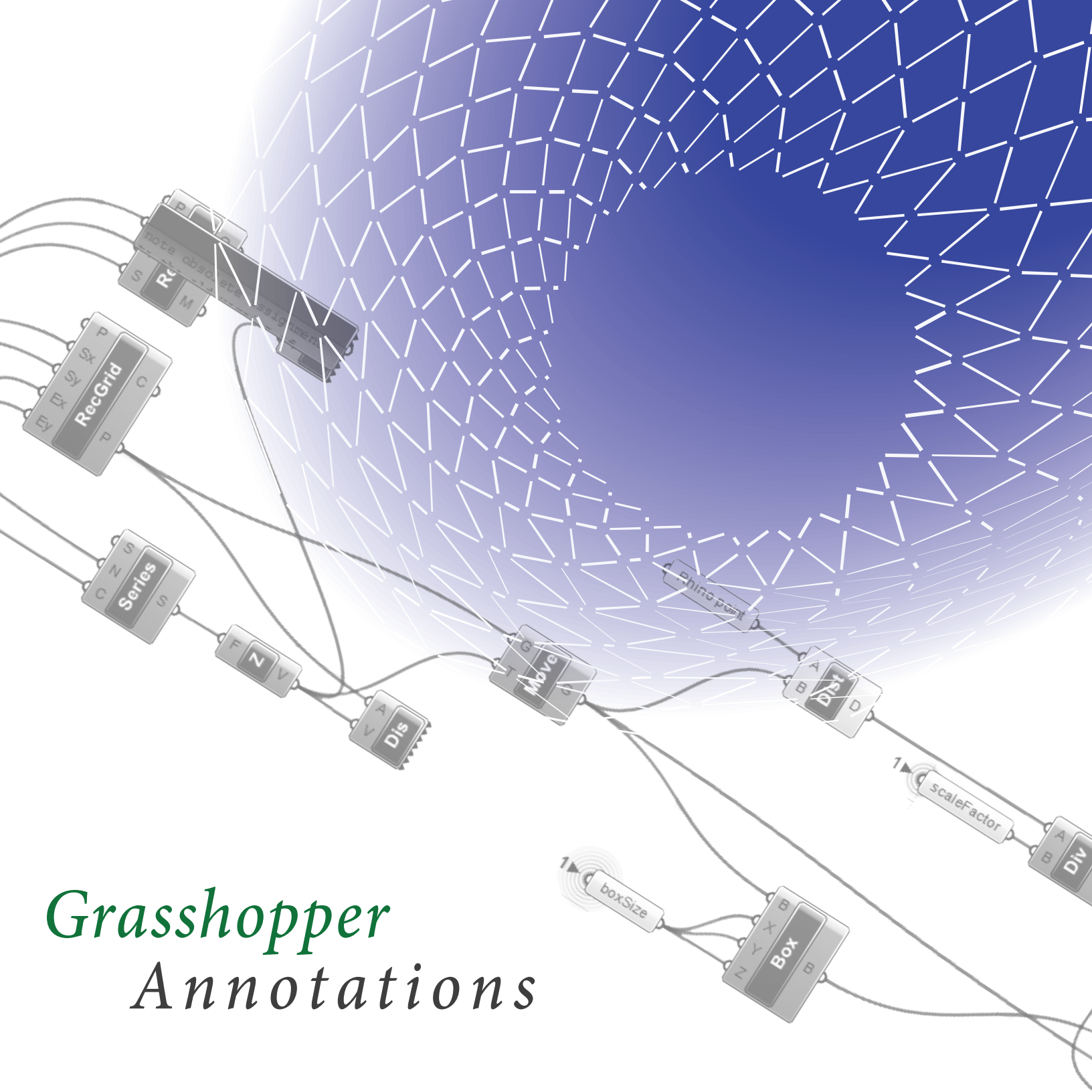


Grasshopper Annotations

Contributors: Andrew Payne
Rajaa Issa
Zubin Khabazi
Kyle Talbott

Written and produced for Independent Study with the Honors College and the School of Architecture and Urban Planning at the University of Wisconsin, Milwaukee.
July 2010

Verge Domain Publishing
All rights reserved

Kevin Hinz
6068 Templeton Drive
Waunakee, WI 53597 USA
kevin_h_us@yahoo.com

ISBN 978-0-578-07439-9

Thank you to the SARUP students and professors who provided advice and analysis during the planning, development and execution of the following work. I appreciate your help, patience and the assistance provided to write the following scripts. Special thanks to Andrew Payne, Rajaa Issa, Zubin Khabazi and Kyle Talbott for making their intellectual property available for annotation.

Kevin Hinz
kevin_h_us@yahoo.com

Contents

Parametric Notation

This publication is assembled as an elaboration of exercises provided in Andrew Payne and Rajaa Issa's *Grasshopper Primer*. In addition, lessons outlined in *Algorithmic Modeling with Grasshopper* by Zubin Khabazi, are referenced as well as scripts written for studio projects during Professor Kyle Talbott's *Microcosm Studio*. The goal is to provide information and insight encountered while completing the Grasshopper tutorials with intention to share the experiences encountered during the learning process.

Grasshopper is an intuitive parametric modeling plug-in running on the Rhinoceros platform. As with any new learning experience, prior education and explanation is paramount to knowledge retention. So much knowledge is available that it can be difficult to discern the exact solution and come away with specific information. This publication is probably no different. However, as an addition to the extensive work assembled by the afore mentioned authors, *Grasshopper Annotations* brings analysis to the table during the administration of tutorials. Explanations provided in the accompanying source files provide insight into the interface organization as well as a breakdown outlining design intentions and how they relate to the construction of each script. Throughout the course of the book and dvd, you will find suggestions and solutions to potential problems for peculiar encounters. I encourage readers to include their own notes at the conclusion of each chapter to share additional acquired experiences.

Because dates are significant factors to the evolutionary progression of changing programs and the information that accompanies them, each of the source files corresponding to the following chapters have been dated. Links to the tools used to construct and assemble this publication are available in the dvd accompanying the printed work and also listed in the bibliography in the conclusion of this document. Other time sensitive specifications, outlining exterior works cited and programs used are as follows:

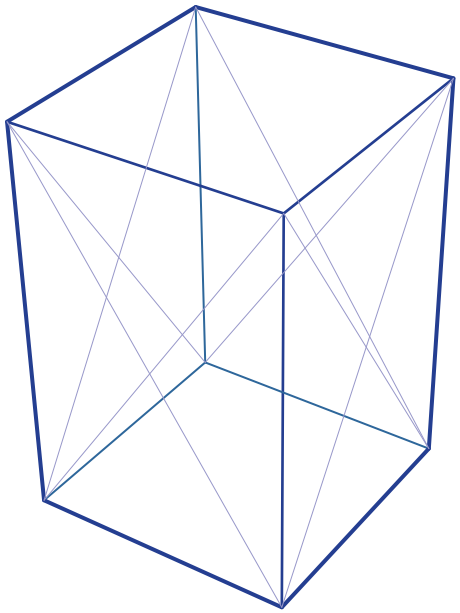
Rhinoceros 4.0, SR7	http://download.rhino3d.com/rhino/4.0/evaluation/download/
Grasshopper Plugin version 0.6.0059	http://www.grasshopper3d.com/page/download-1
Grasshopper Primer for version 0.6.0007, 2009	http://www.liftarchitects.com/downloads/
Generative Algorithms using Grasshopper, 2010	http://download.mcneel.com/s3/mcneel/grasshopper/1.0/docs/en/Generative%20Algorithms.pdf

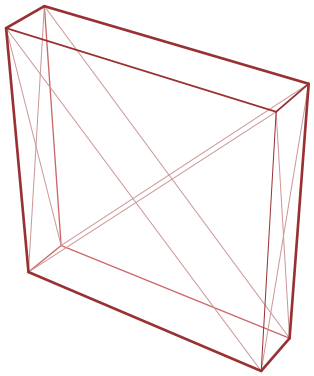
Note: recent versions of Grasshopper often develop issues that prevent the use of scripts written on previous versions. One such encounter happened during the production of this publication because I updated to plugin version 0.6.0055 while working; the issue was minor. The most recent current stable Grasshopper build is now 0.6.0059 but development is to build 0.7.0036 as of July, 2010.

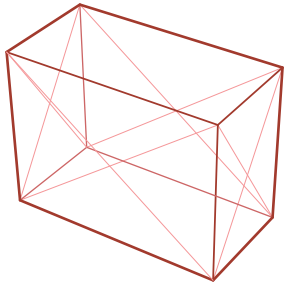
Draw a Parametric Box

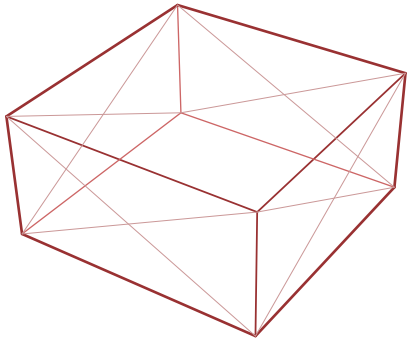
This exercise provides an introduction to the creation and subsequent manipulation of basic geometry. Although the exercise appears at first elementary, the theory behind what is occurring has powerful potential.

I recommend exploring the theories presented here as a way to practice and thus grasp how geometry is determined from spacial orientation.



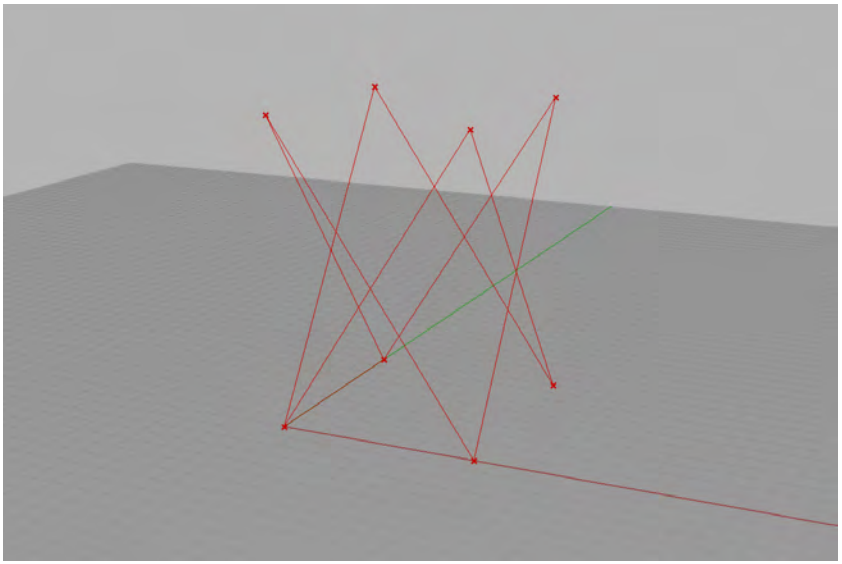


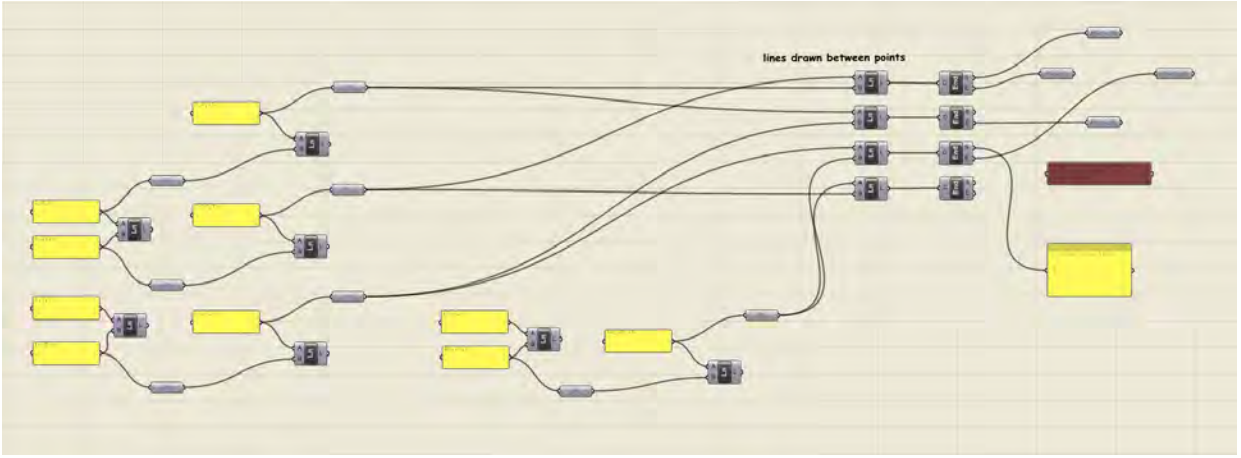


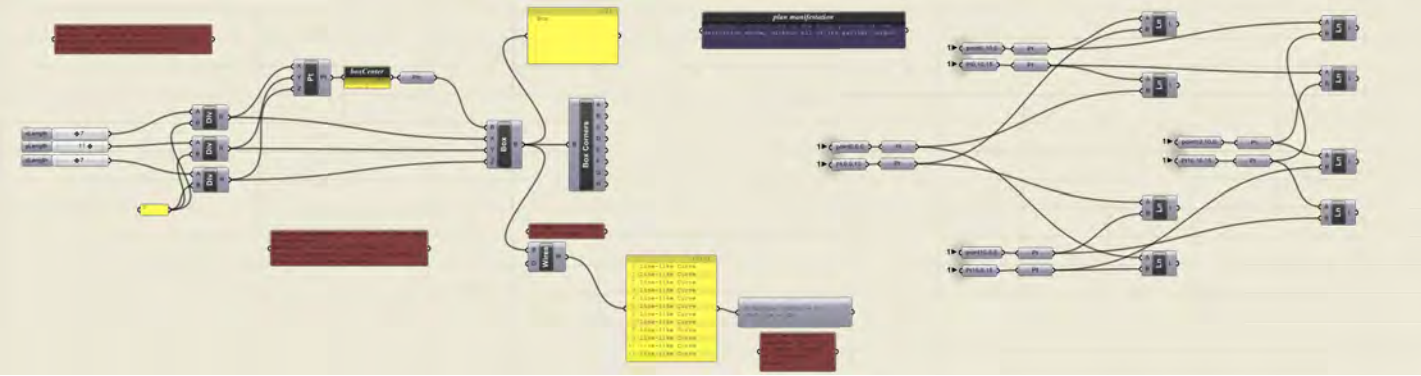


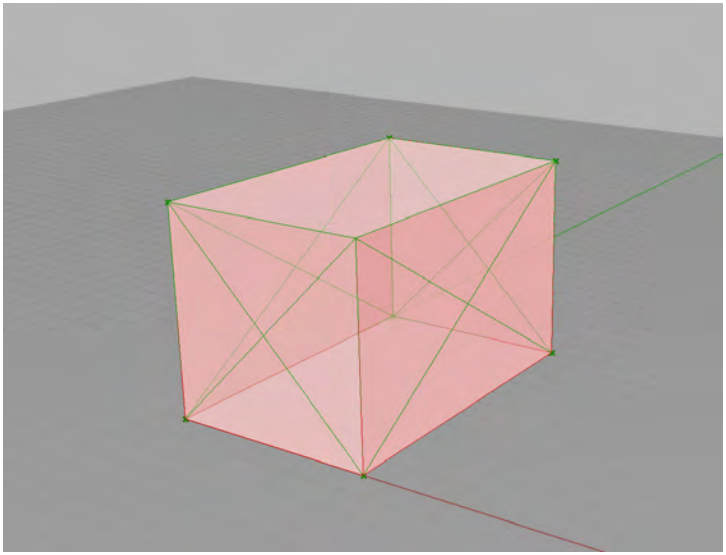
source_files
01_BasicGeometry
31Jan2010_spacialOrientation_KevinHinz.3dm
31Jan2010_spacialOrientation_KevinHinz.ghx

The definition below uses an array of points from Rhino to form the base of the box. Note how Grasshopper interacts with the persistent data in Rhino to develop a parametric series of crosses dependent on the input parameters.





