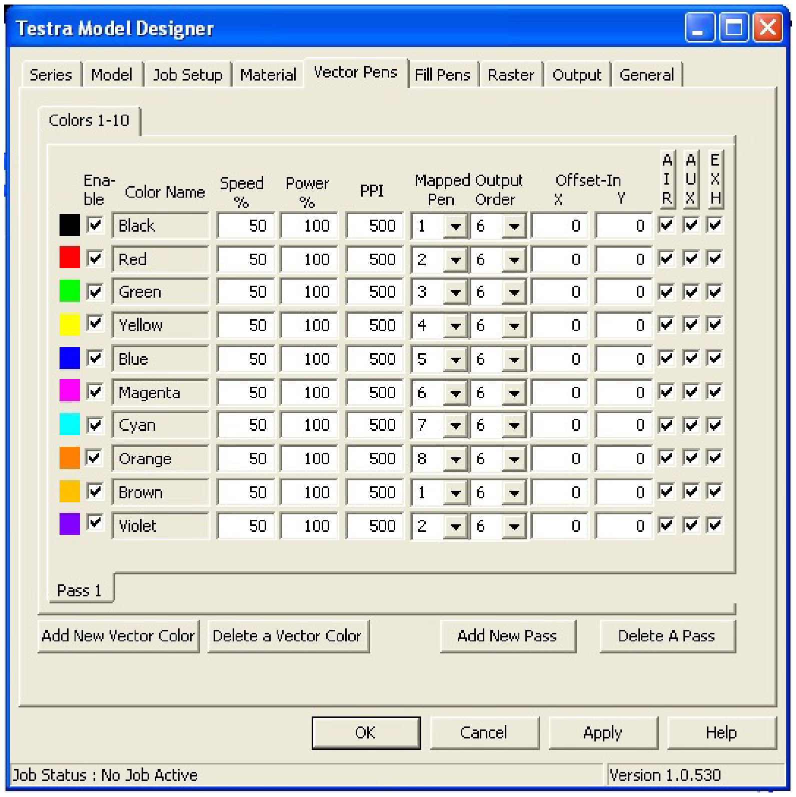
Fig. 135 - Laser Cutter Operating

First place I looked for a cutter was the Canadian distributor for the machines the school used. The Universal reseller was then in Mississauga and I made an appointment to learn about their products and understand the cost. They gave me an hour and it was very helpful. When the price list made it to the table my jaw dropped. I couldn't believe it. A 60W 18"x24" table would cost $35K,130W 24"x60" $75K! This seemed ridiculous to me. I asked that they open the case and show me what made them worth this amount of cash. There was a lot of hand-waving, and a lot of "proprietary this" and "proprietary that", but much of what was I saw inside was air. It was pretty clear to me that these were heavily overpriced and that I should find a way to pull all the fud(fear/uncertainty/doubt) out of these tools. Looking at other manufacturers in Canada(Trotec, Epilog..etc.) I found more of the same. This market seemed either ready for a revolution or I simply didn't understand its complexity. I aimed to find out.

I searched through a few laser cutting fora(cnczone, sawmillcreek, instructables, hackaday, openbuilds...etc.) and came across a machine maker in Florida who had been selling his particular design to material pattern cutters all over the world. His plans were inexpensive and Jerry Condon, of Emission Technologies, was willing to deal with me in Canada. He sold me the plans(US$2K) and explained to me the bunk that is the current laser cutter/engraving industry. His help was instrumental in assisting me to build and assemble his flavour of cutter for myself.

I ordered a 6' long 130Watt CO_2 laser tube, liquid chiller and power supply from a supplier in China(Can$4K), ordered optics from Spectra Physics in California(US$1K), steppers and gantry parts from Jerry in Florida(US$2K). I had the housing welded out of stainless by a Mennonite fabricator in Wallenstein($CAN3K) who had done some work for me before. The brains of the unit was an off-the-shelf stepper controller($US1.5K) from Testra in Arizona. As luck would have it, this was the exact controller used in those Universal cutters I started my journey with. End cost ~Can$12K for a 130W CO_2 24"x48" cutting table. I contacted the folks at Universal to have a talk about pricing and value, they didn't seem open to the conversation(surprise, but understandable I guess). They had been pulling the wool over peoples eyes for a long time, and no one likes change. Since building the machine, costs for laser tubes have dropped considerably(CamFive130W CO_2 ~ US$1.1K) with many going to solid state lasers and the benefits these diode packs offer. I'm guessing a bit of a mini-revolution has been happening these past few years. Access has changed.

After assembly and testing, I've become quite adept at maintaining this tool. It cuts most anything that 130W will allow. At a light wavelength of 10.6µm, this means


Testra Model Designer

Series | Model | Job Setup | Material | Vector Pens | Fill Pens | Raster | Output | General

Colors 1-10

	Ena-ble	Color Name	Speed %	Power %	PPI	Mapped Output Pen	Mapped Output Order	Offset-In X	Offset-In Y	AIR	AUX	EXH
⬛	✓	Black	50	100	500	1	6	0	0	✓	✓	✓
🟥	✓	Red	50	100	500	2	6	0	0	✓	✓	✓
🟩	✓	Green	50	100	500	3	6	0	0	✓	✓	✓
🟨	✓	Yellow	50	100	500	4	6	0	0	✓	✓	✓
🟦	✓	Blue	50	100	500	5	6	0	0	✓	✓	✓
🟪	✓	Magenta	50	100	500	6	6	0	0	✓	✓	✓
🟦	✓	Cyan	50	100	500	7	6	0	0	✓	✓	✓
🟧	✓	Orange	50	100	500	8	6	0	0	✓	✓	✓
🟨	✓	Brown	50	100	500	1	6	0	0	✓	✓	✓
🟪	✓	Violet	50	100	500	2	6	0	0	✓	✓	✓

Pass 1

Add New Vector Color | Delete a Vector Color | Add New Pass | Delete A Pass

OK | Cancel | Apply | Help

Job Status : No Job Active

Version 1.0.530

Fig. 136 - Testra Driver Software Interface

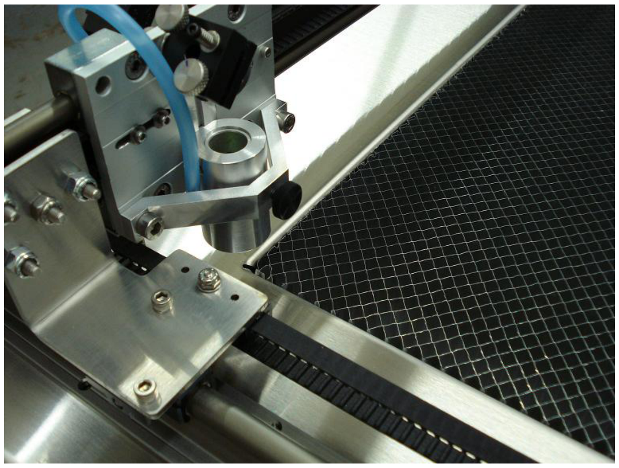


Fig. 137 - Emission - Open Laser Head w. Collimator

anything organic is fair game. It's been used by many groups from our year in undergrad for large jobs or for those needing a tool off hours or those who just want to save some cash. There are issues with the mirror mounts drifting over time but this has been manageable.

The laser has been instrumented to do two things, cut or raster with the most common use being the former. It is relatively simple to set the system to cut; set all paths within a single layer in Autocad, or Adobe Illustrator and then assign a speed, power level, and laser pulse value to the layer. Download the vectorized paths to the cutter(ensuring a path thickness of 0.0mm), and run with the a layer's properties. Easy. When running a raster, the image must be on its own layer again and then all the same values are assigned to that layer(except that a non-zero path thickness is set) and then the cutter will move through the image scan line by scan line, modulating the laser at whatever dpi(dots per inch) was configured. Smaller dpi equals a less detailed image but faster output as it has less data and area to cover.

It became very clear, early on, that a raster plot was not the way to go for this investigation. Firstly, I needed to ensure a clean continuous annealed region to ensure the consistency of the pneumatic bladder and raster output lacked this ability consistently. Secondly, it would simply take too long at higher(and necessary) dpi levels. This was a deal-breaker for the chosen method; if it was slower than our manual process, it was out.

Vector paths would work well if we could find the proper power/speed/dpi balance that would melt, but not cut or burn. This would prove to be an interesting balance to find. The laser system is composed of a static laser tube, coupled with small 1" mirrors that focus the laser's beam to enter a collimating tube before it hits the material surface when positioned at any location on the 24"x48" table. This final stage takes an invisible infra-red beam with the sectional profile of a dime and refocuses it to a profile of 0.1mm(at best).

There is a difference when configuring dpi in vector mode with the laser system. The dpi parameter is controlled by a physical wire that uses pulse width modulation(pwm) to affect the laser's power supply. When the pulse is "0"(0 Volts) the laser stays quiet and does not fire, and when it's "1"(+5 Volts) it fires. The pwm line voltage changes at a rate consistent to give the laser the ability to meet the dpi needed during raster operations, but during vector mode, the pwm value need only be used to maintain the output power at an average level needed for its operation.

This is an important concept to include here because the laser system is not a

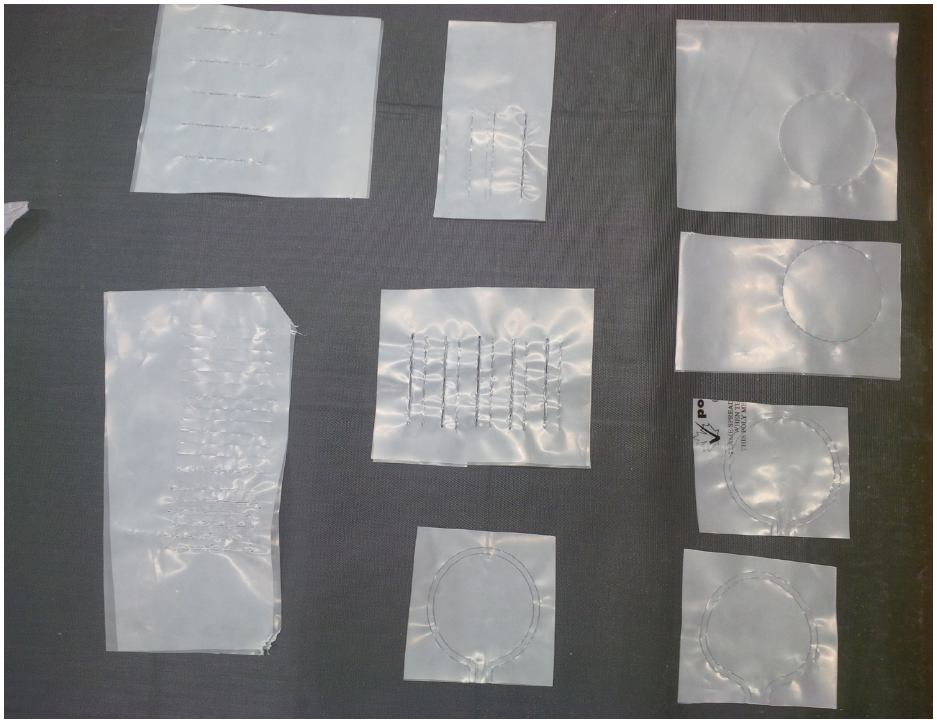


Fig. 138 - Early Laser Fusion Investigations

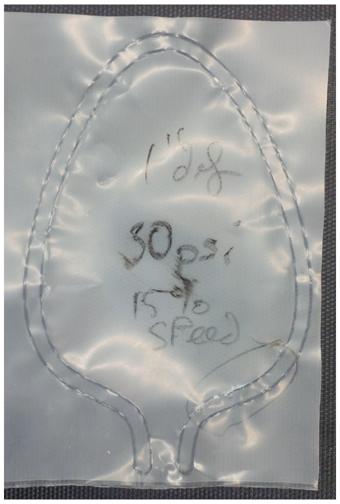


Fig. 139 - First successful Pressure Test

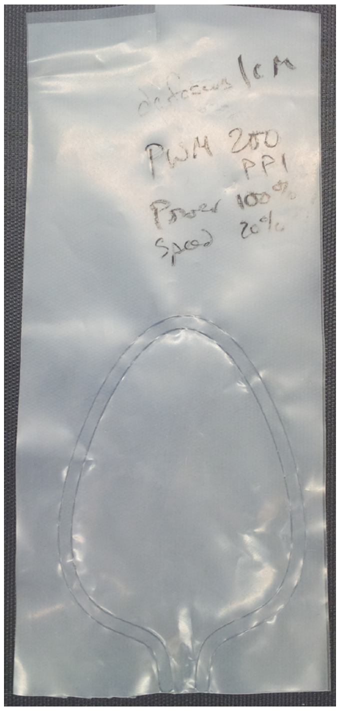


Fig. 140- Laser Refinement and Testing

blank check. Only so much energy is given to the CO^2 gas by the laser's power supply per second of operation. If I pulse the laser too quickly, I can exhaust the output by withdrawing too much too fast. I need to care about pulsing as much as I care about overall power; the two are intimately linked.

I performed a number of tests, all were about producing one plastic welded line. I began with low power and worked the config up until there was too much, then I paired back the pwm parameter until the line appeared like a dashed weld. Then I upped this parameter and slowed the laser down until it either melted through or smoked too much. All the while ensuring the optics weren't negatively affected(smoke deposit reduces reflected power and heats up the $125 gold-plated mirror to the point of failure). Also throughout this process Air Assist(a nozzle function) was disabled as it cooled down the plastic too much and hindered the whole experiment. Normally this function could be used to clean out the area being cut and remove any gasses, but I felt this could be done manually. At the end of this process, a consistent, clean, and very strong line was produced. The process showed much promise.

The test shape throughout this calibration-phase evolved as did our parameters for the laser. Our first shapes were purely circular and crude and had too many corners/edges and this proved to be an issue in creating integrity with the final envelopes...so, corners were out. I also rounded off the test air-entryway and also added a second/offset curve to reduce stress on the inflated volume. I quickly discovered that if p-lines were used, their endpoints were still seen by the laser encoding software and this always caused the laser to "dwell" long enough the system to "punch through." Not a catastrophic issue because the area around the hole was sealed, but not a clean solution. I moved to used connected splines from that moment on. No internal points reduced punching.

The resultant shape(left) had the added benefit of closing the in-bound air aperture when under pressure as well. This proved to be of great benefit when building more complex geometries later in the investigation. I found that the greater the contained pressure, the better the seal; a great side-effect.

Collimation or no Collimation

One added parameter became necessary when doing pressure tests on the pneumatic cells I was creating. They would fail at relatively low psi(5-ish) due to material issues. The 6mil poly would itself fatigue in areas where the melting had been applied. The characteristic of the material only became present after the laser had

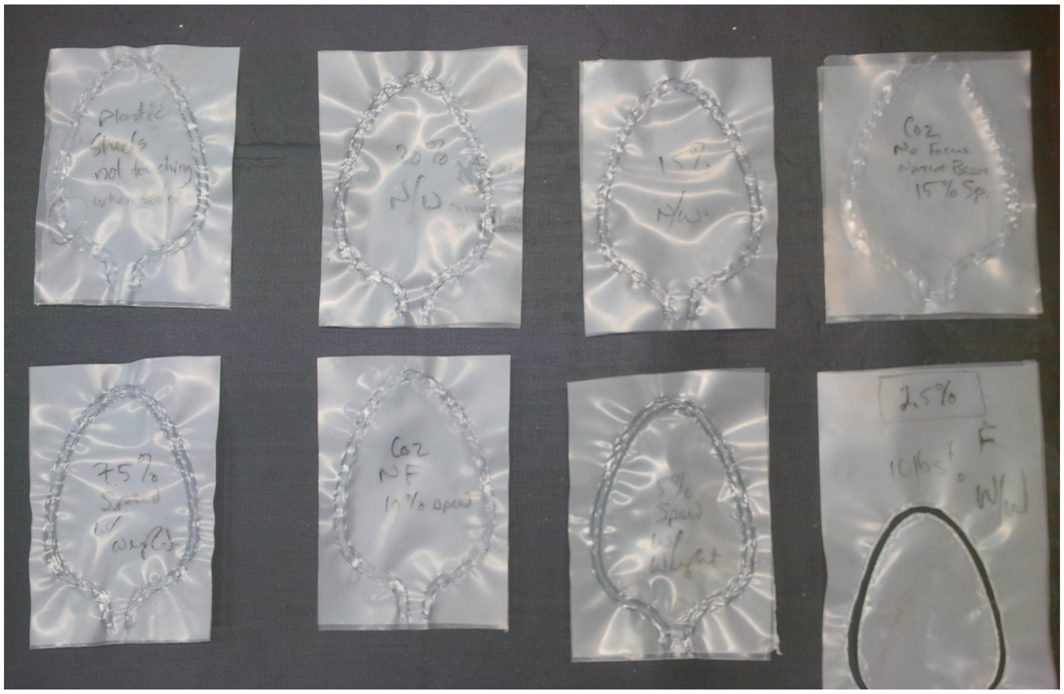


Fig. 141 - Various Collimated Tests - PWM & Power/Speed

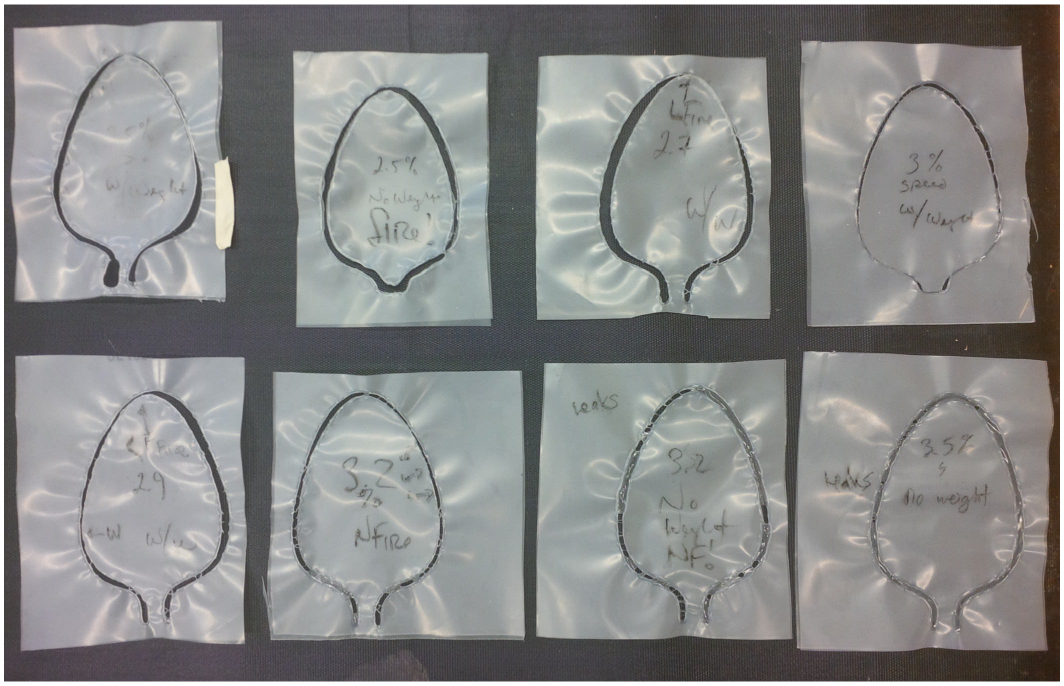


Fig. 142 - Open Beam Laser Tests (No Collimation)

changed the state of the plastic. It was as if the added heat had weakened the plastic generally. The parameter that needed to change was beam diameter.

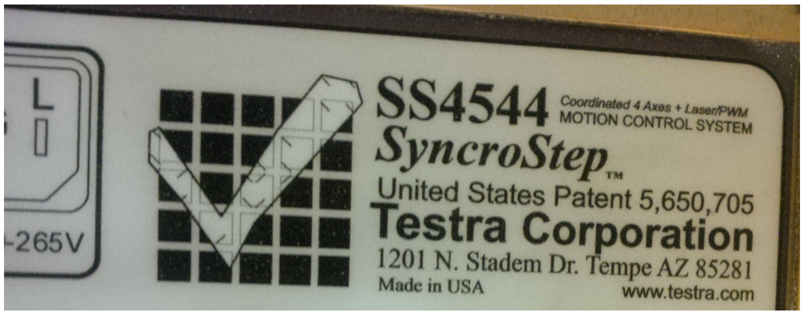
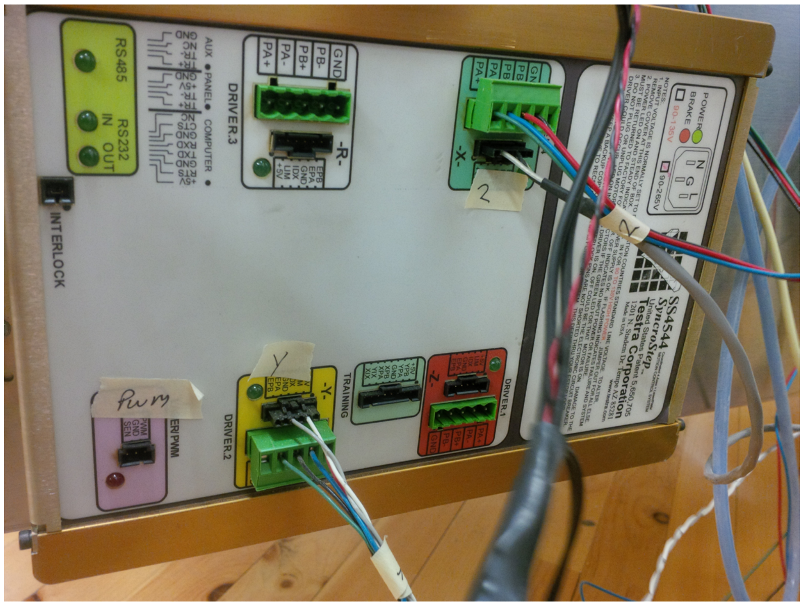
If I could affect a larger section of the plastic, its combined strength might be enough to hold greater pressure. Initially I simply defocused the beam from .1mm to 2mm or 3mm. The change was dramatic(chamber pressure good till 20psi) but I questioned, what if I could make it larger? Would the benefit increase with beam diameter linearly? A decrease in focus meant greater beam diameter hitting the poly which meant greater power needed. This quickly became problematic to tune. Although there was limited benefit, the system became unstable quickly. I opted for one final change.

What if I removed collimation completely? It's dime-sized beam is the best possible output from the laser tube natively. And there would also be a power benefit by not using the final stage lensing. My trials are documented lower left(opposite). Open beam tests would produce a wonderfully beaded edge that seemed to have no upper bound on pressure(tested to well above 100 psi).

There were a couple major issues however; the process consumed the poly around the area and deposited it at the joint in a bead. While this was good for the joint, it wasn't good for the whole process. I needed to "print" a network of connected chambers and having the process cut them out from the plastic fabric was not going to be of benefit.

The second issue was a major one; the poly could easily catch fire during the weld. This was a show stopper. While parameters were good for slowly changing curves, where the curve had the highest rate of change, the laser controller would not appropriately reduce the pwm frequency enough and the system would be over-powered. With a native beam, this resulted in the plastic catching fire in numerous places. Fire being a bad thing, I looked back at previous steps and saw this effect in all the runs.

Due to its vintage, the controller clearly has an issue in tight curves. I suspect its spline approximation is not so wonderful either(bezier implementations were quite new in the 70s). While this is not an issue in cutting or raster operation, I clearly hit an implementation snag for this investigation. I needed uniform power application at all points on the curve.


SS4544
Coordinated 4 Axes + Laser/PWM MOTION CONTROL SYSTEM
SyncroStep™
United States Patent 5,650,705
Testra Corporation
1201 N. Stadem Dr. Tempe AZ 85281
Made in USA
www.testra.com
-265V


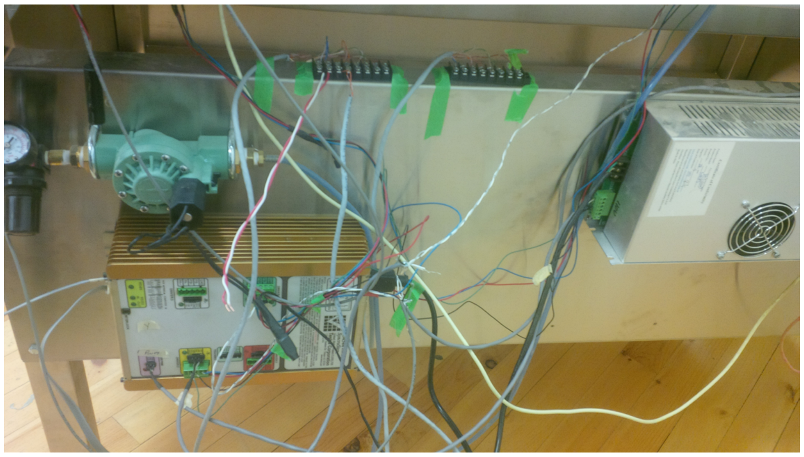


Fig. 143 - Existing Testra Controller - Transition Connectors

Controller - Original System - a.k.a "The Elephant in the Room"

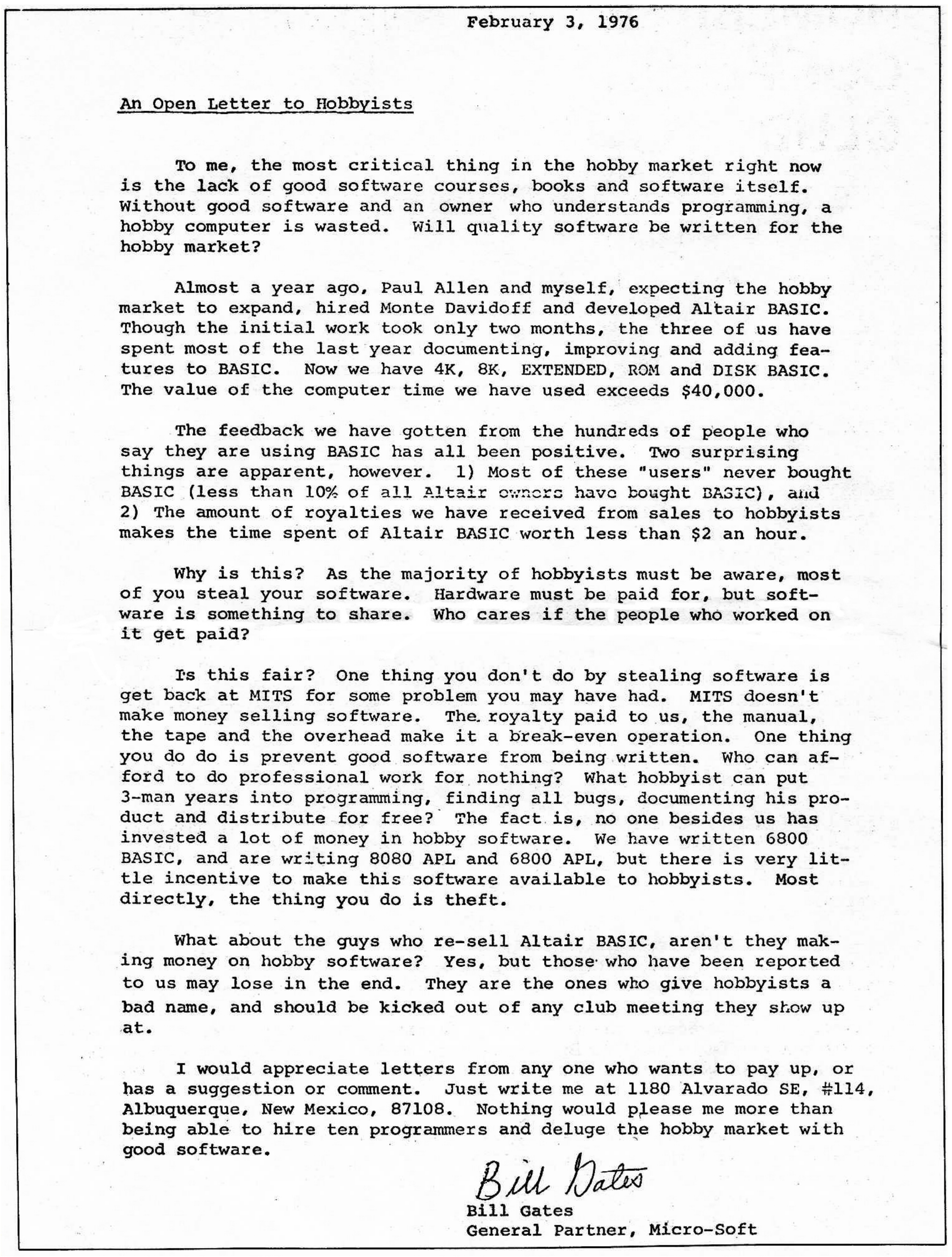
The original controller was manufactured by the Testra Corporation in Tempe Arizona. Best I can see, one very gruff designer built a very simple four-channel stepper controller to manage CNC operations in most any situation. Doing a patent search lead me to see this design was based on many dating back to the cold war era.

The insides are mainly TTL(1970s) technology and this was coupled with a well developed printer driver that is able to manage output from Autodesk software, managing vector layers and rasters well for its time. Testra is still selling this same exact product that they began producing back in the 70s, and surprisingly, still unchanged. This is the root of my issue.

Ist problem - When attempting to emboss a relatively small raster image on acrylic recently, I had to rely on the unit's now geriatric designer to support me in a memory upgrade. He had designed the controller to use (then state of the art) 30pin simm memory (with parity). These chips were scarce even when they were current, being designed mainly for the embedded controller and high-end server market. Basically not many were made. The (then current) memory controller would only allow for 24-bits of address space with an 8-bit word, leaving each sim offering 16Megs of storage. And with the controller only using 2 slots, that left the controller only supporting 32Megs worth of raster images and that's not very large(at 8-bits per pixel intensity). It seemed a memory upgrade would only buy a bit more time for the aging controller. After all, there aren't many Intel 286 or 386 computers in service still these days; manufacturers aren't scrambling over themselves to support this market.

2nd problem - The software driver is tied to the O/S. Testra has stopped releasing new drivers in 2009. Being tied to an older piece of driver software can be nothing but problematic when trying to interface with it using current apps. Crashes and odd artefacts have become more common as we begin to push the boundaries of what the system was designed to do.

3nd problem - The software handles complex curves the way complex curves were handled in the late 70s, as a number of short poly-lines(and not using splines). That means that curves can get choppy when they change quickly. This creeps into the output as holes. The laser "dwells" at the poly-line endpoints during processing just long enough add heat at these points more than while it's tracing out a line. This becomes a substantial issue when trying to control the cutting process closely as noted previously. There's a "hole" in the travelling pwm algorithm.



February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates

Bill Gates
General Partner, Micro-Soft

Fig. 144 - Bill Gates' Famous Letter to Computer Hobbyists

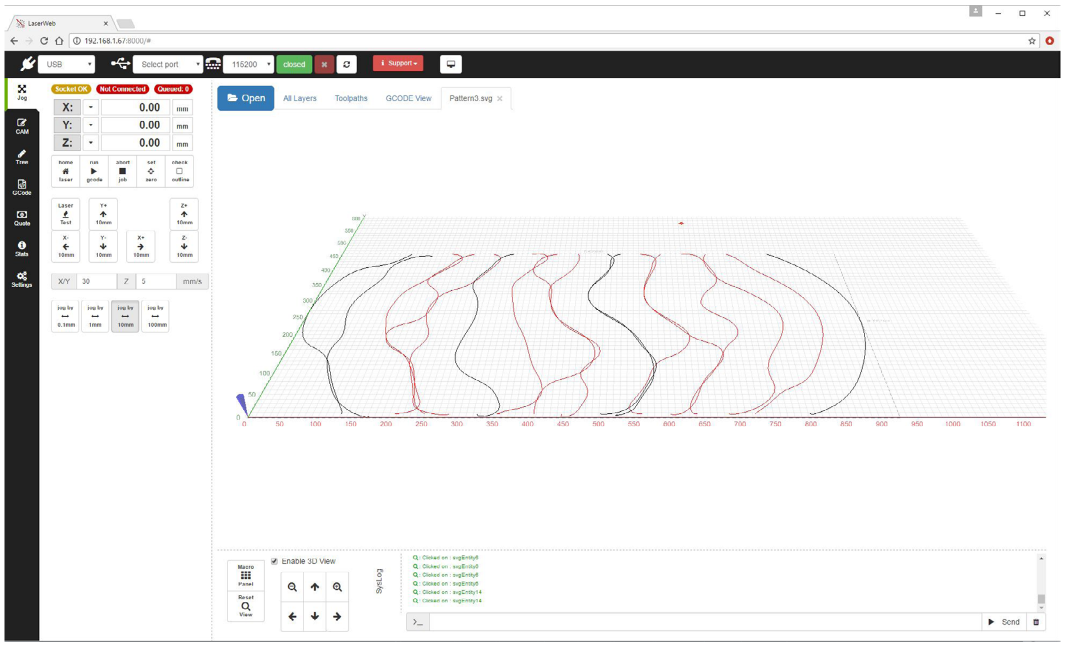
Looking to Industry, even Universal has moved to a more current, software-based, solution rather than relying entirely on an embedded system. Based on these ongoing issues, and a severe reluctance to depend upon a proprietary closed solution, I decided to embrace a more open-source answer to managing the laser. While basic functionality is still acceptable for cutting, a refresh is needed for rasters and the annealing process I'm investigating here.