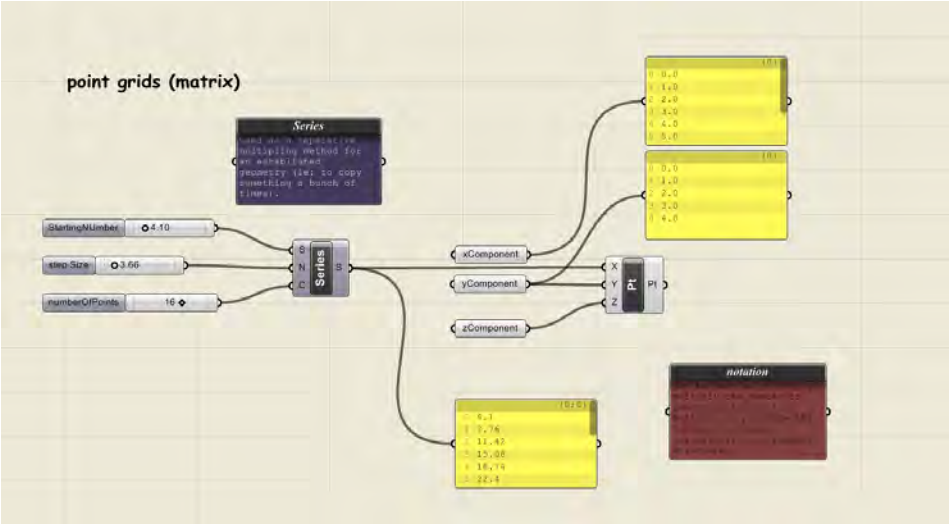




A multitude of component choices provide numerous ways to assemble data. At left, the orientating points lines are shown in green to highlight the box boundaries. Note in the source file how points can both be used to determine the corners of the box and be derived from the box as separate entities. Once other exercises are mastered, the re-used geometry can be reassembled into more complex forms[1]. For example, a simplified version of above could be a column based on a point grid from chapter 3, Matrix and Attractors. Thus, all columns would act together according to the specifications set here.

source_files
01_BasicGeometry
31Jan2010_spacialOrientation_KevinHinz.3dm
31Jan2010_spacialOrientation_KevinHinz.ghx

[1]See also ch. 11 in the *Grasshopper Primer* for more complex applications



Series , Range and Attraction

Here, the operator has an opportunity to investigate the orientation of the regular grid. Again, complexity is kept to a minimum but as in chapter 1, understanding simple organizational patterns is a preemption to understanding more complex algorithmic organization; a strong foundation of knowledge is required for success.

When working with the Grasshopper files provided, use the preview mode to limit visible data for each of the included four definitions: Attractor Lines, Point Lists and Curves, Point Grids (Matrix), and Attractors, point list display.

Attractor Lines explore the Range component to draw a basic grid. Range is used as a uniform multiplier to repeat data in a regular pattern. Its location in the Grasshopper interface (Logic / Sets / Range) is rooted in the description of its function (a logical uniform set of repeatable data inputs). As you may well be aware, Grasshopper components are organized intuitively according to their functional capabilities. Practice and experience will strengthen your navigation skills within the program.

The chapters Point Lists and Curves help layout the order behind data lists and curve lists while Point Grids explore the Series. Series allows the introduction of complexity to the ordering system requiring three inputs: first number, step size and number of values; repetitive generation of data can be easily controlled.

source_files
O2_Matrix_and_Attractors
O3Feb2010_PointMatrix_KevinHinz.3dm
O3Feb2010_PointMatrix_KevinHinz.ghx

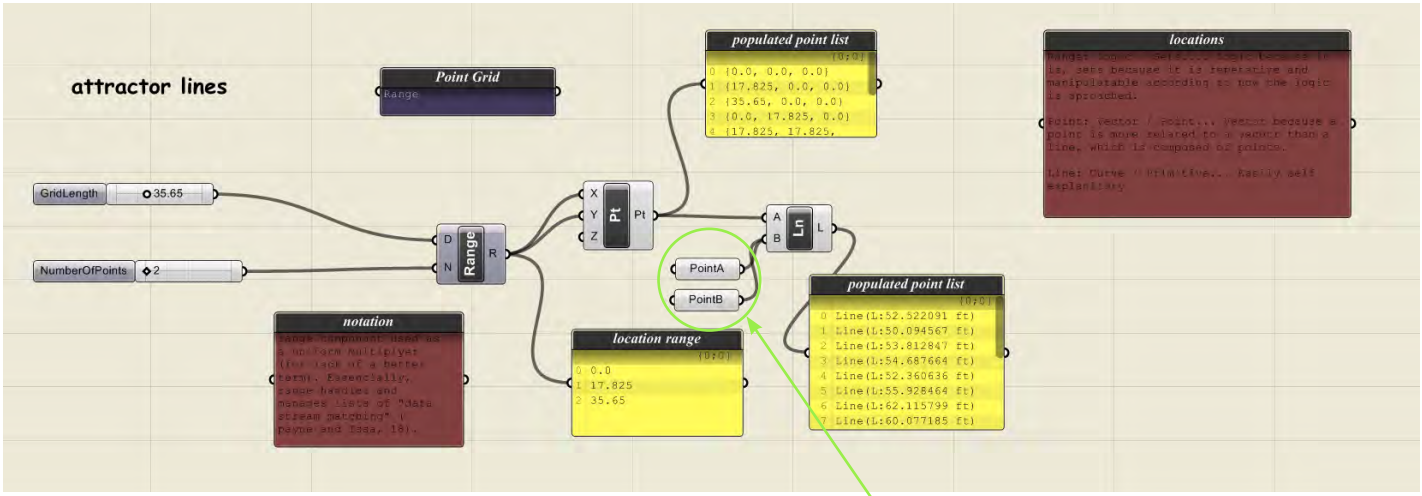


figure 01

persistent data assignment

In figures 01 and 02, volatile data (geometry in Grasshopper) is working congruently with persistent data (geometry in Rhino). When changes are made in Rhino to the assigned point locations, the volatile data is automatically updated to represent the relative relationship.

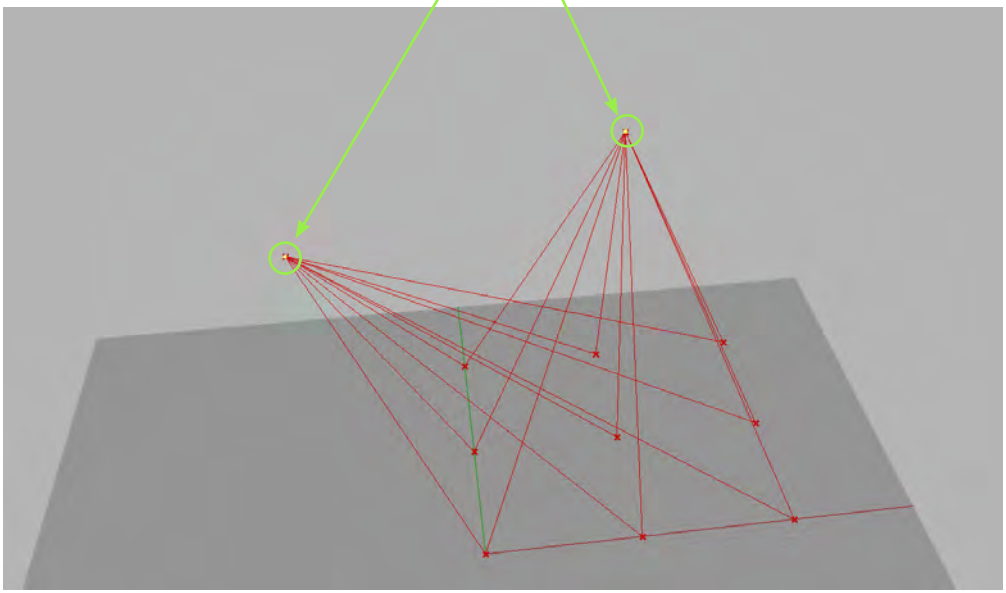
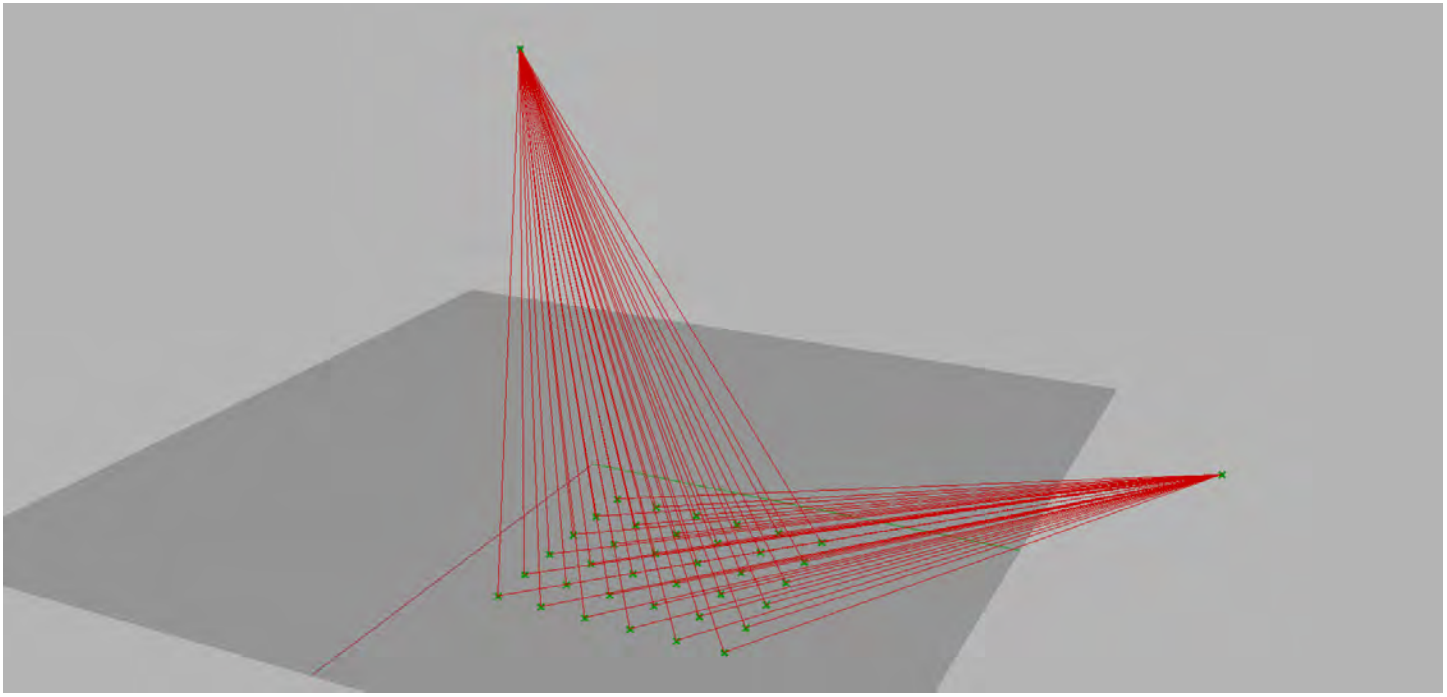
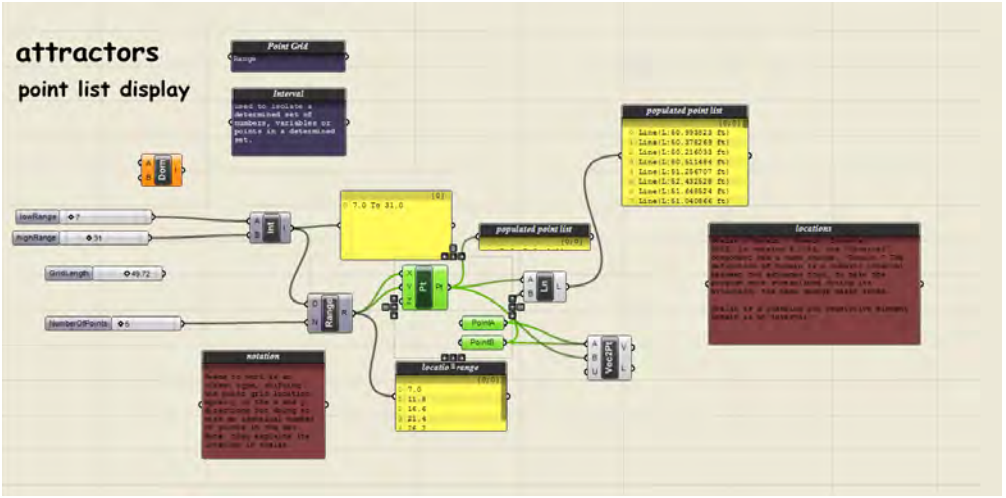


figure 02



Interval is utilized to set up parameters for a subsequent point grid (in green) that exists within a defined domain. Note that a name change has occurred during production of the plug-in: what was “Interval” previously, is now titled “Domain.” Notation and logical explanation of the change can be found in the .ghx file.

This definition combines volatile and persistent data resulting in a frame work that can be utilized as an organizational system for parametrically complex designs.



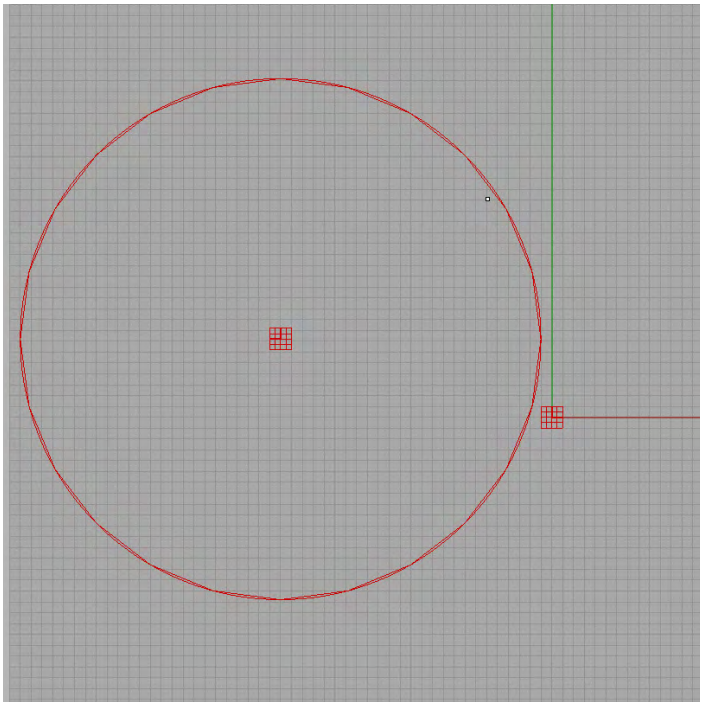
source_files
O2_Matrix_and_Attractors
O3Feb2010_PointMatrix_KevinHinz.3dm
O3Feb2010_PointMatrix_KevinHinz.ghx

Geometric Evaluation

Conditional Statements are one of the most powerful transformation tools available in scripting. Think of it as a set of emergency instructions to be initiated when a predetermined variable reaches a predetermined size. Combine this statement with reoccurring or repetitive computative data and you can often produce unexpected geometry.

The exercises represented here and in chapter 7 of the *Grasshopper Primer* explore input / output relationships: one of the strongest core potentials of parametric design. Combining the Series and Range components from exercise 2 with conditional statements assists design development that is dependent upon object size, physical space location and designer intent.

As before, this exercise is simple; the notated explanation is even less complex. It is up to the user to implement the tools presented.



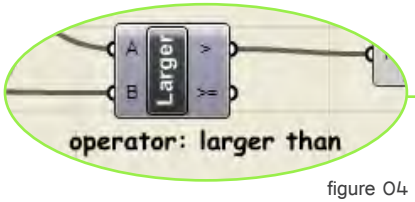


figure 04

source_files
03_ConditionalStatements
05Feb2010_spacialOrientation_KevinHinz.3dm
05Feb2010_spacialOrientation_KevinHinz.ghx

As a conditional statement, Dispatch (figure 03) is used to evaluate incoming data lists. Its location, Logic / List / Dispatch, describes its function. In this example, when the variableX < 25 = false, the polygon is centered at the left plane, consequently, if variableX > 25, the polygon shifts to the right plane (figure 04).

To introduce complexity, the script below draws a polygonal shape that shares its number of sides with the its radius in integers (figure 05). A circle circumscribes the polygon having an similar radius. Note the polygon's radius input and the circle's input come form the same source; each is dependent on the other.

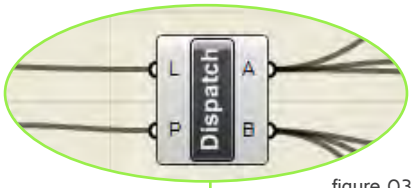
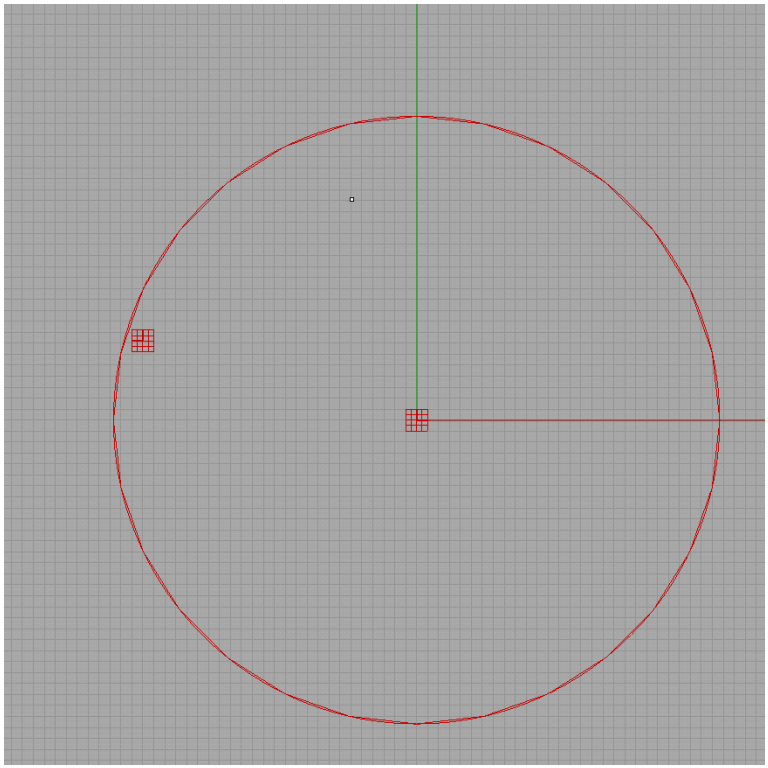
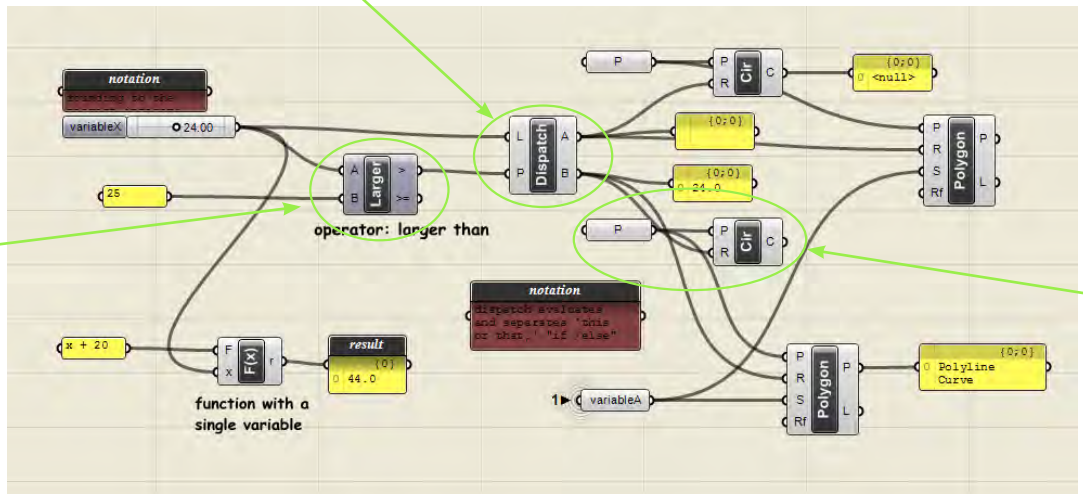


figure 03





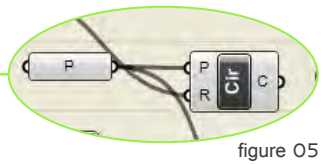
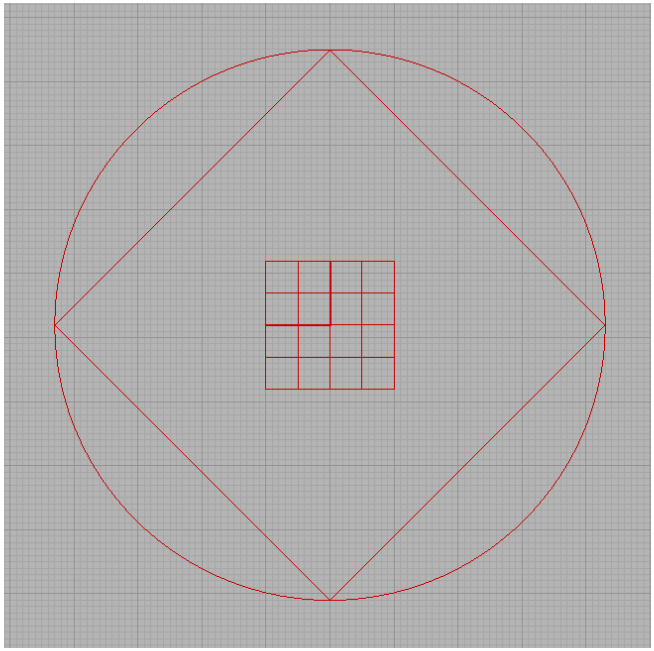
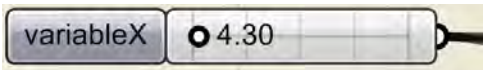
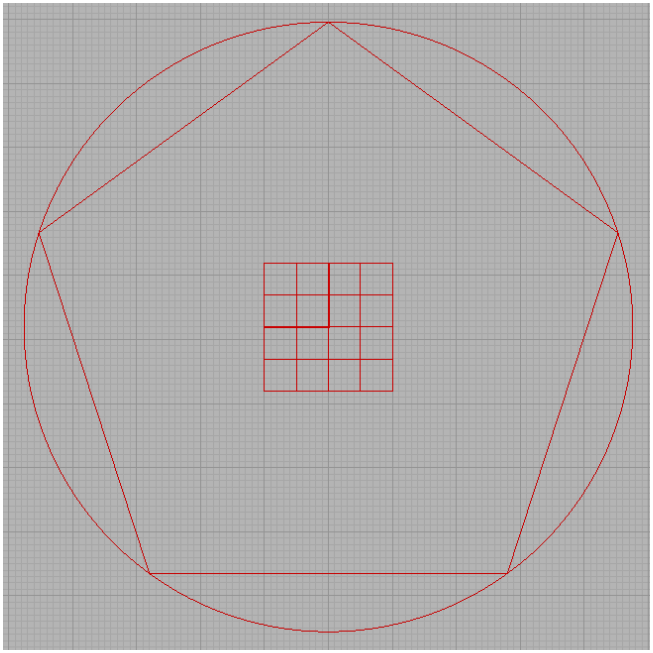


figure 05

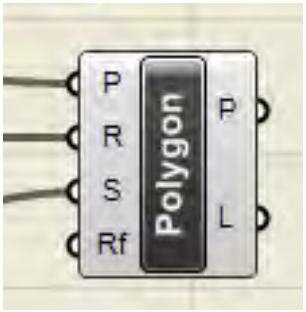






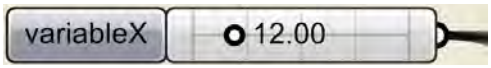
Each component requires specific types of data for input: either integers (whole numbers) or doubles (decimal numbers). You should be familiar with them by now. Because the number of sides on a polygon must be an integer, Grasshopper converts the decimal to the nearest whole number while leaving the radius as determined. Note the images above.

Not utilized here is the Rf input, or corner fillet radius. By combining any one of a number of equations, it would be easy to derive an input value dependent on variableX.



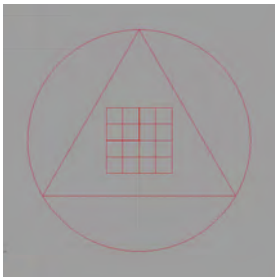
Definitions often produce the most interesting results when geometry is built slowly, upon itself. Simple actions work as instructions in a recipe.

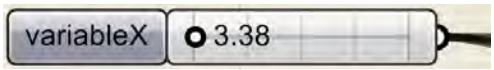
source_files
- 03_ConditionalStatements
  - 05Feb2010_spacialOrientation_KevinHinz.3dm
  - 05Feb2010_spacialOrientation_KevinHinz.ghx

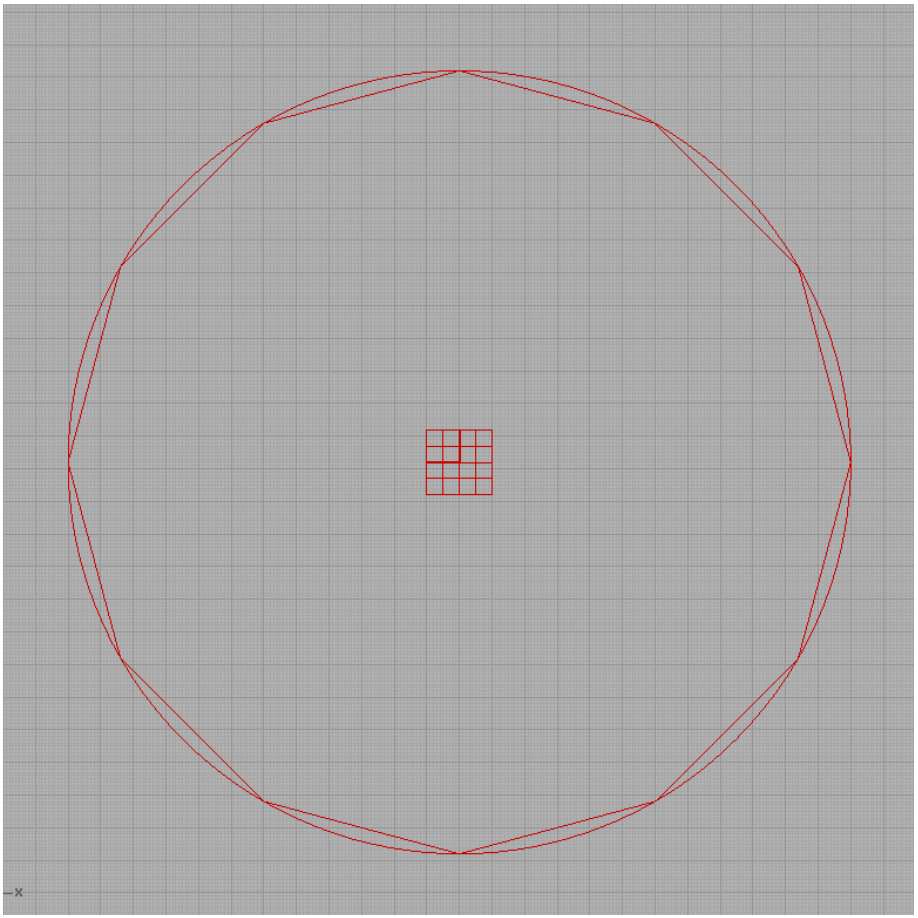


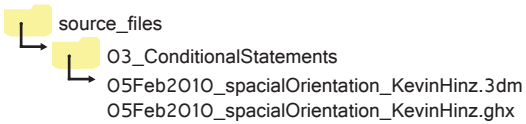
The illustration below shows a triangle having a radius of 3.38 units circumscribed by a circle sharing the same radius. variableX is converted to the nearest integer, 3.

At right, the radius and polygon sides are identical to draw a dodecagon.





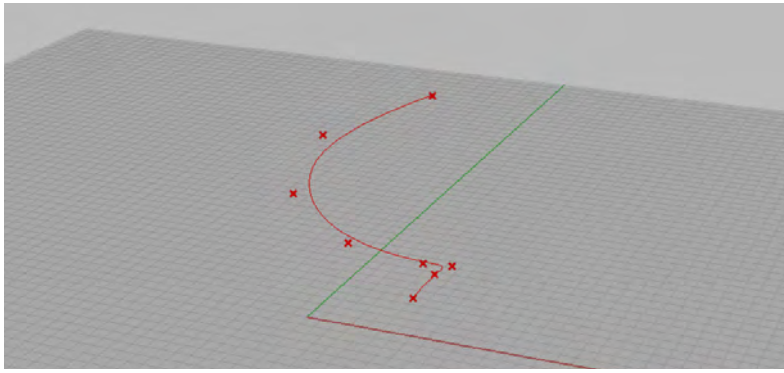


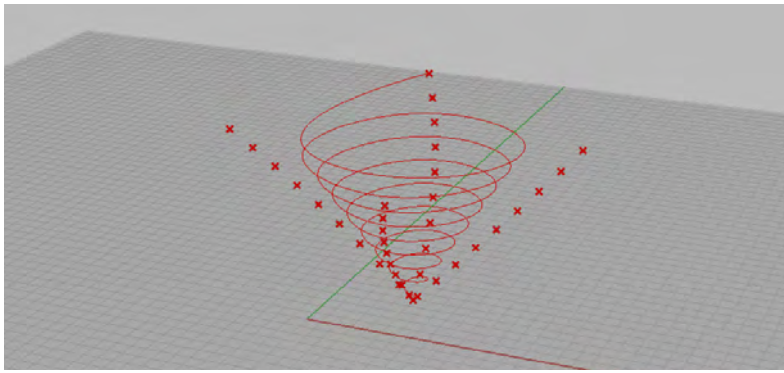

source_files
03_ConditionalStatements
05Feb2010_spacialOrientation_KevinHinz.3dm
05Feb2010_spacialOrientation_KevinHinz.ghx

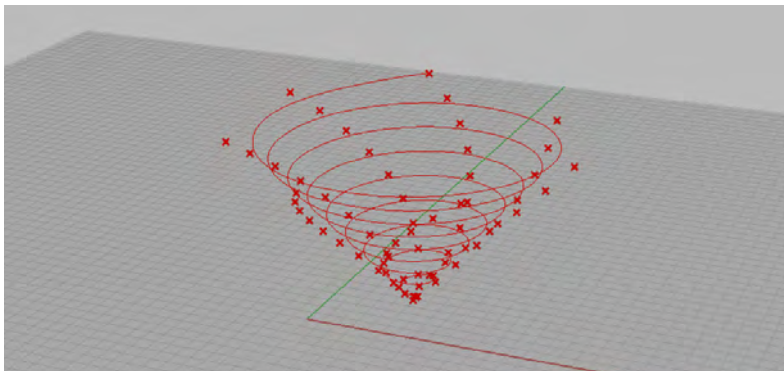
Spiral Curves

Detailed instructions of this exercise can be found in chapter 7.4 of the *Grasshopper Primer* - Functions & Numeric Data. The ability of Grasshopper to transpose and manipulate data into three dimensions is most astounding. This exercise sets up a framework of points that can be used in a variety of different applications. A curve drawn through the point list provides a visual reference to the structure of the list.

Once achieved, the points could be references for more complex geometry or manipulated into an organizing structure with more elegant forms. I suggest exploring this example further by introducing the Pipe or Loft Surface components.