LaserWeb

The Maker movement's effects have been felt in all areas of digital fabrication. Its influence has not been welcome more than in the area of computer programs. Democratizing software has been the very antithesis of Bill Gates' beliefs since he first broke away from the free software movement of the 70s in his now infamous "Open Letter to Hobbyists"(Gates 1976, 2).

In his letter, he called those who share software "thieves," and asked that those who sold his work to others to pay up. While his position made sense when thinking of those who invest their time and effort to implement business solutions, it was I think directed at the wrong audience. There was no GNU GPL at the time and those who enthusiastic to innovate in the computer industry were busy borrowing from the successes of each other to push the movement forward in an open contract of sharing. It surprises me to think that Gates, who benefited massively from this association with hobbyists, would then turn coat and accuse them of being criminals. Was it OK to innovate only for a few?

Today, intellectual property laws allow for solutions to be essentially "locked up," making it illegal to repair your own iphone(Moody 2015) or John Deer tractor(Masnick 2015). While both benefited heavily from open innovation in their humble beginnings, each starting in someone's garage, borrowing technology(Deer - a broken saw blade, Wozniak - the Silicon Homebrew Computer Club) and know-how from those around them. Having to be tied to embedded systems, using closed protocols, is protectionist and has walled off access to the engraving and cutting markets. I experienced this first-hand in my meetings with Universal when shopping for my machine. As I felt then, I still do now; the cost and veiling of these solutions inhibits innovation. When substantial value is offered by the right solution, the money will follow. Until then, I feel sharing should be the default operating mode, fuelling innovation through access.
This is a very long-winded soap-box lecture on my stance. Laserweb embodies the



```
Q : Clicked on : svgEntity6
Q : Clicked on : svgEntity6
Q : Clicked on : svgEntity6
Q : Clicked on : svgEntity6
Q : Clicked on : svgEntity14
Q : Clicked on : svgEntity14
```

Fig. 145 - LaserWeb Interface - Pneumatic Path View

```
G0 X691.2153 Y6.0400
G1 F1200 X691.2153 Y6.0400 Z0.0000
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
G1 F1200 X691.2153 Y6.0400 Z0.0000 S1.00
G1 F1200 X686.7868 Y7.3386 Z0.0000 S1.00
```

Fig. 146 - Path G-Code Listing - Laserweb

type of thinking I think is necessary for a healthier engraving industry, open access and an open development system. Based on the GPL, all Laserweb code is available on GitHub(van der Walt 2016a) and anyone can join the effort. The project was initiated by Peter van der Walt, a Systems Engineer in Durban, South Africa. The project has gone through many iterations as the momentum has grown. The project is in its fourth major iteration, with each major version including more functionality and breadth of support for controlling hardware as more developers join the effort. The current operating version is Laserweb 3. It supports control of both laser/CNC/and 3D printing. Openbuilds describes it as a:

> "Node.js based, Windows/Linux/Mac/Raspberry Pi/Vagrant supported, host software for Lasercutters/Engravers running Marlin/Smoothieware/Grbl/LasaurGrbl with integrated parametric Gcode generators, Raster support, as well as Raster and Vector Engraving. SVG and DXF supported for cutting, PNG, BMP, JPEG support for raster engraving." (van der Walt 2016b)

This solution is ultimately democratized as the hardware, software, and know-how are no longer locked up. The Laserweb project offers even more, a vibrant Maker community(van der Walt 2016) that collectively responds to user issues, driving them to completion through directed education. This drives down cost and increases quality as solutions are vetted across a wide audience, and ensuring changes to the product are heard and incorporated with buy-in and support of actual users. Win, win, win...etc. You get the picture.

Laserweb has been written to be hosted on many flavours of Linux, and more importantly for this investigation, it can be hosted on the tiny $35 Raspberry Pi microcomputer. Running a fully supported version of Linux(Raspian), the Pi is a Quad-Core ARM-based 64-bit computer running at 1.2 GHz. To give you a sense of how things compare, the original Apple Macintosh was a 32-bit(Motorola 68000) Computer running at 8 MHz. The Pi is literally 300 times faster than the first Mac. And at $2500US adjusted price for each Macintosh, that results in each Raspberry Pi being equal to approximately $750,000 worth of 1984 home computing, and it fits in your pocket.

As a Node.js solution, Laserweb resides as a web application served up by the Pi, so it can be controlled remotely. Our existing solution using Testra is embedded and depends upon a dedicated Windows-7 machine connected through to the controller over a Serial RS232a connector as a printer port. The Testra max's out at 64Kb/s while the laserweb solution uses either USB or Ethernet; both having transfer rates in the MB/s range. A newer, faster, interface allows for interactive fetching of g-code instructions from Laserweb as needed, and this puts it head and shoulders above the one-shot printer driver used by the Testra.

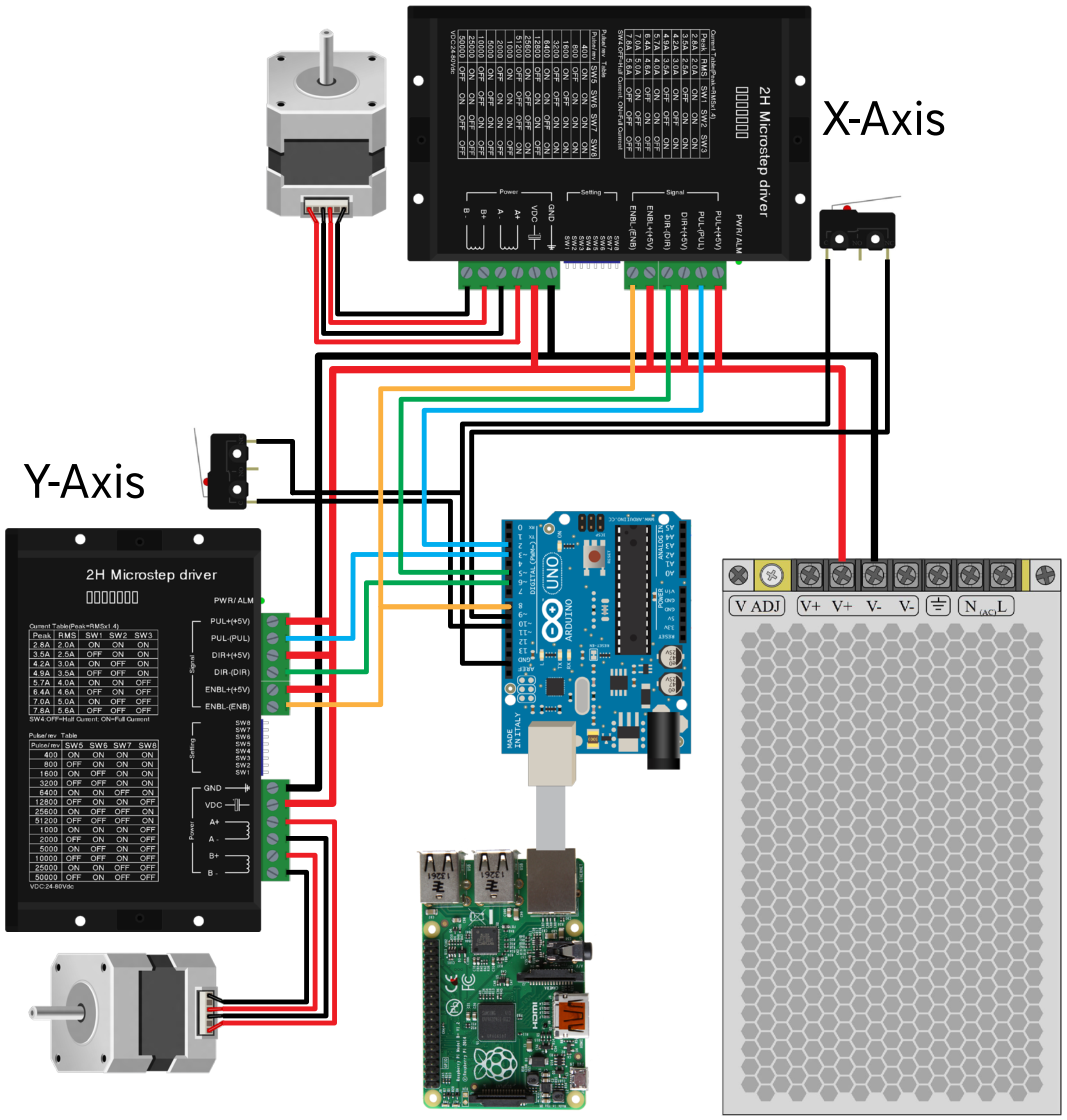


Fig. 147 - GRBL Controller Version - Connected Block Schematic

Controller - GRBL Version - Thought Into Action

The G-code created by Laserweb has to be translated into mechanical action somehow. Here we enter the realm of motors, signals and switches. Grbl(simply short for "Gerbil") is an open-source CNC implementation written in an optimized version of the "C" language(Jeon). It was initially authored by Simen Svale Skogsrud, a designer in Norway at Bengler(bengler.no) who specializes in what he calls "broad spectrum tinkering"(Skogsrud 2017). Grbl is implemented to run on the simple Arduino Uno project system.

The laser system needs precise movement of the stepper motors to locate the optics that direct the infrared beam's energy through to the material surface. The optics are located on moving linear bearings that make up a gantry which slides with minimal friction over the activity field(defined by a Cartesian X and Y coordinate).

The stepper motors react to electrical pulses which turn the motor either left or right in precise steps, allowing for movement within a very small fraction of an inch. In absence of a pulse controlling movement, the system also can lock an axis in place by activating the magnets within the motors, restricting movement of either gantry(like the brakes in a car).

Limit sensors inform the system when it reaches its extremes in absolute position. To optimize hardware, the system uses only one limit switch per axis as it knows the length of each axis(X-48"/Y-24"). As the system counts off it's distance from one of each axes' extremes, it can know when the non-monitored end is reached(within precision limits)

Configuration of this system is done through the Arduino Interactive Development Environment(IDE). Most parameters that apply to the physical implementation of steppers and laser control are managed via the Grbl's "config.h" file. Changes made here are compiled through the IDE and uploaded to the flash memory of the Arduino Uno through my PC's USB port.

Initially, using only a painted piece of plywood, I assembled the circuit for testing. I elected to try to match the connectors existing in the system to make swapping back easy. This proved to be problematic as these connectors are no longer readily available. Simple non-locking pin connectors were used in their place temporarily.

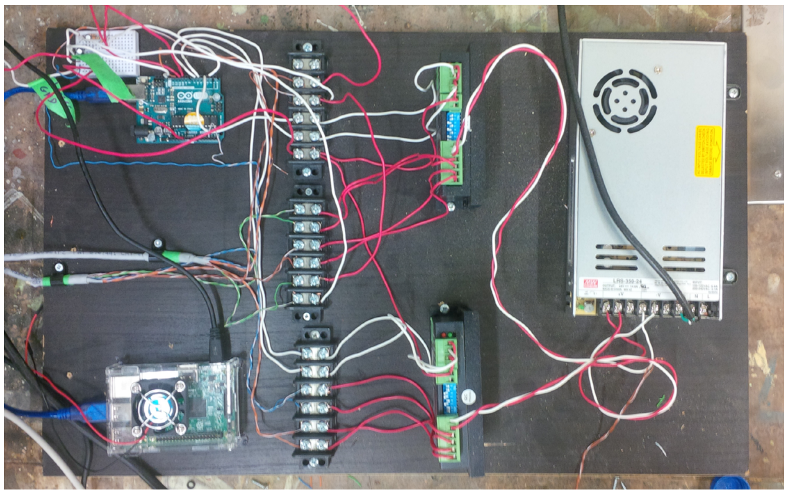


Fig. 148 - Assembled GRBL-based Proof of Concept

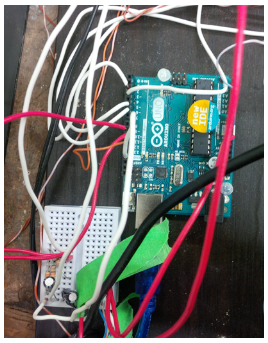


Fig. 149- End-Stop Noise Filter

Fig. 150 - 15A-24V Power Supply

Fig. 151 - Raspberry Pi

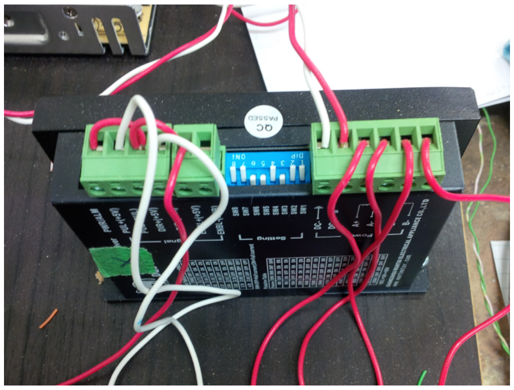


Fig. 152 - Stepper Driver

It was found early-on in the testing that triggering the end-stops cleanly was an issue due to some system-generated high-frequency noise on the signal wires. I had used re-purposed Cat-5 Ethernet twisted-pair wire to limit interference but it seemed the switch itself did not transition cleanly. I assembled a low-pass noise filter(Image-Opposite) to siphon off these errant signals to quiet the system down. This worked quite well and I was able to get the system to find “home” cleanly during its initial power-up calibration(gcode-G28).

The Arduino Uno control board does not have the ability to drive the stepper motors itself(each requiring up to 3 amps of current) so stepper drivers, activated by a 5Vlogic pulse, were used to pass on the current needed. This relieved the uno of need to match the power demands of stepper loads and reduced the cost of the controller substantially and this allowed the Uno to be powered by USB current.

Testing GRBL, at the then-current code revision 0.9, was quite a challenge. Despite the claims of the developers, I found its performance to be choppy and very hard on the hardware. One of GRBL’s great contributions to this area is its claimed ability to drive steppers cleanly while looking forward at programmed movements, like G-code pipelining. An intelligent controller would take into account planned changes by buffering movement requests; slowing change so as to be forgiving on the structure/belts/motors while offering efficiency.

What I experienced was all too often abrupt changes to the gantry motion, resulting in a jarring that required me to re-tighten its fasteners often. The optics that the gantry is charged in moving, also became out of alignment only after a few steps. The optical mounts did not do well under high-stress. While they can hold the mirrors quite securely, the screws used to adjust alignment did not do well under the differential stress of high-G movements. It didn’t take long for the laser system to become a hazard.

Barring any code updates, I felt it useful to look at Open alternate driver solutions. There were a few(Ramps, Smoothie, Mach3, Marlin ..etc.) but many were still based on the arduino and I suspected that the speed of Arduino implementations may have been an issue. I elected to look at alternate controllers and the Smoothieboard project quickly rose to the surface. Based on the ARM 32-bit Cortex-M3 running at 120MHz, it leaves the Arduino Uno(8-bit 16MHz) in the dust.

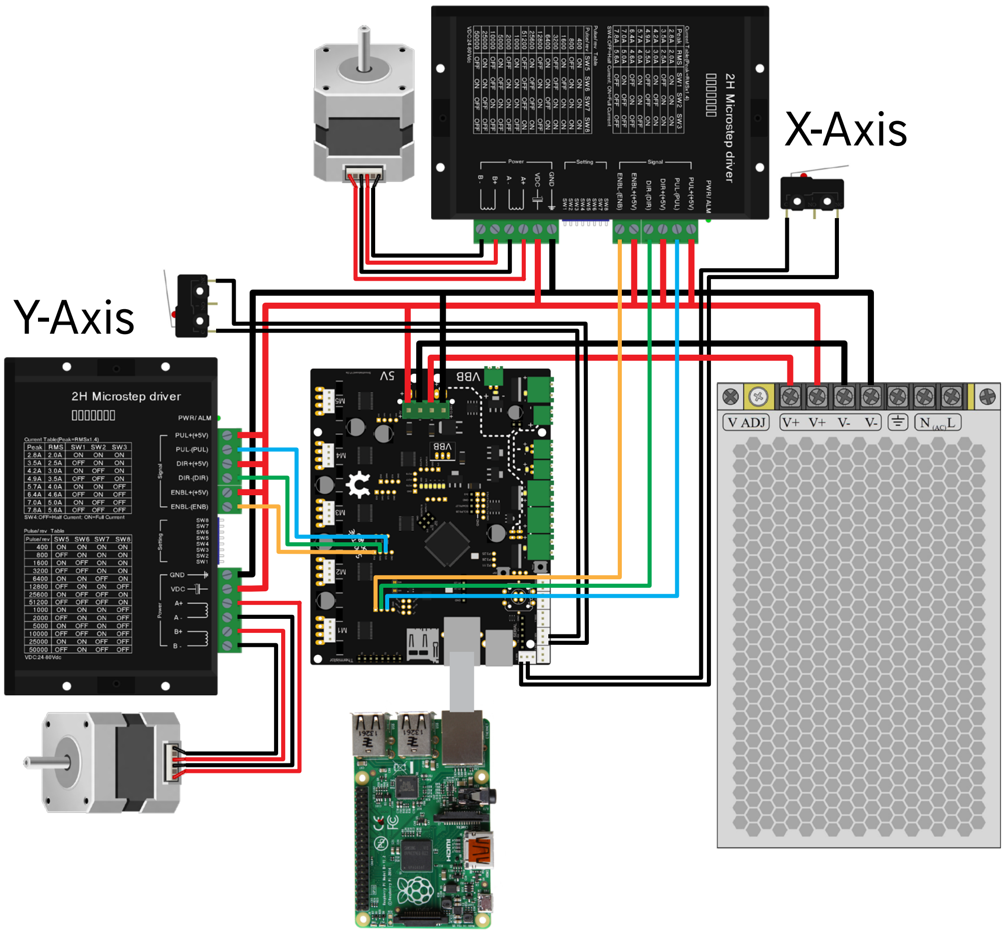


Fig. 153 - Smoothie Solution - Connected Block Schematic

Controller - Smoothieboard Refinement

The Smoothie Project was started as a fork in the GRBL project. As anyone can do, the developers of Smoothie felt that Arduino was inherently limited but wanted to base their work on the successes of GRBL. At the time I was search for a G-code interpreter and stepper driver, Smoothie was still in development. As a 32-bit 120KHz machine, it promised at least a 30X improvement in processing ability and native ability to drive larger steppers with greater precision(steps/microsteps).

While coming with a greater capacity to drive larger steppers, the Smoothie tops out at 2A for its stepper drivers, so I still had to utilize my driver hardware. The Smoothie board I ordered came with 4 channels, allowing for a Z axis, plus one other axis(R1 perhaps for rotation). The Ver 1.0 Smoothie is considered a general-purpose controller; offering the ability to power 3D printers and larger CNC applications.

Implementing the smoothie was as easy as swapping out the hardware and configuring the card to understand the specifics of my Laser table. There's no need for an IDE and compilation step to configure, an on-board sd-card is accessible via USB or Ethernet, or can even be ejected and configured using a separate sd-card loader. It's config.txt file can be directly edited and current boot image swapped out for fixes or updates. As a larger card, it requires more power and there was ample with my current supply.

Smoothie has an integrated buffering on their limit switch channels, so no more false positives or barnacled high-pass filters. I elected to buy a board unpopulated with connectors. There's enough interfacing on board for many applications. I felt I could save some cash by only soldering on what was needed($130US). All the logic is in place for all functions. If I need to re-purpose or add another needed channel to the system(a CO_2 laser coolant flow sensor perhaps).

If you think I'm a convert at this point, you might be right. This board has been thought through. I am very happy to see such a smart Open-Source product being offered to the market.

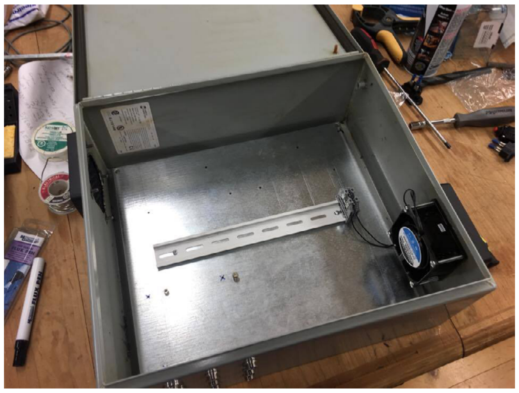


Fig. 154- DIN Rail System in Hammond Case

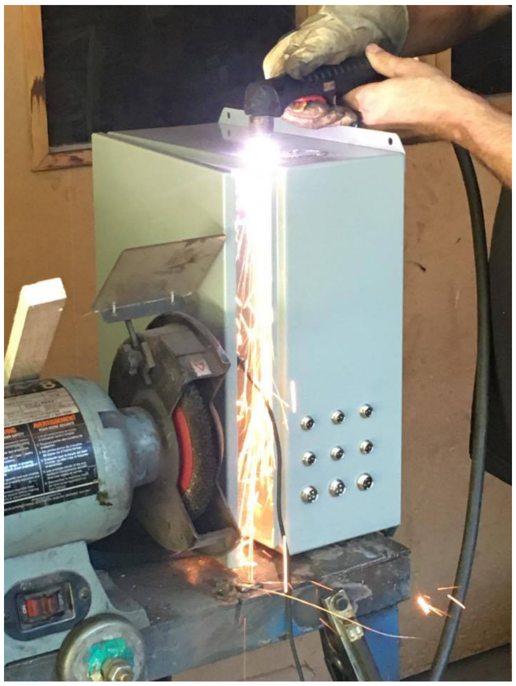


Fig. 155 - Plasma-Cutting of Interface Ports in Case

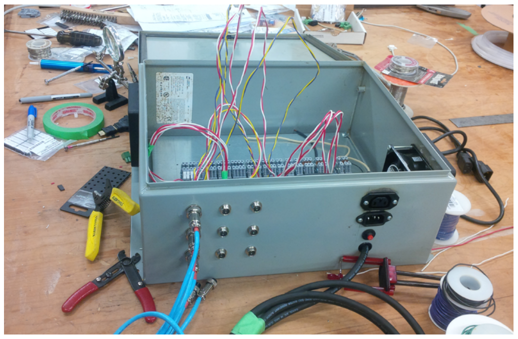


Fig. 156 - Signalling & Power Connection

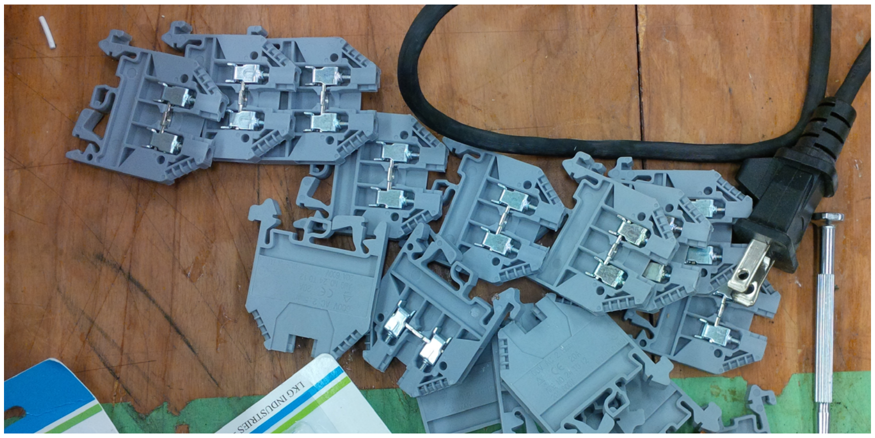


Fig. 157 - DIN Terminating Blocks

Controller Final Build and Implementation

As this was going to live in my workshop(a relatively dirty environment), I figured it made sense to ensure the controller was housed and cooled properly. That meant, a sealed steel commercial cabinet from Hammond Electronics in Guelph, industrial connectors, fusing, and filtered active fan. I was able to source the cabinet from Torbram Electric Supply in Waterloo. They stock some of the more common Hammond components. The cabinet is thick, enamel-coated steel and required hardened steel drill-bits for the industrial connectors' positioning and the use of a plasma cutter for larger holes(opposite).

In this environment vibration can be an issue if other equipment is operating, so I elected to use an industrial DIN-rail system to facilitate connectivity. DIN allows for each wire to be fastened using a recessed, tightened, screw so there is little chance of wiring becoming an issue while in production or maintenance. The Hammond case comes equipped with an offset mount-plate for components. I chose to mount the DIN rail down the centre and organized connectors on one side of the cabinet for easy access. Torbram stocks DIN rail components as well but these were much cheaper in small quantities from Sayal Electronics in Cambridge.

Throughout the build process, it has been important to ensure that moving back to the working, older, controller be possible if there is a failure with the new. To help with this process, I made an additional connector set to allow the older controller to connect to the newer termination system. It was a bit of duplication, but instilling resilient capacity to the laser cutting system will ensure that it will always be available, barring a fundamental failure of the laser or mechanical systems.

The Laserweb system is web-based. It will be necessary and helpful sometimes to walk up to the machine and print a file. Also, control for the system should have some locally positioned instrumentation to allow for laser power changes, job management, cooling alarms/emergency shutdown ...etc. I've included a fused 110V switched power connector mounted in the case to support a future touch-panel that I will mount on the exterior of the machine.

With the control cabinet permanently on the lower portion of the laser table, it can be accessible when needed but to connect through to the Raspberry Pi computer inside the box, a wired Ethernet connector is needed because of the RF insulating capability of the metal case. Without it, the shop's wifi signal will not be able to

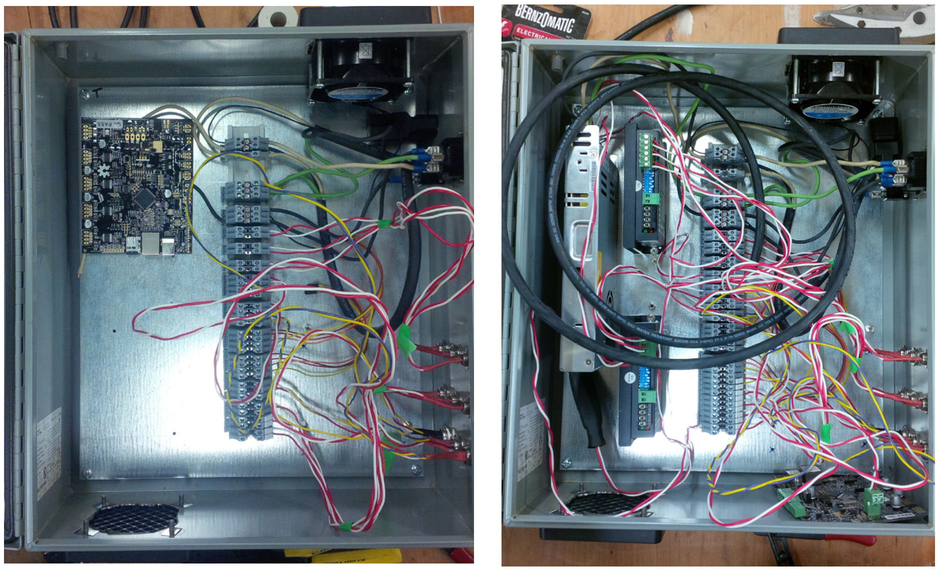


Fig. 158 - Smoothie Controller - Fitting and Assembly

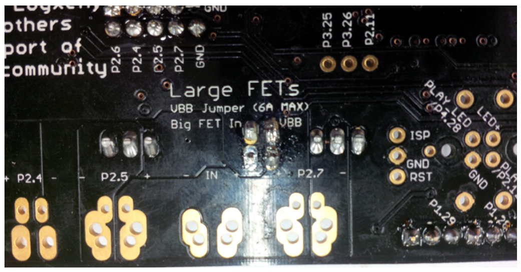


Fig. 159- Smoothie Soldering Repair

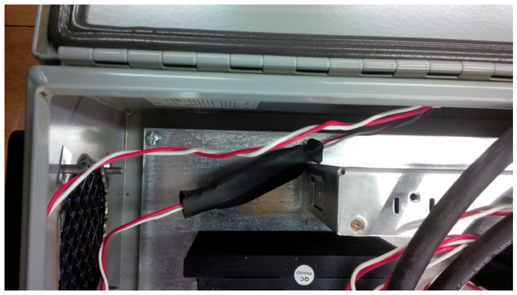


Fig. 160 - Shrink-wrap &
In-line Voltage Reduction(24V->5V)

reach the Pi.

Internally, the components were arranged to allow for maximum air movement for cooling, ease of access for maintenance and wire management, and specially to separate power components from computer components.

There also has to be a sensitivity to inductive interference caused by close proximity of input to output wiring. As I mentioned before, all internal signal wiring is twisted to minimize crosstalk as is done in Ethernet cabling. Twisting ensures that if an electromagnetic pulse effects the wire at one point, the effect is negated by the wiring being in an opposite configuration an inch or so down the signal path. This trick works for high frequency signal induction and is necessary in all conductive network cabling to ensure quality signalling.

Separation of analog and digital signals also ensures there is less chance for inductive affects to cause an issue. With power components(Power supply, stepper drivers) split from digital processing components(Raspberry Pi, Smoothie) I accomplish this within the box. To take this further, inductive effects can be minimized by minimizing how much untwisted lines are bundled together. This is done within the box by allowing analog wiring to be run point-point.

The smoothie's arrival gave me a chance to dust off my soldering skills. I probably should have practiced a bit more before moving straight to the empty production board. I made several mistakes, which also allowed me to dust off my repair and de-soldering skills. I had to repair a trace and refresh the tips on my 20+ year-old Weller soldering station. Thank you Sayal Electronics. Radio Shack is long gone... There are definitely less places that support the home maker-space these days. Long gone are Waterloo Electronics, Orion Electronics is now a travel agency, and once a Mecca of used computer gear/parts/supplies, KW Surplus now sells food and clothing; their once-proud electronic surplus relegated to a dusty corner of one or two forgotten display cases.

Shrink-wrap tube became my friend in this implementation. With all connectors being custom, internal termination also benefited from some proper isolation. Shrink-wrap tubing comes in many sizes and colours and is really the best way to ensure wires stay in place and don't affect one another. Size and slip the rubber piece over the area that needs to be insulated/reinforced, apply a bit of heat from a heat gun/hair dryer/torch and the tubing shrinks around the area tightly. Best invention ever.

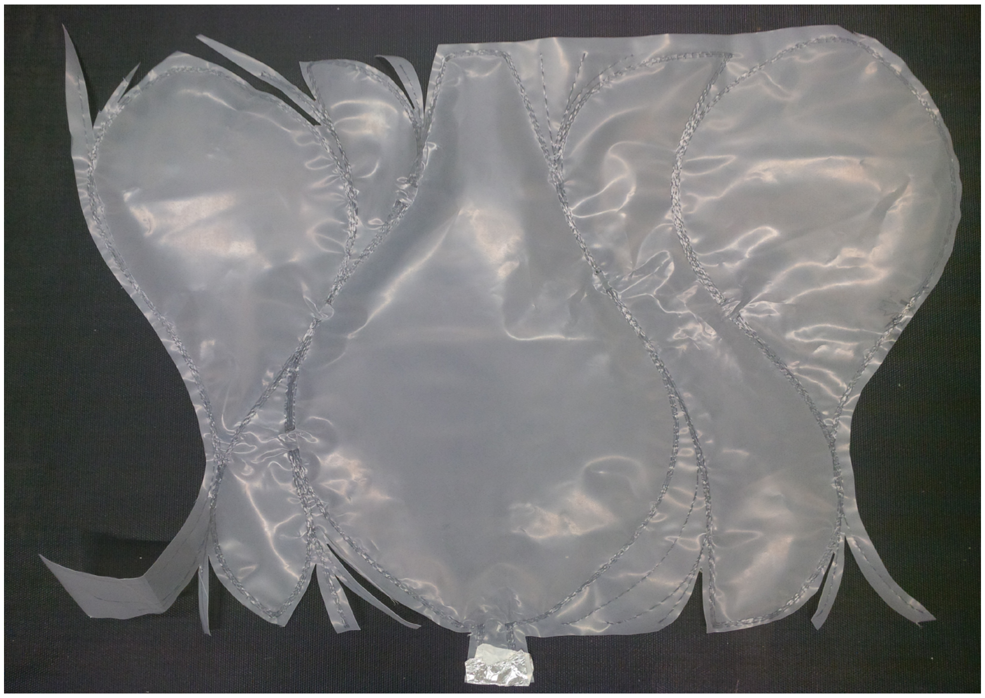
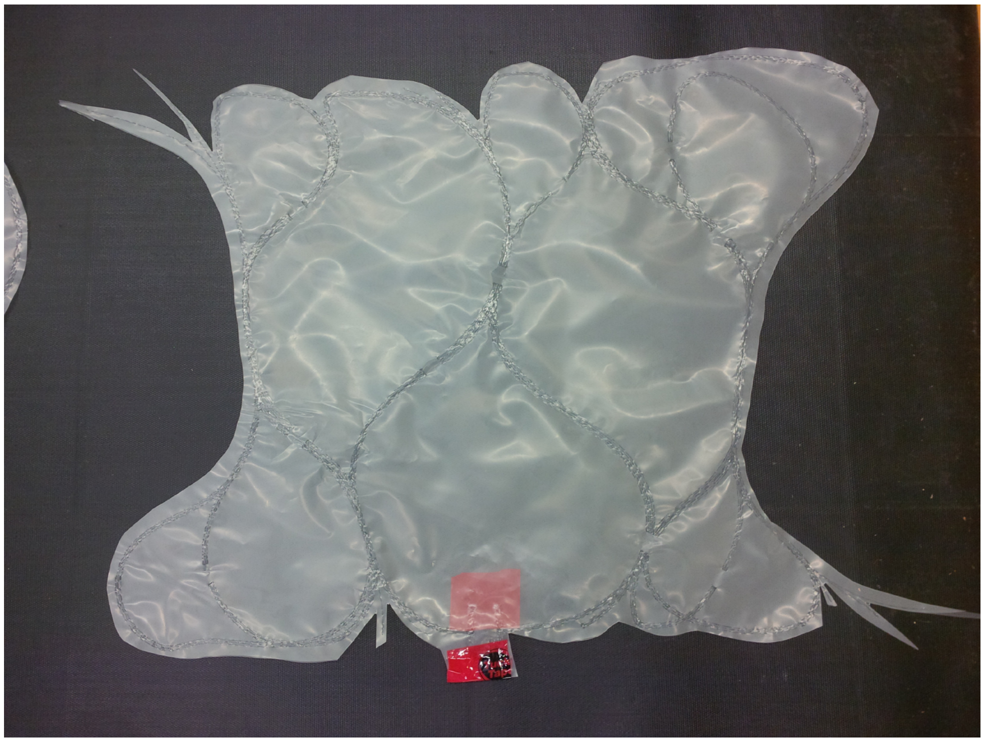



Fig. 161 - Pneumatic Chamber Results - Smoothie Controller

Output - Smoothieboard Successes

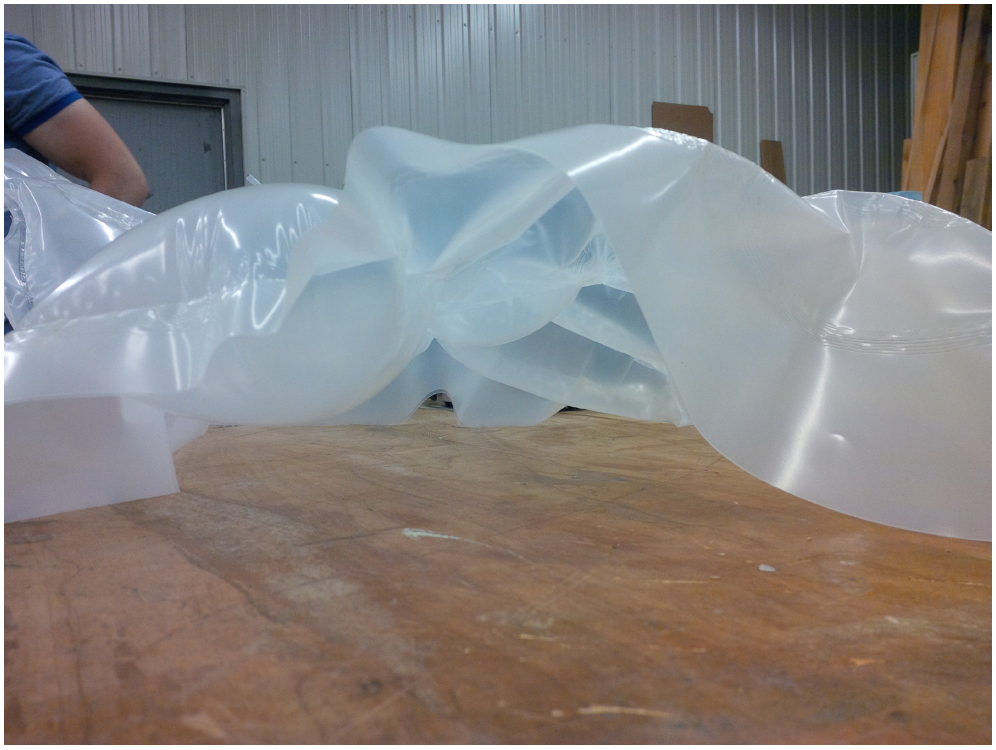
After some minor teething, the system fired up with little issue. End-stop calibration, pwm laser modulation and power setup took a bit of time. The Smoothie system allows for almost any output pin to be modulated via pwm. This really is something. I could modulate the output pins of almost any of the stepper driver lines to control an host of peripherals. This card really is something.

Smoothie does two things that GRBL didn't do, and it does them really well:

> It's look-ahead and state optimization are head and shoulders above the GRBL. I could *hear* the difference immediately. The steppers were almost singing while they moved. The Smoothie has the ability to modulate the signal, using larger steps and smaller microsteps, to ramp up and down the speed of the gantry in an incredibly smooth way(hence "smoothie"). I am amazed how much better the movement happens. It's *seems* more intentional and less forced. Curves are smoother as well. I suspect with better control, comes better geometric output.
>
> Pwm optimization of the laser is finally configured to follow the curves and is modulated to allow for uniform energy release along the curve. There are no more holes and energy transfer doesn't bunch up in tight corners or curves. The resultant pneumatic envelopes seem more consistent and hold greater pressure. They're not perfect, but this process has improved noticeably.

Smoothie does a few things more poorly as well. There may certainly be fixes for these but at the time of this investigation, these were my pain points:

> Smoothie does hiccup sporadically when beginning a run. It forgets its location and attempts to re-home after it has homed already. This causes the gantry to try to move itself past the endstops and off the rail. It's a very disconcerting thing to watch and a real pain to fix. This behaviour, wouldn't normally bother me but the gantry ceramic bushings are effected when forced inappropriately. The more this happens, the sooner I may have to replace them. They are already beginning to crumble.
>
> Seek and calibration speeds have been difficult to manage. This will re-



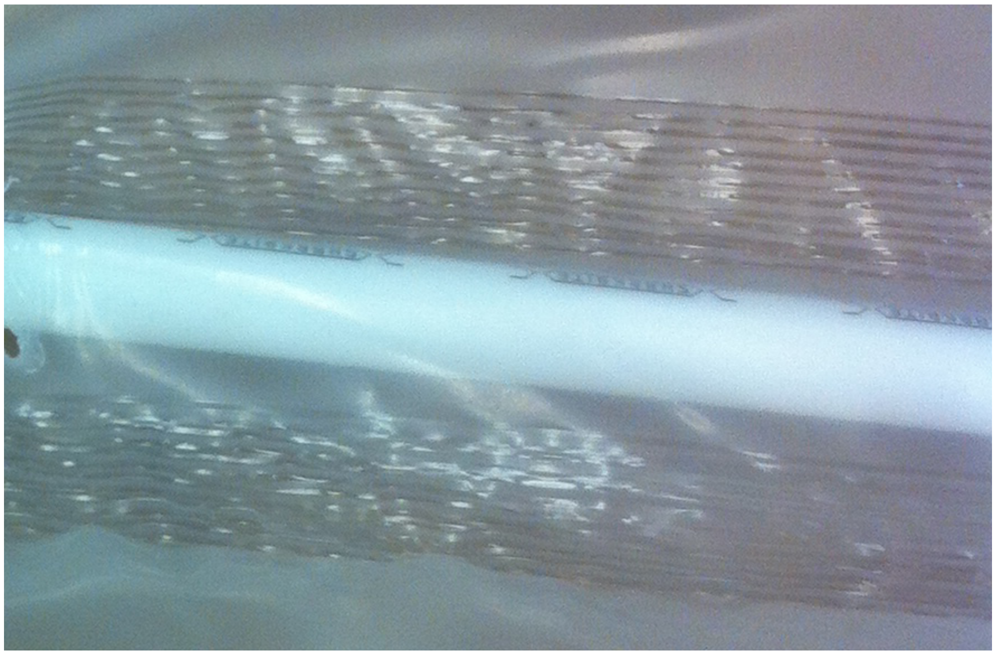


Fig. 162 - Multiple-Line-Offset Pressure Reinforcing

quire more time with the machine. It can be too aggressive when changing speed modes. I will have to consult with the community to see if this is a laserweb issue or if it belongs more within the Smoothie's category.

Results - Pneumatic Structure Proof of Concept

6mm Polyethylene is a fairly rugged material to work with. It's tough and resistant to tearing and puncture within its normal operating temperatures. Thankfully, it's relatively inexpensive and quite easy to obtain. Throughout this investigation, our roll of 6mm poly was in the driver's seat. And while it was always relatively easy to cut, coaxing it to melt to itself was no small feat. Too little heat and it might wrinkle and fuse poorly or not at all, too much and it separates, melting to itself while creating a bead or catching fire - spewing black smoke up to the optics of the cutter(incurring a lengthy cleaning cycle or lens replacement). Air assist was of help initially to keep the optics cleaner, but the air provided cooled the poly too much to be of use.

The air chambers we wished to create were arranged in such a way as to span across the length of the material. To accomplish this I used the Kangaroo add-on from within the Rhino/Grasshopper environment, modelling a similar system of paths that ribs might take across the space. Each path was a collection of points and lines that could be altered parametrically in their span. The lines were allowed to stretch(forming a figure) within limits but would always remain consistent within themselves. When attractive forces were applied to each path while repulsive areas were defined within the line environment, the system would be left to resolve its best configuration. These forces were themselves parametrised and were placed arbitrarily based on my input and intuition. It was through a number of iterations that a set of possible solutions was found.

While initial attempts to fuse the polyethylene were completed manually, it was through the laser annealing process produced in this investigation that allowed a more reliable result. Through iterative trials with the laser cutter, I was able to determine what settings were best at producing a clean seal. It was only through successive pressure tests and adaptations that a reinforcing process was developed to reduce the strain on the material during use. It seemed that once the polyethylene was heated, the regions of adhesion were significantly weaker than the original material and this required a more

Fig. 163 - Pneumatic Structures

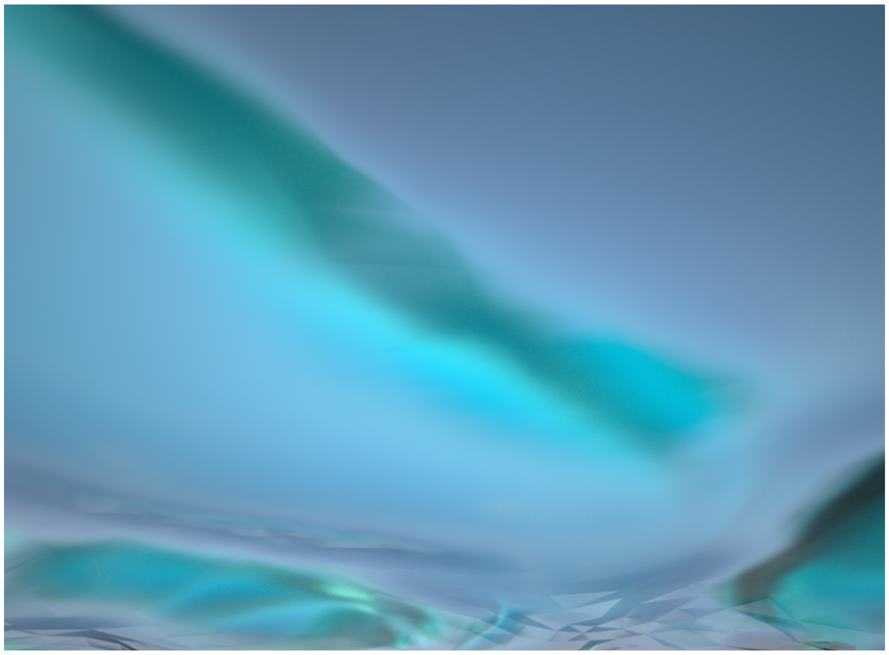
gradual approach to dealing with pressurization and the forces involved. The solution was to add offset paths that enclosed the existing paths in a spline-like manner.

It was determined that the rib paths must be enclosed single loops. We found that if the paths were made of multiple polylines, their endpoints would cause the laser to dwell, if even for a split second at each of these points. Reducing end points reduced blow-through of the material during adhesion, and thus reduced the chance of leaks being formed by the annealing process.

Further issues of laser power modulation in tight curves force me to look at the controller architecture. Due to its age, and lack of documentation concerning configurability, I decided to look for alternatives. A replacement solution that seemed the best fit was an open-source solution named LaserWeb that was hosted on a RaspberryPi computer coupled with an Arduino CNC controller.

I built a test rig, and was able to setup the LaserWeb(Version 3) system. After running a few tests, I determined that the CNC controller board was not a good fit and replaced it with a more advanced controller known for its smoother operating control of the laser's stepper motors. With the system architecture set, I decided to move to a more production-grade implementation and move the components to a shop-grade enclosure with proper cooling and interface protection.

The end result was a more consistent controlling system that offers a web-based interface. It runs off of inexpensive components and an open operating system(Linux). The controller offers greater control and parametrization of the process, and also affords the ability to customize itself for other materials(and processes) more easily.



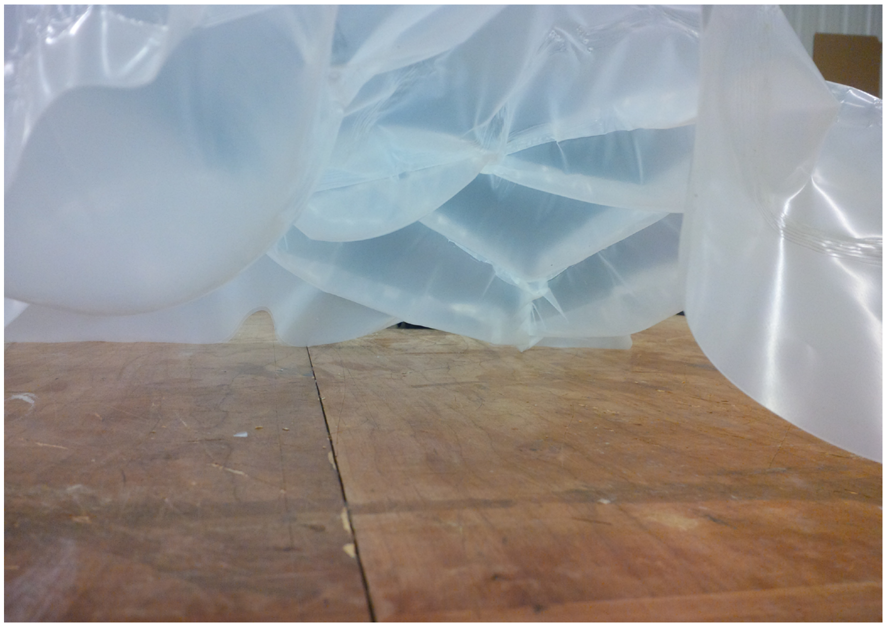


Fig. 164 - Render + Pneumatic Structures

Conclusions - Pneumatic Structure Proof of Concept

As with Sennett's Chef, this portion of the investigation highlighted the entire range of relationships a craftsman might have with their tool. In the first section, I searched for an understanding of the language and characteristics of digital making in the build and commissioning of a delta-based 3D printer where I was introduced to the process and implementation of making in PLA plastic for use in the first two investigations. In the second section, the investigation started with a goal determined by a group project to produce set of analogue experiments in pneumatic structural design. The material, 6mm polyethylene, and the process of plastic welding, guided a number of the parameters within the design of the tool system used to fabricate the tests.

The pneumatic chamber design process was based upon Frei Otto's wool-thread experiments in network optimization. Using Rhino3D/Grasshopper and a hierarchical implementation of the Kangaroo physics solver, I was able to simulate the form we were trying to fabricate. As with anything virtual, the project took a whole new direction when we started to work with the 6mm polyethylene directly.

The materiality of the poly re-shaped my thinking of how fusing processes work. While direct feedback was very possible during refinement, iterative change was seen by the adjustment of parameters of the controller's hardware systems and while these changes were difficult to realize initially(config file changes/process updates, mechanical and vacuum adjustments), I became quite adept at making them quickly as the refinement processes evolved. My goal was to have the capacity to reproduce the paths and process correctly every time it was needed. I was able to reach a point of not really thinking about the changes that were needed; I had reached a point of simply adjusting the system by "feel," minimizing the effort while anticipating the outcome intuitively.

There are better processes for melting plastic in a uniform manner(heat rollers, pin-based systems, etc.) but this system affords me the ability to change the curves easily and quickly, and it is more consistent. It is also a hack of a laser cutting system; the originally configured tool was not meant to do this. I had to get into its head-space for the results to go from mind to hand, from hand to tool, and finally from tool to material. As I slowly improved the control system, the design intent was better realized. With less pin-holing in the output, the pneumatic system could literally hold it's air almost indefinitely. And once welded properly, it's resilience was surprising. Taking the pressure up to 100+psi without issue became the new normal. I had learned how to work with the digital system and the material to achieve and surpass design goals.

The process of making is messy. Through countless iteration, the laser system began to show its tolerance for stress; the mirrors constantly would go out of alignment due to vibration caused by motion trials, the collimation system's lenses had to be cleaned continuously due to the smoke of melting plastic, the gantry rails and gaskets are all but worn due to improper end-stop settings and errors in movement—all necessary when pushing new boundaries in process learning. Sennett states, "[a craftsman] must dwell in error to understand working procedures." (Sennett 2008, 162) This state of material indecision and learning is necessary to support change.

Conclusions

In this thesis, my investigation has travelled from head to hand, from hand to tool, from tool to material and back again. Beginning this work, I was naively looking to understand what was involved in unifying head and hand in the realm of the digital. I've since discovered that there is so much more to consider. Much is the same, but craft within the digital asks for more. It asks the craftsman to focus not only on the use of their tools, but more so on the development of these tools.

Tool and tool making is a craft all its own. In digital design, creation of the tool is a process akin to mind-melding with the processes/materials of making coupled with the technological implementation of what Sennett refers to as "transitional objects"(Sennett, 2009,159). I would agree with his take on this. The virtual models/processes and sketch-physical models stand in for analogues of mental ideals. They're disconnected from the real world in their perfection and only are able to land when making takes place.

The digital modelling of the diatom was a pure mathematical abstraction until the limited real-world demand that its surface be manifold(continuously closed). Driven to ensure the model was makable, the process of design looked to incorporate those explorations that could allow it to exist in the physical world. The tools of making were instrumental in this.

Yes, there are be many phases of making. The digital designer must develop an intimate relationship with the tools of digital making, be they virtual or physical. In each sense, I began to develop prehension(a feeling of anticipation) while making. I began to anticipate next steps without thinking about them. These feelings were more clear when I built/designed the tool myself compared to configuration of an existing tool. Both types worked on me, the designer, but when I already had a mental connection to the construct, be it digital or physical, because I created it, the connection was clearer.

At many instances the system I had implemented resisted scaling due to complexity issues. Whether it was the addition of more than seven levels of differentiation in the diatom Grasshopper model, or the number of distinct line segments that made up the paths in the pneumatic form simulation, or the $O(n^2)$ assumption of the Diatom Coulomb agent implementation, each offered direct resistance during implementation. The result at every turn was an understanding that the direction I had chosen would go no further, and that I needed to think of another direction to go to fulfill my design intent. Resistance spurred ambiguity, and ambiguity spawned imagination. I found this to be a repeatable theme, constantly hitting execution walls. The result spawned better more-optimized code, better choices for design directions, as they uncovered weaknesses in my thinking.

In every project, I developed a sense of what Sennett referred to as "minimum force" thinking(Sennett 2009, 165). Initially, it may have been due to finding the path of least work, but later it became something that manifested without thinking. Minimum force later on became a set of tasks done in an order because it was the most efficient way to an end. I'd stopped thinking about how to prepare path files for the laser system during my pneumatic investigation and only remember the results during our many iterative changes. The tasks of making became quick and efficient because the design iteration process demanded it.

Some of the formal tests embody a philosophy Sennett calls "fit for purpose" procedures or tools(Sennett 2019, 162) where he feels it's important to "dwell in error" because that is the only way to truly understand a system that you are developing. My TensorFlow investigation took more than a month to realize and its development helped me to fully realize at a low level what was necessary for me to bridge head and hand when managing a large number of software agents effectively. I lived within my mental model of an agent management system for more than a month before I came to understand what was necessary to complete the design task. It helped me to make better decision(defer this development until greater programming skill has been acquired) but more so, I gained a marvellous in-depth understanding of when to terminate a design decision. The result allowed me to find and configure a better tool for the task. The digital designer needs to let go of directions that are unfruitful and identify quickly when this condition exists.

Miles Davis is quoted as saying "Be Wrong Strong." As a master craftsman, his work embodied innovation and improvisation. Hacking, the art of using a tool for other, than its designed intent(sometimes for more nefarious purposes), is important to the digital craftsman. It embodies Deleuze's sense of multiplicities and is a very powerful methodology. Craftsman innovate with what they have at their disposal. Their head and hands, adept at making, are able to feel new possibilities when faced with a new problem. Their knowledge of their system affords them an ability to solve problems quickly and without prior restraint. Again, I'm but a digital craftsman in training, but when I needed a digital system that would melt plastic, my close experience with my laser cutting system pointed me to think that it might be able to melt plastic if configured and changed properly. The result, involving successive improvements to the process, was a minor hack. I wouldn't have even thought of it had I not known this tool well.

As a budding digital craftsman, I see that I'm at the very beginning of an exciting journey. The digital craftsman still must put in their time(10,000 hours) to fully understand their craft; the machine does not move this any faster. In fact, I see the digital designer today having to be a master of an uncountably large number of tools and systems. For this investigation alone, I had to become very capable with the tools in this list:

Software	Hardware
Rhino	Switching Power Supplies
Grasshopper	Steppers + Drivers
Realflow	Electrical Wiring and Bus Systems
Maxwell	RaspBerry Pi
Java/Processing	GPU configs - SLI
Mandelbulb	Cable management
Z-Brush	HTC Vive
meshlab	PVA/PLC Processes
ImageJ	Multimeter/Weller Soldering Station
Handbrake	Arduino
Kodon	SmoothieBoard
TiltBrush	Signal Filtration & Buffering
3DS Max	
Maya	
Unreal	
LaserWeb	
Repartier	
Simplify3D	

Where there was no tool, no way to connect, I had to design and build one. This is the true bridge between head and hand for a digital designer. The tool is the connection. It embodies the thinking of design goal and does so in a language that anticipates the characteristics of the material. It takes the virtual construct and translates its intent to the material. In this sense the digital designer must get close to each, and she must learn to speak the languages of each. While Sennett's chef speak to his craft through his hands using his knife as a tool, his craft also works on him—communicating it's materiality, communicating its language. The digital designer, while embodying much of what Sennett's craftsman do, must also take this roll as well.

Bibliography

Alanus, de Insulis,-1202. 2007. Les Proverbez D'Alain, edited by Thomas Maillet, Tony Hunt. Paris: Champion.

Al-Dakoki, Ali. "Digitmakers.Ca Website." https://digitmakers.ca/product/3d-printer-delta-rostock-mini-g2s-diy-kit-with-auto-leveling/., accessed Jan 5, 2016.

Aristotle. 1988. The Politics, edited by Stephen Everson. Cambridge, England]; Cambridge Cambridgeshire]: Cambridge University Press.

Bader, Christoph and Kolb Dominik. "Profile - Descriptive - Facebook." https://www.facebook.com/deskriptiv/about/?ref=page_interna., last modified Jan 19, accessed Jan 14, 2017.

Bader, Christoph and Kolb Dominik. "Deskriptiv - Behance." https://www.behance.net/deskriptiv., last modified Nov 15, accessed Dec, 2015.

Bader, Christoph. "Inspiration Now - Deskriptiv." http://www.inspiration-now.com/double-mesh-by-deskriptiv/., last modified Feb 25, accessed Jan, 2017

Bader, Christoph. "CA - descriptiv." http://www.creativeapplications.net/tag/deskriptiv/., last modified Nov 11, accessed Dec, 2016.

Bader. "Flickr - descriptiv." https://www.flickr.com/photos/deskriptiv/., accessed Jan, 2016.

Bader, Christoph and Neri Oxman. 2016. "Recursive Symmetries for Geometrically Complex and Materially Heterogeneous Additive Manufacturing." Computer-Aided Design 81: 39-47.

Bradbury, Jane. 2004. "Nature's Nanotechnologists: Unveiling the Secrets of Diatoms." PLOS Biology 2 (10): e306.

Brown, Wade and Galen Jones. "Riemann Chair". April 2013.

Cox Eileen J., "Morphology, Cell Wall, Cytology, Ultrastructure and Morphogenetic Studies" in The Diatom World, ed. Joseph Seckbach et al.(Springer Netherlands, 2011), p38.

Cox, Eileen J. 2010. "Discrete Free-Boundary Reaction- Diffusion Model of Diatom Pore Occlusions." Plant Ecology and Evolution 143 (3): 297-306.

Cox, Eileen J., Lisa Willis, and Katie Bentley. 2012. "Integrated Simulation with Experimentation is a Powerful Tool for Understanding Diatom Valve Morphogenesis." BioSystems 109 (3): 450-459.

De Stefano, Luca. 2005. "Nanostructures in Diatom Frustules: Functional Morphology of Valvocopulae in Cocconeidacean Monoraphid Taxa." Journal of Nanoscience and Nanotechnology 5 (1): 15-24.

deskriptiv. 2016."A Unified Approach to Grown Structures." https://www.behance.net/gallery/21605971/Neri-Oxman-Wanderers., accessed Nov 30, 2016, https://www.youtube.com/watch?v=9HI8FerKr6Q.

Dieuwer, F. "Realflow 2013 Tutorial: Flooding an Aquarium<br>." https://www.youtube.com/watch?v=r_XE_8WWJxQ., last modified February 8, accessed Jan 23, 2016.

Diller Scofifio + Renfro. "Blur Building - Diller &amp; Scofidio." http://www.dsrny.com/projects/blur-building., accessed Jan 28, 2016.

Gates, Bill. 1976. "Open Letter to Hobbyists." HomeBrew Computer Club Newsletter Volume 2 Issue 1, January 31, 2.

Gielis, Johan. 2003. "A Generic Geometric Transformation that Unifies a Wide Range of Natural and Abstract Shapes.(INVITED SPECIAL PAPER)(Author Abstract)." The American Journal of Botany 90 (3): 333.

Giulio Piacentino. "Weaverbird – Topological Mesh Editor." http://www.giuliopiacentino.com/weaverbird/., last modified August 31,2011.

Hardy, G. H. An Introduction to the Theory of Numbers. 6th ed. / [revised by D.R. Heath-Brown and J.H. Silverman]. ed. (Oxford ; New York: Oxford University Press,2008) p7-8.

Hegglin, Roman. "Bracelet - Voronoi Style." http://www.thingiverse.com/thing:192211., last modified Nov 28, accessed jan 6, 2016.Licensed under Creative Commons Attribution 3.0 License https://creativecommons.org/licenses/by-sa/3.0

MakerBot Industries, LLC. "Makerbot Thingiverse." http://www.thingiverse.com/., accessed Jan 20, 2016.

Hopkins, Jeffrey, Dalai Lama, Tenzin Gyatso, and Richard Gere. 2005. Meaning of Life. Somerville, MA: Wisdom Publications. http://replace-me/ebraryid=10407976.

Jeon, Sonny. "Grbl Source." https://github.com/grbl/grbl/wiki., accessed Jan 12, 2017.

Lamé, G. 1818. Examen Des Différentes Méthodes Employées Pour Résoudre Les Problèmes De Géométrie.

Liu, R. T. and L. 2006. "Two- Stage Turing Model for Generating Pigment Patterns on the Leopard and the Jaguar." Physical Review.E, Statistical, Nonlinear, and Soft Matter Physics 74 (1): 011914.28-47: John Wiley & Sons, Inc.

Lynn, Greg. "Noah - Divide." http://glform.com/environments/divide-film/., accessed May 05, 2017.

Lynn, Greg and Mark Rappolt. 2008. Form. 1st ed. New York: Rizzoli.

Masnick, Mike. "John Deer Owns the Software in Your Tractor." https://www.techdirt.com/articles/20150421/23581430744/gm-says-that-while-you-may-own-your-car-it-owns-software-it-thanks-to-copyright.shtml., last modified April 23, accessed Feb 17, 2016.

Matsumoto, Dave. "Single Walled Test Piece." http://www.thingiverse.com/thing:1637., last modified Jan 22, accessed Jan 6, 2016. Licensed under Creative Commons Attribution 3.0 License https://creativecommons.org/licenses/by-sa/3.0

Maruyama, Shinichi. "Maruyama - Artist's Statement<br>." http://www.shinichimaruyama.com/statement., accessed Jan 17, 2017.

Moody, Glyn. "Apple Testifies Against "Right to Repair"." https://www.techdirt.com/articles/20170215/02473736716/apple-wants-to-stop-you-fixing-your-iphone-ipad-source-says-it-will-testify-against-right-to-repair-legislation.shtml., accessed Feb 15, 2017.

Miyake, Jun. 2011. Pina - Soundtrack - the here and After Lisa papineau. 380 Grad.

Naqvi, Asif. "Shinichi Maruyama - Early Bio<br>." http://livingdesign.info/2010/09/27/shinichi-maruyama-writing-in-the-sky-kusho-water-sculpture/., last modified Sept 27, accessed Jan 21, 2017.

Oxman,Neri . "Neri Oxman." http://www.materialecology.com/neri-oxman., accessed Nov 30, 2016.

Oxman, Neri. "Design at the Intersection of Technology and Biology." https://www.ted.com/talks/neri_oxman_design_at_the_intersection_of_technology_and_biology., last modified Oct 29, accessed Nov 14, 2015, https://www.youtube.com/watch?v=CVa_IZVzUoc.

NextLimit. "NextLimit<br>." http://www.realflow.com/., accessed Jan 25, 2016.

Painless360. "RC Reviews – Geeetech Rostock G2s Pro 3D Printer (from Banggood.Com) " https://www.youtube.com/watch?v=aMmevEwiDts., last modified Nov 6, accessed Jan 3, 2016.

Piacentino, Giulio. "Weaverbird Grasshopper Plugin<br>." http://www.giuliopiacentino.com/weaverbird/., accessed Jan 24, 2016.

Piker, Daniel. "Kangaroo3d - Interactive Physics/Constraint Solver." http://www.food4rhino.com/app/kangaroo-physics., last modified 18 March, accessed May 10, 2016.

Pina : Film Documentary. By Wim Wenders. Perf. Pina Bausch/Tanztheater Wuppertal. Inter Nationes, 2011. DVD

Prusinkiewicz, Przemyslaw, 1952-. 1996. The Algorithmic Beauty of Plants. New York: Springer-Verlag.

RosenKrantz, Jessica and Louis-Rosenberg, Jesse(Nervous System). “Cellular Lamp.” http://www.thingiverse.com/thing:19104., last modified March 13, accessed Jan 9, 2016.Licensed under Creative Commons Attribution Noncommercial ShareAlike 3.0 https://creativecommons.org/licenses/by-nc-sa/3.0

Seckbach, J. (J. 2011. The Diatom World. Dordrecht: Springer.)

Sennett, Richard, 1943-. 2008. The Craftsman. New Haven: Yale University Press.

Skogsrud, Simen. “Simen Svale Skogsrud.” https://bengler.no/simen., accessed April 16, 2017

Spuybroek, Lars. 2011. Sympathy of Things : Ruskin and the Ecology of Design. Rotterdam]: V2 Publishing : NAi Publishing.

Stoermer, Eugene F., 1934- and Smol, J P (John P). 1999. The Diatoms Applications for the Environmental and Earth Sciences. Cambridge, UK ; New York, NY, USA: Cambridge University Press. (pp 8-18)

Stratasys. 2016.“Bjork Performs in Neri Oxman Designed 3D Printed Mask.” http://blog.stratasys.com/2016/06/30/3d-printed-mask-bjork/., accessed Nov 14, 2016, http://blog.

Thompson, D’Arcy Wentworth. 1951. On Growth and Form. Cambridge: Univ. Press.

Tsiliakos, Marios and Egan, John. “Nudibranch - Digital [Sub]Stance<br>.” https://digitalsubstance.wordpress.com/category/nudibranch/., last modified Feb 19, accessed Jan 20, 2016.

Turing, Alan Mathison, 1912-1954. 1992. Morphogenesis. Amsterdam ; New York : New York, NY, U.S.A.; Amsterdam ; New York: North-Holland ; Distributors for the U.S. and Canada], Elsevier Science Pub. Co; North-Holland.

van der Walt, Peter. “GitHub LaserWeb Repository.” https://github.com/LaserWeb/LaserWeb4., last modified September 25, accessed August 12, 2017.

van der Walt, Peter. “laserWeb 3 Description.” http://www.openbuilds.com/projectresources/laser-web-laser-controller-software.191/)., last modified Feb 15, accessed Jan 10, 2017.