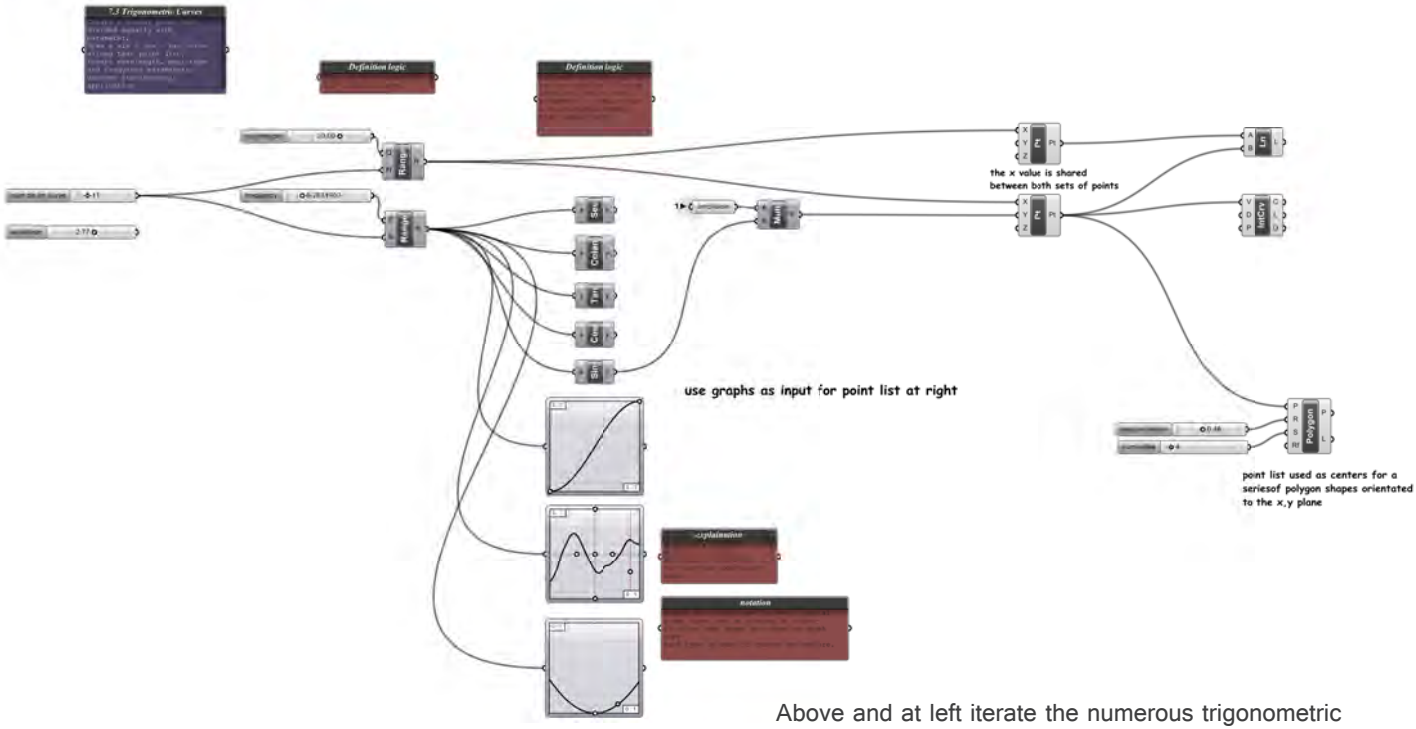


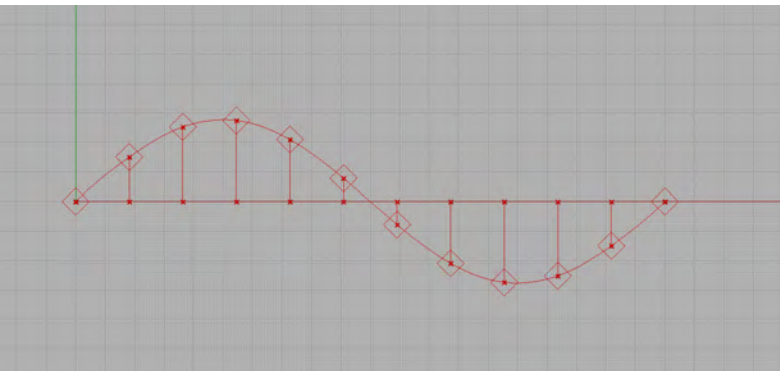




At right, the functions x∗sin(5∗x) and x∗cos(5∗x) are used to generate x and y point coordinates; the same Range input, when introduced as the z input, lifts the spiral into 3 dimensions. Note how the points appear to be organized around the curve (in actuality, the curve is drawn through the point list). As the number of points in within the Range is increased, the list begins to take on a complex structure; depth can be achieved that has potential to manipulate a viewers perspective of the pattern.

source_files
04_CurvesLinesData
13Feb2010_Functions_KevinHinz.3dm
13Feb2010_Functions_KevinHinz.ghx





Above and at left iterate the numerous trigonometric components available in the Grasshopper library. Easy to find and drag onto your canvas, this exercise is best learned by exploring; I would suggest building a similar model as above and manipulating the wavelength, frequency and amplitude parameters to grasp how each operator effects the volatile data. Here you will be introduced to graphs. Graphs can be difficult to understand due to the nature of trigonometry but they can act as a fast and powerful tool to manipulate geometry.

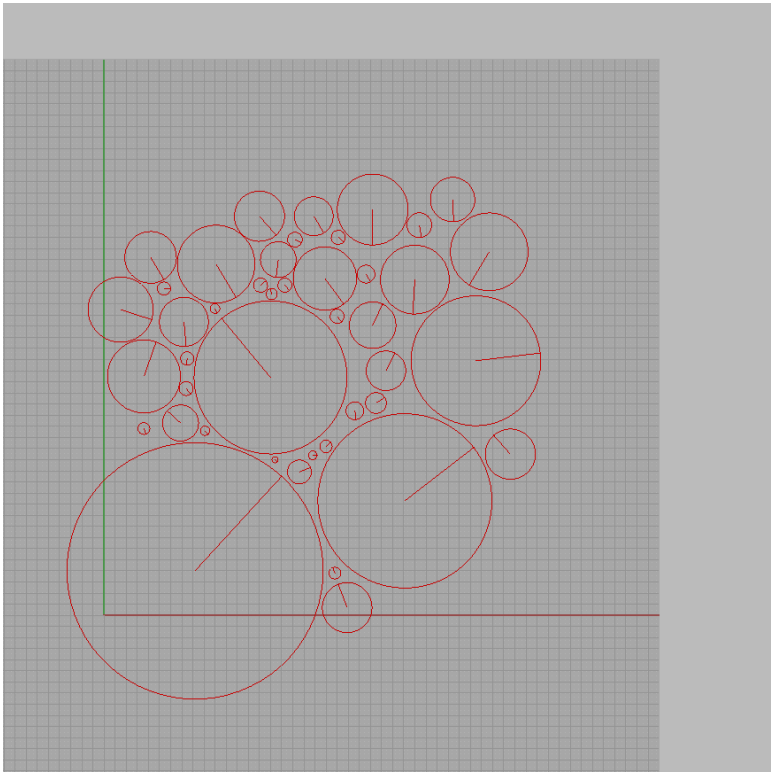
Since the creation of this file, ArcCosine, ArcSine and ArcTangent have been added to the most recent release of Grasshopper; they produce some widely unexpected results as well.

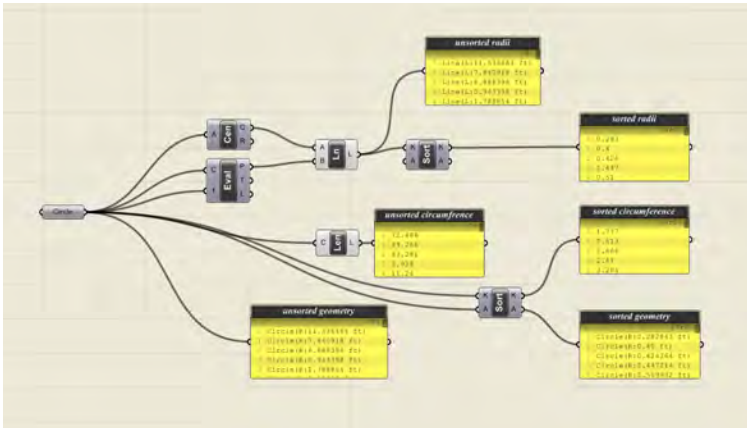
source_files

04_CurvesLinesData

14Feb2010_TrigometricCurves_KevinHinz.3dm

14Feb2010_TrigometricCurves_KevinHinz.ghx



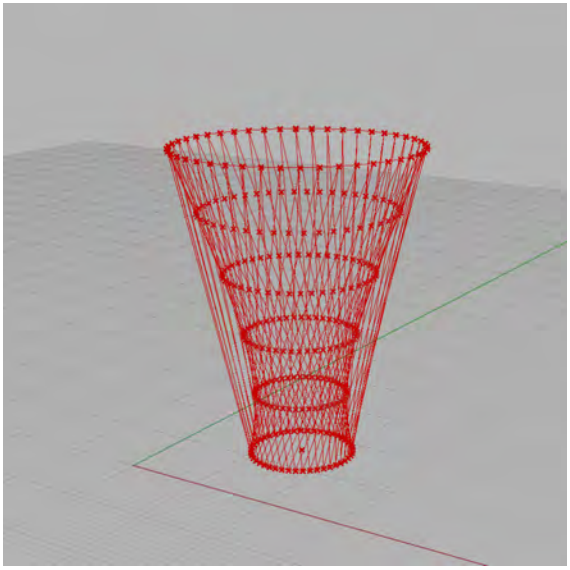


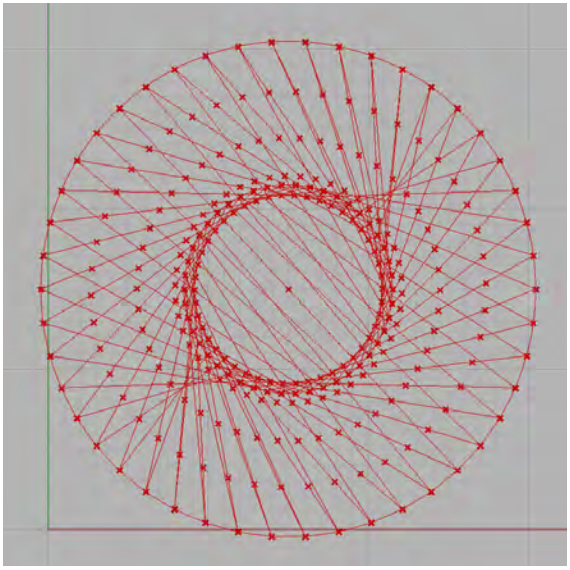
source_files
04_CurvesLinesData
14Feb2010_DataManagment_KevinHinz.3dm
14Feb2010_DataManagment_KevinHinz.ghx

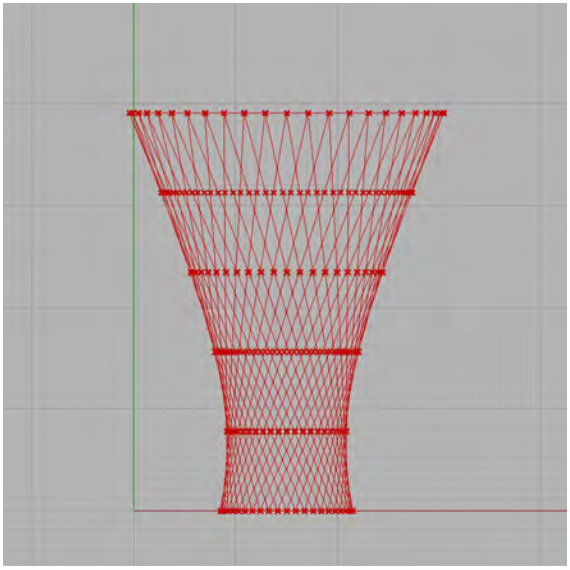
The Sort component is one of the most powerful tools used to manage data in Grasshopper. The component is found in the Logic panel under List, as is most all of Grasshopper's list management components. The definition exemplified orders a group of circles placed randomly with random radii according to radius and circumference. In the source file listed below, you will find additional data manipulation tools such as List, Shift, Cull, Weave, Flatten and many more. Their value will become more evident near the conclusion of this manual.

Authors Payne and Rajaa Issa included an informative drawing created by David Rutten to graphically represent the way Grasshopper manages data[2]; I need not repeat it here. Essentially, data paths are managed much like the trunk of a tree whereas each branch may be subdivided, sheared and / or grafted to additional data structures. As more information is stored, such as point locations and line lengths, paths and numbers are linked and woven together into sub branches, sub-sub branches and so on. Grasshopper makes it easy to view and manipulate data streams as definitions become more complex.

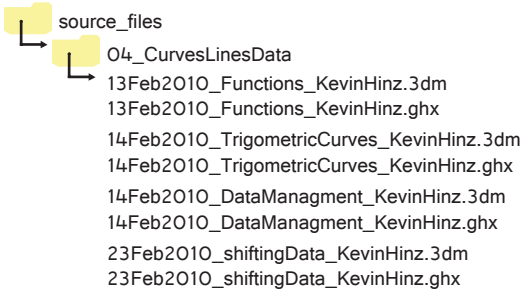
Chapters 8.1, 8.2 and 8.3 of the *Grasshopper Primer* provide very informative definitions for key list management components. Observation is key to interpreting the resulting output. See the source file listed below left for detailed illustrations and observations of data found in the yellow read out panels, such as those seen at right, and the source file (next page) for notation about shifting data. Shift works to take one or more lists and move volatile data according to the input value (in integers); it is as if you simply twist the list as desired.



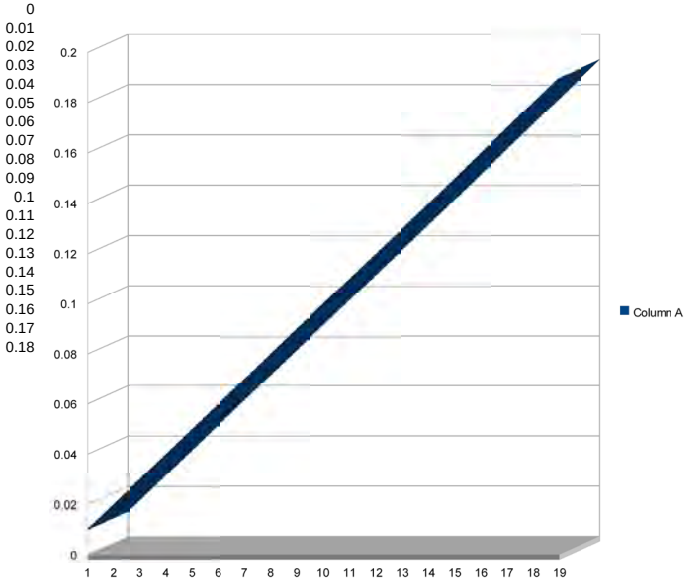


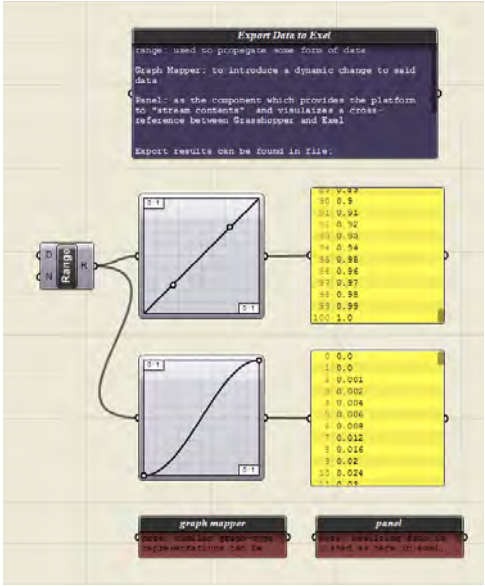


source_files
04_CurvesLinesData
23Feb2010_shiftingData_KevinHinz.3dm
23Feb2010_shiftingData_KevinHinz.ghx


source_files
04_CurvesLinesData
13Feb2010_Functions_KevinHinz.3dm
13Feb2010_Functions_KevinHinz.ghx
14Feb2010_TrigometricCurves_KevinHinz.3dm
14Feb2010_TrigometricCurves_KevinHinz.ghx
14Feb2010_DataManagment_KevinHinz.3dm
14Feb2010_DataManagment_KevinHinz.ghx
23Feb2010_shiftingData_KevinHinz.3dm
23Feb2010_shiftingData_KevinHinz.ghx