Wakin, Daniel J. 2009. Pina Bausch, a German Iconoclast Who Reshaped Dance, Dies at 68.(Obituary)(Photograph). The New York Times, July 1, 2009, Vol.158(54723), p.A31.

ZHANG. 2012. “Bio-Manufacturing Technology Based on Diatom Micro- and Nanostructure.” Chinese Science Bulletin 57 (30): 3836-3849. doi:10.1007/s11434-012-5410-x. http://lib.cqvip.com/qk/86894X/201230/43339470.html.

Zumthor, Peter. 2006. Thinking Architecture. 2., expanded ed. ed. Basel [u.a.]: Birkhäuser.

Appendix

Processing Code - Agent-Based Coulomb Field Particle Simulation

Agent-based dynamic simulation of a charge system with parameterization of magnitudes, agent count and interaction. Program was influenced by the examples in Generative Design by Hartmut Bohnaker.
Menu Library used - ControlP5(Andraes Sclegel - http://www.sojamo.de/libraries/controlP5/)
Image Export Library - TileSaver(Marius Watz - http://workshop.evolutionzone.com - Used by Bohnaker in Generative Design)

Particle Sim - Main - Processing Code

```
/*
Agent-Based Coulomb Particle Simulator - Wade Brown - V2_1
Modules:Main
                    GUI
                    Particle Class
                    TileSaver
 */
/** Interfaces
 *
 * MOUSE
 * position x/y + right drag  : perspective view
 * mouse wheel               : slow zoom
 * shift + mouse wheel        : fast zoom
 *
 * KEYS
 * c                    : cancel point output
 * f                    : toggle freeze simulation
 * m                    : toggle menu open/close
 * o                    : output meshes to dxf
 * p                    : output to png
 * r                    : ribbon
 * s          : display static charges
 * space                : reset sim as set
 * t                    : trail type toggle
 * z                    : disable z-accel calc
 * -                    : remove static charge
 * +                    : add static charge
 * arrow up             : zoom in
 * arrow down           : zoom out
 */
import controlP5.*;//external menu library
import processing.dxf.*;
import java.util.Calendar;
//------ particles ---
int numParticles = 100;//active number of dynamic agents
int maxNumParticles = 500;//Upper limit of agent counr
int numStaticParticles = 2;//start value of static agents
int maxNumStaticParticles = 10;//max number of static agents
int particleTrailLength = 50;//active length of trail
int maxParticleTrailLength = 500;//max length of trail
ArrayList particles;//array of agents
boolean spaceTrue, zDim = true;
//------ agents ------
int spaceSizeX = 600, spaceSizeY = 400, spaceSizeZ = 200; //based field size
int maxSpaceSizeZ = 500;//upper bound on Z for field bounding box
int staticChargeMultiplier = 10; //scale factor to enhance static charges
float baseCharge = -1.602e-19;//charge of an electron in C
float maxHiveFactor = 1.0, hiveFactor = 0.64;//interaction factor for dynamic charges
boolean cull = false, displayStatic = true;
//------ ControlP5 ------
ControlP5 controlP5;
boolean showGUI = false;//don't show menu at start
Slider[] sliders;//array of sliders used in Menu system
Range[] ranges;
//------ mouse interaction ------
int offsetX = 0, offsetY = 0, clickX = 0, clickY = 0, zoom = -450;
float rotationX = 0, rotationY = 0, targetRotationX = 0, targetRotationY = 0, clickRotationX, clickRo-
tationY;
boolean freeze = false, zoomEnhance = false, trails = false, ribbon = true;
int trailWidth = 5;//based trail width for ribbon mode
boolean plus = false, minus = false;//plus/minus default values for adding/removing static charges
float rotator = 0.01;
 //-------- Output stuffs ----------------
boolean exportDXF = false;//mesh output
boolean saveOneFrame = false;//pdf out
int qualityFactor = 3;//pdf out
TileSaver tiler;
boolean bG = true;
void setup(){
 size(1280,800,P3D);//large 3D window interface
 setupGUI(); //enable base config of P Controls
 frameRate(60);
 colorMode(RGB);
 initParticleSystem();//create and fill agent data structures with charges
 tiler=new TileSaver(this);
}
void draw(){
 //numParticles = 200;
 //particleTrailLength = 140;
 particleSim();
}
void particleSim(){
 // for high quality output
 if(tiler==null) return;
```

```
  tiler.pre();
  if (exportDXF) {
    beginRaw(DXF, "output.dxf");
  }
  if ( spaceTrue ) {//control to re-initialize system upon reset
    initParticleSystem();//create and fill agent data structures with charges
    spaceTrue = false;//reset reset
  }
  hint(ENABLE_DEPTH_TEST);//recommended to enable efficient 3D operation(may be reduntant)
(untested)
  smooth();//possibly redundant in Processing 3.0
  lights();//needed for 3D display
  pushMatrix(); //push current coord system on stack
  //------ set view------
  translate(width/2, height/2, zoom);//offset "0" to middle of screen
  //point(0,0,0);//test point-PLO
  //------- Input Works-------
  if (mousePressed && mouseButton==RIGHT) { //mouse system operation for 3D viewing
    offsetX = mouseX-clickX;
    offsetY = mouseY-clickY;
    targetRotationX = min(max(clickRotationX + offsetY/float(width) * TWO_PI,-HALF_PI), HALF_
PI);
    targetRotationY = clickRotationY + offsetX/float(height) * TWO_PI;
  }
  rotationX += (targetRotationX-rotationX)*0.25;
  rotationY += (targetRotationY-rotationY)*0.25;
  rotateX(-rotationX);
  rotateY(rotationY+rotator);
  //rotator += 0.01;
  stroke(150,150,255,255);//box colour an opacity
  strokeWeight(1); //thin it out
  noFill();
  if (bG) {
    background(255);}
    else {
    background(0);}
    //clear screen each iteration
  box(spaceSizeX*2,spaceSizeY*2,spaceSizeZ*2);//playfield
  pushMatrix();//save current ref frame on stack
  drawParticles();//draw agents
  popMatrix();//return to box world
  popMatrix();//return to flat workd
  noLights();//prepare for menu draw
  if (exportDXF) {
    endRaw();
    exportDXF = false;
  }
  drawGUI();//menu me
  // draw next tile for high quality output
  tiler.post();
}
//------ interactions------
//toggles for key sequences
void keyPressed() {
  if (keyCode == UP) zoom += 20;
  if (keyCode == DOWN) zoom-= 20;
  if (keyCode == SHIFT) zoomEnhance = true;
}
void keyReleased() {
  if (key=='f' || key=='F') freeze = !freeze;//f=freeze
  if (key=='+') {//+=add a stat charge
    plus = true;
    addStaticParticle();
  }
  if (key=='-') {//-=get rid of a stat charge
    minus = true;
    removeStaticParticle();
  }
  if (key=='p' || key=='P') tiler.init(timestamp()+".png",qualityFactor);
  if (key=='c' || key=='c') cull = !cull;//point draw toggle
  if (key=='t' || key=='T') trails = !trails;//t=trails
  if (key=='i' || key=='i') bG = !bG;//toggle background
  if (key=='o' || key=='O') exportDXF = true;
  if (key=='r' || key=='R') ribbon = !ribbon;//r=ribbons
  if (key=='s' || key=='S') displayStatic = !displayStatic; //show static charges
  if (key=='z' || key=='Z') zDim = !zDim;
  if (key=='m' || key=='M') {//m=menu
    showGUI = controlP5.getGroup("menu").isOpen();
    showGUI = !showGUI;
  }
  if (key == ' ') spaceTrue = true;//space=re-randomize
  if (keyCode == SHIFT) zoomEnhance =  false;//shift = superzoom
  if (showGUI) controlP5.getGroup("menu").open();
  else controlP5.getGroup("menu").close();
}
void mousePressed(){//enable rptate on right mouse click
  clickX = mouseX;
  clickY = mouseY;
  clickRotationX = rotationX;
  clickRotationY = rotationY;
}
String timestamp() {
  return String.format("%1$ty%1$tm%1$td_%1$tH%1$tM%1$tS", Calendar.getInstance());
}
void mouseWheel(MouseEvent event) {//mouse wheel zoom
  float count = event.getCount();
```

```
//  float step =-10.0; //old - PLO
  count *=-1;
  if (zoomEnhance)
    {zoom += 50*count;}//fast zoom
  else
    {zoom += 10*count;}//slow zoom
 }
```

Particle Sim - GUI - Processing Code

```
void setupGUI() {//menu setup - position/colour/size/variables
  color activeColor = color(0, 130, 164);
  controlP5 = new ControlP5(this);
  //controlP5.setAutoDraw(false);
  controlP5.setColorActive(activeColor);
  controlP5.setColorBackground(color(170));
  controlP5.setColorForeground(color(50));
  controlP5.setColorCaptionLabel(color(50));
  controlP5.setColorValueLabel(color(255));
  ControlGroup ctrl = controlP5.addGroup("menu", 15, 25, 35);
  ctrl.setColorLabel(color(255));
  ctrl.close();
  sliders = new Slider[10];
  ranges = new Range[10];
  int left = 0;
  int top = 5;
  int len = 200;
  int si = 0;
  int ri = 0;
  int posY = 0;
  //"sliders" - variable name used in quotes - create new by copy and offset/change name in quotes
to var name/adjust values to match max/min/defaults
  sliders[si++] = controlP5.addSlider("numParticles", 1,maxNumParticles, numParticles , left,
top+posY+0, len, 15);
  posY += 30;//slider for particle count
  sliders[si++] = controlP5.addSlider("particleTrailLength", 1, maxParticleTrailLength, particleTrail-
Length,left, top+posY+0, len, 15);
  posY += 30;//slider for trail length
  sliders[si++] = controlP5.addSlider("trailWidth", 0, 20, 5, left, top+posY, len, 15);
  posY += 30;//slider for trail width
  sliders[si++] = controlP5.addSlider("staticChargeMultiplier", 0, 100, 25, left, top+posY, len, 15);
  posY += 30;//slider for charge scaling
  sliders[si++] = controlP5.addSlider("spaceSizeZ", 0, maxSpaceSizeZ, 200, left, top+posY, len, 15);
  posY += 30;//slider for arena resizing
  sliders[si++] = controlP5.addSlider("hiveFactor", 0, maxHiveFactor, 0.64, left, top+posY, len, 15);
  posY += 30;//slider for interaction between dynamic charges
  for (int i = 0; i < si; i++) {
    sliders[i].setGroup(ctrl);
    sliders[i].setId(i);
    sliders[i].getCaptionLabel().toUpperCase(true);
    sliders[i].getCaptionLabel().getStyle().padding(4,3,3,3);
    sliders[i].getCaptionLabel().getStyle().marginTop = -4;
    sliders[i].getCaptionLabel().getStyle().marginLeft = 0;
    sliders[i].getCaptionLabel().getStyle().marginRight = -14;
    sliders[i].getCaptionLabel().setColorBackground(0x99ffffff);
  }
  for (int i = 0; i < ri; i++) {
    ranges[i].setGroup(ctrl);
    ranges[i].setId(i);
    ranges[i].getCaptionLabel().toUpperCase(true);
    ranges[i].getCaptionLabel().getStyle().padding(4,3,3,3);
    ranges[i].getCaptionLabel().getStyle().marginTop =-4;
    ranges[i].getCaptionLabel().setColorBackground(0x99ffffff);
  }
}
void drawGUI() {
  controlP5.show();
  controlP5.draw();
}
// called on every change of the gui
void controlEvent(ControlEvent theControlEvent) {
  //needed to reset trail drawing during operation to reduce abhorrent behaviour during transition
  //(else trail loops created during resize of path)
   if (theControlEvent.isController()) {
    if (theControlEvent.getController().getName().equals("particleTrailLength")) {
    for ( int i = 0; i < numParticles ; i++ ) {//get all particles
       Particle p = (Particle) particles.get(i);
          p.indexStart = p.trailIndex;//reset length of trail points, else very bad(Venkman bad)
       }
      }
  }
 }
```

Particle Sim - Particle Class - Processing Code

```
// All that is particle
void drawParticles() {
  //re-initialze particles after they go into E-space(bad)
   for ( int i = 0; i < numParticles ; i++ ) {
   Particle p = (Particle) particles.get(i);//Bounds check
   if ( (p.pos.x <=-spaceSizeX) || (p.pos.x >= spaceSizeX) ||
        (p.pos.y <=-spaceSizeY) || (p.pos.y >= spaceSizeY) ||
        (p.pos.z <=-spaceSizeZ) || (p.pos.z >= spaceSizeZ)   ){//Particle out of bounds
      p.pos.x = random(-spaceSizeX,spaceSizeX); p.vel.x = 0.0; p.acc.x = 0.0;//fill them with naughts
      p.pos.y = random(-spaceSizeY,spaceSizeY); p.vel.y = 0.0; p.acc.y = 0.0;
      if (zDim) {p.pos.z = random(-spaceSizeZ,spaceSizeZ); p.vel.z = 0.0; p.acc.z = 0.0;}
      p.indexStart = p.trailIndex;//reset length of trail points
   }
   if (!freeze) p.updateParticle();//if "f" state chosen
   p.displayParticle();
  }
 }
 void initParticleSystem() {
   // lets start this baby...
   // Populate arraylist initially with particles(agents)
   //   top-filled with static charges
   //   bottom-filled with dynamic
  particles = new ArrayList();
  PVector p;
  for ( int i = 0; i < numStaticParticles; i++ ) {//setup a static particles
     p = new PVector(random(-spaceSizeX,spaceSizeX),random(-spaceSizeY,spaceSizeY),ran-
dom(-spaceSizeZ,spaceSizeZ));//put them in the park
     boolean pState = true;//it is static
     float charge;
     if (randomBoolean()) {
       charge = staticChargeMultiplier *-baseCharge;} //negative - scaled
     else {
       charge = staticChargeMultiplier * baseCharge;//positive - scaled
     }
     particles.add( new Particle( p , pState, charge) );//add it to top of list
  }
  for ( int i = 0; i < maxNumParticles; i++ ) {//setup a whack of mobile particles
     p = new PVector(random(-spaceSizeX,spaceSizeX),random(-spaceSizeY,spaceSizeY),ran-
dom(-spaceSizeZ,spaceSizeZ));//in the park
     boolean pState = false;//dynamic
     float charge;
     if (randomBoolean()) {
       charge =-baseCharge;} //negative
     else {
       charge = baseCharge; //positive
```

```
    }
    particles.add( new Particle( p , pState, charge) ); //add to list after the statics
  }
 }
  void addStaticParticle(){//adds static when "+" pressed...could make initparticlesystem redundant
    plus = false;//reset flag
    PVector p = new PVector(random(-spaceSizeX,spaceSizeX),random(-spaceSizeY,space-
SizeY),random(-spaceSizeZ,spaceSizeZ));
    boolean pState = true;
    float charge;
    if (randomBoolean()) {
      charge = staticChargeMultiplier *-baseCharge;}
    else {charge = staticChargeMultiplier * baseCharge;}
    particles.add(0, new Particle( p , pState, charge) );
    numStaticParticles++;//update static particle count
  }
  void removeStaticParticle(){//opposite of adding them
    minus = false;//reset flag
    particles.remove(0);//arraylist call to remove at position "0"
    numStaticParticles--;//update static particle count
  }
public boolean randomBoolean(){// random boolean function
    return (Math.random() < 0.5);
}
void list() {//dumps particle list (for debug)
      for (int i = 0; i < numParticles; i++ ) {
        Particle current = (Particle) particles.get(i);
        println( "Particle: ", i);
        println( "Position: ", current.pos.x, current.pos.y, current.pos.z );
        println( "Velocity: ", current.vel.x, current.vel.y, current.vel.z );
        println( "Acceleration: ", current.acc.x, current.acc.y, current.acc.z );
        println( "Charge: ", current.charge);
        println( "State: ", current.s, "\n");
      }
}
// Particle class defining active agents with Field System
class Particle {
  //baggage
  PVector pos; //Position in m
  PVector vel; //Velocity in m/s
  PVector acc; //Acceleration in m/s^2
  float charge; //Charge in Coulombs
  boolean s; //Stationary
  float k = 9.09E9; //permitivity constant
  float m = 9.109E-31; //mass of an electron
  PVector[] trail = new PVector[particleTrailLength];//trail data struct
  int trailIndex, indexStart;//keep track of index,start for trail management
  Particle( PVector _location, boolean _state, float _charge) {
```

```
  //Constructor
   acc = new PVector(0,0,0);//acceleration
   vel = new PVector(0,0,0);//speed
   pos = new PVector(0,0,0);//location
   pos = _location;//pass the info
   s = _state;//pass the info
   charge = _charge;// ditto
   trail = new PVector[maxParticleTrailLength];//define and init trail array
   for (int i = 0; i < trail.length;i++) {
     trail[i] = new PVector(0,0,0);//create and populate trail data structure
   }
 }
 void updateParticle() {//short and sweet - where the real work is done
   calcForces();//update force values
   this.vel.add(acc);//add vectors
   this.pos.add(vel);//add vectors
 }
 void calcForces() {
 // the meat - Field system 1
 // calculate forces amd accumulate
   PVector accTotal = new PVector(0,0,0);
   for (int i = 0; i < particles.size(); i++ ) {
     Particle other = (Particle) particles.get(i);
     if ( this.s == true ) {} // I'm static, no field effect.
     else {//ensure vector direction for accel
       float dir;
       float distX = this.pos.x - other.pos.x;
       float distY = this.pos.y - other.pos.y;
       float distZ = this.pos.z - other.pos.z;
       float r1 = sqrt(sq(distX) + sq(distY));
       float theta = atan( distY / distX );
       float phi = atan( distZ / r1 );
       float r = sqrt( sq(distX) + sq(distY) + sq(distZ) );
       float acc = k * this.charge * other.charge / (m * sq(r)); //Coulomb
       if ( this.acc.mag() > 1 ) { this.acc.mult(1/this.acc.mag());} //overall acc limiter
       if ( !other.s ) {acc = acc * hiveFactor;} //limiter for moving charges
       if ( abs(distX) != distX ) { //fix for Cos(-)=Cos(+)
         dir = -1;}
       else {
         dir = 1;}
       if (abs(distX) >= 10) {accTotal.x += dir * acc * cos(theta);}//danger below 10(Venkman)
       if (abs(distY) >= 10) {accTotal.y += dir * acc * sin(theta);}//values grow too quickly to man-
age
       if (abs(distZ) >= 10 && zDim) {accTotal.z += acc * sin(phi);}//change interaction distance to
mitigate inifnties
      }
      this.acc = accTotal;//add it up!
  }
 }
 void displayParticle() {
   //put it
   //displays points/trails/ribbons
   if (( this.s == true ) && ( displayStatic == true) ) {//Static Charges
     this.charge = staticChargeMultiplier*baseCharge;
     pushMatrix();//push ref frame
     noStroke();//ghost it
     translate(this.pos.x,this.pos.y,this.pos.z);//translate point
     sphereDetail(10,10);//define sphere geom for static charge
     fill(150,150,0,192);//light yellow
     sphere(20);//put spheres
     popMatrix();//pop frame
   }
   else {
     if (!freeze) { //if freeze state, don't update trail
      trail[trailIndex%particleTrailLength].set(pos);//Store trail point
      this.trailIndex++;//increment index for points
      if ((trailIndex - indexStart) > particleTrailLength ) indexStart = trailIndex - particleTrailLength; //
adjust indexes if poss out of bounds
     }
     stroke(map(this.acc.mag(),0,0.2,0,100),map(this.acc.mag(),0,0.2,0,100),0);//Point colour
yellow based on accel
     strokeWeight(5);//fatten it up
     if (cull) { point( this.pos.x , this.pos.y ,this.pos.z);} // Output point
     if(trails) {//if "t" pressed
       strokeWeight(2);//thin trails
       for ( int j = trailIndex ; j >= indexStart ; j-- ){
       if (trailIndex >= indexStart) {
         stroke((map(j,indexStart,trailIndex,0,255)),0,0, 127 );//disappearing trails
         point( trail[j%(particleTrailLength)].x ,trail[j%(particleTrailLength)].y, trail[j%(particleTrail-
Length)].z);//output trail points
       }
       }
     }
     if (ribbon) {//ribbon me
       displayParticleRibbon();}
     noStroke();
   }
 }
 void displayParticleRibbon() {
   //algorithm from Generative Design
   fill(192,0,0,192);//red transparent
   noStroke();
   beginShape(QUAD_STRIP);//ribbon
   for ( int j = trailIndex-1 ;  j > indexStart ; j-- ) {//pick a point in trail
     PVector v1 = PVector.sub(trail[j%particleTrailLength],trail[(j+1)%particleTrailLength]);//create a
vector in plane
```

```
      PVector v2 = PVector.add(trail[(j+1)%particleTrailLength],trail[j%particleTrailLength]);//create a
second in plane
      PVector v3 = v1.cross(v2);  //create a vector perp to plane
      v2.set(v3);//store it
      //v2 = v1.cross(v3);
      //v1.normalize();
      v2.normalize();//resize it to unity
      //v3.normalize();
      //v1.mult(theWidth);
      v2.mult(trailWidth);//widen trail
      //v3.mult(theWidth);
      vertex(trail[j%particleTrailLength].x+v2.x,trail[j%particleTrailLength].y+v2.y,trail[j%particleTrail-
Length].z+v2.z);//make ribbon 1-2
      vertex(trail[j%particleTrailLength].x-v2.x,trail[j%particleTrailLength].y-v2.y,trail[j%particleTrail-
Length].z-v2.z);//make ribbon 2-3
      }
    endShape();//make it!
    }
}
```

Particle Sim - TileSaver - Processing Code

```
// M_3_4_01_TOOL.pde
// GUI.pde, Mesh.pde, TileSaver.pde
//
// Generative Gestaltung, ISBN: 978-3-87439-759-9
// First Edition, Hermann Schmidt, Mainz, 2009
// Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
// Copyright 2009 Hartmut Bohnacker, Benedikt Gross, Julia Laub, Claudius Lazzeroni
//
// http://www.generative-gestaltung.de
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.
// TileSaver.pde - v0.12 2007.0326
// Marius Watz - http://workshop.evolutionzone.com
//
// Class for rendering high-resolution images by splitting them into
// tiles using the viewport.
//
// Builds heavily on solution by "surelyyoujest":
// http://processing.org/discourse/yabb_beta/YaBB.cgi?
// board=OpenGL;action=display;num=1159148942
class TileSaver {
  public boolean isTiling=false,done=true;
  public boolean doSavePreview=false;
  PApplet p;
  float FOV=60; // initial field of view
  float cameraZ, width, height;
  int tileNum=10,tileNumSq; // number of tiles
  int tileImgCnt, tileX, tileY, tilePad;
  boolean firstFrame=false, secondFrame=false;
  String tileFilename,tileFileextension=".png";
  PImage tileImg;
  float perc,percMilestone;
  // The constructor takes a PApplet reference to your sketch.
  public TileSaver(PApplet _p) {
    p=_p;
  }
  // If init() is called without specifying number of tiles, getMaxTiles()
  // will be called to estimate number of tiles according to free memory.
  public void init(String _filename) {
```

```
  init(_filename,getMaxTiles(p.width));
}
// Initialize using a filename to output to and number of tiles to use.
public void init(String _filename,int _num) {
  tileFilename=_filename;
  tileNum=_num;
  tileNumSq=(tileNum*tileNum);
  // Reset tile counters to start over correctly
  tileX = 0;
  tileY = 0;
  width=p.width;
  height=p.height;
  cameraZ=(height/2.0f)/p.tan(p.PI*FOV/360.0f);
  p.println("TileSaver: "+tileNum+" tilesnResolution: "+
    (p.width*tileNum)+"x"+(p.height*tileNum));
  // remove extension from filename
  if(!new java.io.File(tileFilename).isAbsolute())
    tileFilename=p.sketchPath(tileFilename);
  tileFilename=noExt(tileFilename);
  p.createPath(tileFilename);
  // save preview
  if(doSavePreview) p.g.save(tileFilename+"_preview.png");
  // set up off-screen buffer for saving tiled images
  tileImg=new PImage(p.width*tileNum, p.height*tileNum);
  // start tiling
  done=false;
  isTiling=false;
  perc=0;
  percMilestone=0;
  tileInc();
}
// set filetype, default is TGA. pass a valid image extension as parameter.
public void setSaveType(String extension) {
  tileFileextension=extension;
  if(tileFileextension.indexOf(".")==-1) tileFileextension="."+tileFileextension;
}
// pre() handles initialization of each frame.
// It should be called in draw() before any drawing occurs.
public void pre() {
  if(!isTiling) return;
  if(firstFrame) firstFrame=false;
  else if(secondFrame) {
    secondFrame=false;
    // since processing version 1.0.8 (revision 0170) the following line has to be removed,
    //       because updating of the projection works now imediately.
    // tileInc();
  }
  setupCamera();
```

```
}
// post() handles tile update and image saving.
// It should be called at the very end of draw(), after any drawing.
public void post() {
  // If first or second frame, don't update or save.
  if(firstFrame||secondFrame|| (!isTiling)) return;
  // Find image ID from reverse row order
  int imgid=tileImgCnt%tileNum+(tileNum-tileImgCnt/tileNum-1)*tileNum;
  int idx=(imgid+0)%tileNum;
  int idy=(imgid/tileNum);
  // Get current image from sketch and draw it into buffer
  p.loadPixels();
  tileImg.set(idx*p.width, idy*p.height, p.g);
  // Increment tile index
  tileImgCnt++;
  perc=100*((float)tileImgCnt/(float)tileNumSq);
  if(perc-percMilestone>5 || perc>99) {
    p.println(p.nf(perc,3,2)+"% completed. "+tileImgCnt+"/"+tileNumSq+" images saved.");
    percMilestone=perc;
  }
  if(tileImgCnt==tileNumSq) tileFinish();
  else tileInc();
}
public boolean checkStatus() {
  return isTiling;
}
// tileFinish() handles saving of the tiled image
public void tileFinish() {
  isTiling=false;
  restoreCamera();
  // save large image to TGA
  tileFilename+="_"+(p.width*tileNum)+"x"+
    (p.height*tileNum)+tileFileextension;
  p.println("Save: "+
    tileFilename.substring(
  tileFilename.lastIndexOf(java.io.File.separator)+1));
  tileImg.save(tileFilename);
  p.println("Done tiling.n");
  // clear buffer for garbage collection
  tileImg=null;
  done=true;
}
// Increment tile coordinates
public void tileInc() {
  if(!isTiling) {
    isTiling=true;
    firstFrame=true;
    secondFrame=true;
    tileImgCnt=0;
  }
  else {
    if(tileX==tileNum-1) {
      tileX=0;
      tileY=(tileY+1)%tileNum;
    }
    else
      tileX++;
  }
}
// set up camera correctly for the current tile
public void setupCamera() {
  p.camera(width/2.0f, height/2.0f, cameraZ,
  width/2.0f, height/2.0f, 0, 0, 1, 0);
  if(isTiling) {
    float mod=1f/10f;
    p.frustum(width*((float)tileX/(float)tileNum-.5f)*mod,
    width*((tileX+1)/(float)tileNum-.5f)*mod,
    height*((float)tileY/(float)tileNum-.5f)*mod,
    height*((tileY+1)/(float)tileNum-.5f)*mod,
    cameraZ*mod, 10000);
  }
}
// restore camera once tiling is done
public void restoreCamera() {
  float mod=1f/10f;
  p.camera(width/2.0f, height/2.0f, cameraZ,
  width/2.0f, height/2.0f, 0, 0, 1, 0);
  p.frustum(-(width/2)*mod, (width/2)*mod,
 -(height/2)*mod, (height/2)*mod,
  cameraZ*mod, 10000);
}
// checks free memory and gives a suggestion for maximum tile
// resolution. It should work well in most cases, I've been able
// to generate 20k x 20k pixel images with 1.5 GB RAM allocated.
public int getMaxTiles(int width) {
  // get an instance of java.lang.Runtime, force garbage collection
  java.lang.Runtime runtime=java.lang.Runtime.getRuntime();
  runtime.gc();
  // calculate free memory for ARGB (4 byte) data, giving some slack
  // to out of memory crashes.
  int num=(int)(Math.sqrt(
  (float)(runtime.freeMemory()/4)*0.925f))/width;
  p.println(((float)runtime.freeMemory()/(1024*1024))+"/"+
    ((float)runtime.totalMemory()/(1024*1024)));
  // warn if low memory
  if(num==1) {
      p.println("Memory is low. Consider increasing memory settings.");
      num=2;
    }
    return num;
  }
  // strip extension from filename
  String noExt(String name) {
    int last=name.lastIndexOf(".");
    if(last>0)
      return name.substring(0, last);
    return name;
  }
}
```

Flocking Proof of Concept - Processing Code

```
// Flocking proof of concept using Punkiert and based upon Daniel Shiffman's Flocking and Boid
examples
//
/** Interfaces
 *
 * MOUSE
 * position x/y + right drag  : perspective view
 * mouse wheel               : slow zoom
 * shift + mouse wheel        : fast zoom
 *
 * KEYS
 * f                         : toogle freeze simulation
 * arrow up                  : zoom in
 * arrow down                : zoom out
 */
int spaceSizeX = 1600, spaceSizeY = 800, spaceSizeZ = 200  ; //based field size
//int maxSpaceSizeZ = 500;//upper bound on Z for field bounding box
//------ mouse interaction ------
int offsetX = 0, offsetY = 0, clickX = 0, clickY = 0, zoom = -450;
float rotationX = 0, rotationY = 0, targetRotationX = 0, targetRotationY = 0, clickRotationX, clickRo-
tationY;
boolean freeze = false, zoomEnhance = false, trails = true, ribbon = false;
int trailWidth = 5;//based trail width for ribbon mode
boolean plus = false, minus = false;//plus/minus default values for adding/removing static charges
import punktiert.math.Vec;
import punktiert.physics.*;
VPhysics physics;
public void setup() {
  size(1600,1200,P3D);//large 3D window interface
  float fov = PI/3;
  float cameraZ = (height/2.0)/tan(fov/2.0);
  perspective(fov,float(width)/float(height),cameraZ/100,cameraZ*100.0);
  //sphereDetail(5,10);
  //size(800, 600);
  //noStroke();
  fill(0, 255);
  //physics = new VPhysics(width, height);
  Vec boxMin = new Vec(float(-width+10),float(-height+10),float(-spaceSizeZ+10));
  Vec boxMax = new Vec(float(width-10),float(height-10),float(spaceSizeZ-10));
  physics = new VPhysics(boxMin,boxMax,true);//define range
  int amount = 2000;//number of particles
  for (int i = 0; i < amount; i++) {
    Vec pos = new Vec(random(-width,width), random(-height,height), random(-spaceSizeZ,space-
SizeZ));
    float rad = random(5,10);
    // Code to allow the system to have flocking characteristics
```

```
   VBoid p = new VBoid(pos);
   p.swarm.setSeperationScale(rad*.5);//separation region
   p.setRadius(rad);//separation distance(spherical coords)
   p.addBehavior(new BCollision());//they can collide elastically
   p.trail.setInPast(1000);//set trail length
   // Code to "kick" particles into managing a light vortex...shake things up
   //Vec vel = new Vec(0.0,1.0,0.0);
   //p.addVelocity( vel );
   //p.addBehavior(new BVortex(p, 1000, 10000));
   //p.addBehavior(new BVortex(p, 1, 2));//add vortex behaviour to each
   physics.addParticle(p);
  }
  //Create Vortex
  //Vec pos = new Vec(0,0,0);
  //Vec vel = new Vec(0,-10,0);
  //VParticle p = new VParticle(pos);
  //p.addBehavior(new BAttraction(pos,vel,1000,-1000));
  //physics.addParticle(p);
}
void draw(){
   pushMatrix(); //push current coord system on stack
  //------ set view------
  translate(width/2,height/2, zoom);//offset "0" to middle of screen
  //point(0,0,0);//test point-PLO
  //------- Input Works-------
  if (mousePressed && mouseButton==RIGHT) { //mouse system operation for 3D viewing
   offsetX = mouseX-clickX;
   offsetY = mouseY-clickY;
   targetRotationX = min(max(clickRotationX + offsetY/float(width) * TWO_PI,-HALF_PI), HALF_PI);
   targetRotationY = clickRotationY + offsetX/float(height) * TWO_PI;
  }
  rotationX += (targetRotationX-rotationX)*0.25;
  rotationY += (targetRotationY-rotationY)*0.25;
  rotateX(-rotationX);
  rotateY(rotationY);
  //stroke(150,150,255,50);//box colour and opacity
  stroke(128,128,255,25);
  strokeWeight(1); //thin it out
  noFill();
  background(0);//clear screen each iteration
  //box(2*width,2*height,2*spaceSizeZ);//playfield
  pushMatrix();//save current ref frame on stack - go to particle RefWorld
  //drawParticles
  //background(0);
  if (!freeze) physics.update();
  //for (VParticle p : physics.particles) {
  for (int i = 0; i< physics.particles.size(); i++ ) {
   VBoid boid = (VBoid) physics.particles.get(i);
```

```
    //ellipse(p.x, p.y, p.getRadius()*2, p.getRadius()*2);
    //noStroke();
    strokeWeight(1);
    stroke(63);
    fill(0,0,0,63);
    //lights();
    pushMatrix();//Welcome to PointSpace
    translate(boid.x,boid.y,boid.z);
    //sphere( boid.getRadius()*2 );
    box(boid.getRadius()*2);
    popMatrix();
    stroke(200,0,0,75);
    noFill();
    beginShape();
    for (int j=0; j < boid.trail.particles.size(); j+= 10 ) {
      VParticle t = boid.trail.particles.get(j);
      //stroke(200,0,0,75);
      //strokeWeight(j/10);
      //point(t.x,t.y,t.z);
      //strokeWeight(1);
      vertex(t.x,t.y,t.z);
    }
    endShape();
    //popMatrix();//Leaving PointSpace
  }
  popMatrix();//return to box world
  popMatrix();//return to flat workd
  noLights();//prepare for menu draw
  //drawGUI();//menu me
}
 //------ interactions ------
  //toggles for key sequences
  void keyPressed() {
   if (keyCode == UP) zoom += 20;
   if (keyCode == DOWN) zoom -= 20;
   if (keyCode == SHIFT) zoomEnhance = true;
  }
  void keyReleased() {
   if (key=='f' || key=='F') freeze = !freeze;//f=freeze
   //if (key=='+') {//+=add a stat charge
   //  plus = true;
   //  addStaticParticle();
   //}
   //if (key=='-') {//-=get rid of a stat charge
   //  minus = true;
   //  removeStaticParticle();
   //}
   //if (key=='t' || key=='T') trails = !trails;//t=trails
 //if (key=='r' || key=='R') ribbon = !ribbon;//r=ribbons
 //if (key=='m' || key=='M') {//m=menu
 //  showGUI = controlP5.getGroup("menu").isOpen();
 //  showGUI = !showGUI;
 //}
 //if (key == ' ') spaceTrue = true;//space=re-randomize
 if (keyCode == SHIFT) zoomEnhance =  false;//shift = superzoom
 //if (showGUI) controlP5.getGroup("menu").open();
 //else controlP5.getGroup("menu").close();
}
void mousePressed(){//enable rptate on right mouse click
 clickX = mouseX;
 clickY = mouseY;
 clickRotationX = rotationX;
 clickRotationY = rotationY;
}
void mouseWheel(MouseEvent event) {//mouse wheel zoom
 float count = event.getCount();
//  float step =-10.0; //old - PLO
 count *=-1;
 if (zoomEnhance)
   {zoom += 50*count;}//fast zoom
 else
   {zoom += 10*count;}//slow zoom
}
```

Processing Code - Agent-Based Tensor Field Simulation

Generative agent-based form development simulation using an harmonic tensor field. AffectNodes can be added/removed from the simulation through a rudimentary interface; their affect can create an effect on the existing circular/manifold tensor field. Agents can be added/removed dynamically. The tensor field can be displayed or hidden during simulation. Agents that leave the playfield are de-rezed and re-allocated, and placed randomly on the field manifold surface. Trails modes are: none, spline,ribbon. Agents can be displayed or hidden.

Modules:	Main
	SVector Class
	Gui
	FlowField Class
	FieldInteraction Class
	ParticleAgent Class
	AffectNode Class

Flowfield - Module Main

```
// Flowfield        - Tensor Field Agent Simulation
//                  - Module Main
//                  - Written By Wade Brown

import controlP5.*;//external menu library

//------ particles------
int numParticles = 100;//active number of dynamic agents
int maxNumParticles = 500;//Upper limit of agent counr
int numStaticParticles = 1;//start value of static agents
int maxNumStaticParticles = 10;//max number of static agents
int particleTrailLength = 50;//active length of trail
int maxParticleTrailLength = 200;//max length of trail
ArrayList particles;//array of agents
boolean spaceTrue;
//------ agents------
int maxNumAgents = 20000;
int numAgents = 500;
int maxPathLength = 1000;
int pathLength = 50;
boolean agentF = true;
boolean simulationF = true;
int trailWidth = 1;//based trail width for ribbon mode
//------ Workfield Settings------
int spaceSizeX = 600, spaceSizeY = 600, spaceSizeZ = 600; //based field size
//------ ControlP5------
ControlP5 controlP5;
boolean showGUI = false;//don't show menu at start
```

```
Slider[] sliders;//array of sliders used in Menu system
Range[] ranges;
//------ mouse interaction------
int offsetX = 0, offsetY = 0, clickX = 0, clickY = 0, zoom =-450;
float rotationX = 0, rotationY = 0, targetRotationX = 0, targetRotationY = 0, clickRotationX, clickRo-
tationY;
boolean freeze = false, zoomEnhance = false, ribbon = false;
boolean plusF = false, minusF = false;//plus/minus default values for adding/removing static charg-
es
boolean keyPressedF = false;
int translationX = width/2;
int translationY = height/2;
boolean runOnceF = true;
int  trails = 0;
//------ Node interaction------
boolean nodeDisplayF = false;
boolean upArrowF = false;
boolean downArrowF = false;
boolean leftArrowF = false;
boolean rightArrowF = false;
boolean nodeF = false;
ArrayList nodeList = new ArrayList();
int currentNode;
boolean enterF = false;
boolean shiftF = false;
boolean deleteF = false;
boolean randomF = false;
int pathM = 0;
boolean dispFieldF = false;
boolean addKickFa = false;
boolean addKickFb = false;
boolean affectF = false;
boolean displayNodeAffectF = false;
//------ Flow Field------
int resolution = 60  ;
float alphaStep = TWO_PI/resolution, betaStep = TWO_PI/resolution;
FlowField main = new FlowField(resolution);
FlowField support = new FlowField(resolution);
//setup list of autonomous agents
ArrayList agentList = new ArrayList();
void settings() {
  //fullScreen(P3D);
  size(1280,1280,P3D);
}
void setup(){
  //size(1280,1280,P3D);
  //Re-creates the default perspective
  float fov = PI/3.0;
```

```
    float cameraZ = (height/2.0) / tan(fov/2.0);
    perspective(fov, float(width)/float(height),cameraZ/10.0, cameraZ*10.0);
    // Base Screen setup
    background(0);
    frameRate(60);
    colorMode(RGB);
    setupGUI(); //enable base config of ControlP5 Controls
    //------ set view------
    //translate(width/2, height/2, zoom);//offset "0" to middle of screen
    //point(0,0,0);//test point-PLO
    // Setup Field Environment
    support.initFieldSupport();
    main.initFieldManifold(support); //populate the surface field tensor values
    //Drop in a whack of agents into the system
    println("| Initialize Agents");
    for ( int i = 0; i < numAgents; i++ ) {
      float a,b;
      a = random(0,TWO_PI);
      b = random(0,TWO_PI);
      particleAgent agent = new particleAgent(support.innerRadius*2,a,b);
      ////random velocity for new agents
      //float c,d;
      //c = random(0,TWO_PI);
      //d = random(0,TWO_PI);
      //agent.velocity.setSV(c,d,random(-0.01,0.01));//give it a random velocity in spherical coords
      //particleAgent agent = new particleAgent(support.innerRadius*(sin(a)*cos(b)),a,b);//alternate
agent location on manifold type 1
      agentList.add( agent );
    }
    initNodePositionTest(nodeList);//init tensor nodes
}
  void draw() {
    background(0);//clear screen each iteration
    hint(ENABLE_DEPTH_TEST);//recommended to enable efficient 3D operation(may be redunt-
ant)(untested)
    lights();//needed for 3D display
    pushMatrix(); //push current coord system on stack
    //------- Input Works-------
    if (mousePressed && mouseButton==RIGHT) { //mouse system operation for 3D viewing
      offsetX = mouseX-clickX;
      offsetY = mouseY-clickY;
      targetRotationX = min(max(clickRotationX + offsetY/float(width) * TWO_PI,-HALF_PI), HALF_
PI);
      targetRotationY = clickRotationY + offsetX/float(height) * TWO_PI;
    }
    if ( (mousePressed && mouseButton==RIGHT) && ( keyPressedF && keyCode==SHIFT) ) { //
mouse system operation for 3D viewing
      offsetX = mouseX-clickX;
```

```
   offsetY = mouseY-clickY;
   translationX -= ( offsetX*-(zoom*4/3)*.05/(zoom) );
   translationY -= ( offsetY*-(zoom*4/3)*.05/(zoom) );
  }
  if (runOnceF) {
   translationX = width/2;
   translationY = height/2;
   runOnceF = false;
  }
  if ( zoom != 0.0 ) {//check for infinities
   translate( translationX, translationY, zoom );
   //println(translationX, translationY);
  }
  pushMatrix();
  rotationX += (targetRotationX-rotationX)*0.25;
  rotationY += (targetRotationY-rotationY)*0.25;
  rotateX(-rotationX);
  rotateY( rotationY);
  stroke(150,150,255,25);//box colour and opacity
  strokeWeight(1); //thin it out
  noFill();
  //background(0);//clear screen each iteration
  box(spaceSizeX*2,spaceSizeY*2,spaceSizeZ*2);//playfield
  pushMatrix();//save current ref frame on stack
  //update the agent at its location within the flowField
  strokeWeight(3);
  stroke(255,0,0,100);
  if ( simulationF ) {//main sim loop
      for ( int k = 0; k < numAgents; k++ ) {
         particleAgent agent = (particleAgent) agentList.get(k);
         agent.update( main.field[int( agent.location.theta/alphaStep)][int( agent.location.phi/beta-
Step)],
                    support.field[int( agent.location.theta/alphaStep)][int( agent.location.phi/beta-
Step)]);
      if ( agentF ) { agent.display(); }
      }
   }//main sim loop
  //Alter Main field using random walk(averaging)
  if ( randomF ) {randomField();}//keypress "r"
  //Display Tensor Field
  if (dispFieldF) {displayField();}//keypress "f"
  //Display affect nodes
  if ( nodeDisplayF ) {
   strokeWeight(5);
   stroke(0,0,255,200);
   for ( int i = nodeList.size()-1; i >= 0 ; i-- ) {//walk through arraylist of nodes FIFO
    AffectNode node = new AffectNode();
    node = (AffectNode) nodeList.get(i);
```

```
      point(node.location.SVtoPV().x,node.location.SVtoPV().y,node.location.SVtoPV().z);
      //println(i);
      //if (i==0) {
      // println("node 2:");
      // println("Polar:", node.location.r, node.location.theta, node.location.phi);
      // println("Cartesian:", node.location.SVtoPV().x,node.location.SVtoPV().y,node.location.
SVtoPV().z);
      //}
    }
   }
   //enter config mode
   if ( nodeF ) {
    selectNode();//display and allow "+" or "-" moves through existing nodes
    moveNode();//move current node using cursor keys
    changeNodeList();//add/remove nodes
    if ( affectF ) { //change type/size/strength
      alterNodeConfig();
      affectF = !affectF; }
   }
   //Display agent paths
   if ( pathM != 0 ) { // display path
    stroke(0,255,0,50);
    strokeWeight(1);
    for ( int k = 0; k < numAgents; k++ ) {
       particleAgent agent = (particleAgent) agentList.get(k);
       agent.displayPath( pathM );//path mode select
    }
   }
  popMatrix();
  popMatrix();//return to box world
  popMatrix();//return to flat workd
  noLights();//prepare for menu draw
  drawGUI();//menu me(drawn last to appear on top of everything)
}
```

Flowfield - Module SVectorClass

```
// Flowfield        - Tensor Field Agent Simulation
//                  - Module SVectorClass
//                  - Written By Wade Brown
/**
 * A class to describe a two or three dimensional vector using spherical coordinates.
 *
 * Initially based on the pVector class included with basic Processing libraries
 */

public class SVector {
  /** The r component of the vector. */
  public float r;
  /** The theta component of the vector. */
  public float theta;
  /** The phi component of the vector. */
  public float phi;
  /** Array so that this can be temporarily used in an array context */
  //protected ArrayList[] path;
  /**
   * Constructor for an empty vector: r, theta, and phi are set to 0.
   */
  public SVector() {
  }
   ///**
   //* Constructor for a 3D vector in polar coordinates
   //*
   //*/
  //Public SVector(SVector v) {
   // this.r = v.r;
   // this.theta = v.theta;
   // this.phi = v.phi;
   // }
  /**
   * Constructor for a 3D Spherical vector.
   *
   * @param  r the r coordinate.
   * @param  theta the theta coordinate.
   * @param  phi the phi coordinate.
   */
  public SVector(float r, float theta, float phi) {
    this.r = r;
    this.theta = theta;
    this.phi = phi;
  }
  /**
 * Constructor for a 2D vector in polar coordinates: phi coordinate is set to 0.
 *
 * @param  r the r coordinate.
 * @param  theta the theta coordinate.
 */
public SVector(float r, float theta) {
  this.r = r;
  this.theta = theta;
  this.phi = 0.0;
}
/**
 * Set r, theta, and phi coordinates.
 *
 * @param r the r coordinate.
 * @param theta the theta coordinate.
 * @param phi the phi coordinate.
 */
public void setSV(float r, float theta, float phi) {
  this.r = r;
  this.theta = theta;
  this.phi = phi;
}
/**
 * Set r, theta, and phi coordinates from a Spherical Vector3D object.
 *
 * @param v the SVector object to be copied
 */
public void setSV(SVector v) {
  this.r = v.r;
  this.theta = v.theta;
  this.phi = v.phi;
}
/**
 * Set the r, theta (and maybe phi) coordinates using a float[] array as the source.
 * @param source array to copy from
 */
public void setSV(float[] source) {
  if (source.length >= 2) {
    r = source[0];
    theta = source[1];
  }
  if (source.length >= 3) {
    phi = source[2];
  }
}
/**
 * Get a copy of this vector.
 */
```

```
public SVector getSV() {
  return new SVector(r, theta, phi);
}
public float[] getSV(float[] target) {
  if (target == null) {
    return new float[] { r, theta, phi };
  }
  if (target.length >= 2) {
    target[0] = r;
    target[1] = theta;
  }
  if (target.length >= 3) {
    target[2] = phi;
  }
  return target;
}
/**
 * Calculate the magnitude (length) of the vector
 * @return the magnitude of the vector
 */
public float magSV() {
  return (float) r;
}
/**
 * Add a vector to this vector
 * @param v the vector to be added
 */
public void addSV(SVector v) {
  if ( v != null ) {
  float x,y,z,x1,y1,z1;
  x = r * cos(theta) * sin(phi);
  y = r * sin(theta) * sin(phi);
  z = r * cos(phi);
  x1 = v.r * cos(v.theta) * sin(v.phi);
  y1 = v.r * sin(v.theta) * sin(v.phi);
  z1 = v.r * cos(v.phi);
  r = sqrt( sq(x+x1) + sq(y+y1) + sq(z+z1) );
  theta = atan2( (y + y1), (x + x1) );
  phi = acos( (z + z1) / r );
  }
}
public void addSV(float r, float theta, float phi) {
  float x,y,z,x1,y1,z1;
  x = this.r * cos(this.theta) * sin(this.phi);
  y = this.r * sin(this.theta) * sin(this.phi);
  z = this.r * cos(this.phi);
  x1 = r * cos(theta) * sin(phi);
  y1 = r * sin(theta) * sin(phi);
  z1 = r * cos(phi);
  this.r = sqrt( sq(x+x1) + sq(y+y1) + sq(z+z1) );
  this.theta = atan2( (y + y1), (x + x1) );
  this.phi = acos( (z + z1) / this.r );
}
/**
 * Add two vectors
 * @param v1 a vector
 * @param v2 another vector
 * @return a new vector that is the sum of v1 and v2
 */
public SVector addSV(SVector v1, SVector v2) {
  //println("Calling AddSV", 200,200);
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  //println("AddSV v1: ",x1,y1,z1);
  //println("AddSV v2: ",x2,y2,z2);
  SVector target = new SVector( sqrt( sq(x1+x2) + sq(y1+y2) + sq(z1+z2) ),
          atan2( (y1 + y2),(x1 + x2) ),
          acos( (z1 + z2) / sqrt( sq(x1+x2) + sq(y1+y2) + sq(z1+z2) ) ) );
  return target;
}
 /**
 * Add two vectors into a target vector
 * @param v1 a vector
 * @param v2 another vector
 * @param target the target vector (if null, a new vector will be created)
 * @return a new vector that is the sum of v1 and v2
 */
public SVector addSV(SVector v1, SVector v2, SVector target) {
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  if (target == null) {
    target = new SVector( sqrt( sq(x1+x2) + sq(y1+y2) + sq(z1+z2) ),
          atan2( (y1 + y2), (x1 + x2) ),
          acos( (z1 + z2) / sqrt( sq(x1+x2) + sq(y1+y2) + sq(z1+z2)) ) );
  } else {
    target.setSV( sqrt( sq(x1+x2) + sq(y1+y2) + sq(z1+z2) ),
               atan2( (y1 + y2), (x1 + x2) ),
              acos( (z1 + z2) / sqrt(sq(x1+x2) + sq(y1+y2) + sq(z1+z2)) )  );
  }
  return target;
}
/**
 * Subtract a vector from this vector
 * @param v the vector to be subtracted
 */
public void subSV(SVector v) {
  float x,y,z,x1,y1,z1;
  x = this.r * cos(this.theta) * sin(this.phi);
  y = this.r * sin(this.theta) * sin(this.phi);
  z = this.r * cos(this.phi);
  x1 = v.r * cos(v.theta) * sin(v.phi);
  y1 = v.r * sin(v.theta) * sin(v.phi);
  z1 = v.r * cos(v.phi);
  this.r = sqrt( sq(x-x1) + sq(y-y1) + sq(z-z1) );
  this.theta = atan2( (y- y1), (x- x1) );
  this.phi = acos( (z- z1) / this.r );
}
public void subSV(float r, float theta, float phi) {
  float x,y,z,x1,y1,z1;
  x = this.r * cos(this.theta) * sin(this.phi);
  y = this.r * sin(this.theta) * sin(this.phi);
  z = this.r * cos(this.phi);
  x1 = r * cos(theta) * sin(phi);
  y1 = r * sin(theta) * sin(phi);
  z1 = r * cos(phi);
  this.r = sqrt( sq(x-x1) + sq(y-y1) + sq(z-z1) );
  this.theta = atan2( (y- y1), (x- x1) );
  this.phi = acos( (z- z1) / this.r );
}
/**
* Subtract one vector from another
 * @param v1 a vector
 * @param v2 another vector
 * @return a new vector that is v1 - v2
 */
public SVector subSV(SVector v1, SVector v2) {
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  SVector target = new SVector( sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ),
          atan2( (y1 - y2), (x1 - x2) ),
          acos( (z1 - z2) / sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ) ) );
  return target;
}
public SVector subSV(SVector v1, SVector v2, SVector target) {
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  if (target == null) {
    target = new SVector( sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ),
          atan2( (y1 - y2), (x1 - x2) ),
          acos( (z1 - z2) / sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ) ) );
  } else {
    target.setSV( sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ),
              atan2( (y1 - y2), (x1 - x2) ),
              acos( (z1 - z2) / sqrt( sq(x1-x2) + sq(y1-y2) + sq(z1-z2) ) ) );
  }
  return target;
}
/**
 * Multiply this vector by a scalar
 * @param n the value to multiply by
 */
public void multSV(float n) {
  r *= n;
}
/**
 * Multiply a vector by a scalar
 * @param v a vector
 * @param n scalar
 * @return a new vector that is v1 * n
 */
public SVector multSV(SVector v, float n) {
  return multSV(v, n, null);
}
/**
 * Multiply a vector by a scalar, and write the result into a target SVector.
 * @param v a vector
 * @param n scalar
 * @param target SVector to store the result
 * @return the target vector, now set to v1 * n
 */
public SVector multSV(SVector v, float n, SVector target) {
  if (target == null) {
    target = new SVector(v.r*n, v.theta, v.phi);
  } else {
    target.setSV(v.r*n, v.theta, v.phi);
  }
  return target;
}
/**
 * Multiply each element of one vector by the elements of another vector.
 * @param v the vector to multiply by
 * -- Not sure this is of real use -- wb
 */
public void multSV(SVector v) {
  r *= v.r;
  theta *= v.theta;
  phi *= v.phi;
}
/**
 * Multiply each element of one vector by the individual elements of another
 * vector, and return the result as a new SVector.
 */
public SVector multSV(SVector v1, SVector v2) {
  return multSV(v1, v2, null);
}
/**
 * Multiply each element of one vector by the individual elements of another
 * vector, and write the result into a target vector.
 * @param v1 the first vector
 * @param v2 the second vector
 * @param target SVector to store the result
 */
public SVector multSV(SVector v1, SVector v2, SVector target) {
  if (target == null) {
    target = new SVector(v1.r*v2.r, v1.theta*v2.theta, v1.phi*v2.phi);
  } else {
    target.setSV(v1.r*v2.r, v1.theta*v2.theta, v1.phi*v2.phi);
  }
  return target;
}
/**
 * Divide this vector by a scalar
 * @param n the value to divide by
 */
public void divSV(float n) {
  r /= n;
}
/**
 * Divide a vector by a scalar and return the result in a new vector.
 * @param v a vector
 * @param n scalar
 * @return a new vector that is v1 / n
 */
public SVector divSV(SVector v, float n) {
  return divSV(v, n, null);
}
public SVector divSV(SVector v, float n, SVector target) {
  if (target == null) {
    target = new SVector(v.r/n, v.theta, v.phi);
  } else {
    target.setSV(v.r/n, v.theta, v.phi);
  }
  return target;
}
/**
 * Divide each element of one vector by the elements of another vector.
 */
public void divSV(SVector v) {
  r /= v.r;
  theta /= v.theta;
  phi /= v.phi;
}
/**
 * Multiply each element of one vector by the individual elements of another
 * vector, and return the result as a new SVector.
 */
public SVector divSV(SVector v1, SVector v2) {
  return divSV(v1, v2, null);
}
/**
 * Divide each element of one vector by the individual elements of another
 * vector, and write the result into a target vector.
 * @param v1 the first vector
 * @param v2 the second vector
 * @param target SVector to store the result
 */
public SVector divSV(SVector v1, SVector v2, SVector target) {
  if (target == null) {
    target = new SVector(v1.r/v2.r, v1.theta/v2.theta, v1.phi/v2.phi);
  } else {
    target.setSV(v1.r/v2.r, v1.theta/v2.theta, v1.phi/v2.phi);
  }
  return target;
}
/**
 * Calculate the Euclidean distance between two points (considering a point as a vector object)
 * @param v another vector
 * @return the Euclidean distance between
 */
public float distSV(SVector v) {
  float x1,y1,z1,x2,y2,z2;
  x1 = r * cos(theta) * sin(phi);
  y1 = r * sin(theta) * sin(phi);
  z1 = r * cos(phi);
  x2 = v.r * cos(v.theta) * sin(v.phi);
  y2 = v.r * sin(v.theta) * sin(v.phi);
  z2 = v.r * cos(v.phi);
  float dx = x1 - x2;
  float dy = y1 - y2;
  float dz = z1 - z2;
  return (float) sqrt(dx*dx + dy*dy + dz*dz);
}
/**
 * Calculate the Euclidean distance between two points (considering a point as a vector object)
 * @param v1 a vector
 * @param v2 another vector
 * @return the Euclidean distance between v1 and v2
 */
public float distSV(SVector v1, SVector v2) {
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  float dx = x1 - x2;
  float dy = y1 - y2;
  float dz = z1 - z2;
  return (float) sqrt(dx*dx + dy*dy + dz*dz);
}
/**
 * Calculate the dot product with another vector
 * @return the dot product
 */
public float dotSV(SVector v) {
  return r*v.r*(sin(theta)*(sin(v.theta)*cos(phi-v.phi)+cos(theta)*cos(v.theta)));
}
public float dotSV(float r, float theta, float phi) {
  return this.r*r*(sin(this.theta)*(sin(theta)*cos(this.phi-phi)+cos(this.theta)*cos(theta)));
}
public float dotSV(SVector v1, SVector v2) {
    return v1.r*v2.r*(sin(v1.theta)*(sin(v2.theta)*cos(v1.phi-v2.phi)+cos(v1.theta)*cos(v2.theta)));
}
/**
 * Return a vector composed of the cross product between this and SVector v.
 */
public void crossSV(SVector v) {
  float x1,y1,z1,x2,y2,z2;
  x1 = r * cos(theta) * sin(phi);
  y1 = r * sin(theta) * sin(phi);
  z1 = r * cos(phi);
  x2 = v.r * cos(v.theta) * sin(v.phi);
  y2 = v.r * sin(v.theta) * sin(v.phi);
  z2 = v.r * cos(v.phi);
  float cross_r = y1 * z2 - y2 * z1;
  float cross_theta = z1 * x2 - z2 * x1;
  float cross_phi = x1 * y2 - x2 * y1;
  this.r = cross_r;
  this.theta = cross_theta;
  this.phi = cross_phi;
  //return crossSV(v, null);
}
/**
 * Perform cross product between this and another vector, and store the
 * result in 'target'. If target is null, a new vector is created.
 */
public SVector crossSV(SVector v, SVector target) {
  float x1,y1,z1,x2,y2,z2;
  x1 = r * cos(theta) * sin(phi);
  y1 = r * sin(theta) * sin(phi);
  z1 = r * cos(phi);
  x2 = v.r * cos(v.theta) * sin(v.phi);
  y2 = v.r * sin(v.theta) * sin(v.phi);
  z2 = v.r * cos(v.phi);
  float cross_r = y1 * z2 - y2 * z1;
  float cross_theta = z1 * x2 - z2 * x1;
  float cross_phi = x1 * y2 - x2 * y1;
  if (target == null) {
    target = new SVector(cross_r, cross_theta, cross_phi);
  } else {
    target.setSV(cross_r, cross_theta, cross_phi);
  }
  return target;
}
public SVector crossSV(SVector v1, SVector v2, SVector target) {
  float x1,y1,z1,x2,y2,z2;
  x1 = v1.r * cos(v1.theta) * sin(v1.phi);
  y1 = v1.r * sin(v1.theta) * sin(v1.phi);
  z1 = v1.r * cos(v1.phi);
  x2 = v2.r * cos(v2.theta) * sin(v2.phi);
  y2 = v2.r * sin(v2.theta) * sin(v2.phi);
  z2 = v2.r * cos(v2.phi);
  float cross_r = y1 * z2 - y2 * z1;
  float cross_theta = z1 * x2 - z2 * x1;
  float cross_phi = x1 * y2 - x2 * y1;
  if (target == null) {
    target = new SVector(cross_r, cross_theta, cross_phi);
  } else {
    target.setSV(cross_r, cross_theta, cross_phi);
  }
  return target;
}
/**
 * Normalize the vector to length 1 (make it a unit vector)
 */
public void normalizeSV() {
  float m = magSV();
  if (m != 0 && m != 1) {
    divSV(m);
  }
}
/**
 * Normalize this vector, storing the result in another vector.
 * @param target Set to null to create a new vector
 * @return a new vector (if target was null), or target
 */
public SVector normalizeSV(SVector target) {
  if (target == null) {
    target = new SVector();
  }
  float m = magSV();
  if (m > 0) {
    target.setSV(r/m, theta, phi);
  } else {
    target.setSV(r, theta, phi);
  }
  return target;
}
/**
 * Limit the magnitude of this vector
 * @param max the maximum length to limit this vector
 */
public void limitSV(float max) {
  if ( magSV() > max) {
    normalizeSV();
    multSV(max);
  }
}
/**
 * Calculate the angle of rotation for this vector (only 2D vectors)
 * @return the angle of rotation
```

```
  */
 public float heading2DSV() {
   return theta;
 }
 /**
  * Calculate the angle between two vectors, using the dot product
  * @param v1 a vector
  * @param v2 another vector
  * @return the angle between the vectors
  */
 public float angleBetweenSV(SVector v1, SVector v2) {
   float x1,y1,z1,x2,y2,z2;
   x1 = v1.r * cos(v1.theta) * sin(v1.phi);
   y1 = v1.r * sin(v1.theta) * sin(v1.phi);
   z1 = v1.r * cos(v1.phi);
   x2 = v2.r * cos(v2.theta) * sin(v2.phi);
   y2 = v2.r * sin(v2.theta) * sin(v2.phi);
   z2 = v2.r * cos(v2.phi);
   double dot = x1 * x2 + y1 * y2 + z1 * z2;
   double v1mag = sqrt(x1 * x1 + y1 * y1 + z1 * z1);
   double v2mag = sqrt(x2 * x2 + y2 * y2 + z2 * z2);
   return (float) acos((float)(dot / (v1mag * v2mag)));
 }
/**
  * Convert vector in spherical coordinates to cartesian
  * SVector-> PVector
  */
 public PVector SVtoPV() {
   float x,y,z;
   x = this.r * cos(this.theta) * sin(this.phi);
   y = this.r * sin(this.theta) * sin(this.phi);
   z = this.r * cos(this.phi);
   PVector result = new PVector(x,y,z);
   return result;
 }
/*
* Plot SVector
*/
 public void plotSV() {
   PVector vOut;
   vOut = this.SVtoPV();
   point(vOut.x, vOut.y, vOut.z);
 }
}//class definition
```