[2]See chapter 8 in the *Grasshopper Primer*





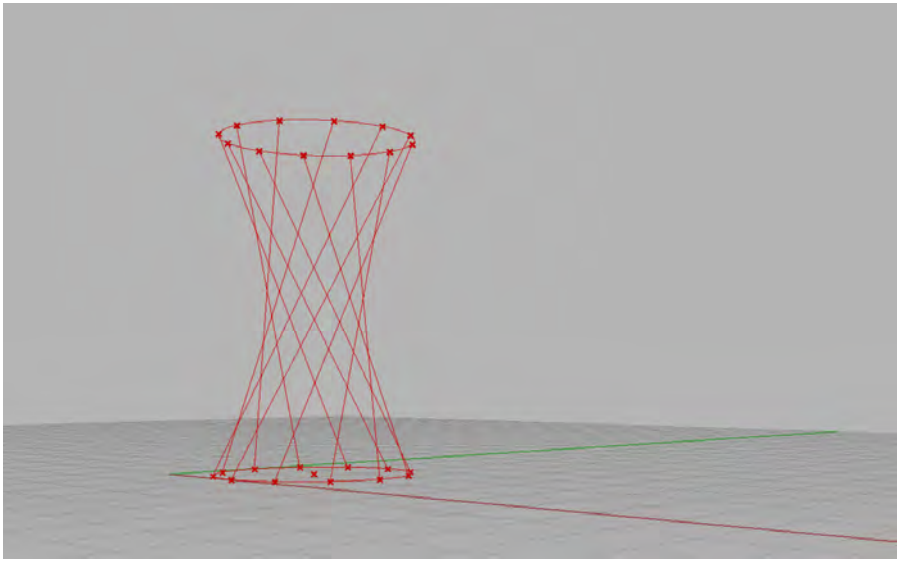
From Grasshopper to Excel

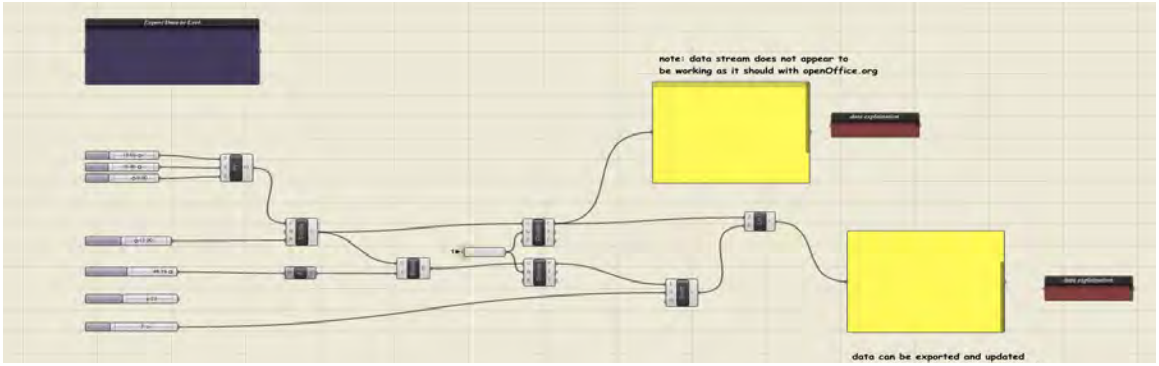
Data produced in Grasshopper can often times be analyzed and utilized by other software packages such as Microsoft's Excel. Spread sheets are very functional documents, able to tabulate material quantities, produce project estimates and calculate costs. In addition, data exports, such as x, y, z coordinates, can be read by robotic arms to place materials. Original source files are listed below however, I recommend using the *Grasshopper Primer* to build your own from scratch; the exercise is simple. If you are looking for more challenging tutorials, look to Zubin Khabazi's *Algorithmic Modeling*, chapter 8 Fabrication.

In the upper right image, this page, you will see a graph mapper attached to a display panel; note the exported data at left from the .csv stream file. Near right you are the line lengths and start point locations for the columnar structure on the next page. My experiences with exporting data were more successful to Microsoft's Excel than the OpenOffice's Calc. The interface appears more friendly in Microsoft and the linked files readily update and reload with ease.

source_files
05_DataCommunication
24Feb2010_ExportingData_KevinHinz.3dm
24Feb2010_ExportingData_KevinHinz.ghx[3]

Line(L:52.863312 in)	{28.3	15	0.0}
Line(L:52.863313 in)	{27.807	18.588	0.0}
Line(L:52.863312 in)	{26.364	21.91	0.0}
Line(L:52.863313 in)	{24.078	24.72	0.0}
Line(L:52.863312 in)	{21.119	26.809	0.0}
Line(L:52.863313 in)	{17.706	28.022	0.0}
Line(L:52.863313 in)	{14.092	28.269	0.0}
Line(L:52.863313 in)	{10.546	27.532	0.0}
Line(L:52.863313 in)	{7.33	25.866	0.0}
Line(L:52.863313 in)	{4.683	23.393	0.0}
Line(L:52.863312 in)	{2.801	20.299	0.0}
Line(L:52.863312 in)	{1.824	16.811	0.0}
Line(L:52.863312 in)	{1.824	13.189	0.0}
Line(L:52.863313 in)	{2.801	9.701	0.0}
Line(L:52.863312 in)	{4.683	6.607	0.0}
Line(L:52.863312 in)	{7.33	4.134	0.0}
Line(L:52.863313 in)	{10.546	2.468	0.0}
Line(L:52.863313 in)	{14.092	1.731	0.0}
Line(L:52.863313 in)	{17.706	1.978	0.0}
Line(L:52.863313 in)	{21.119	3.191	0.0}
Line(L:52.863313 in)	{24.078	5.28	0.0}
Line(L:52.863313 in)	{26.364	8.09	0.0}
Line(L:52.863313 in)	{27.807	11.412	0.0}



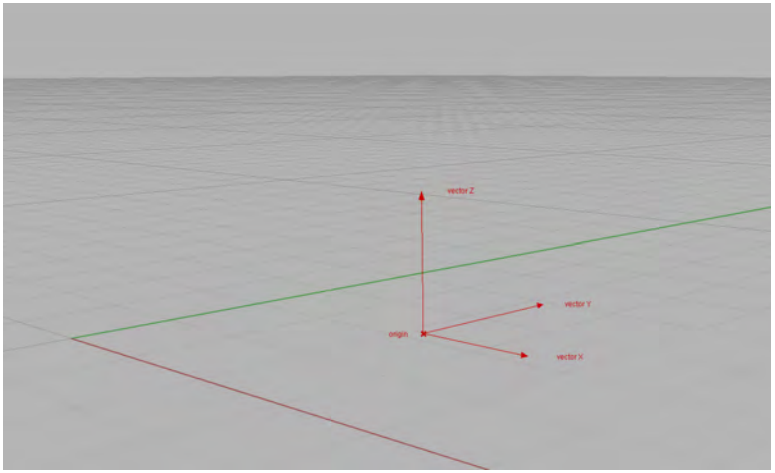


The columnar structure is used to iterate the export process; better resolution is found in the source file but I recommend completing the process on your own. Essentially, this exercise explores the use of spreadsheets. The only thing new in Grasshopper is to choose the stream content selection with a right-mouse over the panel display: that easy.


source_files
05_DataCommunication
24Feb2010_ExportingData_KevinHinz.3dm
24Feb2010_ExportingData_KevinHinz.ghx

[3]see also the .csv files for streamed data output

Distance Factors

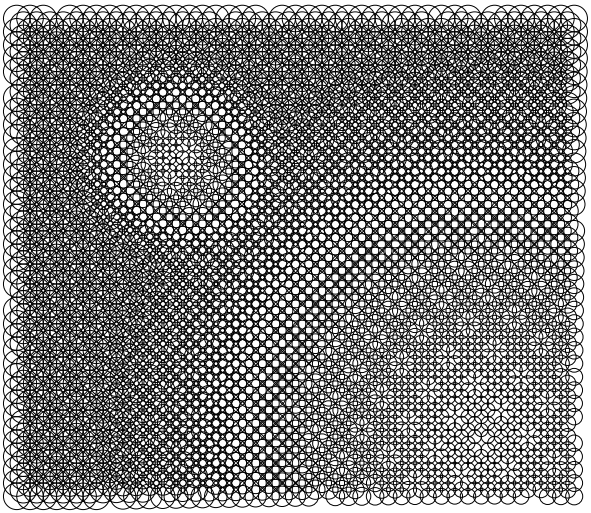


Understanding vector is crucial to building most anything with Grasshopper. Vectors have an origin, orientation and a magnitude, three things needed to create parametric proximity relationships.
The image at right, below, illustrates an attractor script build from a point grid and series of circles. Each radius is adjusted according to the distance between a circles center point and two chosen reaction points". The number of reaction points is easily modifiable as is the reaction point itself.
For example, if one wanted a curve to be inflectional geometry, you would need to establish a point on that curve to measure to. Thus, you would need a closest point on curve component (Crv CP) shown below.

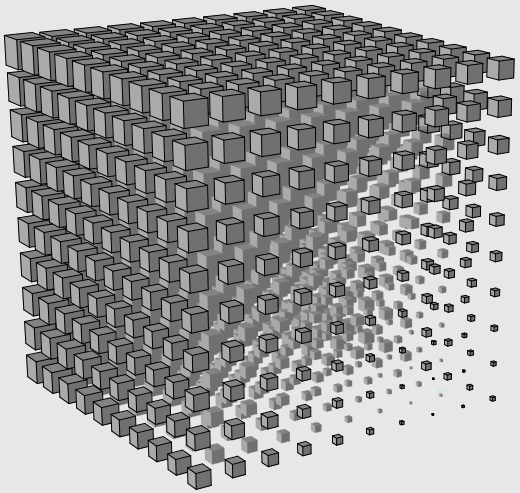
In either case, the logic behind the attractor script is simple:

- -create a point grid to draw some modifiable shape
- -assign some form of attractor
- -evaluate a distance to the attractor
- -introduce a user defined variable to control the rate of influence
- -insert a minimum or maximum cut off to control shape size
- -draw shape according to determined distance parameters

Attractor scripts can be powerful tools to create gradient geometrical forms; none would be possible without the vector: origin, orientation and magnitude.

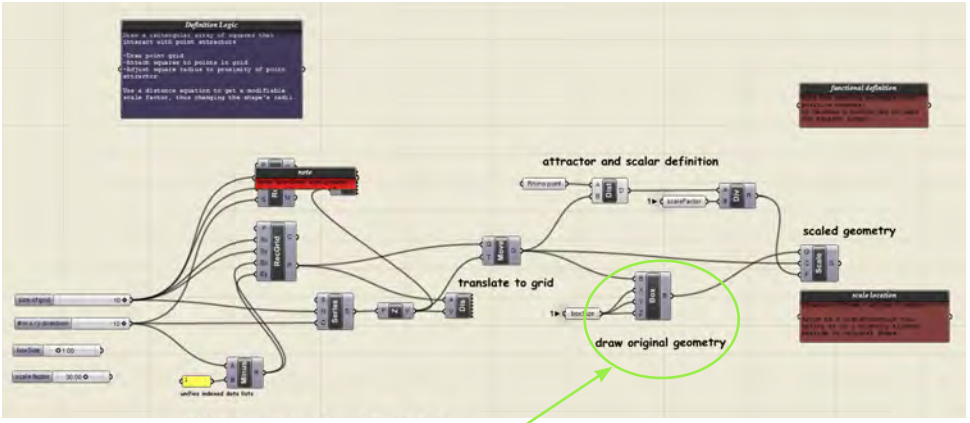


source_files
06_VectorBasics
24Feb2010_Vectors101_KevinHinz.3dm
24Feb2010_Vectors101_KevinHinz.ghx

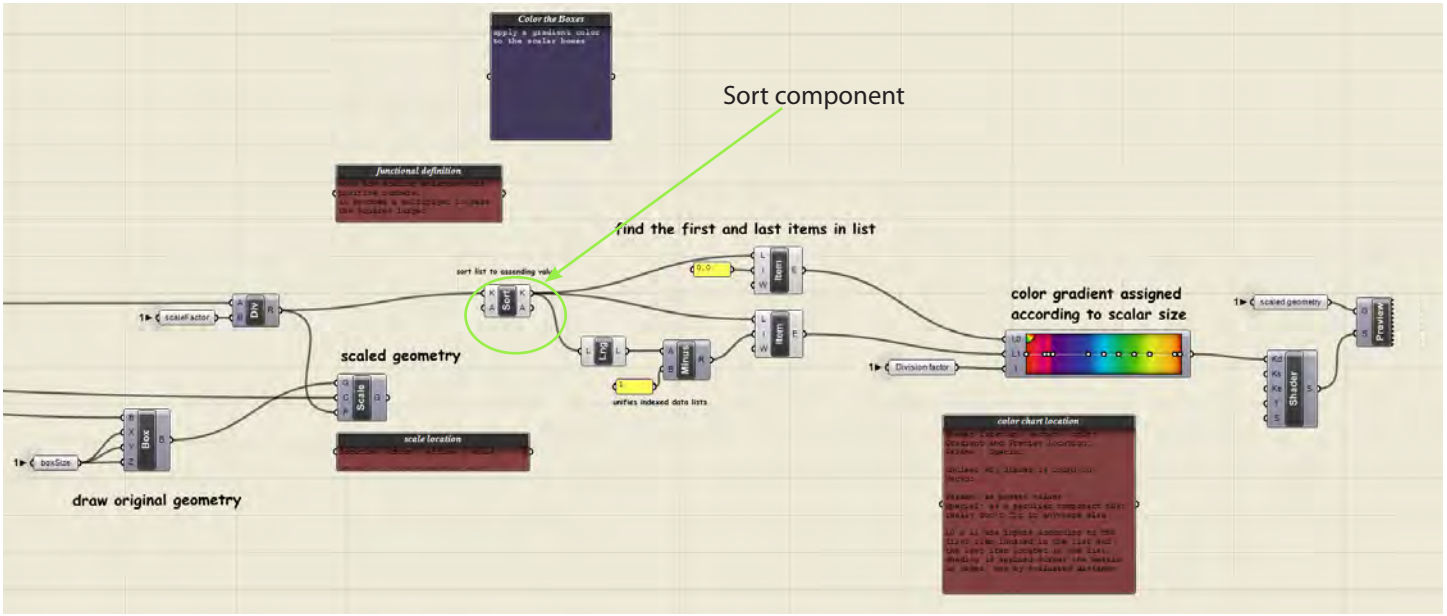


The example above shows how an attractor can be applied to a three dimensional form. While nearly identical to the previous exercise, the script to the right draws geometry first, then magnifies it according to the distance between its center and the attractor[5].

Here, scalar computation is also applied to object coloring.

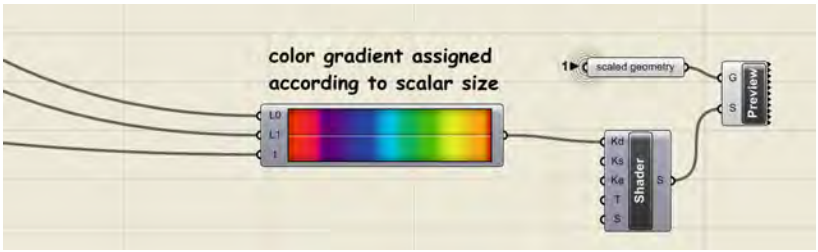


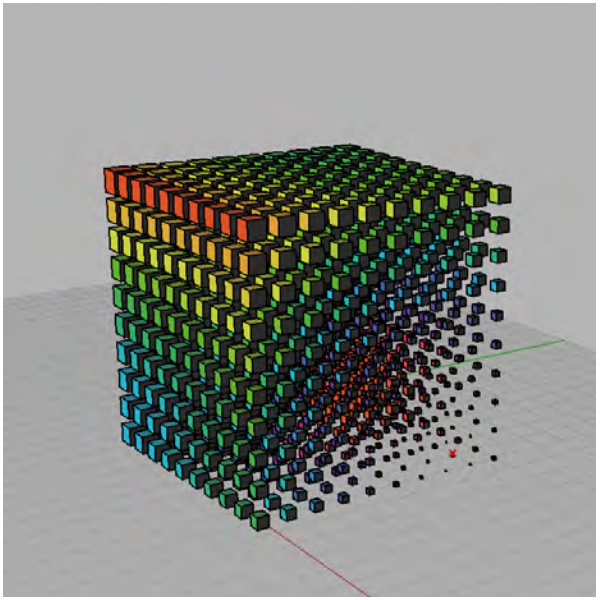
source_files
06_VectorBasics
06Mar2010_ScalarColor_KevinHinz.3dm
06Mar2010_ScalarColor_KevinHinz.ghx



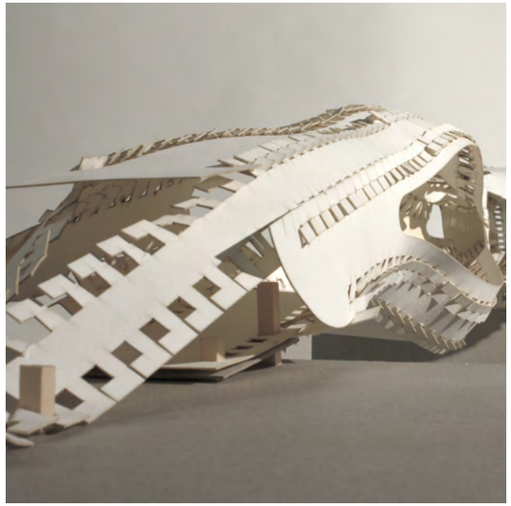
The Sort component becomes critical when applying the color gradient. Input items represent the scaled distances between matrix points and the attractor point; the values are ordered according to its position in the matrix. Because we want the color to be graduated according to distance, values must be reordered. If Sort is removed from the equation, and the first and last list items inputted directly into the shader, only the cubes on the extremities show a color gradient; the data is ordered in very much in a different way.

Note that because the color gradient is determined by the attractor, and not the cube, it operates as its own system.



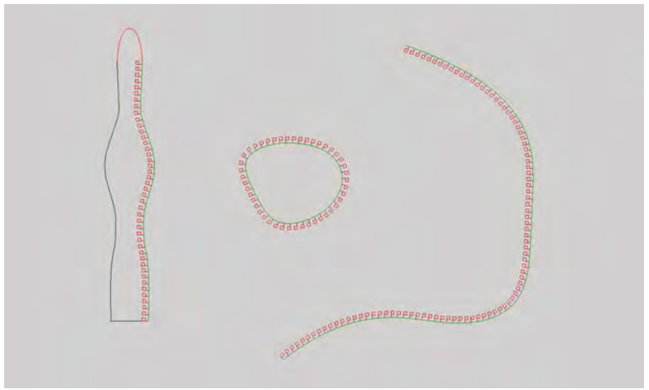


source_files
06_VectorBasics
06Mar2010_ScalarColor_KevinHinz.3dm
06Mar2010_ScalarColor_KevinHinz.ghx



Computation and Material

The bridge model above is design by conventional means but manufactured with scripting. A simple connection (below) was uncovered during material exploration; Grasshopper was used to apply the zipper connection to irregular surface edges and develop refined curves for an elegant structure flow. Conventional modeling tested the design, digital modeling clarified the plan and scripting applied the geometrical framework to polish the layout and simplify the construction processes. The final application and production of the model would be intensely laborious.

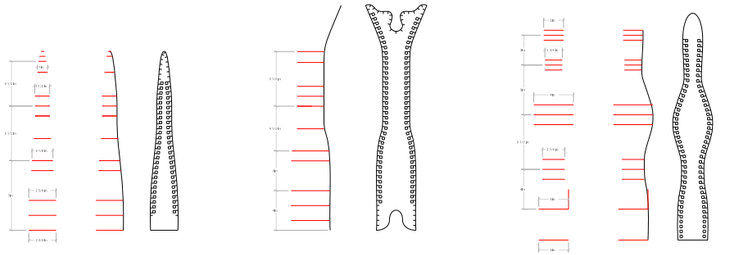


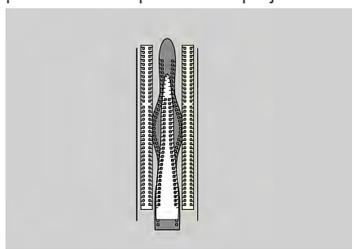
Following the connection discovery, manual iterations exploring width and degree of curvature (in bridge components) produced variations in bridge formation. Final iterations were used as templates for point parameters then refined with Grasshopper's Bezier Spline component.

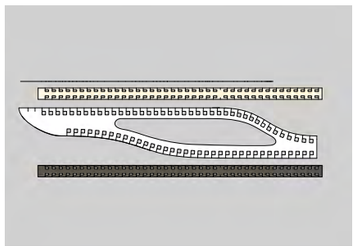
The real power of computation is demonstrated at right. Highlighted in green, three very different curves are used to demonstrate the script's versatility. A 1/4" square is attached to the end of a 1/4" line which is orientated perpendicular to the desired curve[6].

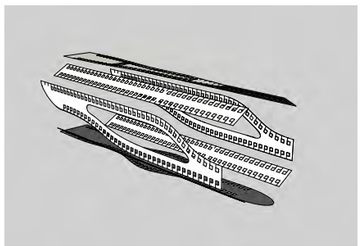
Where it would take countless hours of applying geometric theory, once built, the script will accomplish in seconds.

Final bridge component parameters were documented for explanation purposes but design approximations are completed manually. The project demonstrates how conventional modeling technics can be combined with digital modeling and scripting to produce a comprehensive project.









source_files
- 06_VectorBasics
  - zipperConnection
    - 22Feb2010_BridgeConnection_KevinHinz.3dm
    - 22Feb2010_BridgeConnection_KevinHinz.ghx

[6] see the source file and page 34 for a detailed account of the logic

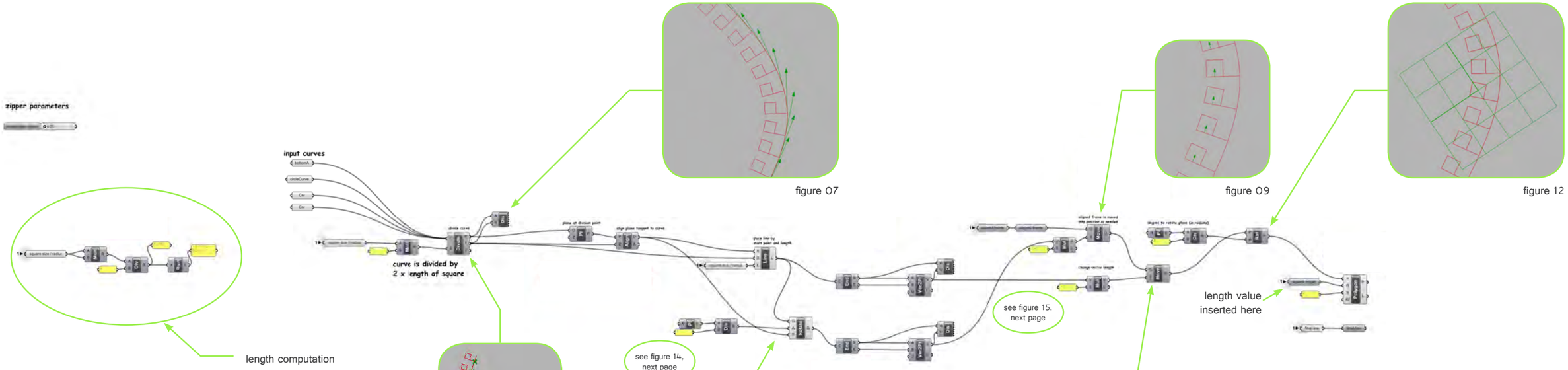


figure 07

figure 09

figure 12

figure 06

figure 08

figure 10

figure 11

figure 06: divide any curve according to length between divisions
Divide: Logic - Curve - Division

figure 07: display tangent vector at division point
Display: Vector - Vector

figure 08: rotate above vector to perpendicular
Rotate: XForm - Euclidian

figure 09: get vector used to move frame for polygon center
Move: XForm - Euclidian

figure 10: moved frame

figure 11: vector showing frame center

figure 12: final rotated frame

A simple series of steps organize the application of this “zipper”-like definition: divide a line according to length / get a line perpendicular to the tangent at each division point / place a square at the end of the perpendicular line. Placing the square proves to be the most complex; Grasshopper draws polygons from their center, in a diagonal rather than from the center by their side edge. More than 50% of the script (and 75% of the time it took to write) is dedicated to drawing the square polygon. Because Grasshopper is organized logically, components are easy to find, if you know what you want to do and what the mathematical theory is behind the transformation. Note the length computation at the beginning of the script; the definition expresses the Pythagorean Theorem and how the location (listed under the figure descriptions at left) is directly related to how the component functions.

source_files
- 06_VectorBasics
  - zipperConnection
    - 22Feb2010_BridgeConnection_KevinHinz.3dm
    - 22Feb2010_BridgeConnection_KevinHinz.ghx

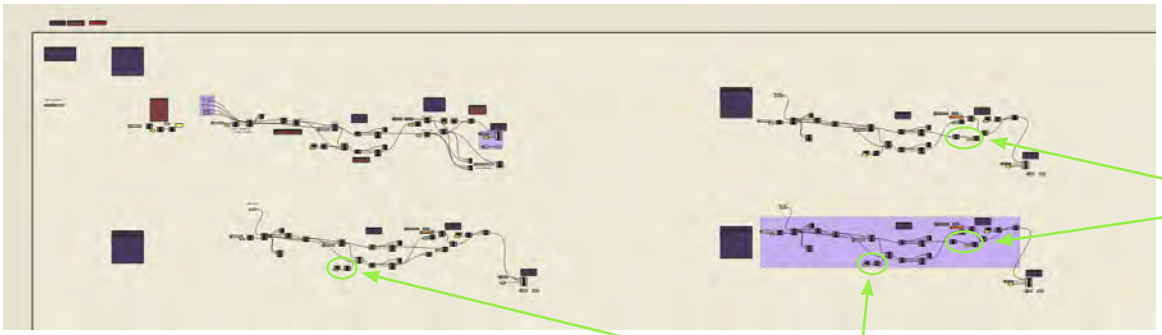


figure 13

Figure 13, found in the source file listed below, illustrates how the definition was transformed according to the script's application. Small changes, such as the rotation reversal shown in figure 14, modify the zippers' orientation according to the designer's desire (shown above at right). Note that Grasshopper requires the angle of rotation to be in Radians, hence, the Pi / 2 (or Pi / -2 in this example) to rotate the vector 90 degrees. Figure 15 illustrates a vector reversal for the two definitions above at left.

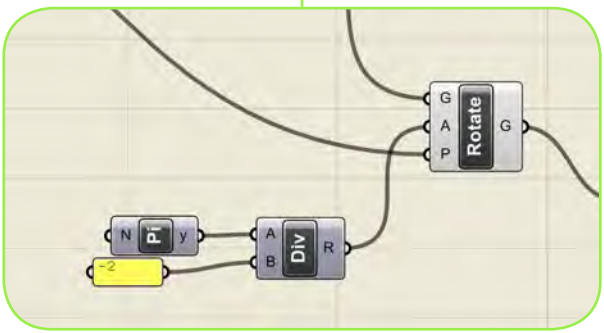


figure 14

One each of the 4 definitions above are used to apply the script left or right and above or below.

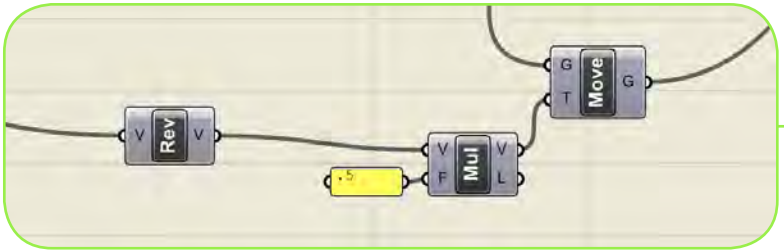
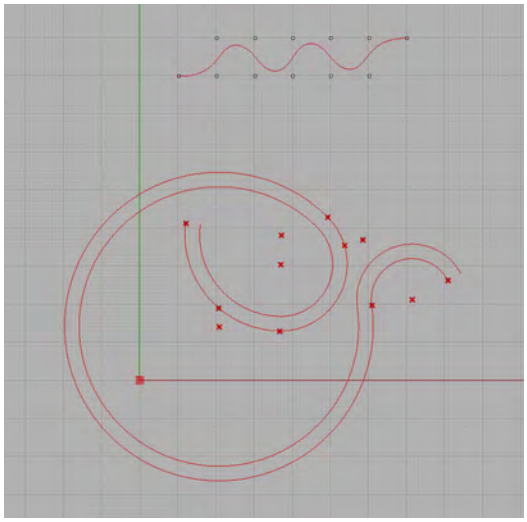


figure 15

source_files
06_VectorBasics
zipperConnection
22Feb2010_BridgeConnection_KevinHinz.3dm
22Feb2010_BridgeConnection_KevinHinz.ghx


source_files
06_VectorBasics
24Feb2010_Vectors101_KevinHinz.3dm
24Feb2010_Vectors101_KevinHinz.ghx
06Mar2010_ScalarColor_KevinHinz.3dm
06Mar2010_ScalarColor_KevinHinz.ghx
zipperConnection
22Feb2010_BridgeConnection_KevinHinz.3dm
22Feb2010_BridgeConnection_KevinHinz.ghx

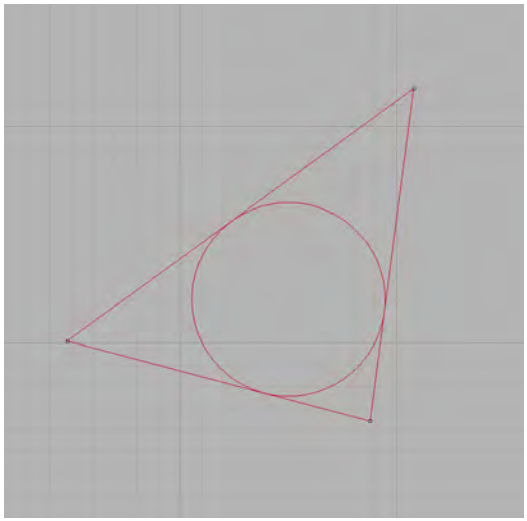
[4]see ch 2 of this publication for additional matrix and attractor information
[5]see the source file for detailed explanation of how the components operate
[6]see the source file and page 34 for a detailed account of the logic



Curve Types

You will find a working model of curve types and degree tests in the source file listed below. Because of its mathematical complexity, I will suggest researching more complete information about NURBS curves found at http://en.wikipedia.org/wiki/NURBS.

At right, top, are two drawings; above, you will find a simple curve with a varying slide parameter attached. Further inspection will shed light onto one way curve structure can be manipulated. The lower, more fanciful curve, is constructed from a series of arcs. Because a thorough explanation is provided, I recommend seeing the *Grasshopper Primer*, pg. 67 - chapter 10, for detailed description of available curve types in Grasshopper.



At right, lower, is a testament to intuitive navigation within the Grasshopper interface. Working on a challenge presented by a classmate, we struggled in our attempts to determine the geometrical relationships required to inscribe a circle within any triangle. Writing the script in Visual Basic presented the greatest challenge because the logic required information regarding how the data was collected. In other words, we needed to know which edge was orientated which way to make the set of instructions work. Grasshopper calculated this for us. Soon after trying to build the script using primitive geometric relationships, I uncovered the component InCircle, located at Curve / Primitive / InCircle.

Organization of Grasshopper components can be understood if the analysis of the final geometrical relationship is understood; the two go hand in hand.