Flowfield - Module Gui

```
/*
* Flowfield            - Tensor Field Agent Simulation
*                      - Module Gui
*                      - Written By Wade Brown
*
*         Gui scaffold code using the ControlP5 Library. Menu items added/removed to
* suit the needs of this investigation.
*
* Interfaces
*
* MOUSE
* position x/y + right drag  : perspective view
* mouse wheel                : slow zoom
* shift + mouse wheel        : fast zoom
*
* KEYS
*         m              : toggle menu open/close
*         p              : path display
*         r              : ribbon
*         arrow up       : zoom in
*         arrow down     : zoom out
*         =              : Enable affect node
*         d/D            : display Affect Nodes
*         a/A            : agent display
*         s/S            : freeze/run simulation
*         f/F            : tensor field display
*         r/R            : random walk to alter tensor field
*         .              : toggle affect node change mode
*                        arrows   : move activated affect node(phi/theta)
*                        -        : remove affect node
*                        +        : add affect node
*/


void setupGUI() {//menu setup - position/colour/size/variables
  color activeColor = color(0, 130, 164);
  controlP5 = new ControlP5(this);
  //controlP5.setAutoDraw(false);
  controlP5.setColorActive(activeColor);
  controlP5.setColorBackground(color(170));
  controlP5.setColorForeground(color(50));
  controlP5.setColorCaptionLabel(color(50));
  controlP5.setColorValueLabel(color(255));
  ControlGroup ctrl = controlP5.addGroup("menu", 15, 25, 35);
  ctrl.setColorLabel(color(255));
  ctrl.close();
  sliders = new Slider[10];
  ranges = new Range[10];
  int left = 0;
  int top = 5;
  int len = 200;
  int si = 0;
  int ri = 0;
  int posY = 0;
  //"sliders" - variable name used in quotes - create new by copy and offset/change name in quotes
to var name/adjust values to match max/min/defaults
  sliders[si++] = controlP5.addSlider("numAgents", 1,maxNumAgents, numAgents , left, top+po-
sY+0, len, 15);
  posY += 30;//slider for particle count
  sliders[si++] = controlP5.addSlider("pathLength", 1, maxPathLength, pathLength,left, top+posY+0,
len, 15);
  posY += 30;//slider for trail length
  sliders[si++] = controlP5.addSlider("trailWidth", 0, 20, 1, left, top+posY, len, 15);
  posY += 30;//slider for trail width
  sliders[si++] = controlP5.addSlider("spaceSizeZ", 0, spaceSizeZ, spaceSizeZ, left, top+posY, len,
15);
  posY += 30;//slider for arena resizing
  for (int i = 0; i < si; i++) {
    sliders[i].setGroup(ctrl);
    sliders[i].setId(i);
    sliders[i].getCaptionLabel().toUpperCase(true);
    sliders[i].getCaptionLabel().getStyle().padding(4,3,3,3);
    sliders[i].getCaptionLabel().getStyle().marginTop = -4;
    sliders[i].getCaptionLabel().getStyle().marginLeft = 0;
    sliders[i].getCaptionLabel().getStyle().marginRight = -14;
    sliders[i].getCaptionLabel().setColorBackground(0x99ffffff);
  }
  for (int i = 0; i < ri; i++) {
    ranges[i].setGroup(ctrl);
    ranges[i].setId(i);
    ranges[i].getCaptionLabel().toUpperCase(true);
    ranges[i].getCaptionLabel().getStyle().padding(4,3,3,3);
    ranges[i].getCaptionLabel().getStyle().marginTop = -4;
    ranges[i].getCaptionLabel().setColorBackground(0x99ffffff);
  }
}
void drawGUI() {
  controlP5.show();
  controlP5.draw();
}
// called on every change of the gui
void controlEvent(ControlEvent theControlEvent) {
  //needed to reset trail drawing during operation to reduce abhorrent behaviour during transition
  //(else trail loops created during resize of path)
```

```
   if (theControlEvent.isController()) {
     if (theControlEvent.getController().getName().equals("numAgents")) {
       updateAgentCount();
     }
      //if (theControlEvent.getController().getName().equals("particleTrailLength")) {
      //for ( int i = 0; i < numParticles ; i++ ) {//get all particles
      //   Particle p = (Particle) particles.get(i);
      //   p.indexStart = p.trailIndex;//reset length of trail points, else very bad(Venkman bad)
      //   }
      //  }
  }
}
/** Interfaces
 *
 * MOUSE
 * position x/y + right drag  : perspective view
 * mouse wheel                : slow zoom
 * shift + mouse wheel        : fast zoom
 *
 * KEYS
 * m                          : toogle menu open/close
 * f                          : toggle freeze simulation
 * space                      : re-randomize with current settings
 * t                          : trail
 * r                          : ribbon
 *-                           : remove static charge
 * +                          : add static charge
 * arrow up                   : zoom in
 * arrow down                 : zoom out
 */
//------ interactions------
//toggles for key sequences
void keyPressed() {
  keyPressedF = true;
  if (keyCode == UP) {upArrowF = true;} else { upArrowF = false; }
  if (keyCode == DOWN) {downArrowF = true;} else { downArrowF = false; }
  if (keyCode == RIGHT) {rightArrowF = true;} else { rightArrowF = false; }
  if (keyCode == LEFT) {leftArrowF = true;} else { leftArrowF = false; }
  //if (keyCode == SHIFT) {zoomEnhance = true; shiftF = true;} else {shiftF = false;}
  if (key == SHIFT) {zoomEnhance = true; shiftF = true;} else {shiftF = false;}
  if (keyCode == ENTER) {enterF = true;} else { enterF = false; }
  if (keyCode == DELETE) {deleteF = true;}
}
void keyReleased() {
  keyPressedF = false;
  if (key=='=') affectF = !affectF;
  if (key=='d' || key=='D') displayNodeAffectF = !displayNodeAffectF;
  if (key=='n' || key=='N') nodeDisplayF = !nodeDisplayF;//n=node display
  if (key=='p' || key=='P') {pathM += 1; pathM = pathM%4;}//p=path display/mode
  if (key=='a' || key=='A') agentF = !agentF;//agent display
  if (key=='f' || key=='F') dispFieldF = !dispFieldF;//f=field
  if (key=='s' || key=='S') simulationF = !simulationF;//s=simulation
  if (key=='.') nodeF = !nodeF;//.=toggle Affect Node mode
  if (key=='+') {plusF = true;} else {plusF = false;}
  if (key=='-') {minusF = true;} else {minusF = false;}
  if (key=='r' || key=='R') randomF = !randomF;//r=randomF walk to alter tensor
  if (key=='m' || key=='M') {//m=menu
    showGUI = controlP5.getGroup("menu").isOpen();
    showGUI = !showGUI;
  }
  if (key == DELETE) {deleteF = true;}
  if (key == ' ') spaceTrue = true;//space=re-randomize
  if (showGUI) controlP5.getGroup("menu").open();
  else controlP5.getGroup("menu").close();
}
void mousePressed(){//enable rptate on right mouse click
  clickX = mouseX;
  clickY = mouseY;
  clickRotationX = rotationX;
  clickRotationY = rotationY;
}
void mouseWheel(MouseEvent event) {//mouse wheel zoom
  float count = event.getCount();
//  float step =-10.0; //old- PLO
  count *=-1;
  if (zoomEnhance)
    {zoom += 50*count;}//fast zoom
  else
    {zoom += 25*count;}//slow zoom
}
```

Flowfield - Module FlowField Class

```
// Flowfield        - Tensor Field Agent Simulation
//                  - Module Flowfield Class
//                  - Written By Wade Brown

//  Class defines tensor fields and manifold shape for initialization


public class FlowField {// A flow field is a two-dimensional array of PVectors.
     SVector[][] field;
     int resolution;
     float alphaStep,betaStep;
     float innerRadius,outerRadius;
    public FlowField(int _r) {//Constructor
      resolution = _r;//Choose a precision
      alphaStep = TWO_PI/_r;//Divide XY space up into equal regions
      betaStep = TWO_PI/_r;//Divide YZ space up into equal regions
      field = new SVector[int(_r)+1][int(_r)+1];//Define the space metric
      innerRadius = 200;//Vector harmonic function offset variable
      outerRadius = 1;//Flow function magnitude
     }
    void initField() {//Create and initialize flow data structure
      println("| Init Field");
      int x = 0;
      for ( float i = 0; i < TWO_PI ; i+= alphaStep ) {
        int y = 0;
          for ( float j = 0; j < TWO_PI ; j+= betaStep ) {
            SVector a = new SVector(outerRadius,i,j);
            SVector b = new SVector(outerRadius,i+betaStep,j);
            field[x][y] = new SVector() ;
            field[x][y].setSV( field[x][y].subSV(a,b) );
            y++;
          }
          x++;
      }
    }
    void initField2() {//Create and initialize flow data structure
      println("| Init Field 2");
      float rad = outerRadius;
      float t = 0.0;
      float p = 0.0;
      float factor = 1;
      int x = 0;
      for ( float i = 0; i < TWO_PI ; i+= alphaStep ) {
        int y = 0;
          for ( float j = 0; j < TWO_PI ; j+= betaStep ) {
            rad = rad + random(-1,1);
            t = t + random( -alphaStep*factor, alphaStep*factor );
            p = p + random( -betaStep*factor, betaStep*factor );
            field[x][y] = new SVector(rad,t,p);
            y++;
          }
          x++;
      }
    }
     void initField3() {//Create and initialize flow data structure - RANDOM
      println("| Init Field 3");
      int x = 0;
      for ( float i = 0; i < TWO_PI ; i+= alphaStep ) {
        int y = 0;
          for ( float j = 0; j < TWO_PI ; j+= betaStep ) {
            //field[x][y] = new PVector( cos(i)*sin(j), sin(i)*sin(j), cos(j) ).setMag(radius);
            //SVector a = new SVector(outerRadius,i,j);
            //SVector b = new SVector(outerRadius,i+betaStep,j);
            field[x][y] = new SVector() ;
            SVector randVector = new SVector(-outerRadius + 2 * random(outerRadius), random(TWO_
PI), random(TWO_PI));
            field[x][y].setSV( randVector ) ;
            y++;
          }
          x++;
      }
     }
     void initFieldManifold( FlowField manifold ) {//Create and initialize flow data structure - Vectors
tangent to mainfold surface and aligned
      println("| Init field on manfold");
      int x = 0;
      for ( float i = 0; i < TWO_PI - alphaStep ; i+= alphaStep ) {
        int y = 0;
          for ( float j = 0; j < TWO_PI - betaStep ; j+= betaStep ) {
            SVector a = new SVector(manifold.field[x][y].r,manifold.field[x][y].theta, manifold.field[x][y].
phi);
            SVector b = new SVector(manifold.field[(x+1)%resolution][y].r,manifold.field[(x+1)%resolu-
tion][y].theta, manifold.field[(x+1)%resolution][y].phi);
            a.subSV(b);
            a.normalizeSV();
            a.limitSV(.0005);//limit initial field tensor
            //a.limitSV(0.001);//put upper bound on initial tensor field
            //a.multSV(random(0.0,1.0));
            field[x][y] = new SVector();
            field[x][y].setSV(a);
            y++;
          }
          x++;
      }
     }
     void initFieldSupport() {//Create and initialize flow surface
      println("| Init field support");
      int x = 0;
      for ( float i = 0; i < TWO_PI - alphaStep ; i+= alphaStep ) {
        int y = 0;
          for ( float j = 0; j < TWO_PI - betaStep ; j+= betaStep ) {
            SVector a = new SVector( innerRadius*2, i, j);//surface function
            //SVector a = new SVector( (innerRadius*(sin(i)+cos(j))), i, j);//surface function
            field[x][y] = new SVector() ;
            field[x][y].setSV( a );
            y++;
          }
          x++;
      }
     }
public SVector lookup(float _alpha, float _beta) {// Access Flow-field data
        float alpha = _alpha%TWO_PI;
        float beta = _beta%TWO_PI;
        return(field[int(resolution/TWO_PI*alpha)][int(resolution/TWO_PI*beta)]);
      }
   }
```

Flowfield - Module FieldInteraction

```
// Flowfield         - Tensor Field Agent Simulation
//                   - Module FieldInteraction
//                   - Written By Wade Brown

//  Module used to bring agents and tensor fields together for interaction. Would normally use a
Class if it were not for the need to bring two classes together. This system seems to work without
breaking the object structure of Java.


void displayField() {// Dump flow data graphically
      //println("| Display Field");
      stroke(255,60,60,50);
      line(0,0,0,50,0,0);
      line(0,0,0,0,50,0);
      line(0,0,0,0,0,50);
      int x = 0;
      for ( float i = 0; i <= TWO_PI-alphaStep ; i+= alphaStep ) {
        int y = 0;
        for ( float j = 0; j <= PI-betaStep ; j+= betaStep ) {
          stroke(255,255,255,60);
          strokeWeight(1);
          SVector a = new SVector(main.field[x][y].r,main.field[x][y].theta, main.field[x][y].phi);
          SVector b = new                SVector(support.field[x][y].r,support.field[x][y].theta,support.
field[x][y].phi);
          SVector c = new SVector();
          c.setSV(a);
          c.normalizeSV();
          c.multSV(5);
          line(b.SVtoPV().x,b.SVtoPV().y,b.SVtoPV().z,c.SVtoPV().x + b.SVtoPV().x, c.SVtoPV().y +
b.SVtoPV().y, c.SVtoPV().z  + b.SVtoPV().z );
          stroke(255,255,255,50);
          strokeWeight(2);
          point( b.SVtoPV().x,b.SVtoPV().y,b.SVtoPV().z );
          y++;
        }
        x++;
      }
     }
     void updateAgentCount() {
     // match agentList length to number of agents
       if (agentList.size() < numAgents) {//lengthen the list
          for ( int i = agentList.size(); i < numAgents; i++ ) {
             float a,b;
             a = random(0,TWO_PI);
             b = random(0,TWO_PI);
             particleAgent agent = new particleAgent(support.innerRadius*2,a,b);
          agentList.add( agent );
        }
    }
    else if (agentList.size() > numAgents) {//shorten the list
         for ( int i = agentList.size()-1; i > numAgents-1; i-- ) {
           agentList.remove(i);
         }
    }
  }
  void initNodePositionTest( ArrayList nodes ){
       SVector a = new SVector(1,0,0);//define affect field value
       a.addSV(support.field[int (PI/6/alphaStep) ][int(PI/betaStep)]);//bring to manifold surface
       AffectNode n = new AffectNode(a,5,10,false,1);//generate affect node
       nodes.add(n);//add node to modelist
       //SVector b = new SVector(1,0,0);
       //b.addSV(support.field[int (PI/alphaStep) ][int (PI/betaStep) ]);
       //AffectNode m = new AffectNode(b,30,5,true);
       //nodes.add(m);
       SVector c = new SVector(1,0,0);
       c.addSV(support.field[0][int (PI/4/betaStep) ]);
       AffectNode o = new AffectNode(c,4,1,true,1);
       nodes.add(o);
       SVector d = new SVector(1,0,0);
       d.addSV(support.field[int ((3/4*PI)/alphaStep) ][int (PI/betaStep) ]);
       AffectNode p = new AffectNode(d,6,3,false,1);
       nodes.add(p);
  }
  void pathDisplay() {//Method - Display Agent Path
       strokeWeight(1);
       noStroke();
       fill(0,255,0,50);
       for ( int k = 0; k < numAgents; k++ ) {
         particleAgent agent = (particleAgent) agentList.get(k);
         agent.displayPath(pathM);
       }
   }
void nodeDisplay( boolean current ) {
   strokeWeight(5);
   if (current) {
     stroke(255,255,50,100);
   } else {
     strokeWeight(5);
     stroke(0,0,255,100);
   }
   for ( int i = nodeList.size()-1; i >= 0 ; i-- ) {//walk through arraylist of nodes FIFO
     AffectNode node = new AffectNode();
     node = (AffectNode) nodeList.get(i);
     point(node.location.SVtoPV().x,node.location.SVtoPV().y,node.location.SVtoPV().z);
   }
}
void selectNode() { //find and select current node
   //println("selectNode");
   //int currentNode = 0;
   //displaynodes
   for ( int i = nodeList.size()-1; i >= 0 ; i-- ) {
     AffectNode node = new AffectNode();
     node = (AffectNode) nodeList.get(i);
     if ( node.current ) {
       stroke(255,255,50,255); }//yellow centre point
     else { //else blue
       stroke(0,0,255,255);//blue centre point
     }
     point( node.location.SVtoPV().x, node.location.SVtoPV().y, node.location.SVtoPV().z );
     if ( node.current == true ) {
         currentNode = i;
         if ( plusF ) { // make next node up current
           plusF = !plusF;
           node.current = false; //current node is marked not current
           currentNode ++;
           if (i==nodeList.size()-1) { currentNode = 0;}
           AffectNode nodeUp = new AffectNode();
           nodeUp = (AffectNode) nodeList.get(currentNode); //get next positive node member
           nodeUp.current = true;
           nodeList.set( currentNode, nodeUp );
           //println("plus");
         } else if ( minusF ) {
           minusF = !minusF;
           node.current = false; //current node is marked not current
           currentNode --;
           if (i==0) { currentNode = nodeList.size()-1;}
           AffectNode nodeDown = new AffectNode();
           nodeDown = (AffectNode) nodeList.get(currentNode); //get next positive node member
           nodeDown.current = true;
           nodeList.set(currentNode, nodeDown );
           //println("minus");
           //addnode
         } else {
           //println("none");
         }
         //println("CurrentNode",currentNode);
     }
   }
}
void moveNode() { //movr current selected node using cursor keys while in config mode
     //println("moveNode");
     AffectNode node = new AffectNode();
```

```
        node = (AffectNode) nodeList.get(currentNode);
        //if ( (node.current == boolean(currentNode) ) && (node.current == true) ) { //doublecheck
that it is the current node that we are operating on
          if (  (node.current == true) ) {
            if ( upArrowF ) { // move node up
              upArrowF = !upArrowF;
              node.location.phi += betaStep;
              if ( (shiftF==true) && (upArrowF==true) ) { node.location.phi += 5 * betaStep; shiftF =
!shiftF; upArrowF = !upArrowF; }
              if (node.location.phi >= TWO_PI ) { node.location.phi = node.location.phi%TWO_PI; }
            }
            if ( downArrowF ) { // move node down
              downArrowF = !downArrowF;
              node.location.phi -= betaStep;
              if ( shiftF ) { node.location.phi -= 5 * betaStep; shiftF = !shiftF; upArrowF = !upArrowF; }
              if (node.location.phi < 0 ) { node.location.phi = TWO_PI - abs(node.location.phi); }
            }
            if ( rightArrowF ) { // move node right
              rightArrowF = !rightArrowF;
              node.location.theta += alphaStep;
              if ( shiftF ) { node.location.theta += 5 * alphaStep; shiftF = !shiftF; upArrowF = !upArrowF; }
              if (node.location.theta >= TWO_PI ) { node.location.theta = node.location.phi%TWO_PI; }
            }
            if ( leftArrowF ) { // move node left
              leftArrowF = !leftArrowF;
              node.location.theta -= alphaStep;
              if ( shiftF ) { node.location.theta -= 5 * alphaStep; shiftF = !shiftF; upArrowF = !upArrowF; }
              if (node.location.theta < 0 ) { node.location.theta = TWO_PI - abs(node.location.theta); }
            }
        }
        nodeList.set(currentNode, node );
        //println("--|", currentNode, boolean(currentNode), node.location.r,node.location.theta, node.
location.phi, node.current);
  }
  void changeNodeList() {
    if ( enterF == true ) {
        //println("EnterF");
        //println("addNode");
        SVector a = new SVector(1,0,0);//define affect field value
        a.addSV(support.field[0][0]);//bring to manifold surface
        AffectNode n = new AffectNode(a,5,10,false,1);//generate affect node
        nodeList.add(n);//add node to modelist
        enterF = false;
    }
    if ( deleteF == true ) {
      //println("DeleteF");
      //println("removeNode");
      if (nodeList.size() > 1) {
        nodeList.remove(currentNode);
        currentNode = nodeList.size()-1;
        AffectNode node = new AffectNode();
        node = (AffectNode) nodeList.get(currentNode);
        node.current = true;
        nodeList.set(currentNode, node );
        deleteF = false;
      }
    }
  }
  //void alterNodeConfig() {//change type/size/strength "Affect"
  //   AffectNode node = new AffectNode();
  //   node = (AffectNode) nodeList.get(currentNode);
  //   //map 1
  //   float posTheta,posPhi;
  //   for ( int h = 1; h <= node.size ; h++ ) {
  //     for ( float i = -PI; i >= PI ; i += alphaStep ) {
  //           posTheta = map(i,-PI,PI,-alphaStep*node.size+thetapos, AlphaStep*node.size+thetapos);
  //           posPhi = map( j,-PI ,PI , -betaStep*node.size+phipos, betaStep*node.size+phipos);
  //           pos = size * cos(i) + sin(j);
  //     }
  //   }
  //}
  void randomField() {
        int posAlpha = int(random( resolution )), posBeta = int(random( resolution ));
        int posAlphaPrev,posBetaPrev;
        int dir = 0;
        SVector a = new SVector(), b = new SVector(), c = new SVector(), d= new SVector();
        for ( int i = 0; i < 1000; i ++) {
         dir = int(random(7));
         posAlphaPrev = posAlpha; posBetaPrev = posBeta;
         if ( dir == 0 ) {
           posAlpha += 1;if (posAlpha >= resolution) posAlpha = posAlpha%resolution;}//mod the
result
         else if ( dir == 1 ) {
           posAlpha += 1;if (posAlpha >= resolution) posAlpha = posAlpha%resolution;//mod the
result
           posBeta += 1;if (posBeta >= resolution) posBeta = posBeta%resolution;}//mod the result
         else if ( dir == 2 ) {
           posBeta += 1;if (posBeta >= resolution) posBeta = posBeta%resolution;}//mod the result
         else if ( dir == 3 ) {
           posAlpha -= 1;if (posAlpha < 0) posAlpha = resolution-1;//mod the result
           posBeta += 1;if (posBeta >= resolution) posBeta = posBeta%resolution;}//mod the result
         else if ( dir == 4 ) {
           posAlpha -= 1;if (posAlpha < 0) posAlpha = resolution-1;}//mod the result
         else if ( dir == 5 ) {
           posAlpha -= 1;if (posAlpha < 0) posAlpha = resolution-1;//mod the result
           posBeta -= 1; if (posBeta < 0 ) posBeta = resolution-1;}//mod the result
        else if ( dir == 6 ) {
          posBeta -= 1; if (posBeta < 0 ) posBeta = resolution-1;}//mod the result
        else if ( dir == 7 ) {
          posAlpha += 1;if (posAlpha > resolution) posAlpha = posAlpha%resolution;//mod the result
          posBeta -= 1; if (posBeta < 0 ) posBeta = resolution-1;}//mod the result
         //add the SVector based on path to the tensor field
         a.setSV(support.field[posAlpha][posBeta].r,support.field[posAlpha][posBeta].theta, support.
field[posAlpha][posBeta].phi);//get original support tensor
         b.setSV(support.field[posAlphaPrev][posBetaPrev].r,support.field[posAlphaPrev][posBetaP-
rev].theta,support.field[posAlphaPrev][posBetaPrev].phi); //next pos suport tensor
         //c.setSV(main.field[posAlpha][posBeta].r,main.field[posAlpha][posBeta].theta, main.field[pos-
Alpha][posBeta].phi); //get field tensor
         //d.subSV(b,a);//determine direction SVector
         b.subSV(a);
         d.setSV(b);
         //println(i,a.magSV(),b.magSV(),d.magSV());
         //println(i,posAlpha,posBeta,posAlphaPrev,posBetaPrev);
         ////d.normalizeSV();//cap at mag(1)
         d.multSV(0.01);//limit
         //c.addSV(d);//add the SVector to existing
         //main.field[posAlpha][posBeta].setSV(c); //replace tensor field with altered version
        //case 0 - upup
        if ( (posAlpha+1) >= resolution ) {//posAlpha out of bounds
          main.field[(posAlpha+1)%resolution][posBeta].addSV(d);
          //main.field[(posAlpha+1)%resolution][posBeta].normalizeSV();
        }
        else {main.field[posAlpha+1][posBeta].addSV(d);
             //main.field[posAlpha+1][posBeta].normalizeSV();
        }
        //case 1 - upright
        if ( ( (posAlpha+1)  >= resolution  ) || ( (posBeta+1) >= resolution  ) ) {//either out of bounds
          main.field[(posAlpha+1)%resolution][(posBeta+1)%resolution].addSV(d);
          //main.field[(posAlpha+1)%resolution][(posBeta+1)%resolution].normalizeSV();
        }
        else {main.field[posAlpha+1][posBeta+1].addSV(d);
             //main.field[posAlpha+1][posBeta+1].normalizeSV();
        } //all good
        //case 2 - right
        if ( (posBeta+1) >= resolution ) {//posBeta out of bounds
          main.field[posAlpha][(posBeta+1)%resolution].addSV(d);
          //main.field[posAlpha][(posBeta+1)%resolution].normalizeSV();
        }
        else { main.field[posAlpha][posBeta+1].addSV(d);
              //main.field[posAlpha][posBeta+1].normalizeSV();
        }//all good
        //case 3 - downright
        if ( ((posAlpha-1) <= 0) && ((posBeta+1) >= resolution) ) {//both out of bounds
          main.field[resolution-1][(posBeta+1)%resolution].addSV(d);
```

```
  //main.field[resolution-1][(posBeta+1)%resolution].normalizeSV();
}
else if ( (posAlpha-1) <= 0 ) {//posAlpha out of bounds
  main.field[resolution-1][posBeta+1].addSV(d);
  //main.field[resolution-1][posBeta+1].normalizeSV();
}
else if ( (posBeta+1) >= resolution  ){//posBeta out of bounds
  main.field[posAlpha-1][(posBeta+1)%resolution].addSV(d);
  //main.field[posAlpha-1][(posBeta+1)%resolution].normalizeSV();
}
else { main.field[posAlpha-1][posBeta+1].addSV(d);
      //main.field[posAlpha-1][posBeta+1].normalizeSV();
}//all good
//case 4 - downdown
if ( posAlpha <= 0 ) { //alpha out of bounds
  main.field[resolution-1][posBeta].addSV(d);
  //main.field[posAlpha+1][posBeta].normalizeSV();
}
else { main.field[posAlpha-1][posBeta].addSV(d);
      //main.field[posAlpha-1][posBeta].normalizeSV();
}//all good
//case 5 - downleft
if ( ((posAlpha-1) <= 0) && ((posBeta-1) <= 0 )) { //both out of bounds
  main.field[resolution-1][resolution-1].addSV(d);
  //main.field[resolution-1][resolution-1].normalizeSV();
}
else if ( (posAlpha-1) <=0 ) { //posAlpha out of bounds
  main.field[resolution-1][posBeta-1].addSV(d);
  //main.field[resolution-1][posBeta-1].normalizeSV();
}
else if ( (posBeta-1) <= 0 ) {//posBeta out of bounds
  main.field[posAlpha-1][resolution-1].addSV(d);
  //main.field[posAlpha-1][resolution-1].normalizeSV();
}
else { main.field[posAlpha-1][posBeta-1].addSV(d);
      //main.field[posAlpha-1][posBeta-1].normalizeSV();
}//all good
//case 6 - leftleft
if ( (posBeta-1) <= 0 ) { //posBeta out of bounds
  main.field[posAlpha][resolution-1].addSV(d);
  //main.field[posAlpha][resolution-1].normalizeSV();
}
else { main.field[posAlpha][posBeta-1].addSV(d);
      //main.field[posAlpha][posBeta-1].normalizeSV();
}//all good
//case 7 - leftup
if (( (posAlpha+1) >= resolution ) && ((posBeta-1) <=0 ) ) {//both out of bounds
  main.field[posAlpha%resolution][resolution-1].addSV(d);
```

```
          //main.field[posAlpha%resolution][resolution-1].normalizeSV();
        }
        else if ((posAlpha+1) >= resolution ) {//posAlpha out of bounds
          main.field[posAlpha%resolution][posBeta-1].addSV(d);
          //main.field[posAlpha%resolution][posBeta-1].normalizeSV();
        }
        else if ((posBeta <= 0)) {//posBeta out of bounds
          main.field[posAlpha+1][resolution-1].addSV(d);
          //main.field[posAlpha+1][resolution-1].normalizeSV();
        }
        else {main.field[posAlpha+1][posBeta-1].addSV(d);
             //main.field[posAlpha+1][posBeta-1].normalizeSV();
        }//all good
       }
   }
void alterNodeConfig() {//change type/size/strength
     ArrayList<PVector> currentAffectField;
     currentAffectField = new ArrayList();
     AffectNode node = (AffectNode) nodeList.get(currentNode);
     int posAlpha,posBeta;
     if ( node.current == true ) {
       for ( int i = 2; i <= node.size; i++ ){//ring up based on .size
         for ( float j = 0 ; j <= TWO_PI; j += (TWO_PI/10) ){
           //Oscillate to create rings of affect
           posAlpha = int( (i * cos(j))  + node.location.theta/alphaStep );
           if ( posAlpha >= resolution ) { //check boundaries
             posAlpha=posAlpha%resolution;}
           else if ( posAlpha < 0 ) {
             posAlpha = resolution - abs(posAlpha);}
           posBeta = int( (i * sin(j)) + node.location.phi/betaStep );
           if ( posBeta >= resolution ) { //check boundaries
             posBeta=posBeta%resolution;}
           else if ( posBeta < 0 ) {
             posBeta = resolution - abs(posBeta); }
           SVector a = new SVector( node.location.r, node.location.theta, node.location.phi );
           SVector b = new SVector();
                   //PVector p1 = new PVector();
                   //PVector p2 = new PVector();
                   //PVector p3 = new PVector();
                   //p1.set(a.SVtoPV());
                   //p2.set(b.SVtoPV());
                   //p3.set(p1.sub(p2));
                   //float r, theta, phi;
                   //r = sqrt( sq(p3.x) + sq(p3.y) + sq(p3.z) );
                   //theta = atan2(p3.y , p3.x);
                   //println("z,y:",p3.z,p3.y, acos( p3.z / p3.y ));
                   //phi = acos( p3.z / p3.y );
                   //SVector result = new SVector(r,theta,phi);
```

```
                    //result.normalizeSV();
                    //result.multSV(10000);
                    //println(result.r,result.theta,result.phi);
              b.setSV(support.field[posAlpha][posBeta].r,support.field[posAlpha][posBeta].theta,support.
field[posAlpha][posBeta].phi);//affect circle point
              a.subSV(b);
              //a.crossSV(b);
              SVector c = new SVector();
              c.setSV(a);
              c.limitSV(.5);
              //c.multSV(10);
              //point( support.field[posAlpha][posBeta].SVtoPV().x,support.field[posAlpha][posBeta].
SVtoPV().y,support.field[posAlpha][posBeta].SVtoPV().z);
            PVector fieldPoint = new PVector(support.field[posAlpha][posBeta].SVtoPV().x,support.
field[posAlpha][posBeta].SVtoPV().y,support.field[posAlpha][posBeta].SVtoPV().z);
            currentAffectField.add(fieldPoint);
            //main.field[posAlpha][posBeta].addSV(c);//create effect
            main.field[posAlpha][posBeta].addSV(c);//create effect
          }//for
        }//for
        if ( displayNodeAffectF ) {
          for (int i=0;i<currentAffectField.size();i++) { //persistent display of node field
            PVector fieldPoint = new PVector();
            fieldPoint = currentAffectField.get(i);
            stroke(255,255,255,163);
            point(fieldPoint.x,fieldPoint.y,fieldPoint.z);
          }
        }
        else {
           currentAffectField.clear();//else flush it
        }//if
      }//if
} //alterNodeConfig
void alterNodeConfig2() {//change type/size/strength
      //println("alterNodeConfig");
      ArrayList<PVector> currentAffectField;
      currentAffectField = new ArrayList();
      AffectNode node = (AffectNode) nodeList.get(currentNode);
      int posAlpha,posBeta;
      if ( node.current == true ) { //doublecheck we are only acting on the "current" node
        for ( int i = 1; i <= node.size; i++ ){//ring up based on .size
          for ( float j = 0 ; j <= TWO_PI; j += (TWO_PI/60) ){
            //Oscillate to create rings of affect
            posAlpha = int( (i * cos(j))  + node.location.theta );
            if ( posAlpha >= resolution ) { //check boundaries
              posAlpha=posAlpha%resolution;}
            else if ( posAlpha < 0 ) {
              posAlpha = resolution - abs(posAlpha);}
            posBeta= int( (i * sin(j)) + node.location.phi );
            if ( posBeta >= resolution ) { //check boundaries
              posBeta=posBeta%resolution;}
            else if ( posBeta < 0 ) {
              posBeta = resolution - abs(posBeta); }
            SVector a = new SVector(node.location.r, node.location.theta,node.location.phi);
            SVector b = new SVector(support.field[posAlpha][posBeta].r,support.field[posAlpha][posBe-
ta].theta,support.field[posAlpha][posBeta].phi);
            SVector c = new SVector();
            //point( support.field[posAlpha][posBeta].SVtoPV().x,support.field[posAlpha][posBeta].
SVtoPV().y,support.field[posAlpha][posBeta].SVtoPV().z);
            PVector fieldPoint = new PVector(support.field[posAlpha][posBeta].SVtoPV().x,support.
field[posAlpha][posBeta].SVtoPV().y,support.field[posAlpha][posBeta].SVtoPV().z);
            currentAffectField.add(fieldPoint);
            //main.field[posAlpha][posBeta].addSV(c);//create effect
            main.field[posAlpha][posBeta].setSV(c);//create effect
          }//for
        }//for
        if ( displayNodeAffectF ) {
          for (int i=0;i<currentAffectField.size();i++) { //persistent display of node field
            PVector fieldPoint = new PVector();
            fieldPoint = currentAffectField.get(i);
            stroke(255,255,255,63);
            point(fieldPoint.x,fieldPoint.y,fieldPoint.z);
          }
        }
        else {
          currentAffectField.clear();//else flush it
        }//if
      }//if
} //alterNodeConfig2
```

Flowfield - Module particleAgent Class

```
// Flowfield        - Tensor Field Agent Simulation
//                  - Module particleAgent Class
//                  - Written By Wade Brown

// Class needed to define agents within the system. Euler integration used to managed particle
dynamics.


public class particleAgent {
  SVector location;
  SVector velocity;
  SVector acceleration;
  ArrayList<SVector> path;
  float maxforce;
  float maxspeed;
  public particleAgent(float _r,float _alpha, float _beta) {
    acceleration = new SVector();
    velocity = new SVector();
    location = new SVector(_r,_alpha,_beta);
    path = new ArrayList();
    //Arbitrary values for maxspeed and force.
    maxspeed = 10;
    //maxforce = 10;
  }
   //Our standard “Euler integration” motion model
  void update(SVector accel, SVector surface) {
    //add old position to trail Arraylist "path"
    SVector oldLocation = new SVector();
    oldLocation.setSV(location);
    path.add(0,oldLocation);//add to beginning of list
    if ( path.size() > pathLength ) {//manage pathlength
      for ( int i = path.size()-1; i > pathLength; i-- ) {
        path.remove(i);
      }
    }
    //Core of the Euler integration..."the roll-up"
    velocity.addSV(accel) ;
    velocity.limitSV(maxspeed);//cap angular speed
    location.addSV(velocity);
    // MOD location to keep within 0<-->TWO_PI so location array bounds are not exceeded
    location.theta = location.theta%TWO_PI;//mod theta to keep position within 0-2PI
    if ( location.theta < 0 ) {location.theta = TWO_PI - abs(location.theta);}
    location.phi = location.phi%TWO_PI;
    if (location.phi < 0 ) {location.phi = TWO_PI - abs(location.phi);}
    //clear acceleration in preparation for recalculation at next step
    acceleration.multSV(0);
    // Is the agent in-bounds? If yes, kill and respawn in a random location on the manifold
    if ( (abs(location.SVtoPV().x) > spaceSizeX) ||
         (abs(location.SVtoPV().y) > spaceSizeY) ||
         (abs(location.SVtoPV().z) > spaceSizeZ)   ) {
       location.setSV(300,random(0,TWO_PI),random(0,TWO_PI));//relocate
       velocity.multSV(0);//wipe its velocity
       path.clear();//delete its history
    }
  }
  //Newton's second law; we could divide by mass if we wanted.
  void applyForce(SVector force) {
    acceleration.addSV(force);
  }
  //Our seek steering force algorithm
  //if looking to steer to a point
  void seek(SVector target) {
    SVector desired = new SVector();
    desired.subSV(target,location);
    desired.normalizeSV();
    desired.multSV(maxspeed);
    SVector steer = new SVector();
    steer.subSV(desired,velocity);
    steer.limitSV(maxforce);
    applyForce(steer);
  }
  void display() {
  //Displays a point represented by the current vector location
    pushMatrix();
    point(location.SVtoPV().x, location.SVtoPV().y, location.SVtoPV().z );
    popMatrix();
  }
  void displayPath(int displayMode) {
    if ( displayMode == 1 ) { //line
      if ( path.size() > 2 ) { // path not too short
        beginShape();
        for ( int i = 1; i < path.size()-1 ; i++ ) {
          vertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).SVtoPV().z );
        }
      }
      endShape();
    } else if ( displayMode == 2 ) { //spline
    if ( path.size() > 2 ) { // path not too short
      beginShape();
        for ( int i = 1; i < path.size()-1 ; i++ ) {
          if ( i == 1 ) {curveVertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).SVtoPV().z
);
                      curveVertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).SVtoPV().z );}
          curveVertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).SVtoPV().z );
          if ( i == path.size()-2 ) {curveVertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.
get(i).SVtoPV().z );
                                    curveVertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).
SVtoPV().z );}
         }
       }
     endShape();
     } else if ( displayMode == 3 ) { //quad strip
     noStroke();
     fill(0,255,0,50);
     if ( path.size() > 2 ) { // path not too short
       beginShape(QUAD_STRIP);
         for ( int i = 1; i < path.size()-1 ; i++ ) {
           vertex(path.get(i).SVtoPV().x, path.get(i).SVtoPV().y, path.get(i).SVtoPV().z );
           vertex(path.get(i).SVtoPV().x+trailWidth, path.get(i).SVtoPV().y + trailWidth, path.get(i).
SVtoPV().z );
         }
       }
     endShape();
     noFill();
     }
   }
}
```

Flowfield - Module AffectNodeClass

```
// Flowfield          - Tensor Field Agent Simulation
//                    - Module AffectNode Class
//                    - Written By Wade Brown

// Needed to define AffectNode data structure


public class AffectNode {//affect node class
  SVector location;
  float size;//~amplitude
  float strength;//~magnitude
  ArrayList<SVector> affect;//resultant affect SVector(ideally perp to manifold surf normal)
  ArrayList<SVector> affectLocation;//surface SVector location
  boolean current;//Live flag
  int type;//Type=1(replace),Type=2(Additive),Type=3(Subtractive),Type=4(Multiplicative)
  public AffectNode() {//constructor
  }
  public AffectNode(SVector _location, float _size, float _strength, boolean _current, int _type) {//
constructor
    location = new SVector();
    location.setSV(_location);
    size = _size;
    strength = _strength;
    affect = new ArrayList();
    affectLocation = new ArrayList();
    current = _current;
    type = _type;
  }
  //Future
  void displayNode() {
  }
  void NodePosition(){
    float aPhi=0.0;
    float aTheta=0.0;
    if (upArrowF) { aPhi +=alphaStep;}
    if (downArrowF) { aPhi -=alphaStep;}
    if (leftArrowF) { aTheta -=betaStep;}
    if (rightArrowF) { aTheta +=betaStep;}
  }
}
```

Smoothieboard Config File - Laser Controller Ver 2

```
#Smoothie/Pi Config file - 20160910 - wb
#
#
#
# Smoothieboard configuration file, see http://smoothieware.org/configuring-smoothie
# NOTE Lines must not exceed 132 characters, and '#' characters mean what follows is ignored
## Robot module configurations : general handling of movement G-codes and slicing into moves

# Basic motion configuration
default_feed_rate                    30480          # Default speed (mm/minute) for G1/G2/G3 -
moves(20in/s)
default_seek_rate                    45720          # Default speed (mm/minute) for G0
moves(30in/s)
mm_per_arc_segment                     0.0           # Fixed length for line segments that divide arcs,
0 to disable
#mm_per_line_segment                   5             # Cut lines into segments this size
mm_max_arc_error                      0.01          # The maximum error for line segments that
divide arcs 0 to disable
                                      # note it is invalid for both the above be 0
                                      # if both are used, will use largest segment length based on
radius

# Arm solution configuration : Cartesian robot. Translates mm positions into stepper positions
# See http://smoothieware.org/stepper-motors
alpha_steps_per_mm                      80           # Steps per mm for alpha ( X ) stepper
beta_steps_per_mm                       80           # Steps per mm for beta ( Y ) stepper
gamma_steps_per_mm                        1600          # Steps per mm for gamma ( Z ) stepper

# Planner module configuration : Look-ahead and acceleration configuration
# See http://smoothieware.org/motion-control
acceleration                         3000          # Acceleration in mm/second/second.
#z_acceleration                        500           # Acceleration for Z only moves in mm/s^2, 0 uses
acceleration which is the default. DO NOT SET ON A DELTA
junction_deviation                   0.05           # See http://smoothieware.org/motion-control#-
junction-deviation
#z_junction_deviation                   0.0           # For Z only moves, -1 uses junction_deviation, zero
disables junction_deviation on z moves DO NOT SET ON A DELTA

# Cartesian axis speed limits
x_axis_max_speed                      46000          # Maximum speed in mm/min
y_axis_max_speed                      46000          # Maximum speed in mm/min
z_axis_max_speed                      300           # Maximum speed in mm/min

# Stepper module configuration
# Pins are defined as  ports, and pin numbers, appending "!" to the number will invert a pin
#Â See http://smoothieware.org/pin-configuration and http://smoothieware.org/pinout
alpha_step_pin                         2.0            # Pin for alpha stepper step signal
alpha_dir_pin                          0.5            # Pin for alpha stepper direction, add '!' to reverse
direction
alpha_en_pin                           0.4            # Pin for alpha enable pin
alpha_current                          1.5            # X stepper motor current
alpha_max_rate                         30480.0        # Maximum rate in mm/min

beta_step_pin                          2.1            # Pin for beta stepper step signal
beta_dir_pin                           0.11           # Pin for beta stepper direction, add '!' to reverse
direction
beta_en_pin                            0.10           # Pin for beta enable
beta_current                           1.5            # Y stepper motor current
beta_max_rate                          30480.0        # Maxmimum rate in mm/min

gamma_step_pin                         2.2            # Pin for gamma stepper step signal
gamma_dir_pin                          0.20           # Pin for gamma stepper direction, add '!' to reverse
direction
gamma_en_pin                           0.19           # Pin for gamma enable
gamma_current                          1.5            # Z stepper motor current
gamma_max_rate                         300.0          # Maximum rate in mm/min

## Extruder module configuration
# See http://smoothieware.org/extruder
extruder.hotend.enable                 true           # Whether to activate the extruder module at all. All
configuration is ignored if false
extruder.hotend.steps_per_mm           140            # Steps per mm for extruder stepper
extruder.hotend.default_feed_rate      600            # Default rate ( mm/minute ) for moves
where only the extruder moves
extruder.hotend.acceleration           500            # Acceleration for the stepper motor mm/secÂ²
extruder.hotend.max_speed              50             # Maximum speed in mm/s

extruder.hotend.step_pin               2.3            # Pin for extruder step signal
extruder.hotend.dir_pin                0.22           # Pin for extruder dir signal ( add '!' to reverse
direction )
extruder.hotend.en_pin                 0.21           # Pin for extruder enable signal

# Extruder offset
#extruder.hotend.x_offset              0              # X offset from origin in mm
#extruder.hotend.y_offset              0              # Y offset from origin in mm
#extruder.hotend.z_offset              0              # Z offset from origin in mm

# Firmware retract settings when using G10/G11, these are the defaults if not defined, must be
defined for each extruder if not using the defaults
#extruder.hotend.retract_length        3              # Retract length in mm
#extruder.hotend.retract_feedrate      45             # Retract feedrate in mm/sec
#extruder.hotend.retract_recover_length    0          # Additional length for recover
```

```
#extruder.hotend.retract_recover_feedrate        8            # Recover feedrate in mm/sec (should be
less than retract feedrate)
#extruder.hotend.retract_zlift_length            0            # Z-lift on retract in mm, 0 disables
#extruder.hotend.retract_zlift_feedrate          6000         # Z-lift feedrate in mm/min (Note mm/min
NOT mm/sec)

delta_current                                    1.5          # First extruder stepper motor current

# Second extruder module configuration
#extruder.hotend2.enable                         true         # Whether to activate the extruder module at all.
All configuration is ignored if false
#extruder.hotend2.steps_per_mm                   140          # Steps per mm for extruder stepper
#extruder.hotend2.default_feed_rate              600          # Default rate ( mm/minute ) for moves
where only the extruder moves
#extruder.hotend2.acceleration                   500          # Acceleration for the stepper motor, as of 0.6,
arbitrary ratio
#extruder.hotend2.max_speed                      50           # mm/s

#extruder.hotend2.step_pin                       2.8          # Pin for extruder step signal
#extruder.hotend2.dir_pin                        2.13         # Pin for extruder dir signal ( add '!' to reverse
direction )
#extruder.hotend2.en_pin                         4.29         # Pin for extruder enable signal

#extruder.hotend2.x_offset                       0            # x offset from origin in mm
#extruder.hotend2.y_offset                       25.0         # y offset from origin in mm
#extruder.hotend2.z_offset                       0            # z offset from origin in mm

#epsilon_current                                 1.5          # Second extruder stepper motor current


## Laser module configuration
# See http://smoothieware.org/laser
laser_module_enable                              false        # Whether to activate the laser module at all
laser_module_pwm_pin                             2.5          # This pin will be PWMed to control the laser.
                                                 # Only pins 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 1.18, 1.20, 1.21, 1.23,
1.24, 1.26, 3.25 and 3.26
                                                 # can be used since laser requires hardware PWM, see
http://smoothieware.org/pinout
#laser_module_ttl_pin                            1.30         # This pin turns on when the laser turns on,
and off when the laser turns off.
#laser_module_maximum_power                      1.0          # This is the maximum duty cycle that will
be applied to the laser
#laser_module_minimum_power                      0.0          # This is a value just below the minimum
duty cycle that keeps the laser
                                                 # active without actually burning.
#laser_module_default_power                      0.8          # This is the default laser power that will be
used for cuts if a power has not been specified.  The value is a scale between
                                                 # the maximum and minimum power levels specified above
```

```
#laser_module_pwm_period                    20            # This sets the pwm frequency as the period
in microseconds

## Temperature control configuration
# See http://smoothieware.org/temperaturecontrol

# First hotend configuration
temperature_control.hotend.enable           true            # Whether to activate this ( "hotend" )
module at all.
temperature_control.hotend.thermistor_pin    0.23            # Pin for the thermistor to read
temperature_control.hotend.heater_pin        2.7             # Pin that controls the heater, set to nc if a
readonly thermistor is being defined
temperature_control.hotend.thermistor        EPCOS100K       # See http://smoothieware.org/
temperaturecontrol#toc5
#temperature_control.hotend.beta            4066            # Or set the beta value
temperature_control.hotend.set_m_code        104             # M-code to set the temperature for this
module
temperature_control.hotend.set_and_wait_m_code 109           # M-code to set-and-wait for this
module
temperature_control.hotend.designator        T               # Designator letter for this module
#temperature_control.hotend.max_temp         300             # Set maximum temperature - Will
prevent heating above 300 by default
#temperature_control.hotend.min_temp         0               # Set minimum temperature - Will prevent
heating below if set

# Safety control is enabled by default and can be overidden here, the values show the defaults
# See http://smoothieware.org/temperaturecontrol#runaway
#temperature_control.hotend.runaway_heating_timeout      900  # How long it can take to heat up,
max is 2040 seconds.
#temperature_control.hotend.runaway_cooling_timeout        0  # How long it can take to cool down
if temp is set lower, max is 2040 seconds
#temperature_control.hotend.runaway_range               20   # How far from the set temperature it
can wander, max setting is 63Â°C

# PID configuration
# See http://smoothieware.org/temperaturecontrol#pid
#temperature_control.hotend.p_factor         13.7            # P ( proportional ) factor
#temperature_control.hotend.i_factor         0.097           # I ( integral ) factor
#temperature_control.hotend.d_factor         24              # D ( derivative ) factor

#temperature_control.hotend.max_pwm          64              # Max pwm, 64 is a good value if driving
a 12v resistor with 24v.

# Second hotend configuration
#temperature_control.hotend2.enable           true          # Whether to activate this ( "hotend" )
module at all.
#temperature_control.hotend2.thermistor_pin    0.25          # Pin for the thermistor to read
#temperature_control.hotend2.heater_pin        1.23          # Pin that controls the heater
```

```
#temperature_control.hotend2.thermistor        EPCOS100K      # See http://smoothieware.org/
temperaturecontrol#thermistor
##temperature_control.hotend2.beta             4066           # or set the beta value
#temperature_control.hotend2.set_m_code        104            # M-code to set the temperature for
this module
#temperature_control.hotend2.set_and_wait_m_code 109          # M-code to set-and-wait for this
module
#temperature_control.hotend2.designator        T1             # Designator letter for this module

#temperature_control.hotend2.p_factor          13.7           # P ( proportional ) factor
#temperature_control.hotend2.i_factor          0.097          # I ( integral ) factor
#temperature_control.hotend2.d_factor          24             # D ( derivative ) factor

#temperature_control.hotend2.max_pwm           64             # Max pwm, 64 is a good value if driving
a 12v resistor with 24v.

temperature_control.bed.enable                 true           # Whether to activate this ( "hotend" ) module
at all.
temperature_control.bed.thermistor_pin         0.24           # Pin for the thermistor to read
temperature_control.bed.heater_pin             2.5            # Pin that controls the heater
temperature_control.bed.thermistor             Honeywell100K  # See http://smoothieware.org/
temperaturecontrol#thermistor
#temperature_control.bed.beta                  3974           # Or set the beta value
temperature_control.bed.set_m_code             140            # M-code to set the temperature for this
module
temperature_control.bed.set_and_wait_m_code  190              # M-code to set-and-wait for this
module
temperature_control.bed.designator             B              # Designator letter for this module

# Bang-bang ( simplified ) control
# See http://smoothieware.org/temperaturecontrol#bang-bang
#temperature_control.bed.bang_bang             false          # Set to true to use bang bang control
rather than PID
#temperature_control.bed.hysteresis            2.0            # Set to the temperature in degrees C to use
as hysteresis

## Switch modules
# See http://smoothieware.org/switch

# Switch module for fan control
switch.fan.enable                              true           # Enable this module
switch.fan.input_on_command                    M106           # Command that will turn this switch on
switch.fan.input_off_command                   M107           # Command that will turn this switch off
switch.fan.output_pin                          2.6            # Pin this module controls
switch.fan.output_type                         pwm            # PWM output settable with S parameter in the
input_on_comand
#switch.fan.max_pwm                            255            # Set max pwm for the pin default is 255
```

```
#switch.misc.enable                        true             # Enable this module
#switch.misc.input_on_command              M42              # Command that will turn this switch on
#switch.misc.input_off_command             M43              # Command that will turn this switch off
#switch.misc.output_pin                    2.4              # Pin this module controls
#switch.misc.output_type                   digital          # Digital means this is just an on or off pin

## Temperatureswitch
# See http://smoothieware.org/temperatureswitch
# Automatically toggle a switch at a specified temperature. Different ones of these may be defined
to monitor different temperatures and switch different swithxes
# Useful to turn on a fan or water pump to cool the hotend
#temperatureswitch.hotend.enable              true           #
#temperatureswitch.hotend.designator          T              # first character of the temperature control
designator to use as the temperature sensor to monitor
#temperatureswitch.hotend.switch              misc           # select which switch to use, matches the
name of the defined switch
#temperatureswitch.hotend.threshold_temp      60.0           # temperature to turn on (if rising) or off
the switch
#temperatureswitch.hotend.heatup_poll         15             # poll heatup at 15 sec intervals
#temperatureswitch.hotend.cooldown_poll       60             # poll cooldown at 60 sec intervals

## Endstops
# See http://smoothieware.org/endstops
endstops_enable                              true            # The endstop module is enabled by default and can
be disabled here
#corexy_homing                               false           # Set to true if homing on a hbot or corexy
alpha_min_endstop                            1.24^           # Pin to read min endstop, add a ! to invert if
endstop is NO connected to ground
#alpha_max_endstop                           1.25^           # Pin to read max endstop, uncomment this and
comment the above if using max endstops
alpha_homing_direction                       home_to_min     # Or set to home_to_max and set alpha_
max and uncomment the alpha_max_endstop
alpha_min                                    0               # This gets loaded as the current position after homing
when home_to_min is set
alpha_max                                    200             # This gets loaded as the current position after
homing when home_to_max is set
beta_min_endstop                             1.26^           # Pin to read min endstop, add a ! to invert if
endstop is NO connected to ground
#beta_max_endstop                            1.27^           # Pin to read max endstop, uncomment this and
comment the above if using max endstops
beta_homing_direction                        home_to_min     # Or set to home_to_max and set alpha_
max and uncomment the alpha_max_endstop
beta_min                                     0               # This gets loaded as the current position after homing
when home_to_min is set
beta_max                                     200             # This gets loaded as the current position after homing
when home_to_max is set
gamma_min_endstop                            1.28^           # Pin to read min endstop, add a ! to invert if
endstop is NO connected to ground
```

```
#gamma_max_endstop                         1.29^           # Pin to read max endstop, uncomment this
and comment the above if using max endstops
gamma_homing_direction                     home_to_min      # Or set to home_to_max and set
alpha_max and uncomment the alpha_max_endstop
gamma_min                              0               # This gets loaded as the current position after
homing when home_to_min is set
gamma_max                              200             # This gets loaded as the current position after
homing when home_to_max is set

alpha_max_travel                       500             # Max travel in mm for alpha/X axis when homing
beta_max_travel                        500             # Max travel in mm for beta/Y axis when homing
gamma_max_travel                          500             # Max travel in mm for gamma/Z axis when
homing

# Optional enable limit switches, actions will stop if any enabled limit switch is triggered
#alpha_limit_enable                    false           # Set to true to enable X min and max limit switches
#beta_limit_enable                     false           # Set to true to enable Y min and max limit switches
#gamma_limit_enable                       false           # Set to true to enable Z min and max limit
switches

# Endstops home at their fast feedrate first, then once the endstop is found they home again at their
slow feedrate for accuracy
alpha_fast_homing_rate_mm_s                50              # Alpha/X fast homing feedrate in mm/
second
alpha_slow_homing_rate_mm_s                25              # Alpha/X slow homing feedrate in mm/
second
beta_fast_homing_rate_mm_s                 50              # Beta/Y  fast homing feedrate in mm/
second
beta_slow_homing_rate_mm_s                 25              # Beta/Y  slow homing feedrate in mm/
second
gamma_fast_homing_rate_mm_s                 4              # Gamma/Z fast homing feedrate in mm/
second
gamma_slow_homing_rate_mm_s                 2              # Gamma/Z slow homing feedrate in mm/
second

alpha_homing_retract_mm                   5              # Distance to retract from the endstop after it is
hit for alpha/X
beta_homing_retract_mm                    5              # Distance to retract from the endstop after it is
hit for beta/Y
gamma_homing_retract_mm                     1              # Distance to retract from the endstop after it
is hit for gamma/Z


# Optional enable limit switches, actions will stop if any enabled limit switch is triggered (all are set
for delta)
#alpha_limit_enable                    false           # Set to true to enable X min and max limit switches
#beta_limit_enable                     false           # Set to true to enable Y min and max limit switches
#gamma_limit_enable                       false           # Set to true to enable Z min and max limit
```

```
switches

# Optional order in which axis will home, default is they all home at the same time,
# If this is set it will force each axis to home one at a time in the specified order
#homing_order                    XYZ          # X axis followed by Y then Z last
#move_to_origin_after_home           false        # Move XY to 0,0 after homing
#endstop_debounce_count             100          # Uncomment if you get noise on your
endstops, default is 100
#endstop_debounce_ms                1            # Uncomment if you get noise on your endstops,
default is 1 millisecond debounce
#home_z_first                    true         # Uncomment and set to true to home the Z first,
otherwise Z homes after XY

# End of endstop config
# Delete the above endstop section and uncomment next line and copy and edit Snippets/abc-end-
stop.config file to enable endstops for ABC axis
#include abc-endstop.config

## Z-probe
# See http://smoothieware.org/zprobe
zprobe.enable                    false        # Set to true to enable a zprobe
zprobe.probe_pin                  1.28!^       # Pin probe is attached to, if NC remove the !
zprobe.slow_feedrate               5           # Mm/sec probe feed rate
#zprobe.debounce_count               100          # Set if noisy
zprobe.fast_feedrate               100          # Move feedrate mm/sec
zprobe.probe_height                5           # How much above bed to start probe
#gamma_min_endstop                  nc           # Normally 1.28. Change to nc to prevent
conflict,

# Levelling strategy
# Example for 3-point levelling strategy, see wiki documentation for other strategies
#leveling-strategy.three-point-leveling.enable         true       # a leveling strategy that probes three
points to define a plane and keeps the Z parallel to that plane
#leveling-strategy.three-point-leveling.point1         100.0,0.0   # the first probe point (x,y) optional
may be defined with M557
#leveling-strategy.three-point-leveling.point2         200.0,200.0 # the second probe point (x,y)
#leveling-strategy.three-point-leveling.point3         0.0,200.0   # the third probe point (x,y)
#leveling-strategy.three-point-leveling.home_first     true       # home the XY axis before probing
#leveling-strategy.three-point-leveling.tolerance      0.03       # the probe tolerance in mm, anything
less that this will be ignored, default is 0.03mm
#leveling-strategy.three-point-leveling.probe_offsets  0,0,0      # the probe offsets from nozzle, must
be x,y,z, default is no offset
#leveling-strategy.three-point-leveling.save_plane     false      # set to true to allow the bed plane to
be saved with M500 default is false

## Panel
# See http://smoothieware.org/panel
# Please find your panel on the wiki and copy/paste the right configuration here
```

```
panel.enable                          false           # Set to true to enable the panel code

# Example for reprap discount GLCD
# on glcd EXP1 is to left and EXP2 is to right, pin 1 is bottom left, pin 2 is top left etc.
# +5v is EXP1 pin 10, Gnd is EXP1 pin 9
#panel.lcd                            reprap_discount_glcd     #
#panel.spi_channel                    0               # SPI channel to use  ; GLCD EXP1 Pins 3,5 (MOSI,
SCLK)
#panel.spi_cs_pin                     0.16            # SPI chip select     ; GLCD EXP1 Pin 4
#panel.encoder_a_pin                  3.25!^          # Encoder pin         ; GLCD EXP2 Pin 3
#panel.encoder_b_pin                  3.26!^          # Encoder pin         ; GLCD EXP2 Pin 5
#panel.click_button_pin               1.30!^          # Click button        ; GLCD EXP1 Pin 2
#panel.buzz_pin                       1.31            # Pin for buzzer      ; GLCD EXP1 Pin 1
#panel.back_button_pin                2.11!^          # Back button         ; GLCD EXP2 Pin 8

panel.menu_offset                     0               # Some panels will need 1 here

panel.alpha_jog_feedrate              6000            # X jogging feedrate in mm/min
panel.beta_jog_feedrate               6000            # Y jogging feedrate in mm/min
panel.gamma_jog_feedrate              200             # Z jogging feedrate in mm/min

panel.hotend_temperature              185             # Temp to set hotend when preheat is selected
panel.bed_temperature                 60              # Temp to set bed when preheat is selected

## Custom menus : Example of a custom menu entry, which will show up in the Custom entry.
# NOTE _ gets converted to space in the menu and commands, | is used to separate multiple
commands
custom_menu.power_on.enable           true            #
custom_menu.power_on.name             Power_on        #
custom_menu.power_on.command          M80             #

custom_menu.power_off.enable          true            #
custom_menu.power_off.name            Power_off       #
custom_menu.power_off.command         M81             #


## Network settings
# See http://smoothieware.org/network
network.enable                        false           # Enable the ethernet network services
network.webserver.enable              true            # Enable the webserver
network.telnet.enable                 true            # Enable the telnet server
network.ip_address                    auto            # Use dhcp to get ip address
# Uncomment the 3 below to manually setup ip address
#network.ip_address                   192.168.3.222   # The IP address
#network.ip_mask                      255.255.255.0   # The ip mask
#network.ip_gateway                   192.168.3.1     # The gateway address
#network.mac_override                 xx.xx.xx.xx.xx.xx  # Override the mac address, only do this if
you have a conflict
```

```
## System configuration
# Serial communications configuration ( baud rate defaults to 9600 if undefined )
# For communication over the UART port, *not* the USB/Serial port
uart0.baud_rate                         115200          # Baud rate for the default hardware ( UART )
serial port

second_usb_serial_enable                  false           # This enables a second USB serial port
#leds_disable                           true            # Disable using leds after config loaded
#play_led_disable                        true            # Disable the play led

# Kill button maybe assigned to a different pin, set to the onboard pin by default
# See http://smoothieware.org/killbutton
kill_button_enable                       true            # Set to true to enable a kill button
kill_button_pin                         2.12            # Kill button pin. default is same as pause button 2.12
(2.11 is another good choice)

#msd_disable                            false           # Disable the MSD (USB SDCARD), see http://
smoothieware.org/troubleshooting#disable-msd
#dfu_enable                             false           # For linux developers, set to true to enable DFU

# Only needed on a smoothieboard
# See http://smoothieware.org/currentcontrol
currentcontrol_module_enable              true            # Control stepper motor current via the
configuration file
```