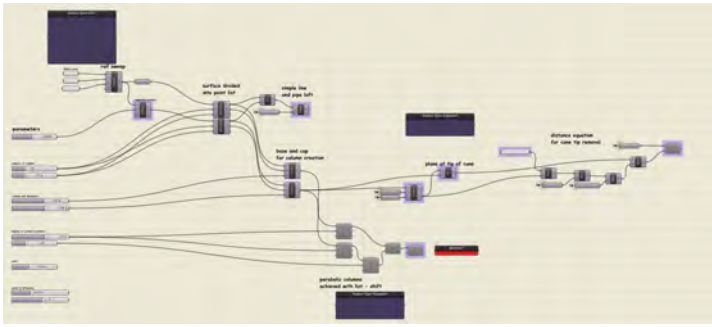
source_files
07_Curves_and_Surfaces
06Mar2010_curveTypes_KevinHinz.3dm
06Mar2010_curveTypes_KevinHinz.ghx

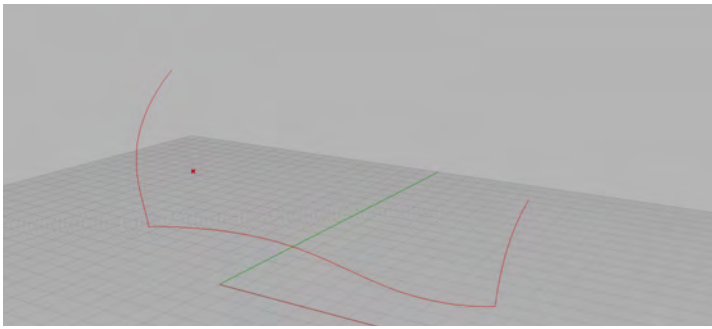
Surface Manipulations

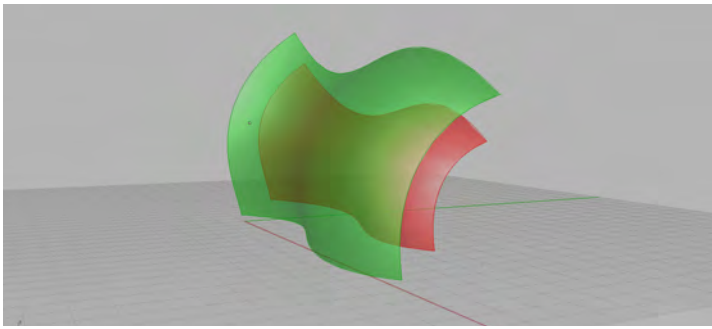
At first glance, the script at right appears easy to follow and understand. However, the example quickly becomes complex to the point of freezing the CPU used to construct it. Strangely enough, perhaps due to the way Rhino and Grasshopper compute data, the definition does not overwhelm the system when assembled from scratch; problems arise only when tweaking parameters and testing iterations after the script is built. Have patience when investigating this definition.

Beginning with three double curves, a surface is lofted between before being offset. Examples of a columnar structure, similar to a peristyle, can be found in the lower portion of the script. By combining a point grid (see following page) with a complex curve, what is typically a simple organized structural system, can easily become something much more fluid and dynamic.

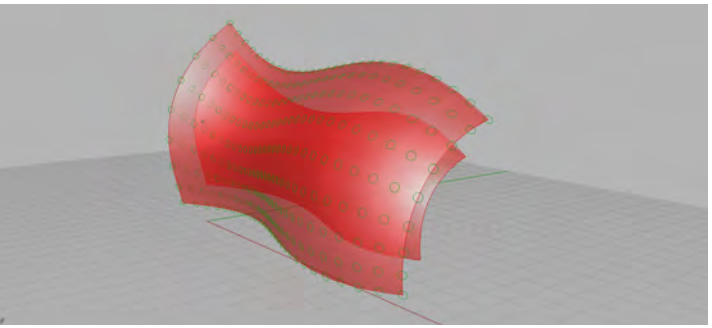
The aim of experimentation outlined in this chapter and the next, point towards a reintroduction of complexity and movement into architecture: two interjections that have been shunned since the beginning of the modern movement.



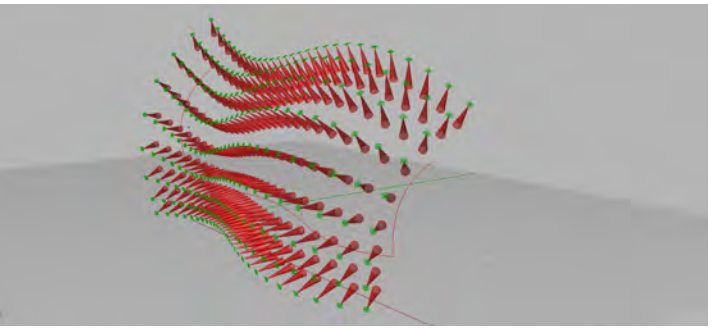




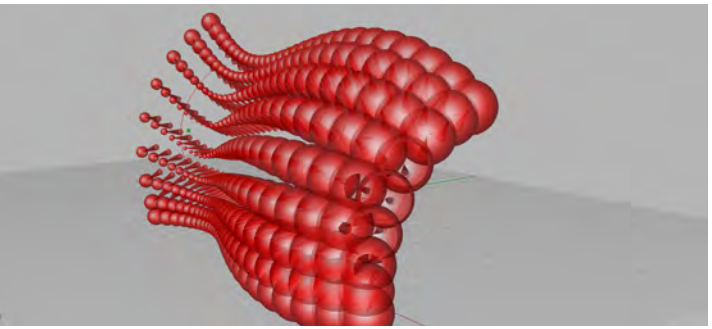
source_files
07_Curves_and_Surfaces
06Mar2010_surfaceRelationships_KevinHinz.3dm
06Mar2010_surfaceRelationships_KevinHinz.ghx



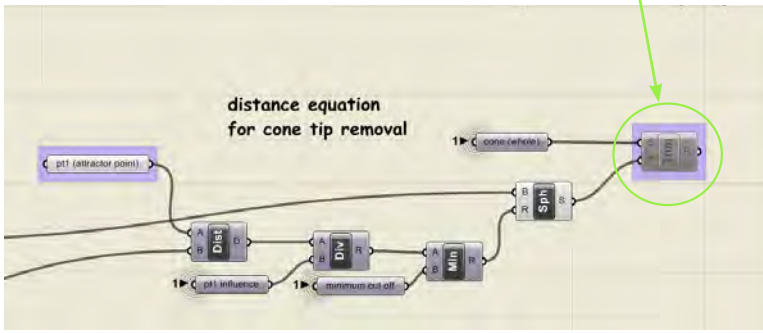
At left, above, you will see the application of a point grid with a radius controlled circle applied. Form here, the script found in the source file takes two directions: one to design a set of columns between the complex surface, the other to articulate a uniquely texture surface. Follow the illustrations (and the majority of the definition) for a combination of the exercises outlined in the chapters presented so far.



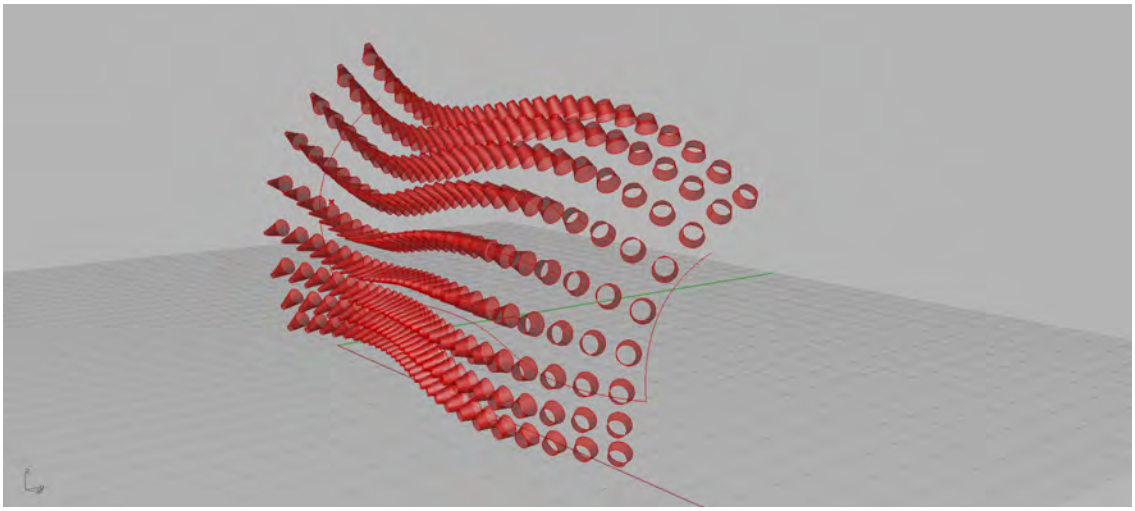
At left, center, cones are placed on the point grid. On each cone tip, a plan is used to set up for the spheres shown at left, below. Looking back to chapter 2, this script copies and pastes (ctrl + C , ctrl + V) the earlier attractor script to combine and build the definition's complexity.



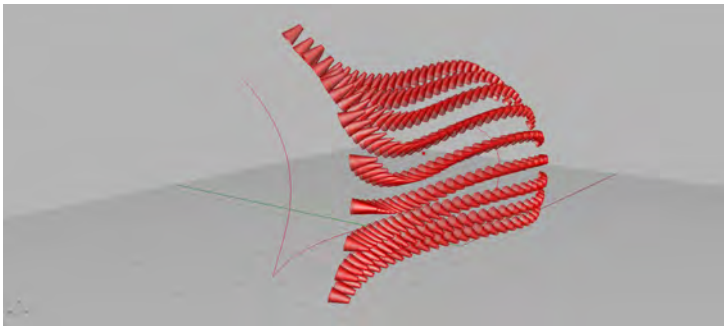
As mentioned earlier, the computation soon challenges power of standard computers thus, the final component is disabled.



source_files
07_Curves_and_Surfaces
06Mar2010_surfaceRelationships_KevinHinz.3dm
06Mar2010_surfaceRelationships_KevinHinz.ghx

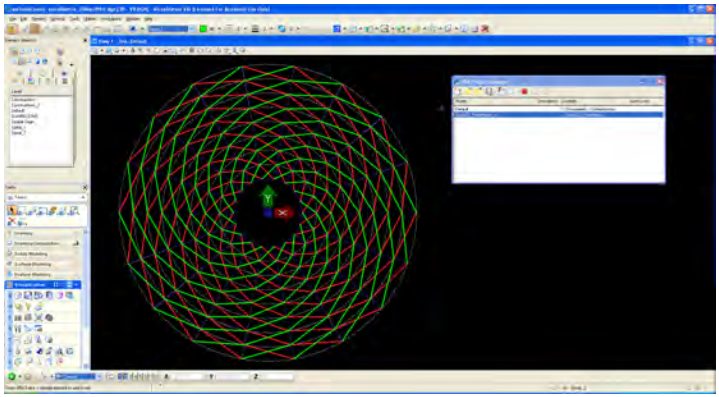


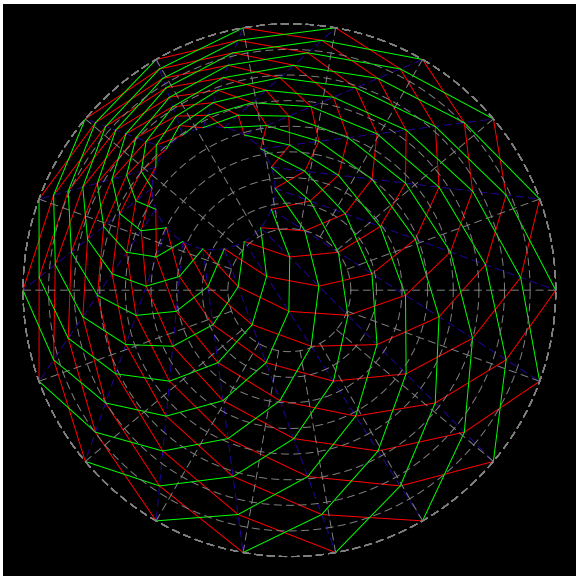
Once the final Trim component is enabled (and the computer gets done thinking...) the individual cones/ unique shape is revealed.



Options to further develop and complete this design could include placing the cones atop the columns from earlier or even joining the cones to your columns and baking in Rhino before meshing, flattening, cutting and constructing a paper model of the design. Evolution of the idea is only as large as you take the thought.

source_files
07_Curves_and_Surfaces
06Mar2010_curveTypes_KevinHinz.3dm
06Mar2010_curveTypes_KevinHinz.ghx
06Mar2010_surfaceRelationships_KevinHinz.3dm
06Mar2010_surfaceRelationships_KevinHinz.ghx





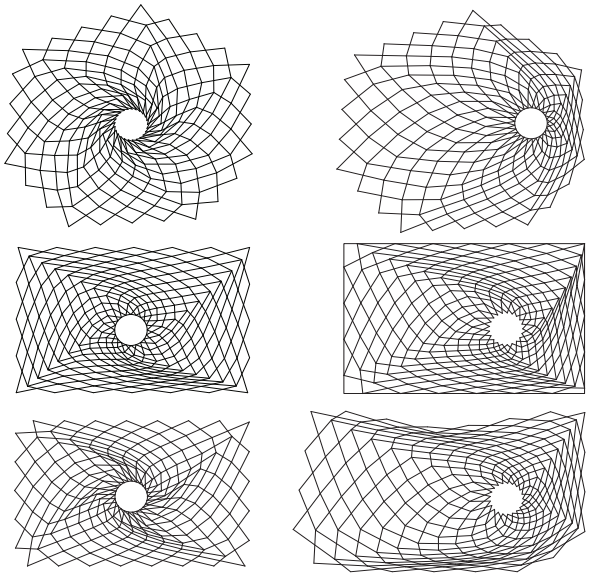
Twisted Lines

Biological inspiration was behind the design of this spiraled pattern (inset below). A graphical inspection of the system revealed a simple organization of divisions and a shift in orientation to distribute a spiraled matrix across a surface. The plans provided here deal with propagation in two dimensions. However, the theory is sound and can be applied to 3D.

Above, the spiral matrix is written with Visual Basic (VB) in MicroStation by Professor Kyle Talbott. A series of circle divisions are achieved with rotation of a line which are then subdivided themselves.

The matrix is critical in the script to record points of intersection that will connect later as the instructions carry on. Iterations are nearly endless but the real life application, interaction with existing conditions, severely complicate the script writing process.

The circular shape from above is irrelevant in the taxonomy shown at right. Contrasting the VB script, Grasshopper uses the same logic which is easily modified to local parameters. In addition, by directly interacting with the Rhinoceros modeling platform, the 2d plan for this script changes and adapts easily to the designers wishes.



source_files
- 08_SpiralMatrix
  - 28Mar2010_PolarMatrix_KyleTalbott.dgn
  - 28Mar2010_PolarMatrix_v3_KyleTalbott.mvba

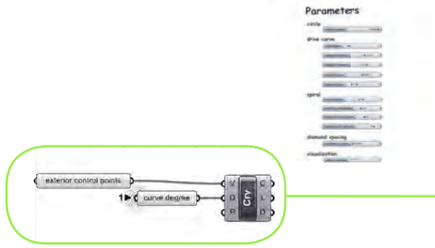


figure 13

Grasshopper interacts with environmental parameters from the beginning (figure 13). Versatility is increased in figure 14 with the addition of a Bezier curve to subtly introduce a wave of variation.
The end result is a plan which dynamically expresses balance, without the rigidity of the initial VB script.

Because Grasshopper uses plug-in components to build each script, variation and change, from a circle to a polygon or a line to a curve, can be substituted at will to develop active and intriguing architectural solutions.