GRBL Config File - Laser Controller Ver 1

```
/*
  config.h - compile time configuration
  Part of Grbl

  Copyright (c) 2012-2016 Sungeun K. Jeon for Gnea Research LLC
  Copyright (c) 2009-2011 Simen Svale Skogsrud

  Grbl is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  Grbl is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with Grbl.  If not, see <http://www.gnu.org/licenses/>.
*/

// This file contains compile-time configurations for Grbl's internal system. For the most part,
// users will not need to directly modify these, but they are here for specific needs, i.e.
// performance tuning or adjusting to non-typical machines.

// IMPORTANT: Any changes here requires a full re-compiling of the source code to propagate
them.

#ifndef config_h
#define config_h
#include "grbl.h" // For Arduino IDE compatibility.


// Define CPU pin map and default settings.
// NOTE: OEMs can avoid the need to maintain/update the defaults.h and cpu_map.h files and use
only
// one configuration file by placing their specific defaults and pin map at the bottom of this file.
// If doing so, simply comment out these two defines and see instructions below.
#define DEFAULTS_GENERIC
#define CPU_MAP_ATMEGA328P // Arduino Uno CPU

// Serial baud rate
// #define BAUD_RATE 230400
#define BAUD_RATE 115200
```

```
// Define realtime command special characters. These characters are 'picked-off' directly from the
// serial read data stream and are not passed to the grbl line execution parser. Select characters
// that do not and must not exist in the streamed g-code program. ASCII control characters may be
// used, if they are available per user setup. Also, extended ASCII codes (>127), which are never in
// g-code programs, maybe selected for interface programs.
// NOTE: If changed, manually update help message in report.c.

#define CMD_RESET 0x18 // ctrl-x.
#define CMD_STATUS_REPORT '?'
#define CMD_CYCLE_START '~'
#define CMD_FEED_HOLD '!'

// NOTE: All override realtime commands must be in the extended ASCII character set, starting
// at character value 128 (0x80) and up to 255 (0xFF). If the normal set of realtime commands,
// such as status reports, feed hold, reset, and cycle start, are moved to the extended set
// space, serial.c's RX ISR will need to be modified to accomodate the change.
// #define CMD_RESET 0x80
// #define CMD_STATUS_REPORT 0x81
// #define CMD_CYCLE_START 0x82
// #define CMD_FEED_HOLD 0x83
#define CMD_SAFETY_DOOR 0x84
#define CMD_JOG_CANCEL  0x85
#define CMD_DEBUG_REPORT 0x86 // Only when DEBUG enabled, sends debug report in '{}'
braces.
#define CMD_FEED_OVR_RESET 0x90         // Restores feed override value to 100%.
#define CMD_FEED_OVR_COARSE_PLUS 0x91
#define CMD_FEED_OVR_COARSE_MINUS 0x92
#define CMD_FEED_OVR_FINE_PLUS  0x93
#define CMD_FEED_OVR_FINE_MINUS  0x94
#define CMD_RAPID_OVR_RESET 0x95        // Restores rapid override value to 100%.
#define CMD_RAPID_OVR_MEDIUM 0x96
#define CMD_RAPID_OVR_LOW 0x97
// #define CMD_RAPID_OVR_EXTRA_LOW 0x98 // *NOT SUPPORTED*
#define CMD_SPINDLE_OVR_RESET 0x99      // Restores spindle override value to 100%.
#define CMD_SPINDLE_OVR_COARSE_PLUS 0x9A
#define CMD_SPINDLE_OVR_COARSE_MINUS 0x9B
#define CMD_SPINDLE_OVR_FINE_PLUS 0x9C
#define CMD_SPINDLE_OVR_FINE_MINUS 0x9D
#define CMD_SPINDLE_OVR_STOP 0x9E
#define CMD_COOLANT_FLOOD_OVR_TOGGLE 0xA0
#define CMD_COOLANT_MIST_OVR_TOGGLE 0xA1

// If homing is enabled, homing init lock sets Grbl into an alarm state upon power up. This forces
// the user to perform the homing cycle (or override the locks) before doing anything else. This is
// mainly a safety feature to remind the user to home, since position is unknown to Grbl.
#define HOMING_INIT_LOCK // Comment to disable

// Define the homing cycle patterns with bitmasks. The homing cycle first performs a search mode
// to quickly engage the limit switches, followed by a slower locate mode, and finished by a short
// pull-off motion to disengage the limit switches. The following HOMING_CYCLE_x defines are
executed
// in order starting with suffix 0 and completes the homing routine for the specified-axes only. If
// an axis is omitted from the defines, it will not home, nor will the system update its position.
// Meaning that this allows for users with non-standard cartesian machines, such as a lathe (x then
z,
// with no y), to configure the homing cycle behavior to their needs.
// NOTE: The homing cycle is designed to allow sharing of limit pins, if the axes are not in the same
// cycle, but this requires some pin settings changes in cpu_map.h file. For example, the default
homing
// cycle can share the Z limit pin with either X or Y limit pins, since they are on different cycles.
// By sharing a pin, this frees up a precious IO pin for other purposes. In theory, all axes limit pins
// may be reduced to one pin, if all axes are homed with seperate cycles, or vice versa, all three
axes
// on separate pin, but homed in one cycle. Also, it should be noted that the function of hard limits
// will not be affected by pin sharing.
// NOTE: Defaults are set for a traditional 3-axis CNC machine. Z-axis first to clear, followed by X &
Y.
#define HOMING_CYCLE_0 (1<<Z_AXIS)                // REQUIRED: First move Z to clear workspace.
#define HOMING_CYCLE_1 ((1<<X_AXIS)|(1<<Y_AXIS))  // OPTIONAL: Then move X,Y at the
same time.
// #define HOMING_CYCLE_2                         // OPTIONAL: Uncomment and add axes mask to
enable

// NOTE: The following are two examples to setup homing for 2-axis machines.
// #define HOMING_CYCLE_0 ((1<<X_AXIS)|(1<<Y_AXIS))  // NOT COMPATIBLE WITH
COREXY: Homes both X-Y in one cycle.

// #define HOMING_CYCLE_0 (1<<X_AXIS)  // COREXY COMPATIBLE: First home X
// #define HOMING_CYCLE_1 (1<<Y_AXIS)  // COREXY COMPATIBLE: Then home Y

// Number of homing cycles performed after when the machine initially jogs to limit switches.
// This help in preventing overshoot and should improve repeatability. This value should be one or
// greater.
#define N_HOMING_LOCATE_CYCLE 1 // Integer (1-128)

// Enables single axis homing commands. $HX, $HY, and $HZ for X, Y, and Z-axis homing. The full
homing
// cycle is still invoked by the $H command. This is disabled by default. It's here only to address
// users that need to switch between a two-axis and three-axis machine. This is actually very rare.
// If you have a two-axis machine, DON'T USE THIS. Instead, just alter the homing cycle for two-ax-
es.
// #define HOMING_SINGLE_AXIS_COMMANDS // Default disabled. Uncomment to enable.

// After homing, Grbl will set by default the entire machine space into negative space, as is typical
// for professional CNC machines, regardless of where the limit switches are located. Uncomment
this
```

```
// define to force Grbl to always set the machine origin at the homed location despite switch orien-
tation.
// #define HOMING_FORCE_SET_ORIGIN // Uncomment to enable.

// Number of blocks Grbl executes upon startup. These blocks are stored in EEPROM, where the
size
// and addresses are defined in settings.h. With the current settings, up to 2 startup blocks may
// be stored and executed in order. These startup blocks would typically be used to set the g-code
// parser state depending on user preferences.
#define N_STARTUP_LINE 2 // Integer (1-2)

// Number of floating decimal points printed by Grbl for certain value types. These settings are
// determined by realistic and commonly observed values in CNC machines. For example, position
// values cannot be less than 0.001mm or 0.0001in, because machines can not be physically more
// precise this. So, there is likely no need to change these, but you can if you need to here.
// NOTE: Must be an integer value from 0 to ~4. More than 4 may exhibit round-off errors.
#define N_DECIMAL_COORDVALUE_INCH 4 // Coordinate or position value in inches
#define N_DECIMAL_COORDVALUE_MM   3 // Coordinate or position value in mm
#define N_DECIMAL_RATEVALUE_INCH  1 // Rate or velocity value in in/min
#define N_DECIMAL_RATEVALUE_MM    0 // Rate or velocity value in mm/min
#define N_DECIMAL_SETTINGVALUE    3 // Decimals for floating point setting values
#define N_DECIMAL_RPMVALUE        0 // RPM value in rotations per min.

// If your machine has two limits switches wired in parallel to one axis, you will need to enable
// this feature. Since the two switches are sharing a single pin, there is no way for Grbl to tell
// which one is enabled. This option only effects homing, where if a limit is engaged, Grbl will
// alarm out and force the user to manually disengage the limit switch. Otherwise, if you have one
// limit switch for each axis, don't enable this option. By keeping it disabled, you can perform a
// homing cycle while on the limit switch and not have to move the machine off of it.
// #define LIMITS_TWO_SWITCHES_ON_AXES

// Allows GRBL to track and report gcode line numbers.  Enabling this means that the planning
buffer
// goes from 16 to 15 to make room for the additional line number data in the plan_block_t struct
// #define USE_LINE_NUMBERS // Disabled by default. Uncomment to enable.

// Upon a successful probe cycle, this option provides immediately feedback of the probe coordi-
nates
// through an automatically generated message. If disabled, users can still access the last probe
// coordinates through Grbl '$#' print parameters.
#define MESSAGE_PROBE_COORDINATES // Enabled by default. Comment to disable.

// Enables a second coolant control pin via the mist coolant g-code command M7 on the Arduino
Uno
// analog pin 4. Only use this option if you require a second coolant control pin.
// NOTE: The M8 flood coolant control pin on analog pin 3 will still be functional regardless.
// #define ENABLE_M7 // Disabled by default. Uncomment to enable.
// This option causes the feed hold input to act as a safety door switch. A safety door, when trig-
gered,
// immediately forces a feed hold and then safely de-energizes the machine. Resuming is blocked
until
// the safety door is re-engaged. When it is, Grbl will re-energize the machine and then resume on
the
// previous tool path, as if nothing happened.
// #define ENABLE_SAFETY_DOOR_INPUT_PIN // Default disabled. Uncomment to enable.

// After the safety door switch has been toggled and restored, this setting sets the power-up delay
// between restoring the spindle and coolant and resuming the cycle.
#define SAFETY_DOOR_SPINDLE_DELAY 4.0 // Float (seconds)
#define SAFETY_DOOR_COOLANT_DELAY 1.0 // Float (seconds)

// Enable CoreXY kinematics. Use ONLY with CoreXY machines.
// IMPORTANT: If homing is enabled, you must reconfigure the homing cycle #defines above to
// #define HOMING_CYCLE_0 (1<<X_AXIS) and #define HOMING_CYCLE_1 (1<<Y_AXIS)
// NOTE: This configuration option alters the motion of the X and Y axes to principle of operation
// defined at (http://corexy.com/theory.html). Motors are assumed to positioned and wired exactly
as
// described, if not, motions may move in strange directions. Grbl requires the CoreXY A and B
motors
// have the same steps per mm internally.
// #define COREXY // Default disabled. Uncomment to enable.

// Inverts pin logic of the control command pins based on a mask. This essentially means you can
use
// normally-closed switches on the specified pins, rather than the default normally-open switches.
// NOTE: The top option will mask and invert all control pins. The bottom option is an example of
// inverting only two control pins, the safety door and reset. See cpu_map.h for other bit definitions.
// #define INVERT_CONTROL_PIN_MASK CONTROL_MASK // Default disabled. Uncomment to
disable.
// #define INVERT_CONTROL_PIN_MASK ((1<<CONTROL_SAFETY_DOOR_BIT)|(CONTROL_
RESET_BIT)) // Default disabled.

// Inverts select limit pin states based on the following mask. This effects all limit pin functions,
// such as hard limits and homing. However, this is different from overall invert limits setting.
// This build option will invert only the limit pins defined here, and then the invert limits setting
// will be applied to all of them. This is useful when a user has a mixed set of limit pins with both
// normally-open(NO) and normally-closed(NC) switches installed on their machine.
// NOTE: PLEASE DO NOT USE THIS, unless you have a situation that needs it.
// #define INVERT_LIMIT_PIN_MASK ((1<<X_LIMIT_BIT)|(1<<Y_LIMIT_BIT))
// Default disabled. Uncomment to enable.

// Inverts the spindle enable pin from low-disabled/high-enabled to low-enabled/high-disabled.
Useful
// for some pre-built electronic boards.
// NOTE: If VARIABLE_SPINDLE is enabled(default), this option has no effect as the PWM output
```

```
and
// spindle enable are combined to one pin. If you need both this option and spindle speed PWM,
// uncomment the config option USE_SPINDLE_DIR_AS_ENABLE_PIN below.
// #define INVERT_SPINDLE_ENABLE_PIN // Default disabled. Uncomment to enable.

// Inverts the selected coolant pin from low-disabled/high-enabled to low-enabled/high-disabled.
Useful
// for some pre-built electronic boards.
// #define INVERT_COOLANT_FLOOD_PIN // Default disabled. Uncomment to enable.
// #define INVERT_COOLANT_MIST_PIN // Default disabled. Note: Enable M7 mist coolant in
config.h

// When Grbl powers-cycles or is hard reset with the Arduino reset button, Grbl boots up with no
ALARM
// by default. This is to make it as simple as possible for new users to start using Grbl. When homing
// is enabled and a user has installed limit switches, Grbl will boot up in an ALARM state to indicate
// Grbl doesn't know its position and to force the user to home before proceeding. This option
forces
// Grbl to always initialize into an ALARM state regardless of homing or not. This option is more for
// OEMs and LinuxCNC users that would like this power-cycle behavior.
// #define FORCE_INITIALIZATION_ALARM // Default disabled. Uncomment to enable.

// At power-up or a reset, Grbl will check the limit switch states to ensure they are not active
// before initialization. If it detects a problem and the hard limits setting is enabled, Grbl will
// simply message the user to check the limits and enter an alarm state, rather than idle. Grbl will
// not throw an alarm message.
#define CHECK_LIMITS_AT_INIT

// ---------------------------------------------------------------------------------------
// ADVANCED CONFIGURATION OPTIONS:

// Enables code for debugging purposes. Not for general use and always in constant flux.
// #define DEBUG // Uncomment to enable. Default disabled.

// Configure rapid, feed, and spindle override settings. These values define the max and min
// allowable override values and the coarse and fine increments per command received. Please
// note the allowable values in the descriptions following each define.
#define DEFAULT_FEED_OVERRIDE           100 // 100%. Don't change this value.
#define MAX_FEED_RATE_OVERRIDE          200 // Percent of programmed feed rate (100-255).
Usually 120% or 200%
#define MIN_FEED_RATE_OVERRIDE           10 // Percent of programmed feed rate (1-100).
Usually 50% or 1%
#define FEED_OVERRIDE_COARSE_INCREMENT   10 // (1-99). Usually 10%.
#define FEED_OVERRIDE_FINE_INCREMENT      1 // (1-99). Usually 1%.

#define DEFAULT_RAPID_OVERRIDE  100 // 100%. Don't change this value.
#define RAPID_OVERRIDE_MEDIUM    50 // Percent of rapid (1-99). Usually 50%.
#define RAPID_OVERRIDE_LOW       25 // Percent of rapid (1-99). Usually 25%.
```

```
// #define RAPID_OVERRIDE_EXTRA_LOW 5 // *NOT SUPPORTED* Percent of rapid (1-99).
Usually 5%.

#define DEFAULT_SPINDLE_SPEED_OVERRIDE    100 // 100%. Don't change this value.
#define MAX_SPINDLE_SPEED_OVERRIDE        200 // Percent of programmed spindle speed
(100-255). Usually 200%.
#define MIN_SPINDLE_SPEED_OVERRIDE         10 // Percent of programmed spindle speed (1-
100). Usually 10%.
#define SPINDLE_OVERRIDE_COARSE_INCREMENT  10 // (1-99). Usually 10%.
#define SPINDLE_OVERRIDE_FINE_INCREMENT     1 // (1-99). Usually 1%.

// When a M2 or M30 program end command is executed, most g-code states are restored to their
defaults.
// This compile-time option includes the restoring of the feed, rapid, and spindle speed override
values
// to their default values at program end.
#define RESTORE_OVERRIDES_AFTER_PROGRAM_END // Default enabled. Comment to disable.

// The status report change for Grbl v1.1 and after also removed the ability to disable/enable most
data
// fields from the report. This caused issues for GUI developers, who've had to manage several
scenarios
// and configurations. The increased efficiency of the new reporting style allows for all data fields to
// be sent without potential performance issues.
// NOTE: The options below are here only provide a way to disable certain data fields if a unique
// situation demands it, but be aware GUIs may depend on this data. If disabled, it may not be
compatible.
#define REPORT_FIELD_BUFFER_STATE // Default enabled. Comment to disable.
#define REPORT_FIELD_PIN_STATE // Default enabled. Comment to disable.
#define REPORT_FIELD_CURRENT_FEED_SPEED // Default enabled. Comment to disable.
#define REPORT_FIELD_WORK_COORD_OFFSET // Default enabled. Comment to disable.
#define REPORT_FIELD_OVERRIDES // Default enabled. Comment to disable.
#define REPORT_FIELD_LINE_NUMBERS // Default enabled. Comment to disable.

// Some status report data isn't necessary for realtime, only intermittently, because the values don't
// change often. The following macros configures how many times a status report needs to be
called before
// the associated data is refreshed and included in the status report. However, if one of these value
// changes, Grbl will automatically include this data in the next status report, regardless of what the
// count is at the time. This helps reduce the communication overhead involved with high frequency
reporting
// and agressive streaming. There is also a busy and an idle refresh count, which sets up Grbl to
send
// refreshes more often when its not doing anything important. With a good GUI, this data doesn't
need
// to be refreshed very often, on the order of a several seconds.
// NOTE: WCO refresh must be 2 or greater. OVR refresh must be 1 or greater.
#define REPORT_OVR_REFRESH_BUSY_COUNT 20  // (1-255)
```

```
#define REPORT_OVR_REFRESH_IDLE_COUNT 10  // (1-255) Must be less than or equal to the
busy count
#define REPORT_WCO_REFRESH_BUSY_COUNT 30  // (2-255)
#define REPORT_WCO_REFRESH_IDLE_COUNT 10  // (2-255) Must be less than or equal to the
busy count

// The temporal resolution of the acceleration management subsystem. A higher number gives
smoother
// acceleration, particularly noticeable on machines that run at very high feedrates, but may nega-
tively
// impact performance. The correct value for this parameter is machine dependent, so it's advised
to
// set this only as high as needed. Approximate successful values can widely range from 50 to 200
or more.
// NOTE: Changing this value also changes the execution time of a segment in the step segment
buffer.
// When increasing this value, this stores less overall time in the segment buffer and vice versa.
Make
// certain the step segment buffer is increased/decreased to account for these changes.
#define ACCELERATION_TICKS_PER_SECOND 100

// Adaptive Multi-Axis Step Smoothing (AMASS) is an advanced feature that does what its name
implies,
// smoothing the stepping of multi-axis motions. This feature smooths motion particularly at low
step
// frequencies below 10kHz, where the aliasing between axes of multi-axis motions can cause
audible
// noise and shake your machine. At even lower step frequencies, AMASS adapts and provides even
better
// step smoothing. See stepper.c for more details on the AMASS system works.
#define ADAPTIVE_MULTI_AXIS_STEP_SMOOTHING  // Default enabled. Comment to disable.

// Sets the maximum step rate allowed to be written as a Grbl setting. This option enables an error
// check in the settings module to prevent settings values that will exceed this limitation. The maxi-
mum
// step rate is strictly limited by the CPU speed and will change if something other than an AVR
running
// at 16MHz is used.
// NOTE: For now disabled, will enable if flash space permits.
// #define MAX_STEP_RATE_HZ 30000 // Hz

// By default, Grbl sets all input pins to normal-high operation with their internal pull-up resistors
// enabled. This simplifies the wiring for users by requiring only a switch connected to ground,
// although its recommended that users take the extra step of wiring in low-pass filter to reduce
// electrical noise detected by the pin. If the user inverts the pin in Grbl settings, this just flips
// which high or low reading indicates an active signal. In normal operation, this means the user
// needs to connect a normal-open switch, but if inverted, this means the user should connect a
// normal-closed switch.
// The following options disable the internal pull-up resistors, sets the pins to a normal-low
// operation, and switches must be now connect to Vcc instead of ground. This also flips the mean-
ing
// of the invert pin Grbl setting, where an inverted setting now means the user should connect a
// normal-open switch and vice versa.
// NOTE: All pins associated with the feature are disabled, i.e. XYZ limit pins, not individual axes.
// WARNING: When the pull-ups are disabled, this requires additional wiring with pull-down resis-
tors!
//#define DISABLE_LIMIT_PIN_PULL_UP
//#define DISABLE_PROBE_PIN_PULL_UP
//#define DISABLE_CONTROL_PIN_PULL_UP

// Sets which axis the tool length offset is applied. Assumes the spindle is always parallel with
// the selected axis with the tool oriented toward the negative direction. In other words, a positive
// tool length offset value is subtracted from the current location.
#define TOOL_LENGTH_OFFSET_AXIS Z_AXIS // Default z-axis. Valid values are X_AXIS, Y_AXIS,
or Z_AXIS.

// Enables variable spindle output voltage for different RPM values. On the Arduino Uno, the spindle
// enable pin will output 5V for maximum RPM with 256 intermediate levels and 0V when disabled.
// NOTE: IMPORTANT for Arduino Unos! When enabled, the Z-limit pin D11 and spindle enable pin
D12 switch!
// The hardware PWM output on pin D11 is required for variable spindle output voltages.
#define VARIABLE_SPINDLE // Default enabled. Comment to disable.

// Used by variable spindle output only. This forces the PWM output to a minimum duty cycle when
enabled.
// The PWM pin will still read 0V when the spindle is disabled. Most users will not need this option,
but
// it may be useful in certain scenarios. This minimum PWM settings coincides with the spindle rpm
minimum
// setting, like rpm max to max PWM. This is handy if you need a larger voltage difference between
0V disabled
// and the voltage set by the minimum PWM for minimum rpm. This difference is 0.02V per PWM
value. So, when
// minimum PWM is at 1, only 0.02 volts separate enabled and disabled. At PWM 5, this would be
0.1V. Keep
// in mind that you will begin to lose PWM resolution with increased minimum PWM values, since
you have less
// and less range over the total 255 PWM levels to signal different spindle speeds.
// NOTE: Compute duty cycle at the minimum PWM by this equation: (% duty cycle)=(SPINDLE_
PWM_MIN_VALUE/255)*100
// #define SPINDLE_PWM_MIN_VALUE 5 // Default disabled. Uncomment to enable. Must be
greater than zero. Integer (1-255).

// By default on a 328p(Uno), Grbl combines the variable spindle PWM and the enable into one pin
to help
// preserve I/O pins. For certain setups, these may need to be separate pins. This configure option
```

```
uses
// the spindle direction pin(D13) as a separate spindle enable pin along with spindle speed PWM on
pin D11.
// NOTE: This configure option only works with VARIABLE_SPINDLE enabled and a 328p processor
(Uno).
// NOTE: Without a direction pin, M4 will not have a pin output to indicate a difference with M3.
// NOTE: BEWARE! The Arduino bootloader toggles the D13 pin when it powers up. If you flash Grbl
with
// a programmer (you can use a spare Arduino as "Arduino as ISP". Search the web on how to wire
this.),
// this D13 LED toggling should go away. We haven't tested this though. Please report how it goes!
// #define USE_SPINDLE_DIR_AS_ENABLE_PIN // Default disabled. Uncomment to enable.

// Alters the behavior of the spindle enable pin with the USE_SPINDLE_DIR_AS_ENABLE_PIN
option . By default,
// Grbl will not disable the enable pin if spindle speed is zero and M3/4 is active, but still sets the
PWM
// output to zero. This allows the users to know if the spindle is active and use it as an additional
control
// input. However, in some use cases, user may want the enable pin to disable with a zero spindle
speed and
// re-enable when spindle speed is greater than zero. This option does that.
// NOTE: Requires USE_SPINDLE_DIR_AS_ENABLE_PIN to be enabled.
// #define SPINDLE_ENABLE_OFF_WITH_ZERO_SPEED // Default disabled. Uncomment to
enable.

// With this enabled, Grbl sends back an echo of the line it has received, which has been pre-parsed
(spaces
// removed, capitalized letters, no comments) and is to be immediately executed by Grbl. Echoes
will not be
// sent upon a line buffer overflow, but should for all normal lines sent to Grbl. For example, if a user
// sendss the line 'g1 x1.032 y2.45 (test comment)', Grbl will echo back in the form '[echo: G1X-
1.032Y2.45]'.
// NOTE: Only use this for debugging purposes!! When echoing, this takes up valuable resources
and can effect
// performance. If absolutely needed for normal operation, the serial write buffer should be greatly
increased
// to help minimize transmission waiting within the serial write protocol.
// #define REPORT_ECHO_LINE_RECEIVED // Default disabled. Uncomment to enable.

// Minimum planner junction speed. Sets the default minimum junction speed the planner plans to
at
// every buffer block junction, except for starting from rest and end of the buffer, which are always
// zero. This value controls how fast the machine moves through junctions with no regard for accel-
eration
// limits or angle between neighboring block line move directions. This is useful for machines that
can't
// tolerate the tool dwelling for a split second, i.e. 3d printers or laser cutters. If used, this value
// should not be much greater than zero or to the minimum value necessary for the machine to
work.
#define MINIMUM_JUNCTION_SPEED 0.0 // (mm/min)

// Sets the minimum feed rate the planner will allow. Any value below it will be set to this minimum
// value. This also ensures that a planned motion always completes and accounts for any float-
ing-point
// round-off errors. Although not recommended, a lower value than 1.0 mm/min will likely work in
smaller
// machines, perhaps to 0.1mm/min, but your success may vary based on multiple factors.
#define MINIMUM_FEED_RATE 1.0 // (mm/min)

// Number of arc generation iterations by small angle approximation before exact arc trajectory
// correction with expensive sin() and cos() calcualtions. This parameter maybe decreased if there
// are issues with the accuracy of the arc generations, or increased if arc execution is getting
// bogged down by too many trig calculations.
#define N_ARC_CORRECTION 12 // Integer (1-255)

// The arc G2/3 g-code standard is problematic by definition. Radius-based arcs have horrible
numerical
// errors when arc at semi-circles(pi) or full-circles(2*pi). Offset-based arcs are much more accu-
rate
// but still have a problem when arcs are full-circles (2*pi). This define accounts for the floating
// point issues when offset-based arcs are commanded as full circles, but get interpreted as ex-
tremely
// small arcs with around machine epsilon (1.2e-7rad) due to numerical round-off and precision
issues.
// This define value sets the machine epsilon cutoff to determine if the arc is a full-circle or not.
// NOTE: Be very careful when adjusting this value. It should always be greater than 1.2e-7 but not
too
// much greater than this. The default setting should capture most, if not all, full arc error situations.
#define ARC_ANGULAR_TRAVEL_EPSILON 5E-7 // Float (radians)

// Time delay increments performed during a dwell. The default value is set at 50ms, which provides
// a maximum time delay of roughly 55 minutes, more than enough for most any application.
Increasing
// this delay will increase the maximum dwell time linearly, but also reduces the responsiveness of
// run-time command executions, like status reports, since these are performed between each
dwell
// time step. Also, keep in mind that the Arduino delay timer is not very accurate for long delays.
#define DWELL_TIME_STEP 50 // Integer (1-255) (milliseconds)

// Creates a delay between the direction pin setting and corresponding step pulse by creating
// another interrupt (Timer2 compare) to manage it. The main Grbl interrupt (Timer1 compare)
// sets the direction pins, and does not immediately set the stepper pins, as it would in
// normal operation. The Timer2 compare fires next to set the stepper pins after the step
// pulse delay time, and Timer2 overflow will complete the step pulse, except now delayed
// by the step pulse time plus the step pulse delay. (Thanks langwadt for the idea!)
// NOTE: Uncomment to enable. The recommended delay must be > 3us, and, when added with the
// user-supplied step pulse time, the total time must not exceed 127us. Reported successful
// values for certain setups have ranged from 5 to 20us.
// #define STEP_PULSE_DELAY 10 // Step pulse delay in microseconds. Default disabled.

// The number of linear motions in the planner buffer to be planned at any give time. The vast
// majority of RAM that Grbl uses is based on this buffer size. Only increase if there is extra
// available RAM, like when re-compiling for a Mega2560. Or decrease if the Arduino begins to
// crash due to the lack of available RAM or if the CPU is having trouble keeping up with planning
// new incoming motions as they are executed.
// #define BLOCK_BUFFER_SIZE 16 // Uncomment to override default in planner.h.

// Governs the size of the intermediary step segment buffer between the step execution algorithm
// and the planner blocks. Each segment is set of steps executed at a constant velocity over a
// fixed time defined by ACCELERATION_TICKS_PER_SECOND. They are computed such that the
planner
// block velocity profile is traced exactly. The size of this buffer governs how much step
// execution lead time there is for other Grbl processes have to compute and do their thing
// before having to come back and refill this buffer, currently at ~50msec of step moves.
// #define SEGMENT_BUFFER_SIZE 6 // Uncomment to override default in stepper.h.

// Line buffer size from the serial input stream to be executed. Also, governs the size of
// each of the startup blocks, as they are each stored as a string of this size. Make sure
// to account for the available EEPROM at the defined memory address in settings.h and for
// the number of desired startup blocks.
// NOTE: 80 characters is not a problem except for extreme cases, but the line buffer size
// can be too small and g-code blocks can get truncated. Officially, the g-code standards
// support up to 256 characters. In future versions, this default will be increased, when
// we know how much extra memory space we can re-invest into this.
// #define LINE_BUFFER_SIZE 80  // Uncomment to override default in protocol.h

// Serial send and receive buffer size. The receive buffer is often used as another streaming
// buffer to store incoming blocks to be processed by Grbl when its ready. Most streaming
// interfaces will character count and track each block send to each block response. So,
// increase the receive buffer if a deeper receive buffer is needed for streaming and avaiable
// memory allows. The send buffer primarily handles messages in Grbl. Only increase if large
// messages are sent and Grbl begins to stall, waiting to send the rest of the message.
// NOTE: Grbl generates an average status report in about 0.5msec, but the serial TX stream at
// 115200 baud will take 5 msec to transmit a typical 55 character report. Worst case reports are
// around 90-100 characters. As long as the serial TX buffer doesn't get continually maxed, Grbl
// will continue operating efficiently. Size the TX buffer around the size of a worst-case report.
// #define RX_BUFFER_SIZE 128 // (1-254) Uncomment to override defaults in serial.h
// #define TX_BUFFER_SIZE 100 // (1-254)

// A simple software debouncing feature for hard limit switches. When enabled, the interrupt
// monitoring the hard limit switch pins will enable the Arduino's watchdog timer to re-check
// the limit pin state after a delay of about 32msec. This can help with CNC machines with
// problematic false triggering of their hard limit switches, but it WILL NOT fix issues with
// electrical interference on the signal cables from external sources. It's recommended to first
// use shielded signal cables with their shielding connected to ground (old USB/computer cables
// work well and are cheap to find) and wire in a low-pass circuit into each limit pin.
// #define ENABLE_SOFTWARE_DEBOUNCE // Default disabled. Uncomment to enable.

// Configures the position after a probing cycle during Grbl's check mode. Disabled sets
// the position to the probe target, when enabled sets the position to the start position.
// #define SET_CHECK_MODE_PROBE_TO_START // Default disabled. Uncomment to enable.

// Force Grbl to check the state of the hard limit switches when the processor detects a pin
// change inside the hard limit ISR routine. By default, Grbl will trigger the hard limits
// alarm upon any pin change, since bouncing switches can cause a state check like this to
// misread the pin. When hard limits are triggered, they should be 100% reliable, which is the
// reason that this option is disabled by default. Only if your system/electronics can guarantee
// that the switches don't bounce, we recommend enabling this option. This will help prevent
// triggering a hard limit when the machine disengages from the switch.
// NOTE: This option has no effect if SOFTWARE_DEBOUNCE is enabled.
// #define HARD_LIMIT_FORCE_STATE_CHECK // Default disabled. Uncomment to enable.

// Adjusts homing cycle search and locate scalars. These are the multipliers used by Grbl's
// homing cycle to ensure the limit switches are engaged and cleared through each phase of
// the cycle. The search phase uses the axes max-travel setting times the SEARCH_SCALAR to
// determine distance to look for the limit switch. Once found, the locate phase begins and
// uses the homing pull-off distance setting times the LOCATE_SCALAR to pull-off and re-engage
// the limit switch.
// NOTE: Both of these values must be greater than 1.0 to ensure proper function.
// #define HOMING_AXIS_SEARCH_SCALAR  1.5 // Uncomment to override defaults in limits.c.
// #define HOMING_AXIS_LOCATE_SCALAR  10.0 // Uncomment to override defaults in limits.c.

// Enable the '$RST=*', '$RST=$', and '$RST=#' eeprom restore commands. There are cases where
// these commands may be undesirable. Simply comment the desired macro to disable it.
// NOTE: See SETTINGS_RESTORE_ALL macro for customizing the `$RST=*` command.
#define ENABLE_RESTORE_EEPROM_WIPE_ALL         // '$RST=*' Default enabled. Comment to
disable.
#define ENABLE_RESTORE_EEPROM_DEFAULT_SETTINGS // '$RST=$' Default enabled. Com-
ment to disable.
#define ENABLE_RESTORE_EEPROM_CLEAR_PARAMETERS // '$RST=#' Default enabled. Com-
ment to disable.

// Defines the EEPROM data restored upon a settings version change and `$RST=*` command.
Whenever the
// the settings or other EEPROM data structure changes between Grbl versions, Grbl will automat-
ically
// wipe and restore the EEPROM. This macro controls what data is wiped and restored. This is
useful
// particularily for OEMs that need to retain certain data. For example, the BUILD_INFO string can
be
// written into the Arduino EEPROM via a seperate .INO sketch to contain product data. Altering this
```

// macro to not restore the build info EEPROM will ensure this data is retained after firmware upgrades.
// NOTE: Uncomment to override defaults in settings.h
// #define SETTINGS_RESTORE_ALL (SETTINGS_RESTORE_DEFAULTS | SETTINGS_RESTORE_PARAMETERS | SETTINGS_RESTORE_STARTUP_LINES | SETTINGS_RESTORE_BUILD_INFO)

// Enable the '$I=(string)' build info write command. If disabled, any existing build info data must
// be placed into EEPROM via external means with a valid checksum value. This macro option is useful
// to prevent this data from being over-written by a user, when used to store OEM product data.
// NOTE: If disabled and to ensure Grbl can never alter the build info line, you'll also need to enable
// the SETTING_RESTORE_ALL macro above and remove SETTINGS_RESTORE_BUILD_INFO from the mask.
// NOTE: See the included grblWrite_BuildInfo.ino example file to write this string seperately.
#define ENABLE_BUILD_INFO_WRITE_COMMAND // '$I=' Default enabled. Comment to disable.

// AVR processors require all interrupts to be disabled during an EEPROM write. This includes both
// the stepper ISRs and serial comm ISRs. In the event of a long EEPROM write, this ISR pause can
// cause active stepping to lose position and serial receive data to be lost. This configuration
// option forces the planner buffer to completely empty whenever the EEPROM is written to prevent
// any chance of lost steps.
// However, this doesn't prevent issues with lost serial RX data during an EEPROM write, especially
// if a GUI is premptively filling up the serial RX buffer simultaneously. It's highly advised for
// GUIs to flag these gcodes (G10,G28.1,G30.1) to always wait for an 'ok' after a block containing
// one of these commands before sending more data to eliminate this issue.
// NOTE: Most EEPROM write commands are implicitly blocked during a job (all '$' commands). However,
// coordinate set g-code commands (G10,G28/30.1) are not, since they are part of an active streaming
// job. At this time, this option only forces a planner buffer sync with these g-code commands.
#define FORCE_BUFFER_SYNC_DURING_EEPROM_WRITE // Default enabled. Comment to disable.

// In Grbl v0.9 and prior, there is an old outstanding bug where the `WPos:` work position reported
// may not correlate to what is executing, because `WPos:` is based on the g-code parser state, which
// can be several motions behind. This option forces the planner buffer to empty, sync, and stop
// motion whenever there is a command that alters the work coordinate offsets `G10,G43.1,G92,G54-59`.
// This is the simplest way to ensure `WPos:` is always correct. Fortunately, it's exceedingly rare
// that any of these commands are used need continuous motions through them.
#define FORCE_BUFFER_SYNC_DURING_WCO_CHANGE // Default enabled. Comment to disable.

// By default, Grbl disables feed rate overrides for all G38.x probe cycle commands. Although this
// may be different than some pro-class machine control, it's arguable that it should be this way.
// Most probe sensors produce different levels of error that is dependent on rate of speed. By
// keeping probing cycles to their programmed feed rates, the probe sensor should be a lot more

```
// repeatable. If needed, you can disable this behavior by uncommenting the define below.
// #define ALLOW_FEED_OVERRIDE_DURING_PROBE_CYCLES // Default disabled. Uncomment
to enable.

// Enables and configures parking motion methods upon a safety door state. Primarily for OEMs
// that desire this feature for their integrated machines. At the moment, Grbl assumes that
// the parking motion only involves one axis, although the parking implementation was written
// to be easily refactored for any number of motions on different axes by altering the parking
// source code. At this time, Grbl only supports parking one axis (typically the Z-axis) that
// moves in the positive direction upon retracting and negative direction upon restoring position.
// The motion executes with a slow pull-out retraction motion, power-down, and a fast park.
// Restoring to the resume position follows these set motions in reverse: fast restore to
// pull-out position, power-up with a time-out, and plunge back to the original position at the
// slower pull-out rate.
// NOTE: Still a work-in-progress. Machine coordinates must be in all negative space and
// does not work with HOMING_FORCE_SET_ORIGIN enabled. Parking motion also moves only in
// positive direction.
// #define PARKING_ENABLE  // Default disabled. Uncomment to enable

// Configure options for the parking motion, if enabled.
#define PARKING_AXIS Z_AXIS // Define which axis that performs the parking motion
#define PARKING_TARGET -5.0 // Parking axis target. In mm, as machine coordinate [-max_trav-
el,0].
#define PARKING_RATE 500.0 // Parking fast rate after pull-out in mm/min.
#define PARKING_PULLOUT_RATE 100.0 // Pull-out/plunge slow feed rate in mm/min.
#define PARKING_PULLOUT_INCREMENT 5.0 // Spindle pull-out and plunge distance in mm.
Incremental distance.
                                // Must be positive value or equal to zero.

// Enables a special set of M-code commands that enables and disables the parking motion.
// These are controlled by `M56`, `M56 P1`, or `M56 Px` to enable and `M56 P0` to disable.
// The command is modal and will be set after a planner sync. Since it is g-code, it is
// executed in sync with g-code commands. It is not a real-time command.
// NOTE: PARKING_ENABLE is required. By default, M56 is active upon initialization. Use
// DEACTIVATE_PARKING_UPON_INIT to set M56 P0 as the power-up default.
// #define ENABLE_PARKING_OVERRIDE_CONTROL   // Default disabled. Uncomment to enable
// #define DEACTIVATE_PARKING_UPON_INIT // Default disabled. Uncomment to enable.

// This option will automatically disable the laser during a feed hold by invoking a spindle stop
// override immediately after coming to a stop. However, this also means that the laser still may
// be reenabled by disabling the spindle stop override, if needed. This is purely a safety feature
// to ensure the laser doesn't inadvertently remain powered while at a stop and cause a fire.
#define DISABLE_LASER_DURING_HOLD // Default enabled. Comment to disable.

/* ---------------------------------------------------------------------------------------
  OEM Single File Configuration Option

  Instructions: Paste the cpu_map and default setting definitions below without an enclosing
  #ifdef. Comment out the CPU_MAP_xxx and DEFAULT_xxx defines at the top of this file, and
  the compiler will ignore the contents of defaults.h and cpu_map.h and use the definitions
  below.
*/

// Paste CPU_MAP definitions here.

// Paste default settings definitions here.


#endif
```