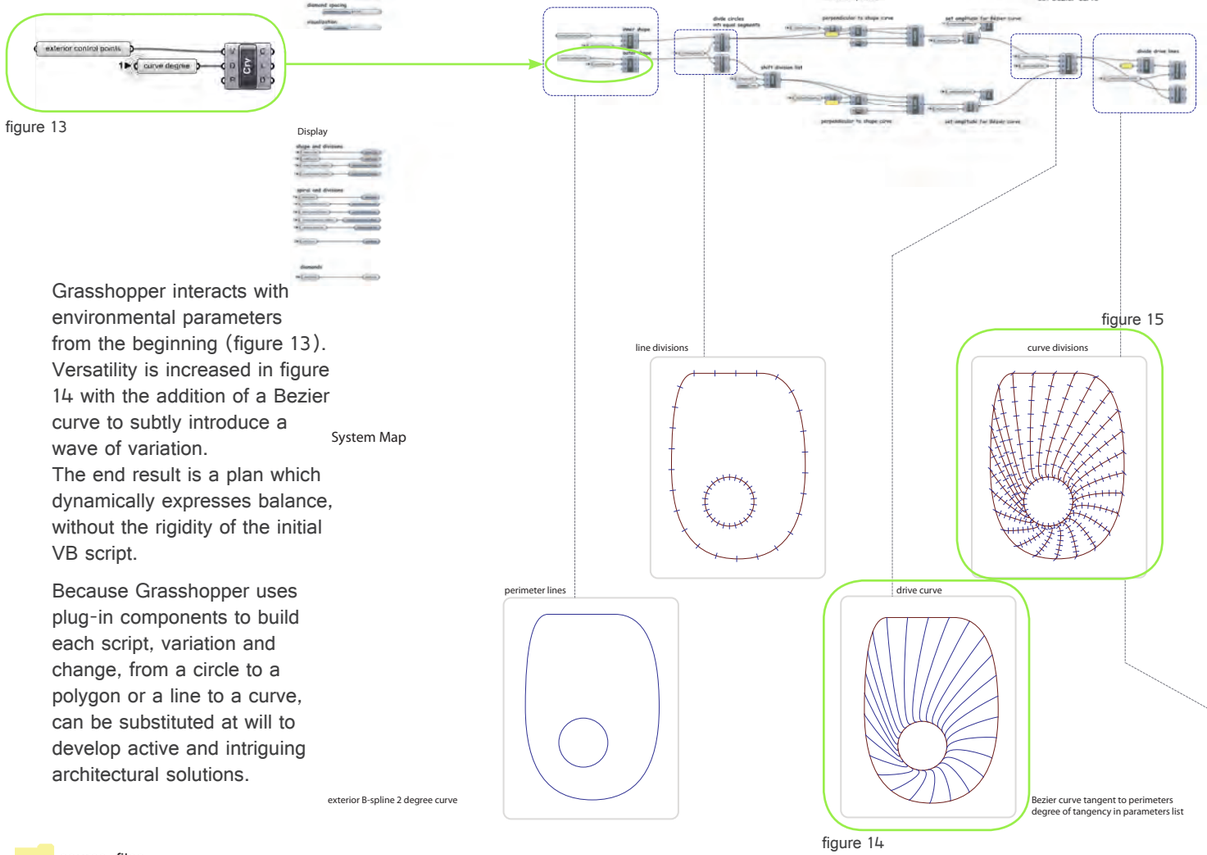


figure 14

figure 15

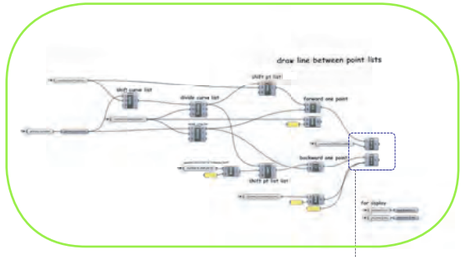


figure 16

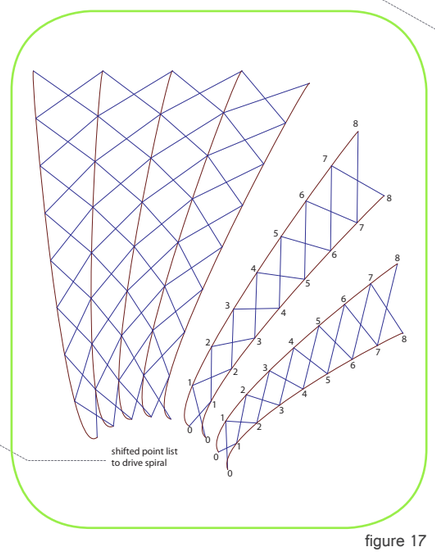
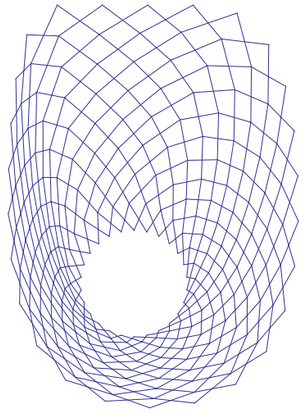


figure 17



The main body of the spiral is shown in figure 16; the degree of intensity the spiral holds occurs here when the point lists are shifted. Note the vertical symmetry; essentially, inverting the number to shift achieves the proper result. To facilitate processing for the entire set, the shift boolean must be set to true.

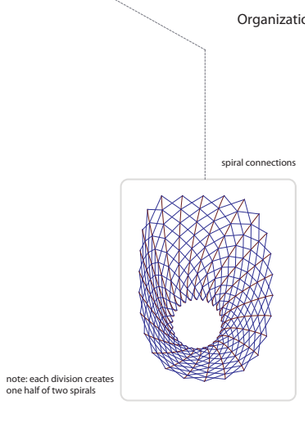


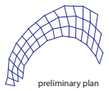
Figure 17 illustrates how similar, in logic, the Grasshopper and VB scripts are; learning how to develop designs in one program facilitates understanding the other[7]. In fact, once practiced in both languages, script power magnifies exponentially. Without experience in RhinoScript, a Visual basic language, the spiral design was limited to equal divisions along the “drive curve” highlighted in figure 17. For the purposes of this application, the plan for a ceiling prototype, the existing divisions proved varied enough to be successful.

source_files
08_SpiralMatrix
19April2010_SpiralMatrix_KevinHinz.3dm
20April2010_SpiralMatrix_KevinHinz.ghx

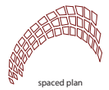
[7]Look to the Grasshopper Primer, ch 14: Visual Basic DotNET for exercises in integrating VB script directly into Grasshopper.

Diamonds: plan

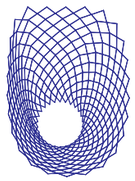
the first full scale prototype uncovered a critical design flaw: spacing for draped material as well as human error in assembly



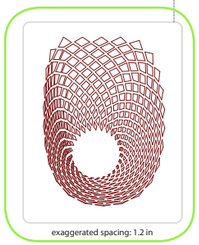
preliminary plan



spaced plan

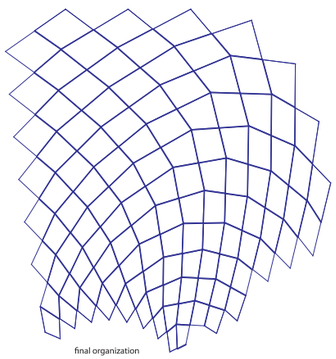


final spacing: 0.2 in average



exaggerated spacing: 1.2 in

figure 18



final organization

Clearance Allowances

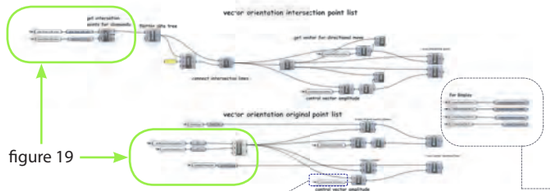


figure 19

Diamonds: odd numbered

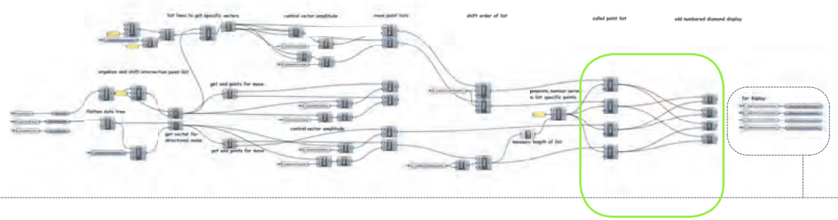
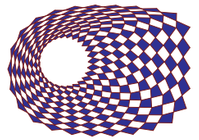
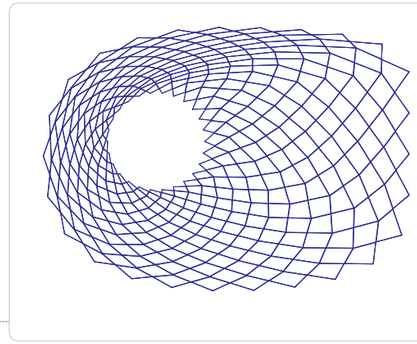
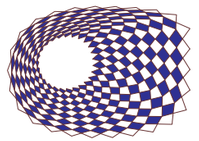


figure 20



Final Plan top view





Diamonds: even numbered

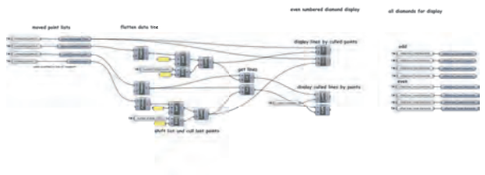


Figure 18 highlights a critical design flaw encountered during production tests. Each polygon in the spiral matrix represents the plan for a physical component to be fabricated. Material thicknesses had to be calculated into the computer designed plan. Space between components had to be added; at this point, lessons from previous exercises, of vector and data management, became apparent.

In essence, intersection points were isolated, related to its neighbor point and moved a determined distance from its point of origin. Figure 19 highlights the beginning of this relationship. Once shifted, the points had to once again be connected; organization of the script quickly develops complexity.

Refer to chapters 6 & 8 of the *Grasshopper Primer* for explanation of the way Grasshopper manages data. The new found point lists had to be re-combined to form a new set of lines; the Flatten component is used to achieve the combination. Once flattened and joined, the list is simple duplicated and shifted before drawing a new plan (figure 20).

Note that as experience in script-writing progresses through out this publication, detailed notations such as the location of the Flatten component (Logic / Tree / Flatten) become redundant to notate. The lack of notation is testament to the intuitive nature of Grasshopper's layout, suggesting that as experience with geometric manipulation and data management grows, so does the speed at which one can navigate the Grasshopper interface. Navigation is improved through analytical thought about what one is attempting to accomplish rather than familiarity of the interface; interaction and overall understanding of the program is likewise cultivated.

source_files
- 08_SpiralMatrix
  - 19April2010_SpiralMatrix_KevinHinz.3dm
  - 20April2010_SpiralMatrix_KevinHinz.ghx

As mentioned earlier, the script outlined here has the added benefit of manifesting as a built project. The computer work outlined above did not take the 3rd dimension but manually adjusted plans and analog jigs were developed closely together with the work in Grasshopper. The combined efforts of material exploration and the digital production of geometry can be attributed to the project's success. Designed as a ceiling prototype, the project is currently on display in a Milwaukee, Wisconsin night club awaiting scheduled shows at galleries across the city.

source_files
04_CurvesLinesData
28Mar2010_PolarMatrix_KyleTalbott.dgn
28Mar2010_PolarMatrix_v3_KyleTalbott.mvba
19April2010_SpiralMatrix_KevinHinz.3dm
20April2010_SpiralMatrix_KevinHinz.ghx

[1]Chapter 11 in the Grasshopper Primer provides more complex applications. Payne and Issa provide a series of basic construction script which can be built upon, manipulated and modified to develop designs such as the surface features illustrated in chapter 7, Curves and Surfaces.

[2]Rutten provides illustrations in chapter 8 of the Grasshopper Primer to demonstrate how Grasshopper combines and displays data streams. Payne and Issa also show the significance of available components to manage data.

[3]See also the .csv, comma separated values, files for streamed data output. Create your own .csv files when recreating the definition.

[4]See chapter 2 of this publication for additional matrix and attractor information. For the definition provided here, the attractor script from the 03Feb2010_PointMatrix file was copied, pasted and reused in this example.

[5]The source file will provide annotations outlining the components' locations and operational purposes; more detailed functional descriptions for the definition are provided.

[6]Step by step descriptions are provided in the 22Feb2010_BridgeConnection . ghx file. Once a design is imagined, the detailed logic, here the square at the end of a line sharing the length of the square edge, can be broken down into segments. These segments are then used to both create geometry and to navigate the Grasshopper interface according to each component needed to produce that geometry.

Works Cited

Payne, Andrew, and Rajaa Issa. "The Grasshopper Primer - Second_Edition_090323." *LIFT architects - Home - [Andrew Payne]*. Web. 27 Nov. 2009. <http://www.liftarchitects.com/journal/2009/3/25/the-grasshopper-primer-second-edition.html>.

The authors provide detailed explanations concerning the parameters and components available in the most recent version of Grasshopper. Exercises and source files accompany these explanations, also available at the above listed site. In conclusion, the authors provide an overview of the Rhino operating platform, discussing the way it structures and manages NURBS geometry. NOTE: The 3rd Edition for version 0.6.0059 is in progress and can be expected to be found at Payne's website when available: *Grasshopper Primer* for version 0.6.0007, 2009 http://www.liftarchitects.com/downloads/

Khabazi, Zubin M. Generative Algorithms with Grasshopper. Web. 26 July 2010. <http://www.grasshopper3d.com/page/tutorials-1>.

Khabazi delivers introductory lessons to explain not only the Grasshopper interface but also the theory behind the generation of algorithmic objects. Methods, component location and data management are discussed before the exercises expand to include complex geometric productions and architectural examples.
Algorithmic Modeling with Grasshopper, 2009 http://download.mcneel.com/s3/mcneel/grasshopper/1.0/docs/en/Generative%20Algorithms.pdf

Additional Resources

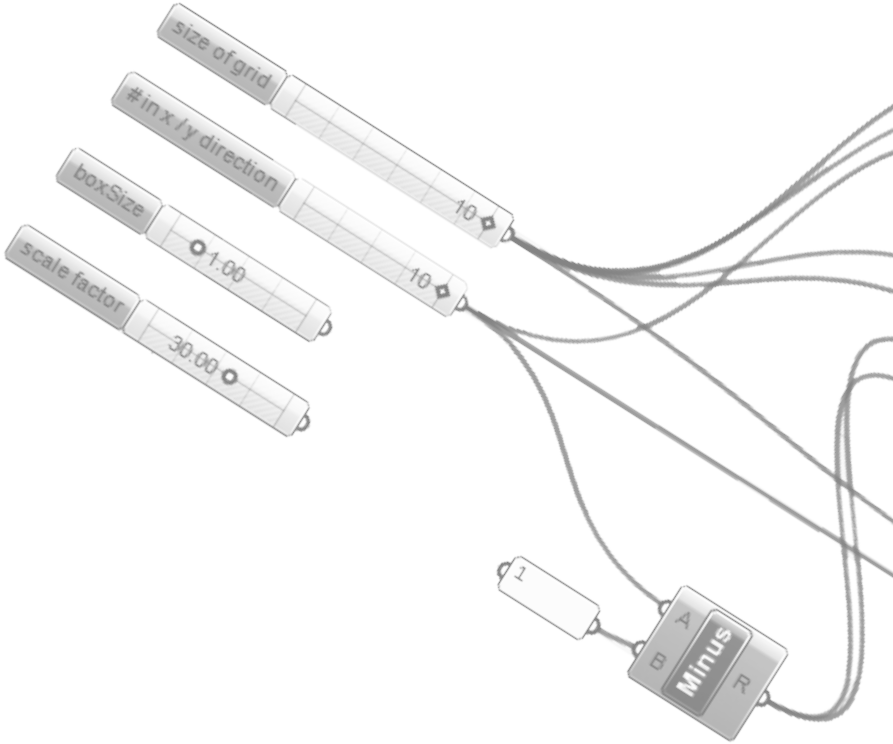
Davidson, Scott. *Grasshopper - Generative Modeling for Rhino*. Web. 27 Nov. 2009. <http://www.grasshopper3d.com/>.

The Grasshopper home page provides a beginning source for all documents and discussions relating officially to the program. Numerous tutorial links, connections to blogs concerning problems and questions can be found at this centralized source.

González, Andrés, with McNeel. "GH." *Grasshopper Visual Introduction*. Mcneel North America. Web. 27 Nov. 2009. <http://web.mac.com/rhino3dtv/GH/GH.html>.

Primary Grasshopper introduction website sponsored by McNeel, producer of the Rhinoceros platform.

The information provided includes but is not limited to the latest tutorials and highly articulated graphical explanations.



ISBN 978-0-578-07439-9
52400>
9 780578 074399